
Dear PyGui

Jonathan Hoffstadt and Preston Cothren

May 13, 2026

ABOUT DPG

1	About DPG	1
1.1	What & Why	1
1.2	Project Information	2
2	Quick Start	3
2.1	First Steps	3
2.2	DPG Structure Overview	4
2.3	Item Usage	7
2.4	Tips & More Resources	14
3	Documentation	19
3.1	Render Loop	19
3.2	Viewport	20
3.3	Primary Window	20
3.4	IO, Handlers, State Polling	20
3.5	Item Creation	23
3.6	Tag System	27
3.7	Item Configuration	29
3.8	Item Callbacks	29
3.9	Values	33
3.10	Containers & Context Managers	34
3.11	Container Slots & Children	38
3.12	Container Stack	39
3.13	Drawing-API	41
3.14	File & Directory Selector	50
3.15	Filter Set	53
3.16	Fonts	54
3.17	Init Files	56
3.18	Menu Bar	57
3.19	Node Editor	59
3.20	Plots	61
3.21	Popups	73
3.22	Simple Plots	75
3.23	Staging	76
3.24	Tables	79
3.25	Textures & Images	89
3.26	Themes	95
3.27	Tooltips	103
3.28	dearpygui.dearpygui	103

4 More	279
4.1 Showcase	279
4.2 Video Tutorials	280
4.3 Glossary	284
Python Module Index	285
Index	287

ABOUT DPG

Dear PyGui is an easy-to-use, dynamic, GPU-Accelerated, cross-platform graphical user interface toolkit(GUI) for Python. It is “built with” *Dear ImGui*.

Features include traditional GUI elements such as buttons, radio buttons, menus and various methods to create a functional layout.

Additionally, DPG has an incredible assortment of dynamic plots, tables, drawings, debugger, and multiple resource viewers.

DPG is well suited for creating simple user interfaces as well as developing complex and demanding graphical interfaces.

DPG offers a solid framework for developing scientific, engineering, gaming, data science and other applications that require fast and interactive interfaces.

1.1 What & Why

1.1.1 What is DPG

DPG is a simple, bloat-free, and powerful Python GUI framework.

DPG is built with *Dear ImGui* and other extensions in order to create a unique retained mode API, as opposed to *Dear ImGui*'s immediate mode paradigm.

Under the hood, DPG uses the immediate mode paradigm allowing for extremely dynamic interfaces. Similar to *PyQt*, DPG does not use native widgets but instead draws the widgets using your computer's graphics card (using DirectX11, Metal, and Vulkan rendering APIs).

In the same manner *Dear ImGui* provides a simple way to create tools for game developers, DPG provides a simple way for python developers to create quick and powerful GUIs for scripts.

1.1.2 Why use DPG

When compared with other Python GUI libraries DPG is unique with:

- GPU rendering
- Multithreaded
- Highly customizable
- Built-in developer tools: theme inspection, resource inspection, runtime metrics
- 70+ widgets with hundreds of widget combinations

- Detailed documentation, examples and support

1.2 Project Information

Socials:

| | [Github Discussions](#) | [Youtube](#) |

Funding:

Repository:

Issues:

CI Builds:

CI Static Analysis:

Platforms:

Header row, column (header rows)	Rendering API	Latest Version
Windows 10	DirectX 11	
macOs	Metal	
Linux	OpenGL 3	
Raspberry Pi 4	OpenGL ES	

QUICK START

If you're ready to start using DPG visit the *First Steps* in tutorials.

The *Tutorials* will provide a great overview and links to each topic in the API Reference for more detailed reading.

However, use the API reference for the most detailed documentation on any specific topic.

2.1 First Steps

Tutorials will give a broad overview and working knowledge of DPG. Tutorials do not cover every detail so refer to the documentation on each topic to learn more.

2.1.1 Installing

Python 3.6 (64 bit) or above is required.

```
pip install dearpygui
```

2.1.2 First Run

Confirm the pip install by running the code block below.

Code:

```
import dearpygui.dearpygui as dpg

dpg.create_context()
dpg.create_viewport(title='Custom Title', width=600, height=300)

with dpg.window(label="Example Window"):
    dpg.add_text("Hello, world")
    dpg.add_button(label="Save")
    dpg.add_input_text(label="string", default_value="Quick brown fox")
    dpg.add_slider_float(label="float", default_value=0.273, max_value=1)

dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

Result:

2.1.3 Demo

DPG has a complete built-in demo/showcase. It is a good idea to look into this demo. The code for this can be found in the repo in the `demo.py` file

Code:

```
import dearpygui.dearpygui as dpg
import dearpygui.demo as demo

dpg.create_context()
dpg.create_viewport(title='Custom Title', width=600, height=600)

demo.show_demo()

dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

Result:

Note: The main script must always:

- Create the context `create_context`
 - Create the viewport `create_viewport`
 - Setup dearpygui `setup_dearpygui`
 - Show the viewport `show_viewport`
 - Start dearpygui `start_dearpygui`
 - Clean up the context `destroy_context`
-

2.2 DPG Structure Overview

A DPG app will have an overall structure as follows:

- Setup
- Context
- Viewport
- Render Loop
- Items
- Primary Window

2.2.1 Setup

All DPG apps must do 3 things:

- Create & Destroy context
- Create & Show Viewport
- Setup & Start DearPyGui

2.2.2 The Context

To access any DPG commands the context must be created with `create_context`. This should be the first DPG command and it's typical to perform this with an import.

Proper clean up of DPG can be done using `destroy_context`.

Creating and destroying the context and also setup and start dearpygui are useful when the DPG needs to be started and stopped multiple times in one python session.

Warning: If `create_context` is not first DPG will not start (and will probably crash).

2.2.3 The Viewport

The viewport is the *window* created by the operating system.

The viewport needs to be explicitly created using `create_viewport` and shown using `show_viewport`

Code:

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Example Window"):
    dpg.add_text("Hello, world")
    dpg.add_button(label="Save")
    dpg.add_input_text(label="string", default_value="Quick brown fox")
    dpg.add_slider_float(label="float", default_value=0.273, max_value=1)

dpg.create_viewport(title='Custom Title', width=600, height=200)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

See also:

For more information on the viewport *Viewport*

2.2.4 The Render Loop

The render loop is responsible for displaying items, partially maintaining state and callbacks.

The render loop is completely handled by the `start_dearpygui` command.

In some cases it's necessary to explicitly create the render loop so you can call python commands that may need to run every frame. Such as per-frame ticker or counter update functions.

Code:

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Example Window"):
    dpg.add_text("Hello, world")
    dpg.add_button(label="Save")
    dpg.add_input_text(label="string", default_value="Quick brown fox")
    dpg.add_slider_float(label="float", default_value=0.273, max_value=1)

dpg.create_viewport(title='Custom Title', width=600, height=200)
dpg.setup_dearpygui()
dpg.show_viewport()

# below replaces, start_dearpygui()
while dpg.is_dearpygui_running():
    # insert here any code you would like to run in the render loop
    # you can manually stop by using stop_dearpygui()
    print("this will run every frame")
    dpg.render_dearpygui_frame()

dpg.destroy_context()
```

Warning: The manual render loop must be created after `setup_dearpygui`

See also:

For more information on the render loop *Render Loop*

2.2.5 Item Overview

DPG can be broken down into **Items**, **UI Items**, **Containers**

Items: Items are anything in the library (i.e. button, registries, windows, etc).

UI Items: Any item in DPG that has a visual component (i.e. button, listbox, window, etc).

Containers: Items that can hold other items. (i.e. window, groups, registries, etc).

2.2.6 The Primary Window

DPG can assign one window to be the *primary window*. The primary window will fill the viewport and always be drawn behind other windows.

Code:

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(tag="Primary Window"):
    dpg.add_text("Hello, world")
    dpg.add_button(label="Save")
    dpg.add_input_text(label="string", default_value="Quick brown fox")
    dpg.add_slider_float(label="float", default_value=0.273, max_value=1)

dpg.create_viewport(title='Custom Title', width=600, height=200)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.set_primary_window("Primary Window", True)
dpg.start_dearpygui()
dpg.destroy_context()
```

See also:

for more information on the primary window *Primary Window*

2.3 Item Usage

2.3.1 Creating Items

Items are created using their *add_**** commands.

All items must have a tag which can either be specified or are automatically generated by DPG.

Tags can be either integers or strings and are used to refer to the item after it has been created.

Items return their tag when they are created.

Warning: Item tags must be unique if specified using the *tag* keyword. Integers 0-10 are reserved for DPG internal items.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Tutorial"):
    b0 = dpg.add_button(label="button 0")
    b1 = dpg.add_button(tag=100, label="Button 1")
    dpg.add_button(tag="Btn2", label="Button 2")
```

(continues on next page)

(continued from previous page)

```
print(b0)
print(b1)
print(dpg.get_item_label("Btn2"))

dpg.create_viewport(title='Custom Title', width=600, height=200)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

Note: Items can be created and delete at runtime see *Item Creation*

See also:

For more information on the creating items:

Item Creation

Tag System

2.3.2 Creating Containers

Below we will add a window, a group and a child window to the code. Items can either be added directly to the context manager or later by specifying the parent.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Tutorial"):
    dpg.add_button(label="Button 1")
    dpg.add_button(label="Button 2")
    with dpg.group():
        dpg.add_button(label="Button 3")
        dpg.add_button(label="Button 4")
        with dpg.group() as group1:
            pass
    dpg.add_button(label="Button 6", parent=group1)
    dpg.add_button(label="Button 5", parent=group1)

dpg.create_viewport(title='Custom Title', width=600, height=400)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

See also:

For more information on containers:

Item Creation

Containers & Context Managers
Container Slots & Children
Container Stack

2.3.3 Configuration, State, Info

DPG items consist of configuration, state and info. (AND value but we will cover that separately)

Each of these can be accessed by their corresponding function

`get_item_configuration` keywords that control its appearance and behavior (label, callback, width, height)

`get_item_state` keywords that reflect its interaction (visible, hovered, clicked, etc)

`get_item_info` keywords that reflect its information (item type, children, theme, etc)

Note: configuration, state and info are broken into separate commands that access each individual keyword, instead of returning the entire dictionary.

Examples:

```
get_item_label  
is_item_hovered  
get_item_children
```

Below we will demonstrate the ways to configure items and how to check their state by viewing them through the item registry tool.

Code:

```
import dearpygui.dearpygui as dpg  
  
dpg.create_context()  
  
with dpg.window(label="Tutorial"):  
  
    # configuration set when button is created  
    dpg.add_button(label="Apply", width=300)  
  
    # user data and callback set any time after button has been created  
    btn = dpg.add_button(label="Apply 2")  
    dpg.set_item_label(btn, "Button 57")  
    dpg.set_item_width(btn, 200)  
  
dpg.show_item_registry()  
  
dpg.create_viewport(title='Custom Title', width=800, height=600)  
dpg.setup_dearpygui()  
dpg.show_viewport()  
dpg.start_dearpygui()  
dpg.destroy_context()
```

See also:

For more information on the these topics:

Item Configuration

IO, Handlers, State Polling

2.3.4 Callbacks

Callbacks give items functionality by assigning a function to run when they are activated and almost all UI Items in DPG can run callbacks.

Functions or methods are assigned as UI item callbacks when an item is created or at a later runtime using `set_item_callback`

Callbacks may have up to 3 arguments in the following order.

sender: the *id* of the UI item that submitted the callback

app_data: occasionally UI items will send their own data (ex. file dialog)

user_data: any python object you want to send to the function

Note: Because they are optional positional arguments you must use the *sender* and *app_data* if you want to use *user_data* standard keyword arguments

Code:

```
import dearpygui.dearpygui as dpg

dpg.create_context()

def button_callback(sender, app_data, user_data):
    print(f"sender is: {sender}")
    print(f"app_data is: {app_data}")
    print(f"user_data is: {user_data}")

with dpg.window(label="Tutorial"):
    # user data and callback set when button is created
    dpg.add_button(label="Apply", callback=button_callback, user_data="Some Data")

    # user data and callback set any time after button has been created
    btn = dpg.add_button(label="Apply 2", )
    dpg.set_item_callback(btn, button_callback)
    dpg.set_item_user_data(btn, "Some Extra User Data")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

See also:

For more information on the item callbacks *Item Callbacks*

2.3.5 Item Values

Almost all UI items have a *value* which can be accessed or set.

All UI items that have a *value* also have the *default_value* parameter which will set the items' initial starting value.

Values can be accessed using *get_value*.

Below is an example of setting the *default_value* for two different items, setting a callback to the items and printing their values.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

def print_value(sender):
    print(dpg.get_value(sender))

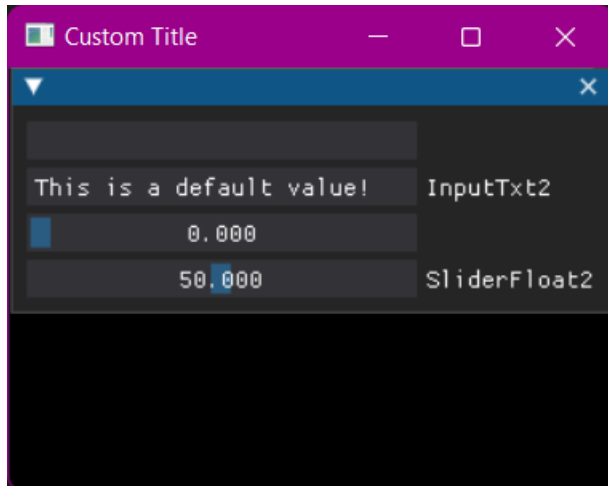
with dpg.window(width=300):
    input_txt1 = dpg.add_input_text()
    # The value for input_text2 will have a starting value
    # of "This is a default value!"
    input_txt2 = dpg.add_input_text(
        label="InputTxt2",
        default_value="This is a default value!",
        callback=print_value
    )

    slider_float1 = dpg.add_slider_float()
    # The slider for slider_float2 will have a starting value
    # of 50.0.
    slider_float2 = dpg.add_slider_float(
        label="SliderFloat2",
        default_value=50.0,
        callback=print_value
    )

    dpg.set_item_callback(input_txt1, print_value)
    dpg.set_item_callback(slider_float1, print_value)

    print(dpg.get_value(input_txt1))
    print(dpg.get_value(input_txt2))
    print(dpg.get_value(slider_float1))
    print(dpg.get_value(slider_float2))

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```



An input item's value is changed by interacting with it. In the above example, moving `slider_float1` slider to 30.55 sets its' value to 30.55.

We can set the position of the slider by changing items' value at runtime using `set_value`.

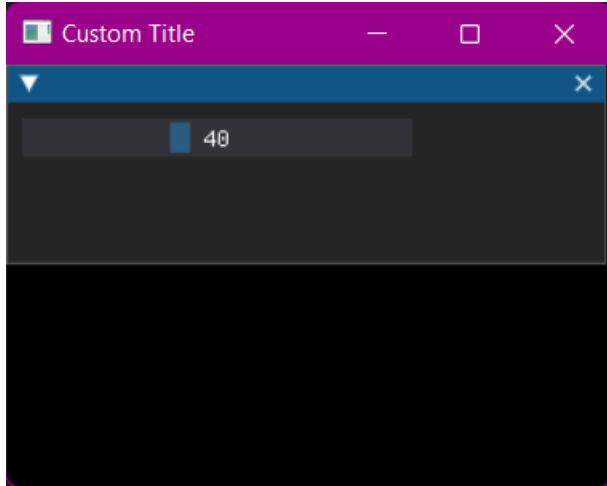
```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(width=300):
    # Creating a slider_int widget and setting the
    # default value to 15.
    dpg.add_slider_int(default_value=15, tag="slider_int")

    # On second thought, we're gonna set the value to 40
    # instead - for no reason in particular...
    dpg.set_value("slider_int", 40)

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```



Note: The values' type depends on the widget. (ex.) `input_int` default value needs to be an integer.

See also:

For more information on item values [Values](#)

2.3.6 Using Item Handlers

UI item handlers listen for events (changes in state) related to a UI item then submit a callback.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

def change_text(sender, app_data):
    dpg.set_value("text item", f"Mouse Button ID: {app_data}")

with dpg.window(width=500, height=300):
    dpg.add_text("Click me with any mouse button", tag="text item")
    with dpg.item_handler_registry(tag="widget handler") as handler:
        dpg.add_item_clicked_handler(callback=change_text)
    dpg.bind_item_handler_registry("text item", "widget handler")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

See also:

For more information on item handlers [IO](#), [Handlers](#), [State Polling](#)

2.4 Tips & More Resources

2.4.1 Developer Tools

DPG includes several tools which can help develop and debug applications.

Code:

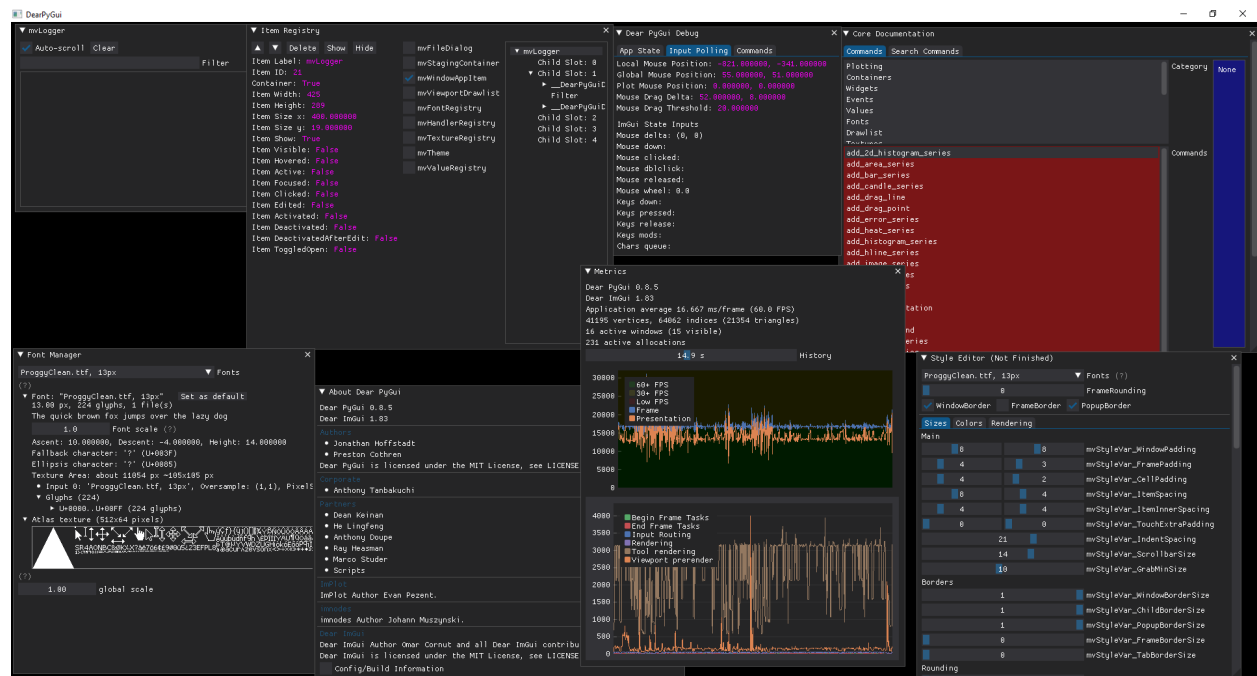
```
import dearpygui.dearpygui as dpg

dpg.create_context()

dpg.show_documentation()
dpg.show_style_editor()
dpg.show_debug()
dpg.show_about()
dpg.show_metrics()
dpg.show_font_manager()
dpg.show_item_registry()

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

Results



2.4.2 Style editor

The built-in style editor allows you to experiment with all style options at runtime to find the exact colors, padding, rounding and other style settings for your application. You can use the sliders to change the settings, which are applied to all items in your app, so you can immediately see what effect the changes have on your GUI.

Code:

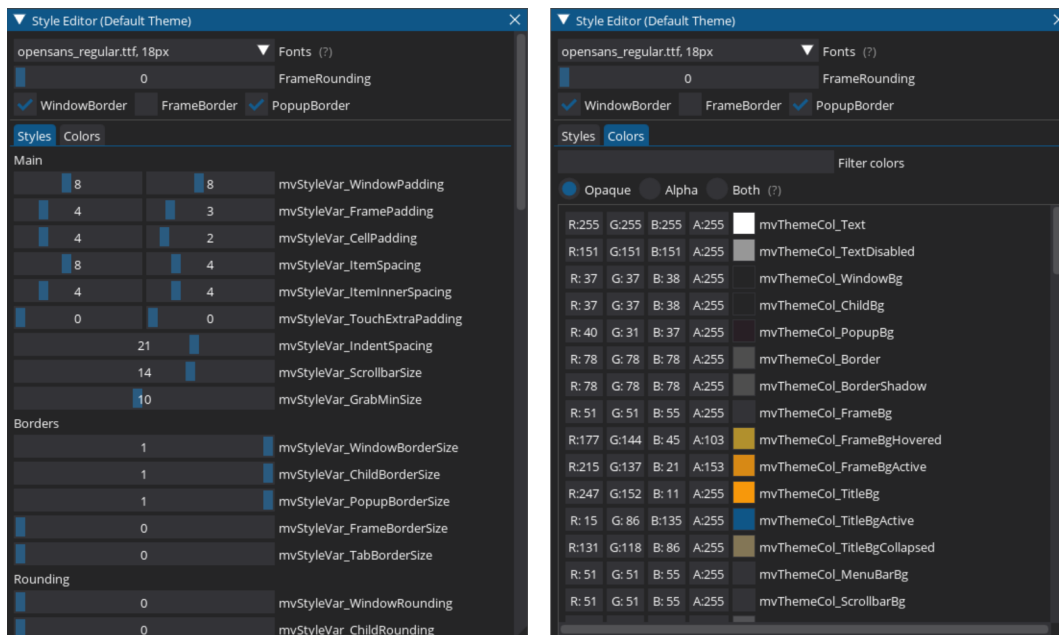
```
import dearpygui.dearpygui as dpg

dpg.create_context()
dpg.create_viewport()
dpg.setup_dearpygui()

dpg.show_style_editor()

dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

Result:



2.4.3 Item registry

The item registry shows all items of a running application in a hierarchical structure. For each item, a number of details, such as its tag ID, are shown.

Code:

```
import dearpygui.dearpygui as dpg

dpg.create_context()
dpg.create_viewport()
```

(continues on next page)

(continued from previous page)

```
dpg.setup_dearpygui()
dpg.show_item_registry()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

Result:

2.4.4 Font manager

The font manager shows all loaded fonts and their appropriate sizes. It allows you to inspect all characters, or glyphs, that are loaded with each font file.

Code:

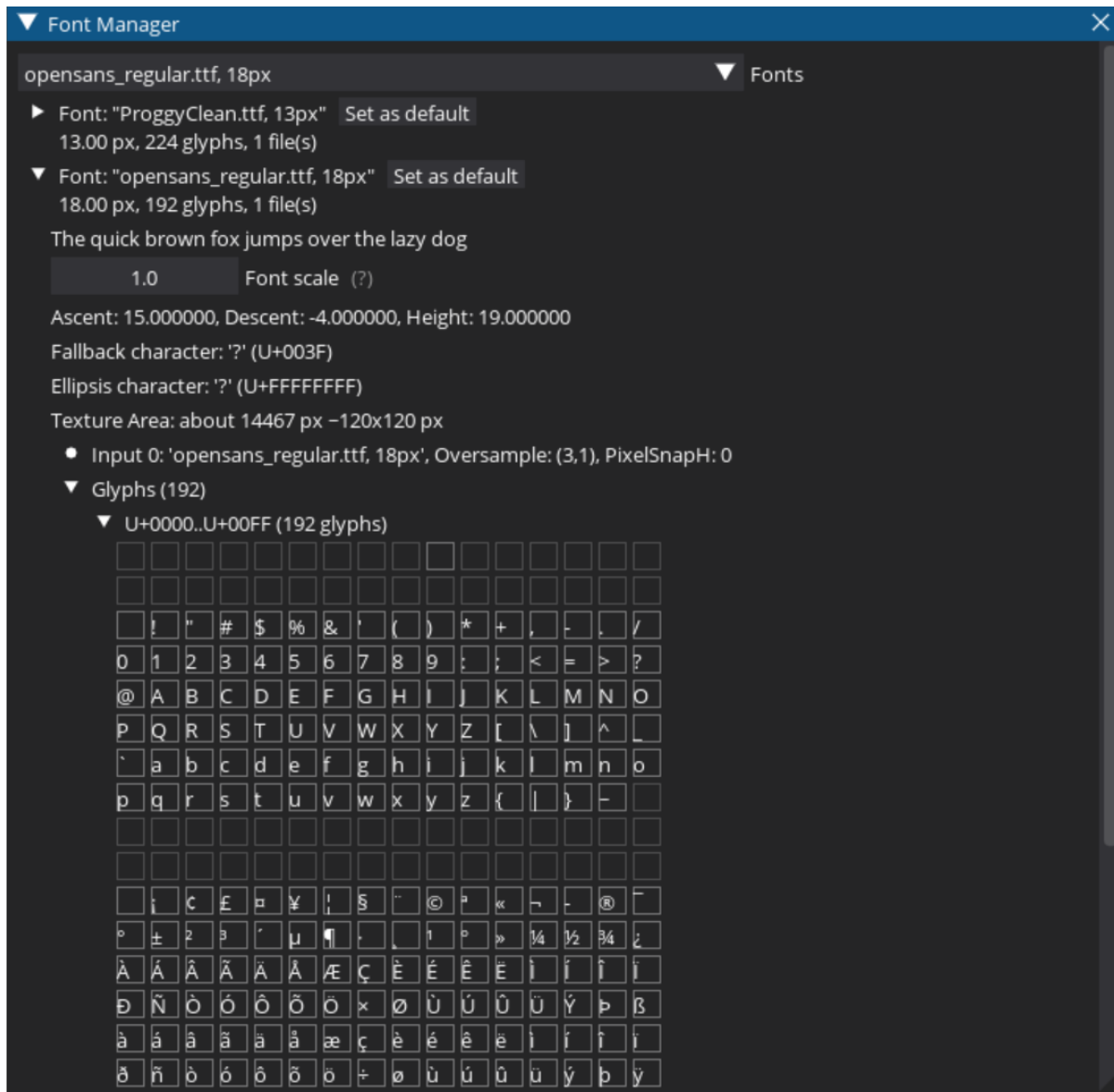
```
import dearpygui.dearpygui as dpg

dpg.create_context()
dpg.create_viewport()
dpg.setup_dearpygui()

dpg.show_font_manager()

dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

Result:



2.4.5 Runtime metrics

Runtime metrics show the performance of your app in real-time. Here is it shown in conjunction with the built-in style editor.

Code:

```
import dearpygui.dearpygui as dpg

dpg.create_context()
dpg.create_viewport(title='Custom Title', width=800, height=600)

dpg.show_style_editor()
dpg.show_metrics()
```

(continues on next page)

(continued from previous page)

```
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

Result:

2.4.6 More Resources

- *Showcase*
- *Video Tutorials*

DOCUMENTATION

Live Demo: A mostly complete showcase of DPG can be found by running the `show_demo` command in the `dearpygui.demo` module.

Internal Documentation: Run `show_documentation`

API Reference Guide: *Online API Reference*

3.1 Render Loop

The render loop (or event loop) runs continuously and is responsible for polling user input and drawing widgets.

Drawing the items is how the DPG appears to update items. DPG does this at the rate of your monitor refresh when `set_viewport_vsync` is set **True**. If `vsync` is set **False** the render loop will run as fast possible.

If you try to run too many computationally expensive operations inside the render loop, you may reduce the frame rate of your application.

For most use cases the render loop does not need to be considered and is completely handled by `start_dearpygui`.

For more advanced use cases full access to the render loop can be accessed like so:

```
import dearpygui.dearpygui as dpg

dpg.create_context()
dpg.create_viewport(title='Custom Title', width=600, height=200)
dpg.setup_dearpygui()

with dpg.window(label="Example Window"):
    dpg.add_text("Hello, world")

dpg.show_viewport()

# below replaces, start_dearpygui()
while dpg.is_dearpygui_running():
    # insert here any code you would like to run in the render loop
    # you can manually stop by using stop_dearpygui()
    print("this will run every frame")
    dpg.render_dearpygui_frame()

dpg.destroy_context()
```

3.2 Viewport

The viewport is what you traditionally call a **window** in other GUI libraries.

In the case of DPG we call the operating system window the *viewport* and the DPG windows as *windows*.

Before calling `start_dearpygui`, you must do the following:

1. Create a viewport, using `create_viewport`.
2. Assign the viewport, using `setup_dearpygui`.
3. Show the viewport, using `show_viewport`.

Once the viewport has been created, you can begin configuring the viewport using `configure_viewport` or the helper commands `set_viewport_***`.

Note: Large/small icon must be set before showing the viewport (i.e. you will need to setup the viewport manually).

3.3 Primary Window

The primary window fills the viewport, resizes with the viewport and remains behind other windows.

A window can be set as the primary window by using the `set_primary_window` command using the required True/False allows the window to be set or unset.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(tag="Primary Window"):
    dpg.add_text("Hello, world")

dpg.create_viewport(title='Custom Title', width=600, height=200)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.set_primary_window("Primary Window", True)
dpg.start_dearpygui()
dpg.destroy_context()
```

3.4 IO, Handlers, State Polling

3.4.1 Handlers

Handlers are items that submit a callback when the specified state of an item changes.

Handlers can be activated or deactivated by showing or hiding them.

Handlers are required to be added to a handler registry.

3.4.2 Item Handlers

Item handlers listen for states related to a specific item.

Events:

- Activated
- Active
- Clicked
- Deactivated
- Deactivated After Edited
- Focus
- Hover
- Resize
- Toggled
- Visible

Item handlers are required to be added to a item handler registry.

Item handler registries can be bound to an item. They can be bound to multiple items to prevent having to duplicate handlers for every item.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

def change_text(sender, app_data):
    dpg.set_value("text item", f"Mouse Button ID: {app_data}")

def visible_call(sender, app_data):
    print("I'm visible")

with dpg.item_handler_registry(tag="widget handler") as handler:
    dpg.add_item_clicked_handler(callback=change_text)
    dpg.add_item_visible_handler(callback=visible_call)

with dpg.window(width=500, height=300):
    dpg.add_text("Click me with any mouse button", tag="text item")
    dpg.add_text("Close window with arrow to change visible state printing to console",
    ↪tag="text item 2")

# bind item handler registry to item
dpg.bind_item_handler_registry("text item", "widget handler")
dpg.bind_item_handler_registry("text item 2", "widget handler")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.4.3 Global Handlers (IO Input)

Global handlers listen for actions not tied to a specific item:

Keys:

- Down
- Press
- Release

Mouse:

- Click
- Double Click
- Down
- Drag
- Move
- Release
- Wheel

Global handlers are required to be added to a handler registry.

Registries provide a grouping aspect to handlers allowing separation by input device. They also provide the ability to turn on and off the entire registry.

For example this can allow a mouse input registry or a keyboard input registry. Registries also give the ability to deactivate all their children handlers by simply hiding the registry.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

def change_text(sender, app_data):
    dpg.set_value("text_item", f"Mouse Button: {app_data[0]}, Down Time: {app_data[1]} ↵
↵seconds")

with dpg.handler_registry():
    dpg.add_mouse_down_handler(callback=change_text)

with dpg.window(width=500, height=300):
    dpg.add_text("Press any mouse button", tag="text_item")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.4.4 Polling Item State

Polling item state is accessible through `get_item_state`. These can be very powerful when combined with handlers as shown below.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

def change_text(sender, app_data):
    if dpg.is_item_hovered("text item"):
        dpg.set_value("text item", f"Stop Hovering Me, Go away!!")
    else:
        dpg.set_value("text item", f"Hover Me!")

with dpg.handler_registry():
    dpg.add_mouse_move_handler(callback=change_text)

with dpg.window(width=500, height=300):
    dpg.add_text("Hover Me!", tag="text item")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.5 Item Creation

DPG can be broken down into **Items**, **UI Items**, **Containers**.

3.5.1 Items

Everything created in DPG is an **item**. New items can be created by calling various `add_***` or `draw_***` functions. These commands return a unique identifier that can be used to later refer to the item. **UI items** and **containers** are also **items** - but not every **item** is necessarily a **UI item** or **container**.

All items have the following optional parameters: *label*, *tag*, *user_data*, and *use_internal_label*. The *tag* is generated automatically or can be specified. A *label* serves as the display name for an item. *user_data* can be any value and is frequently used for **callbacks**.

Note: Event **handlers**, **registries**, **group**, and **themes** are also items. These are under-the-hood items for customizing the functionality, flow, and overall look of your APP.

3.5.2 Containers

Container items that can contain other (allowable) items.

In addition to creating them by calling their corresponding `add_***` function, they can also be created by calling their corresponding context manager.

Note: Containers are more useful (and recommended) when used as context managers.

Below is an example of creating two new **window** items using their context manager and starting the application:

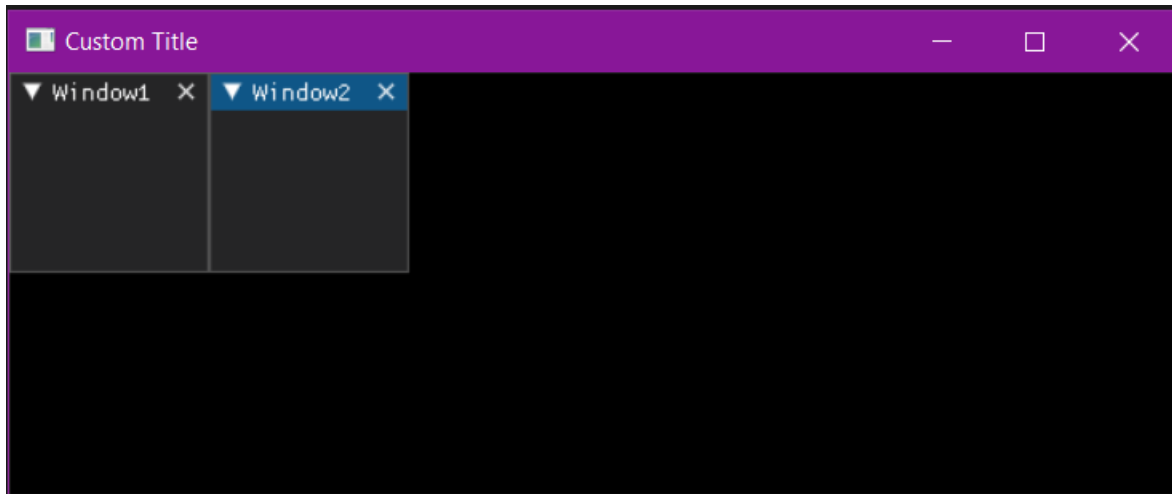
```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Window1", pos=(0,0)):
    pass

with dpg.window(label="Window2", pos=(100,0)):
    pass

dpg.create_viewport(title='Custom Title', width=600, height=200)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```



3.5.3 UI Items

UI items are items that are considered to be a visual and usually interactable element in your user interface.

These include **buttons**, **sliders**, **inputs**, and even other containers such as **windows** and **tree nodes**.

Below is an example for creating a **window** container that contains a few other items:

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Tutorial") as window:
    # When creating items within the scope of the context
    # manager, they are automatically "parented" by the
    # container created in the initial call. So, "window"
    # will be the parent for all of these items.

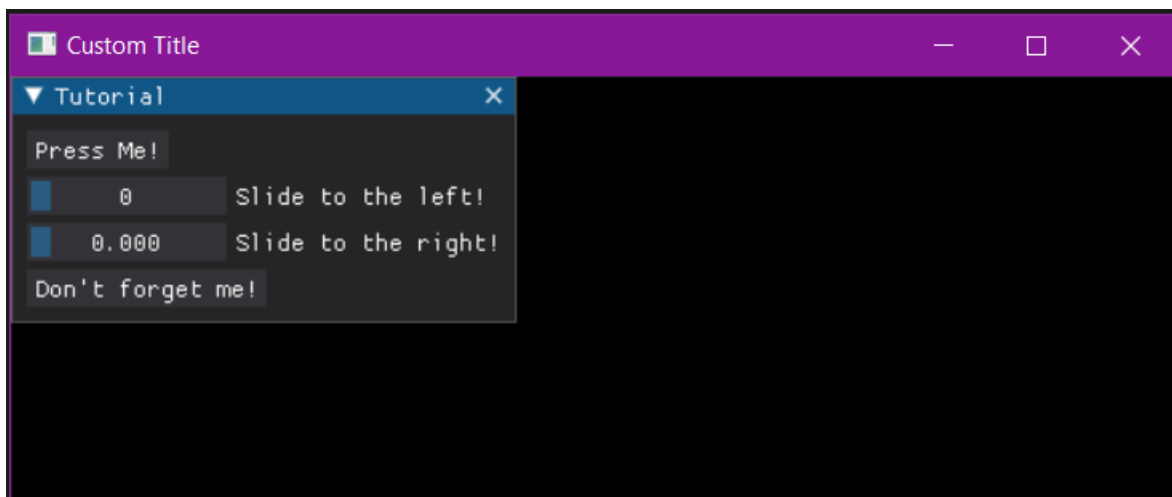
    button1 = dpg.add_button(label="Press Me!")

    slider_int = dpg.add_slider_int(label="Slide to the left!", width=100)
    slider_float = dpg.add_slider_float(label="Slide to the right!", width=100)

    # An item's unique identifier (tag) is returned when
    # creating items.
    print(f"Printing item tag's: {window}, {button1}, {slider_int}, {slider_float}")

# If you want to add an item to an existing container, you
# can specify it by passing the container's tag as the
# "parent" parameter.
button2 = dpg.add_button(label="Don't forget me!", parent=window)

dpg.create_viewport(title='Custom Title', width=600, height=200)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```



3.5.4 Runtime Adding and Deleting

With DPG you can dynamically add, delete, and move items at runtime.

This can be done by using a callback to run the desired item's `add_***` command and specifying the parent the item will belong to.

By using the **before** keyword when adding a item you can control which item in the parent the new item will come before. Default will place the new widget at the end.

Below is an example demonstrating adding and deleting items during runtime:

```
import dearpygui.dearpygui as dpg

dpg.create_context()

def add_buttons():
    global new_button1, new_button2
    new_button1 = dpg.add_button(label="New Button", before="delete_button", tag="new_
↪button1")
    new_button2 = dpg.add_button(label="New Button 2", parent="secondary_window", tag=
↪"new_button2")

def delete_buttons():
    dpg.delete_item("new_button1")
    dpg.delete_item("new_button2")

with dpg.window(label="Tutorial", pos=(200, 200)):
    dpg.add_button(label="Add Buttons", callback=add_buttons)
    dpg.add_button(label="Delete Buttons", callback=delete_buttons, tag="delete_button")

with dpg.window(label="Secondary Window", tag="secondary_window", pos=(100, 100)):
    pass

dpg.create_viewport(title='Custom Title', width=600, height=400)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

Hint: When deleting a container the container and its' children are deleted by default, unless the keyword **children_only** is set to True, i.e.:

```
import dearpygui.dearpygui as dpg

dpg.create_context()

def delete_children():
    dpg.delete_item("window", children_only=True)

with dpg.window(label="Tutorial", pos=(200, 200), tag="window"):
```

(continues on next page)

(continued from previous page)

```
dpg.add_button(label="Delete Children", callback=delete_children)
dpg.add_button(label="Button_1")
dpg.add_button(label="Button_2")
dpg.add_button(label="Button_3")

dpg.create_viewport(title='Custom Title', width=600, height=400)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.6 Tag System

In DPG, all items must have an associated unique ID (UUID) which can either be an integer or a string.

When a item is created, a tag is generated for you automatically. It is your responsibility to store this tag if you intend on interacting with the widget at a later time.

Tags allow for modification of the associated item at runtime.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

unique_id = 0 # to be filled out later

def callback():
    print(dpg.get_value(unique_id))

with dpg.window(label="Example"):
    dpg.add_button(label="Press me (print to output)", callback=callback)
    unique_id = dpg.add_input_int(label="Input")

dpg.create_viewport(title='Custom Title', width=600, height=200)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.6.1 Generated Tags

The previous example could also be handled by generating the tag beforehand like this:

```
import dearpygui.dearpygui as dpg

dpg.create_context()

unique_tag = dpg.generate_uuid()

def callback():
```

(continues on next page)

(continued from previous page)

```
print(dpg.get_value(unique_tag))

with dpg.window(label="Example"):
    dpg.add_button(label="Press me (print to output)", callback=callback)
    dpg.add_input_int(label="Input", tag=unique_tag)

dpg.create_viewport(title='Custom Title', width=600, height=200)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.6.2 Aliases

An alias is a string that takes the place of the regular `int` tag. Aliases can be used anywhere UUID's can be used. It is the user's responsibility to make sure aliases are unique.

A simple example can be seen below:

```
import dearpygui.dearpygui as dpg

dpg.create_context()

def callback():
    print(dpg.get_value("unique_tag"))

with dpg.window(label="Example"):
    dpg.add_button(label="Press me (print to output)", callback=callback)
    dpg.add_input_int(default_value=5, label="Input", tag="unique_tag")

dpg.create_viewport(title='Custom Title', width=600, height=200)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.6.3 Recent Tags

The most recent tag is stored for the last item, container, and root.

This is useful when the last item created may be done at run time, is anonymous or sometimes just for convenience.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Example"):
    with dpg.group():
        dpg.add_button(label="View the Terminal for item tags")
        print(dpg.last_item())
        print(dpg.last_container())
```

(continues on next page)

(continued from previous page)

```
print(dpg.last_root())

dpg.create_viewport(title='Custom Title', width=600, height=200)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.7 Item Configuration

In DPG various configuration options and flags can be set when items are created. There are several options common to all items (i.e. **show**) but most items have specific options.

In order to modify an item's configuration after being created, you can use the `configure_item` command in conjunction with the keyword from the item's `add_***` command. You can also retrieve an item's configuration in the form of a dictionary by using the `get_item_configuration` command.

3.7.1 Example

Simple usage can be found below:

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(width=500, height=300):
    dpg.add_button(enabled=True, label="Press me", tag="item")

    # at a later time, change the item's configuration
    dpg.configure_item("item", enabled=False, label="New Label")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.8 Item Callbacks

Most items have a callback which is submitted to a **queue of callbacks** when the item is interacted with.

Callbacks are used to give functionality to items. Callbacks can either be assigned to the item upon creation or after creation using `set_item_callback` as shown in the code below.

Callbacks in DPG can have up to 3 arguments. The first is usually the sender or item triggering the callback. The second is data sent by DPG for different reasons. The third is reserved for user specified data. We refer to this in general terms as: **sender**, **app_data**, and **user_data**.

Note: Because they are optional positional arguments you must use the *sender* and *app_data* if you want to use *user_data* keyword arguments:

3.8.1 Sender, App_data

sender: argument is used by DPG to inform the callback which item triggered the callback by sending the tag or 0 if trigger by the application.

app_data: argument is used by DPG to send information to the callback i.e. the current value of most basic widgets.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

def button_callback(sender, app_data):
    print(f"sender is: {sender}")
    print(f"app_data is: {app_data}")

with dpg.window(label="Tutorial"):
    dpg.add_button(label="Print to Terminal", callback=button_callback)

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.8.2 User Data

user_data: argument is **Optionally** used to pass your own python data into the function.

The python data can be assigned or updated to the keyword *user_data* when the item is created or after the item is created using *set_item_user_data*

User data can be any python object.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

def button_callback(sender, app_data, user_data):
    print(f"sender is: {sender}")
    print(f"app_data is: {app_data}")
    print(f"user_data is: {user_data}")

with dpg.window(label="Tutorial"):
    # user data set when button is created
    dpg.add_button(label="Print to Terminal", callback=button_callback, user_data="Some_
↪Data")

    # user data and callback set any time after button has been created
```

(continues on next page)

(continued from previous page)

```

dpg.add_button(label="Print to Terminal 2", tag="btn")
dpg.set_item_callback("btn", button_callback)
dpg.set_item_user_data("btn", "Some Extra User Data")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

3.8.3 Drag and Drop callback (and drag payload)

Drag/Drop callback receive its **app_data** from **drag_payload**'s **drag_data**.

user_data (optional) can be specified in both **drag payload** and **target drop item** for further use in Drag/Drop callback.

payload_type (optional) can be any string you like. If not specified, **payload_type**'s default value is '\$\$DPG_PAYLOAD'.

For drop callback to work, **payload_type** must be specified the same in both **drag payload** and **target drop item**.

```

import dearpygui.dearpygui as dpg
dpg.create_context()

def drag_cb(sender, app_data, user_data):
    # sender is btn_drag
    # app_data is btn_drag (value from drag_data)
    # do some configure(drawing_item), animation
    ...

def drop_cb(sender, app_data, user_data):
    # sender is group, app_data is btn_drag
    dpg.move_item(app_data, parent=sender)

with dpg.window():
    with dpg.group(horizontal=True):

        with dpg.group(width=300, drop_callback=drop_cb, payload_type="int"): # user_
            ↪data=??
            dpg.add_text("Group left")
            dpg.add_button(label="not drag this")

        with dpg.group(width=300, drop_callback=drop_cb, payload_type="int"):
            dpg.add_text("Group right")
            dpg.add_button(label="not drag this")
            btn_drag = dpg.add_button(label="drag me to another group then drop", drag_
            ↪callback=drag_cb)

            with dpg.drag_payload(parent=btn_drag, drag_data=btn_drag, payload_type="int"):
                dpg.add_text("dragging a button")

                # parent=btn_drag    --> this payload will appear if dragged from the btn_
            ↪drag

```

(continues on next page)

(continued from previous page)

```

        # drag_data=btn_drag --> btn_drag will be app_data in the above drag_cb and
↪drop_cb
        # payload_type="int" --> btn_drag is an int, specified in this payload and
↪drop target - two group above

dpg.create_viewport()
dpg.setup_dearpygui()
dpg.show_viewport()
while dpg.is_dearpygui_running():
    dpg.render_dearpygui_frame()
dpg.destroy_context()

```

3.8.4 Debugging Callbacks (new in 1.2)

Because callbacks are not run on the main thread, debugging can be a hassle. In 1.2 we added a few utilities to help with this.

By default, Dear PyGui handles the callbacks internally on a worker thread. This allows for optimizations and steady framerate. However, to help with debugging, you can set the new **manual_callback_management** key to **True** with `configure_app`. This will prevent Dear PyGui from handling the callbacks. Instead the callbacks and arguments will be stored. You can then retrieve (and clear) them by calling `get_callback_queue` within your main event loop. This will return a list of “Jobs”. A “Job” is just list with the first item being the callable and the remaining items (up to 3) being the typical arguments. We have also provided `run_callbacks` to properly handle the jobs for simple usage.

Below is a simple example

```

import dearpygui.dearpygui as dpg

dpg.create_context()
dpg.configure_app(manual_callback_management=True)
dpg.create_viewport()
dpg.setup_dearpygui()

def callback(sender, app_data, user_data):
    print("Called on the main thread!")

with dpg.window(label="Tutorial"):
    dpg.add_button(label="Press me", callback=callback)

# main loop
dpg.show_viewport()
while dpg.is_dearpygui_running():
    jobs = dpg.get_callback_queue() # retrieves and clears queue
    dpg.run_callbacks(jobs)
    dpg.render_dearpygui_frame()

dpg.destroy_context()

```

3.9 Values

When an item is created, it creates an associated value by default. Values can be shared between items with the same underlying value type. This is accomplished by using the *source* keyword. One of the benefits of this is to have multiple items control the same value.

Values are retrieved from the value *get_value*.

Values can be changed manually using *set_value*.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Tutorial"):
    dpg.add_checkbox(label="Radio Button1", tag="R1")
    dpg.add_checkbox(label="Radio Button2", source="R1")

    dpg.add_input_text(label="Text Input 1")
    dpg.add_input_text(label="Text Input 2", source=dpg.last_item(), password=True)

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.9.1 Value Items

There are several “Value” items that can be used. These are items that have no visual component. These include:

- **mvBoolValue**
- **mvColorValue**
- **mvDoubleValue**
- **mvDouble4Value**
- **mvFloatValue**
- **mvFloat4Value**
- **mvFloatVectValue**
- **mvIntValue**
- **mvInt4Value**
- **mvSeriesValue**
- **mvStringValue**

Basic usage can be found below:

```
import dearpygui.dearpygui as dpg

dpg.create_context()
```

(continues on next page)

```

with dpg.value_registry():
    dpg.add_bool_value(default_value=True, tag="bool_value")
    dpg.add_string_value(default_value="Default string", tag="string_value")

with dpg.window(label="Tutorial"):
    dpg.add_checkbox(label="Radio Button1", source="bool_value")
    dpg.add_checkbox(label="Radio Button2", source="bool_value")

    dpg.add_input_text(label="Text Input 1", source="string_value")
    dpg.add_input_text(label="Text Input 2", source="string_value", password=True)

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

3.10 Containers & Context Managers

We have added context managers as helpers for most container items.

See also:

For more detail [Container Stack](#)

Core Command	Context Manager
<code>add_table</code>	<code>with table(...):</code>
<code>add_table_row</code>	<code>with table_row(...):</code>
<code>add_window</code>	<code>with window(...):</code>
<code>add_menu_bar</code>	<code>with menu_bar(...):</code>
<code>add_child</code>	<code>with child(...):</code>
<code>add_clipper</code>	<code>with clipper(...):</code>
<code>add_collapsing_header</code>	<code>with collapsing_header(...):</code>
<code>add_colormap_registry</code>	<code>with colormap_registry(...):</code>
<code>add_group</code>	<code>with group(...):</code>
<code>add_node</code>	<code>with node(...):</code>
<code>add_node_attribute</code>	<code>with node_attribute(...):</code>
<code>add_node_editor</code>	<code>with node_editor(...):</code>
<code>add_staging_container</code>	<code>with staging_container(...):</code>
<code>add_tab_bar</code>	<code>with tab_bar(...):</code>
<code>add_tab</code>	<code>with tab(...):</code>
<code>add_tree_node</code>	<code>with tree_node(...):</code>
<code>add_tooltip</code>	<code>with tooltip(...):</code>
<code>add_popup</code>	<code>with popup(...):</code>
<code>add_drag_payload</code>	<code>with payload(...):</code>
<code>add_drawlist</code>	<code>with drawlist(...):</code>
<code>add_draw_layer</code>	<code>with draw_layer(...):</code>
<code>add_viewport_drawlist</code>	<code>with viewport_drawlist(...):</code>

continues on next page

Table 1 – continued from previous page

add_file_dialog	with file_dialog(...):
add_filter_set	with filter_set(...):
add_font	with font(...):
add_font_registry	with font_registry(...):
add_handler_registry	with handler_registry(...):
add_plot	with plot(...):
add_subplots	with subplots(...):
add_texture_registry	with texture_registry(...):
add_value_registry	with value_registry(...):
add_theme	with theme(...):
add_item_pool	with item_pool(...):
add_template_registry	with template_registry(...):

3.10.1 Benefits:

1. Automatically push containers to container stack.
2. Automatically pop containers off container stack.
3. More structured, readable code.

3.10.2 Context Managers:

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Main"):

    with dpg.menu_bar():
        with dpg.menu(label="Themes"):
            dpg.add_menu_item(label="Dark")
            dpg.add_menu_item(label="Light")
            dpg.add_menu_item(label="Classic")

        with dpg.menu(label="Other Themes"):
            dpg.add_menu_item(label="Purple")
            dpg.add_menu_item(label="Gold")
            dpg.add_menu_item(label="Red")

        with dpg.menu(label="Tools"):
            dpg.add_menu_item(label="Show Logger")
            dpg.add_menu_item(label="Show About")

        with dpg.menu(label="Oddities"):
            dpg.add_button(label="A Button")
            dpg.add_simple_plot(label="Menu plot", default_value=(0.3, 0.9, 2.5, 8.9),
↵height=80)

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
```

(continues on next page)

```
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.10.3 Explicit Parental Assignment (using UUIDs):

```
import dearpygui.dearpygui as dpg

dpg.create_context()

w = dpg.add_window(label="Main")

mb = dpg.add_menu_bar(parent=w)

themes = dpg.add_menu(label="Themes", parent=mb)
dpg.add_menu_item(label="Dark", parent=themes)
dpg.add_menu_item(label="Light", parent=themes)

other_themes = dpg.add_menu(label="Other Themes", parent=themes)
dpg.add_menu_item(label="Purple", parent=other_themes)
dpg.add_menu_item(label="Gold", parent=other_themes)
dpg.add_menu_item(label="Red", parent=other_themes)

tools = dpg.add_menu(label="Tools", parent=mb)
dpg.add_menu_item(label="Show Logger", parent=tools)
dpg.add_menu_item(label="Show About", parent=tools)

oddities = dpg.add_menu(label="Oddities", parent=mb)
dpg.add_button(label="A Button", parent=oddities)
dpg.add_simple_plot(label="A menu plot", default_value=(0.3, 0.9, 2.5, 8.9), height=80,
↳parent=oddities)

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.10.4 Explicit Parental Assignment (using aliases):

```
import dearpygui.dearpygui as dpg

dpg.create_context()

dpg.add_window(label="Main", tag="w")

dpg.add_menu_bar(parent="w", tag="mb")

dpg.add_menu(label="Themes", parent="mb", tag="themes")
```

(continues on next page)

(continued from previous page)

```

dpg.add_menu_item(label="Dark", parent="themes")
dpg.add_menu_item(label="Light", parent="themes")

dpg.add_menu(label="Other Themes", parent="themes", tag="other_themes")
dpg.add_menu_item(label="Purple", parent="other_themes")
dpg.add_menu_item(label="Gold", parent="other_themes")
dpg.add_menu_item(label="Red", parent="other_themes")

dpg.add_menu(label="Tools", parent="mb", tag="tools")
dpg.add_menu_item(label="Show Logger", parent="tools")
dpg.add_menu_item(label="Show About", parent="tools")

dpg.add_menu(label="Oddities", parent="mb", tag="Oddities")
dpg.add_button(label="A Button", parent="Oddities")
dpg.add_simple_plot(label="A menu plot", default_value=(0.3, 0.9, 2.5, 8.9), height=80,
↳parent="Oddities")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

3.10.5 Container Stack Operations:

```

import dearpygui.dearpygui as dpg

dpg.create_context()

dpg.push_container_stack(dpg.add_window(label="Main"))

dpg.push_container_stack(dpg.add_menu_bar())

dpg.push_container_stack(dpg.add_menu(label="Themes"))
dpg.add_menu_item(label="Dark")
dpg.add_menu_item(label="Light")
dpg.pop_container_stack()

dpg.push_container_stack(dpg.add_menu(label="Tools"))
dpg.add_menu_item(label="Show Logger")
dpg.add_menu_item(label="Show About")
dpg.pop_container_stack()

# remove menu_bar from container stack
dpg.pop_container_stack()

# remove window from container stack
dpg.pop_container_stack()

dpg.create_viewport(title='Custom Title', width=800, height=600)

```

(continues on next page)

```
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.11 Container Slots & Children

Most items can have child items. Items can only be children to valid **container** items. Some related commands can be found below:

is_item_container checks if an item is a container type

get_item_slot returns the item's slot

get_item_parent returns the item's parent UUID

get_item_children returns an item's children

reorder_items reorders children in a single call

move_item_up moves an item up within its slot

move_item_down moves an item down within its slot

move_item moves an item between containers

set_item_children unstaging a stage into an item's children slot

3.11.1 Slots

Items are stored in target slots within their parent container. Below is the breakdown of slots:

Slot 0: *mvFileExtension*, *mvFontRangeHint*, *mvNodeLink*, *mvAnnotation*, *mvDragLine*, *mvDragRect*, *mvDragPoint*, *mvLegend*, *mvTableColumn*

Slot 1: Most items

Slot 2: Draw items

Slot 3: *mvDragPayload*

To query what slot an item belongs to, use *get_item_slot*.

3.11.2 Basic Example

Below is a simple example that demonstrates some of the above:

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="about", width=400, height=400):
    dpg.add_button(label="Press me")
    dpg.draw_line((0, 10), (100, 100), color=(255, 0, 0, 255), thickness=1)
```

(continues on next page)

(continued from previous page)

```
# print children
print(dpg.get_item_children(dpg.last_root()))

# print children in slot 1
print(dpg.get_item_children(dpg.last_root(), 1))

# check draw_line's slot
print(dpg.get_item_slot(dpg.last_item()))

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

Note: Use the *slot* keyword with *get_item_children* to return just a specific slot.

Note: Use the *slot* and *children_only* keywords with *delete_item* to delete a specific slot of children from a parent.

3.12 Container Stack

Unless an item is a root item, all items need to belong to a valid container. An item's parent is deduced through the following process:

1. If item is a root, no parent needed; finished.
2. Check *before* keyword, if used skip to 5 using parent of "before" item.
3. Check *parent* keyword, if used skip to 5.
4. Check container stack, if used skip to 5.
5. Check if parent is compatible.
6. Check if parent accepts.
7. If runtime, add item using runtime methods; finished.
8. If startup, add item using startup methods; finished.

Container items can be manually pushed onto the container stack using *push_container_stack* and popped off using *pop_container_stack*.

This process is automated when using *Containers & Context Managers*. Below is a simple example demonstrating manual stack operations:

```
import dearpygui.dearpygui as dpg

dpg.create_context()

dpg.push_container_stack(dpg.add_window(label="Tutorial"))
```

(continues on next page)

```
dpg.push_container_stack(dpg.add_menu_bar())

dpg.push_container_stack(dpg.add_menu(label="Themes"))
dpg.add_menu_item(label="Dark")
dpg.add_menu_item(label="Light")
dpg.pop_container_stack()

# remove menu_bar from container stack
dpg.pop_container_stack()

# remove window from container stack
dpg.pop_container_stack()

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.12.1 Explicit Parental Assignment

Parents can be explicitly assigned using the *parent* keyword. This is most often used for adding new items at runtime. The above example can be shown again below using explicit parent assignment:

```
import dearpygui.dearpygui as dpg

dpg.create_context()

dpg.add_window(label="Tutorial", tag="window")

dpg.add_menu_bar(parent="window", tag="menu_bar")

dpg.add_menu(label="Themes", parent="menu_bar", tag="themes")
dpg.add_menu_item(label="Dark", parent="themes")
dpg.add_menu_item(label="Light", parent="themes")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.12.2 Context Managers

Context managers can be used to simplify the above example. All the context managers can be found in the *Containers & Context Managers* but a simple example can be found below:

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Tutorial"):
    with dpg.menu_bar():
        with dpg.menu(label="Themes"):
            dpg.add_menu_item(label="Dark")
            dpg.add_menu_item(label="Light")
            dpg.add_menu_item(label="Classic")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

Benefits 1. Automatically push container to container stack. 2. Automatically pop container off container stack. 3. More structured, readable code.

3.13 Drawing-API

DPG has a low level drawing API that is well suited for primitive drawing, custom widgets or even dynamic drawings.

Drawing commands can be added to containers like drawlist, viewport_drawlist, or a window.

A drawlist item is created by calling *add_drawlist* then items can be added by calling their respective draw commands. The origin for the drawing is in the top left and the y-axis points down.

The coordinate system is right-handed with the x axis point left, y axis point down, and z axis pointing into the screen.

Code

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Tutorial"):

    with dpg.drawlist(width=300, height=300): # or you could use dpg.add_drawlist and
        ↪set parents manually

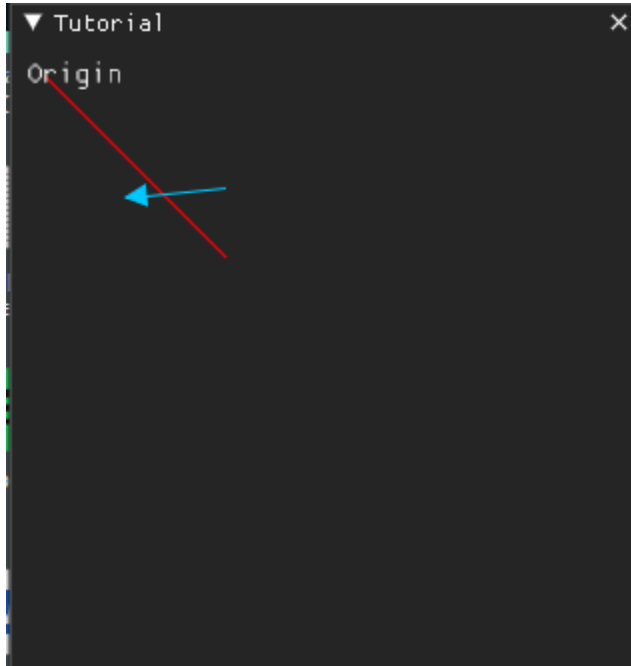
        dpg.draw_line((10, 10), (100, 100), color=(255, 0, 0, 255), thickness=1)
        dpg.draw_text((0, 0), "Origin", color=(250, 250, 250, 255), size=15)
        dpg.draw_arrow((50, 70), (100, 65), color=(0, 200, 255), thickness=1, size=10)

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
```

(continues on next page)

(continued from previous page)

```
dpg.start_dearpygui()
dpg.destroy_context()
```

Results**3.13.1 Layers**

Drawlists can also contain layers. Layers are an effective way to group drawing items for better control of hiding, Z ordering, etc.

New in 1.1. Layers can be used to assist with some 3D operations. See “3D Operations” section below.

Code

```
import dearpygui.dearpygui as dpg

dpg.create_context()

def toggle_layer2(sender):
    show_value = dpg.get_value(sender)
    dpg.configure_item("layer2", show=show_value)

with dpg.window(label="Tutorial"):
    dpg.add_checkbox(label="show layer", callback=toggle_layer2, default_value=True)

    with dpg.drawlist(width=300, height=300):

        with dpg.draw_layer():
            dpg.draw_line((10, 10), (100, 100), color=(255, 0, 0, 255), thickness=1)
            dpg.draw_text((0, 0), "Origin", color=(250, 250, 250, 255), size=15)
            dpg.draw_arrow((50, 70), (100, 65), color=(0, 200, 255), thickness=1, ↵
↪ size=10)
```

(continues on next page)

(continued from previous page)

```

    with dpg.draw_layer(tag="layer2"):
        dpg.draw_line((10, 60), (100, 160), color=(255, 0, 0, 255), thickness=1)
        dpg.draw_arrow((50, 120), (100, 115), color=(0, 200, 255), thickness=1,
↪size=10)

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

3.13.2 Images

Drawlists can display images of types PNG, JPEG, or BMP (See *Textures & Images* for more detail). Images are added using `draw_image`.

Using the keywords **pmin** and **pmax** we can define the upper left and lower right area of the rectangle that the image will be drawn onto the canvas. The image will scale to fit the specified area.

With keywords **uv_min** and **uv_max** we can specify normalized texture coordinates to use just a portion of the area on the image. The default of `uv_min = [0,0]` and `uv_max = [1,1]` will display the entire image while `uv_min = [0,0]` `uv_max = [0.5,0.5]` will only show the first quarter of the drawing.

To be able to demonstrate these features you must update the directory to that of an image on your computer, such as `SpriteMapExample.png`.

Code

```

import dearpygui.dearpygui as dpg

dpg.create_context()

width, height, channels, data = dpg.load_image('SpriteMapExample.png') # 0: width, 1:
↪height, 2: channels, 3: data

with dpg.texture_registry():
    dpg.add_static_texture(width, height, data, tag="image_id")

with dpg.window(label="Tutorial"):

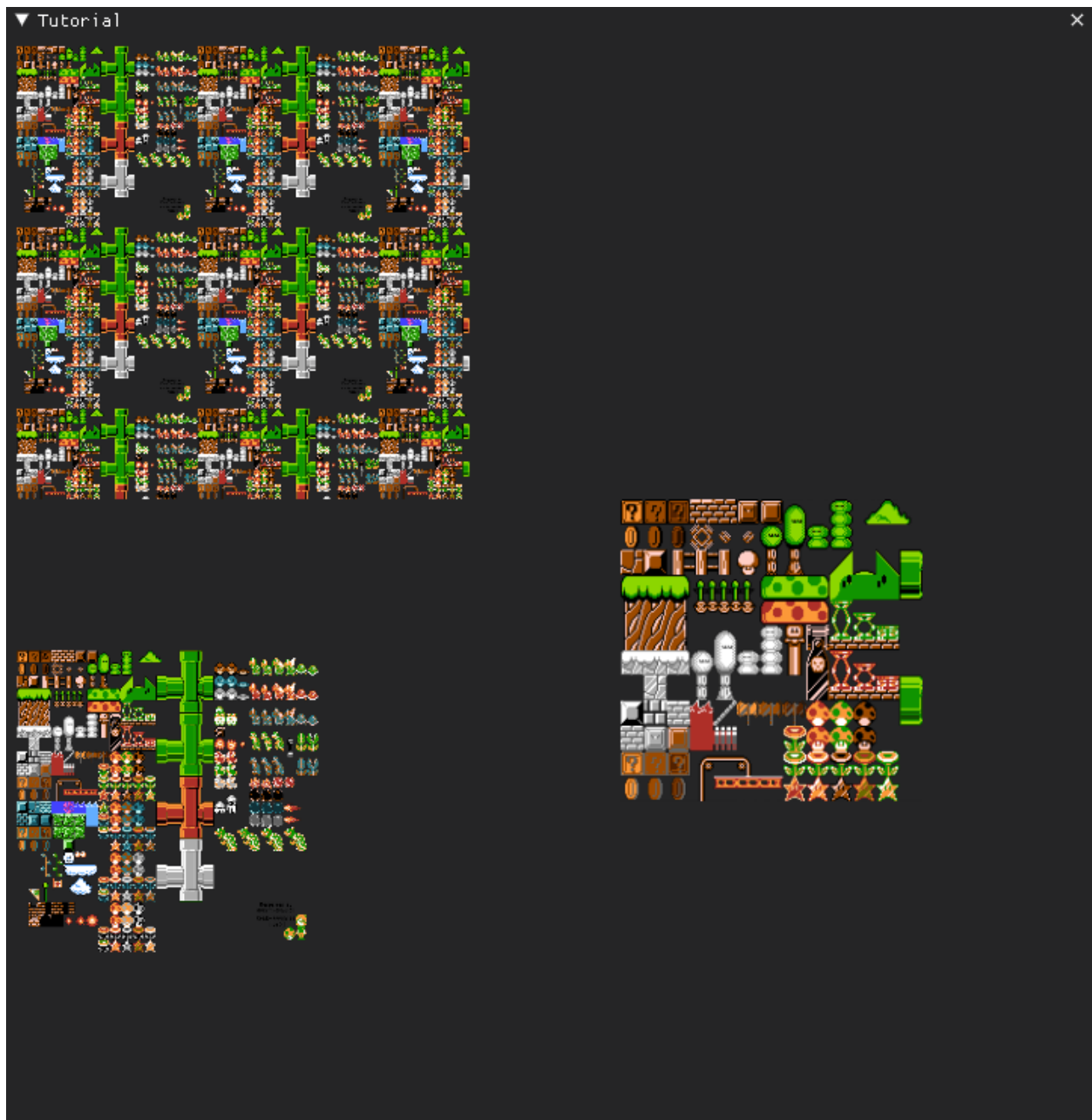
    with dpg.drawlist(width=700, height=700):

        dpg.draw_image("image_id", (0, 400), (200, 600), uv_min=(0, 0), uv_max=(1, 1))
        dpg.draw_image("image_id", (400, 300), (600, 500), uv_min=(0, 0), uv_max=(0.5, 0.
↪5))
        dpg.draw_image("image_id", (0, 0), (300, 300), uv_min=(0, 0), uv_max=(2.5, 2.5))

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

Results



3.13.3 Viewport and Window

You can also use all the same `draw_*` drawings commands with a window as the parent. Similarly you can draw to the viewport foreground or background by using a `viewport_drawlist`.

Code

```
import dearpygui.dearpygui as dpg

dpg.create_context()
```

(continues on next page)

(continued from previous page)

```

# creating font and back viewport drawlists
with dpg.viewport_drawlist():
    dpg.draw_circle((100, 100), 25, color=(255, 255, 255, 255))

dpg.add_viewport_drawlist(front=False, tag="viewport_back")

dpg.draw_circle((200, 200), 25, color=(255, 255, 255, 255), parent="viewport_back")

with dpg.window(label="Tutorial", width=300, height=300):
    dpg.add_text("Move the window over the drawings to see the effects.", wrap=300)
    dpg.draw_circle((100, 100), 25, color=(255, 255, 255, 255))

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

Results

3.13.4 Scene Graph

New in 1.1. For more complex drawing, you can utilize a **draw_node** item. A draw node is used to associate a transformation matrix with a group of draw items. You can use **apply_transform** to apply a transformation matrix to a node. This matrix will be used to multiply each child draw item's points. If a child is another draw node, the matrices will concatenate.

Code

```

import dearpygui.dearpygui as dpg
import math

dpg.create_context()
dpg.create_viewport()
dpg.setup_dearpygui()

with dpg.window(label="tutorial", width=550, height=550):

    with dpg.drawlist(width=500, height=500):

        with dpg.draw_node(tag="root node"):
            dpg.draw_circle([0, 0], 150, color=[0, 255, 0]) # inner_
↪planet orbit
            dpg.draw_circle([0, 0], 200, color=[0, 255, 255]) # outer_
↪planet orbit
            dpg.draw_circle([0, 0], 15, color=[255, 255, 0], fill=[255, 255, 0]) # sun

            with dpg.draw_node(tag="planet node 1"):
                dpg.draw_circle([0, 0], 10, color=[0, 255, 0], fill=[0, 255, 0]) # inner_
↪planet

```

(continues on next page)

```

dpg.draw_circle([0, 0], 25, color=[255, 0, 255]) # moon
↪orbit path

    with dpg.draw_node(tag="planet 1, moon node"):
        dpg.draw_circle([0, 0], 5, color=[255, 0, 255], fill=[255, 0, 255])
↪# moon

        with dpg.draw_node(tag="planet node 2"):
            dpg.draw_circle([0, 0], 10, color=[0, 255, 255], fill=[0, 255, 255]) #
↪outer planet
            dpg.draw_circle([0, 0], 25, color=[255, 0, 255]) #
↪moon 1 orbit path
            dpg.draw_circle([0, 0], 45, color=[255, 255, 255]) #
↪moon 2 orbit path

            with dpg.draw_node(tag="planet 2, moon 1 node"):
                dpg.draw_circle([0, 0], 5, color=[255, 0, 255], fill=[255, 0, 255])
↪# moon 1

            with dpg.draw_node(tag="planet 2, moon 2 node"):
                dpg.draw_circle([0, 0], 5, color=[255, 255, 255], fill=[255, 255,
↪255]) # moon 2

planet1_distance = 150
planet1_angle = 45.0
planet1_moondistance = 25
planet1_moonangle = 45

planet2_distance = 200
planet2_angle = 0.0
planet2_moon1distance = 25
planet2_moon1angle = 45
planet2_moon2distance = 45
planet2_moon2angle = 120

dpg.apply_transform("root node", dpg.create_translation_matrix([250, 250]))
dpg.apply_transform("planet node 1", dpg.create_rotation_matrix(math.pi*planet1_angle/
↪180.0 , [0, 0, -1])*dpg.create_translation_matrix([planet1_distance, 0]))
dpg.apply_transform("planet 1, moon node", dpg.create_rotation_matrix(math.pi*planet1_
↪moonangle/180.0 , [0, 0, -1])*dpg.create_translation_matrix([planet1_moondistance, 0]))
dpg.apply_transform("planet node 2", dpg.create_rotation_matrix(math.pi*planet2_angle/
↪180.0 , [0, 0, -1])*dpg.create_translation_matrix([planet2_distance, 0]))
dpg.apply_transform("planet 2, moon 1 node", dpg.create_rotation_matrix(math.pi*planet2_
↪moon1distance/180.0 , [0, 0, -1])*dpg.create_translation_matrix([planet2_moon1distance,
↪0]))
dpg.apply_transform("planet 2, moon 2 node", dpg.create_rotation_matrix(math.pi*planet2_
↪moon2angle/180.0 , [0, 0, -1])*dpg.create_translation_matrix([planet2_moon2distance,
↪0]))

dpg.show_viewport()
while dpg.is_dearpygui_running():
    dpg.render_dearpygui_frame()

```

(continues on next page)

(continued from previous page)

```
dpg.destroy_context()
```

3.13.5 3D Operations

New in 1.1. Version *1.1* added 3 new options to layers, **perspective_divide**, **depth_clipping**, and **cull_mode**.

When perspective divide is set to **True**, the x, y, and z components of each point are divided by the w component after transformation.

When depth clipping is set to **True**, points will be clipped when they are outside the clip space set using **set_clip_space**. Setting the clip space will scale and transform points. Scaling is based on normalized coordinates (use perspective or orthographic matrices).

Cull mode is used to activate front/back face culling.

Matrices are column major. Post-multiplication is used, for example to scale, then rotate, then transform you use: Transform = Translate * Rotate * Scale.

The following matrix helper functions are provided: - **create_rotation_matrix** - **create_translation_matrix** - **create_scale_matrix** - **create_lookat_matrix** - **create_perspective_matrix** - **create_orthographic_matrix** - **create_fps_matrix**

Code

```
import dearpygui.dearpygui as dpg
import math

dpg.create_context()
dpg.create_viewport()
dpg.setup_dearpygui()

size = 5
vertices = [
    [-size, -size, -size], # 0 near side
    [ size, -size, -size], # 1
    [-size,  size, -size], # 2
    [ size,  size, -size], # 3
    [-size, -size,  size], # 4 far side
    [ size, -size,  size], # 5
    [-size,  size,  size], # 6
    [ size,  size,  size], # 7
    [-size, -size, -size], # 8 left side
    [-size,  size, -size], # 9
    [-size, -size,  size], # 10
    [-size,  size,  size], # 11
    [ size, -size, -size], # 12 right side
    [ size,  size, -size], # 13
    [ size, -size,  size], # 14
    [ size,  size,  size], # 15
    [-size, -size, -size], # 16 bottom side
    [ size, -size, -size], # 17
    [-size, -size,  size], # 18
    [ size, -size,  size], # 19
```

(continues on next page)

```

        [-size, size, -size], # 20 top side
        [ size, size, -size], # 21
        [-size, size, size], # 22
        [ size, size, size], # 23
    ]

colors = [
    [255, 0, 0, 150],
    [255, 255, 0, 150],
    [255, 255, 255, 150],
    [255, 0, 255, 150],
    [ 0, 255, 0, 150],
    [ 0, 255, 255, 150],
    [ 0, 0, 255, 150],
    [ 0, 125, 0, 150],
    [128, 0, 0, 150],
    [128, 70, 0, 150],
    [128, 255, 255, 150],
    [128, 0, 128, 150]
]

with dpb.window(label="tutorial", width=550, height=550):
    with dpb.drawlist(width=500, height=500):
        with dpb.draw_layer(tag="main pass", depth_clipping=True, perspective_
↪divide=True, cull_mode=dpb.mvCullMode_Back):
            with dpb.draw_node(tag="cube"):
                dpb.draw_triangle(vertices[1], vertices[2], vertices[0], color=[0,0,
↪0.0], fill=colors[0])
                dpb.draw_triangle(vertices[1], vertices[3], vertices[2], color=[0,0,
↪0.0], fill=colors[1])
                dpb.draw_triangle(vertices[7], vertices[5], vertices[4], color=[0,0,
↪0.0], fill=colors[2])
                dpb.draw_triangle(vertices[6], vertices[7], vertices[4], color=[0,0,
↪0.0], fill=colors[3])
                dpb.draw_triangle(vertices[9], vertices[10], vertices[8], color=[0,0,
↪0.0], fill=colors[4])
                dpb.draw_triangle(vertices[9], vertices[11], vertices[10], color=[0,
↪0,0.0], fill=colors[5])
                dpb.draw_triangle(vertices[15], vertices[13], vertices[12], color=[0,
↪0,0.0], fill=colors[6])
                dpb.draw_triangle(vertices[14], vertices[15], vertices[12], color=[0,
↪0,0.0], fill=colors[7])
                dpb.draw_triangle(vertices[18], vertices[17], vertices[16], color=[0,
↪0,0.0], fill=colors[8])
                dpb.draw_triangle(vertices[19], vertices[17], vertices[18], color=[0,
↪0,0.0], fill=colors[9])
                dpb.draw_triangle(vertices[21], vertices[23], vertices[20], color=[0,
↪0,0.0], fill=colors[10])

```

(continues on next page)

(continued from previous page)

```

        dpg.draw_triangle(vertices[23], vertices[22], vertices[20], color=[0,
↪0,0.0], fill=colors[11])

x_rot = 10
y_rot = 45
z_rot = 0

view = dpg.create_fps_matrix([0, 0, 50], 0.0, 0.0)
proj = dpg.create_perspective_matrix(math.pi*45.0/180.0, 1.0, 0.1, 100)
model = dpg.create_rotation_matrix(math.pi*x_rot/180.0 , [1, 0, 0])*\  

        dpg.create_rotation_matrix(math.pi*y_rot/180.0 , [0, 1, 0])*\  

        dpg.create_rotation_matrix(math.pi*z_rot/180.0 , [0, 0, 1])

dpg.set_clip_space("main pass", 0, 0, 500, 500, -1.0, 1.0)
dpg.apply_transform("cube", proj*view*model)

dpg.show_viewport()
while dpg.is_dearpygui_running():
    dpg.render_dearpygui_frame()

dpg.destroy_context()

```

3.13.6 3D Operations Limitations

New in 1.1. The drawing API 3D operations are not hardware accelerated (this will be introduced with DearPy3D). This API is for ‘light’ 3D operations. There are a few issues you may come across while performing 3D operation with the drawing API.

Issue 1: Z ordering

With the current API, users are responsible for correct Z ordering. The recommended way to address this is to use “painter’s algorithm”. Basically, just order the items in proper depth order.

This is not the best solution overall. We would prefer to use proper depth buffering but this requires pixel level control which is not practical with this API.

Issue 2: Perspective Texture Correction

When using `draw_image_quad`, viewing the image at sharp angles will deform the image. This is due to texture coordinates being linearly interpolated in normalized device coordinate space.

There is currently no practical solution but an attempt could be made to split the quad into several smaller quads. The actual solution requires pixel level control which is not practical with this API.

Issue 3: Culling

Currently, culling is only setup for triangles.

Dear Py3D and Software Renderer

All of the above issues will resolved in **Dear Py3D**. Before **Dear Py3D**, we will also be introducing a software renderer for **Dear PyGui** that will resolve the above issues.

3.14 File & Directory Selector

The file dialog item can be used to select a single file, multiple files, or a directory. When the user clicks the **Ok** button, the dialog's callback is run. An optional second callback, to be run when the cancel button is clicked, can be provided as a keyword argument.

When OK is clicked, information is passed through the `app_data` argument such as: * file path * file name * current path * current filter (the file type filter)

The simplest case is as a directory picker. Below is the example

Code

```
import dearpygui.dearpygui as dpg

dpg.create_context()

def callback(sender, app_data):
    print('OK was clicked.')
    print("Sender: ", sender)
    print("App Data: ", app_data)

def cancel_callback(sender, app_data):
    print('Cancel was clicked.')
    print("Sender: ", sender)
    print("App Data: ", app_data)

dpg.add_file_dialog(
    directory_selector=True, show=False, callback=callback, tag="file_dialog_id",
    cancel_callback=cancel_callback, width=700, height=400)

with dpg.window(label="Tutorial", width=800, height=300):
    dpg.add_button(label="Directory Selector", callback=lambda: dpg.show_item("file_
    ↪dialog_id"))

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

Note: If no file extensions have been added, the selector defaults to directories.

3.14.1 File Extensions

File extensions are items that are added to the file dialog. You can even set the color of the file extensions. Below is a simple example:

```
import dearpygui.dearpygui as dpg

dpg.create_context()

def callback(sender, app_data, user_data):
    print("Sender: ", sender)
    print("App Data: ", app_data)

with dpg.file_dialog(directory_selector=False, show=False, callback=callback, id="file_
↪dialog_id", width=700 ,height=400):
    dpg.add_file_extension(".*")
    dpg.add_file_extension("", color=(150, 255, 150, 255))
    dpg.add_file_extension("Source files (*.cpp *.h *.hpp){.cpp,.h,.hpp}", color=(0, 255,
↪ 255, 255))
    dpg.add_file_extension(".h", color=(255, 0, 255, 255), custom_text="[header]")
    dpg.add_file_extension(".py", color=(0, 255, 0, 255), custom_text="[Python]")

with dpg.window(label="Tutorial", width=800, height=300):
    dpg.add_button(label="File Selector", callback=lambda: dpg.show_item("file_dialog_id
↪"))

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.14.2 Customizing

File dialogs can be customized with a panel by just adding items to the file dialog as if it were a regular container.

This can allow the creation of a pinned menu, favorites, directory tree, and much more.

Below is an example:

```
import dearpygui.dearpygui as dpg

dpg.create_context()

def callback(sender, app_data):
    print("Sender: ", sender)
    print("App Data: ", app_data)

with dpg.file_dialog(directory_selector=False, show=False, callback=callback, tag="file_
↪dialog_tag", width=700 ,height=400):
    dpg.add_file_extension(".*")
    dpg.add_file_extension("", color=(150, 255, 150, 255))
    dpg.add_file_extension(".cpp", color=(255, 255, 0, 255))
```

(continues on next page)

(continued from previous page)

```

dpg.add_file_extension(".h", color=(255, 0, 255, 255))
dpg.add_file_extension(".py", color=(0, 255, 0, 255))

with dpg.group(horizontal=True):
    dpg.add_button(label="fancy file dialog")
    dpg.add_button(label="file")
    dpg.add_button(label="dialog")
dpg.add_date_picker()
with dpg.child_window(height=100):
    dpg.add_selectable(label="bookmark 1")
    dpg.add_selectable(label="bookmark 2")
    dpg.add_selectable(label="bookmark 3")

with dpg.window(label="Tutorial", width=800, height=300):
    dpg.add_button(label="File Selector", callback=lambda: dpg.show_item("file_dialog_tag
↪"))

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

3.14.3 Selecting Multiple Files

You can select multiple files by setting the `file_count` keyword **Must use Ctrl + click to select multiple files Must use Shift + click to select multiple files**

```

import dearpygui.dearpygui as dpg

dpg.create_context()

def callback(sender, app_data):
    print("Sender: ", sender)
    print("App Data: ", app_data)

with dpg.file_dialog(directory_selector=False, show=False, callback=callback, file_
↪count=3, tag="file_dialog_tag", width=700, height=400):
    dpg.add_file_extension("", color=(255, 150, 150, 255))
    dpg.add_file_extension(".*")
    dpg.add_file_extension(".cpp", color=(255, 255, 0, 255))
    dpg.add_file_extension(".h", color=(255, 0, 255, 255))
    dpg.add_file_extension(".py", color=(0, 255, 0, 255))

    dpg.add_button(label="fancy file dialog")
    with dpg.child_window(width=100):
        dpg.add_selectable(label="bookmark 1")
        dpg.add_selectable(label="bookmark 2")
        dpg.add_selectable(label="bookmark 3")

with dpg.window(label="Tutorial", width=800, height=300):

```

(continues on next page)

(continued from previous page)

```

    dpg.add_button(label="File Selector", callback=lambda: dpg.show_item("file_dialog_tag
↪"))

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

3.15 Filter Set

The filter set item is a container that can be used to filter its children based on their *filter_key*.

Most items have a *filter_key* keyword that can be set when creating the item. Filtering is based on the set value of the filter set.

The easiest way to understand this is by considering the example below

Code

```

import dearpygui.dearpygui as dpg

dpg.create_context()

def callback(sender, filter_string):
    dpg.set_value("filter_id", filter_string)

with dpg.window(label="about", width =500, height=300):
    dpg.add_input_text(label="Filter (inc, -exc)", callback=callback)
    with dpg.filter_set(id="filter_id"):
        dpg.add_text("aaa1.c", filter_key="aaa1.c", bullet=True)
        dpg.add_text("bbb1.c", filter_key="bbb1.c", bullet=True)
        dpg.add_text("ccc1.c", filter_key="ccc1.c", bullet=True)
        dpg.add_text("aaa2.cpp", filter_key="aaa2.cpp", bullet=True)
        dpg.add_text("bbb2.cpp", filter_key="bbb2.cpp", bullet=True)
        dpg.add_text("ccc2.cpp", filter_key="ccc2.cpp", bullet=True)
        dpg.add_text("abc.h", filter_key="abc.h", bullet=True)
        dpg.add_text("hello, world", filter_key="hello, world", bullet=True)

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

Note:

- Display everything with ""
- Display lines containing xxx with "xxx"
- Display lines containing xxx or yyy with "xxx,yyy"

- Hide lines containing xxx with “-xxx”
-

3.16 Fonts

DPG embeds a copy of ‘ProggyClean.ttf’ (by Tristan Grimmer), a 13 pixels high, pixel-perfect font used by default. ProggyClean does not scale smoothly, therefore it is recommended that you load your own file when using DPG in an application aiming to look nice and wanting to support multiple resolutions.

You do this by loading external .TTF/.OTF files. In the `Assets` folder you can find an example of a otf font.

3.16.1 Readme First

All loaded fonts glyphs are rendered into a single texture atlas. Rendering occurs on-the-fly as characters are encountered in UI strings (item labels, input contents, etc. - anything that gets displayed in the viewport).

You can use the Fonts Manager tool `show_font_manager` to browse your fonts and understand what’s going on if you have an issue.

3.16.2 Font Loading Instructions

To add your own fonts, you must first create a font registry to add fonts to. Next, add fonts to the registry.

For a font to take effect, it must be either bound globally with `bind_font` or bound to a particular item with `bind_item_font`. In the latter case, the font is “inherited” by all child item if it is bound to a container. To remove font binding and revert to the default font (or inherit from the parent item), call `bind_font/bind_item_font` again, passing 0 in the `font` argument.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

# add a font registry
with dpg.font_registry():
    # first argument ids the path to the .ttf or .otf file
    default_font = dpg.add_font("NotoSerifCJKjp-Medium.otf", 20)
    second_font = dpg.add_font("NotoSerifCJKjp-Medium.otf", 10)

dpg.bind_font(default_font)

with dpg.window(label="Font Example", height=200, width=200):
    dpg.add_button(label="Default font")
    b2 = dpg.add_button(label="Secondary font")
    dpg.add_button(label="default")

    # set font of specific widget
    dpg.bind_item_font(b2, second_font)

dpg.show_font_manager()

dpg.create_viewport(title='Custom Title', width=800, height=600)
```

(continues on next page)

(continued from previous page)

```
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.16.3 Loading Specific Unicode Characters

Note: This section previously described how to load specific characters from font file so that they are displayed in the UI correctly. Since DPG version 2.3, character glyphs are rendered automatically and there is no need to specify them in advance. This section is no longer relevant, and the functions *add_font_chars*, *add_font_range*, and *add_font_range_hint* are obsolete and do nothing if you call them.

3.16.4 Remapping Characters

For convenience, you can remap a character to a different character code and use this replacement in your text strings. In particular, it might be useful with icon fonts. Here is an example of how to do such remapping.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.font_registry():
    with dpg.font("NotoSerifCJKjp-Medium.otf", 20) as font1:
        # remap to %
        dpg.add_char_remap(0x3084, 0x0025)

dpg.show_font_manager()

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.16.5 Where to find unicode character codes?

Unicode Characters

3.17 Init Files

Init files are used to preserve the following data between application sessions

- window positions
- window sizes
- window collapse state
- window docking
- table column widths
- table column ordering
- table column visible state
- table column sorting state

Note: Init files use the tag of the window. Make sure the tag does not change between sessions by generating the tag beforehand or specifying it as a string.

3.17.1 Creating init files

Use `save_init_file` while your application is running.

Note: windows and tables can individually opt out of having their settings saved with the `no_saved_settings` keyword.

3.17.2 Loading init files

Use `configure_app` before creating the viewport.

Below is an example of using **init** files to preserve settings between sessions.

- Position the windows
- Press the save button and the init file will be saved in the current working directory
- Restart the app and see your windows in the previous positions.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

def save_init():
    dpg.save_init_file("dpg.ini")

dpg.configure_app(init_file="dpg.ini") # default file is 'dpg.ini'
with dpg.window(label="about", tag="main window"):
    dpg.add_button(label="Save Window pos", callback=lambda: save_init)

with dpg.window(label="about", tag="side window"):
    dpg.add_button(label="Press me")
```

(continues on next page)

(continued from previous page)

```
dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.18 Menu Bar

DPG contains two types of Menu Bars

Viewport Menu Bar Attached to the viewport

Menu Bar: Attached to a specified window

Other than being attached to different parents they both act the same.

A typical menu bar consists of the following components:

Menu Bar: The main menu ribbon. Used to contain menus.

Menu: Popup windows that are used to contain items in a collapsable fashion.

Menu Item: Items that can run callbacks and display a checkmark.

3.18.1 Usage

Items added to the Menu Bar are displayed from left to right. Items added to the menus are displayed from top to bottom.

Menus can be nested inside other menus.

Any widget can be added to a menu.

Viewport Menu Bar Example

```
import dearpygui.dearpygui as dpg

def print_me(sender):
    print(f"Menu Item: {sender}")

dpg.create_context()
dpg.create_viewport(title='Custom Title', width=600, height=200)

with dpg.viewport_menu_bar():
    with dpg.menu(label="File"):
        dpg.add_menu_item(label="Save", callback=print_me)
        dpg.add_menu_item(label="Save As", callback=print_me)

        with dpg.menu(label="Settings"):
            dpg.add_menu_item(label="Setting 1", callback=print_me, check=True)
            dpg.add_menu_item(label="Setting 2", callback=print_me)

    dpg.add_menu_item(label="Help", callback=print_me)
```

(continues on next page)

```
    with dpg.menu(label="Widget Items"):
        dpg.add_checkbox(label="Pick Me", callback=print_me)
        dpg.add_button(label="Press Me", callback=print_me)
        dpg.add_color_picker(label="Color Me", callback=print_me)

dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

Menu Bar Example

```
import dearpygui.dearpygui as dpg

dpg.create_context()

dpg.create_viewport(title='Custom Title', width=600, height=200)

def print_me(sender):
    print(f"Menu Item: {sender}")

with dpg.window(label="Tutorial"):
    with dpg.menu_bar():
        with dpg.menu(label="File"):
            dpg.add_menu_item(label="Save", callback=print_me)
            dpg.add_menu_item(label="Save As", callback=print_me)

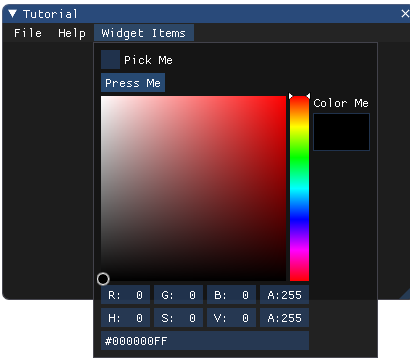
            with dpg.menu(label="Settings"):
                dpg.add_menu_item(label="Setting 1", callback=print_me, check=True)
                dpg.add_menu_item(label="Setting 2", callback=print_me)

            dpg.add_menu_item(label="Help", callback=print_me)

        with dpg.menu(label="Widget Items"):
            dpg.add_checkbox(label="Pick Me", callback=print_me)
            dpg.add_button(label="Press Me", callback=print_me)
            dpg.add_color_picker(label="Color Me", callback=print_me)

dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

Results



3.19 Node Editor

A Node Editor presents an editable schematic or graph, displaying nodes and the connections between their attributes. It allows you to view, modify, and create new node connections.

You can see an example below

There are 4 main components

1. **Node Editor** - the area in which the nodes are located
2. **Nodes** - the free floating “windows” which contains attributes
3. **Attributes** - the collections of widgets with pins to create links to/from. Can be input, output, or static.
4. **Links** - the connections between attributes

Attributes can contain any UI Items. When a user clicks and drags a node’s attribute the node editor’s callback is ran. DPG sends the attributes’ tags through the `_app_data_` argument of the callback.

It’s the developer’s responsibility to create the link.

Below is a basic example. You can grab an output pin and connect it to an input pin. You can detach a link by **ctrl** clicking the link and dragging it.

Code

```
import dearpygui.dearpygui as dpg

dpg.create_context()

# callback runs when user attempts to connect attributes
def link_callback(sender, app_data):
    # app_data -> (link_id1, link_id2)
    dpg.add_node_link(app_data[0], app_data[1], parent=sender)

# callback runs when user attempts to disconnect attributes
def delink_callback(sender, app_data):
    # app_data -> link_id
    dpg.delete_item(app_data)

with dpg.window(label="Tutorial", width=400, height=400):
```

(continues on next page)

(continued from previous page)

```

with dpg.node_editor(callback=link_callback, delink_callback=delink_callback):
    with dpg.node(label="Node 1"):
        with dpg.node_attribute(label="Node A1"):
            dpg.add_input_float(label="F1", width=150)

        with dpg.node_attribute(label="Node A2", attribute_type=dpg.mvNode_Attr_
↔Output):
            dpg.add_input_float(label="F2", width=150)

    with dpg.node(label="Node 2"):
        with dpg.node_attribute(label="Node A3"):
            dpg.add_input_float(label="F3", width=200)

        with dpg.node_attribute(label="Node A4", attribute_type=dpg.mvNode_Attr_
↔Output):
            dpg.add_input_float(label="F4", width=200)

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

3.19.1 Selection Querying

You can retrieve selected nodes and links (and clear this selections with the following commands)

```

dpg.get_selected_nodes(editor_id)
dpg.get_selected_links(editor_id)
dpg.clear_selected_nodes(editor_id)
dpg.clear_selected_links(editor_id)

```

3.19.2 Node Attribute Types

The following constants can be used in the *attribute_type* argument for node attributes

Attribute
mvNode_Attr_Input (default)
mvNode_Attr_Output
mvNode_Attr_Static

3.19.3 Node Attribute Pin Shapes

The following constants can be used in the *shape* argument for node attributes

Shape
mvNode_PinShape_Circle
mvNode_PinShape_CircleFilled (default)
mvNode_PinShape_Triangle
mvNode_PinShape_TriangleFilled
mvNode_PinShape_Quad
mvNode_PinShape_QuadFilled

3.19.4 Associated Items

- **mvNode**
- **mvNodeAttribute**
- **mvNodeLink**

3.20 Plots

Plots are composed of multiple components.

Y-axis: This is a container and is the parent of all the data series that are added to the plot. The plot can have multiple Y-axis at one time (up to 3).

X-axis: This is the x data scale (only 1 x axis is allowed).

Series: These are the containers for the data you wish to display. Data series need to be added as a child of a Y-axis to be displayed on the plot. There are many different types of data series available. Series also can contain UI Items that will be displayed when right-clicking the series label in the legend as a context menu.

Legend (optional): This is a normal legend and also allows the user to toggle which data series are visible.

Plots have some functionality built in:

Toggle Data Series: Click on the legend name of the desired data series to toggle

Settings: Double Right Click

Pan Plot: Click & Drag on plot

Pan Axis: Click & Drag on Axis

Zoom: Scroll Mouse Wheel

Zoom Axis: Hover Axis & Scroll Mouse Wheel

Zoom Region: Right Click & Drag

Zoom Extents: Double Click

Zoom Axis Area: Shift + Right Click & Drag

```
import dearpygui.dearpygui as dpg
from math import sin
```

(continues on next page)

```

dpg.create_context()

# creating data
sindatax = []
sindatay = []
for i in range(0, 500):
    sindatax.append(i / 1000)
    sindatay.append(0.5 + 0.5 * sin(50 * i / 1000))

with dpg.window(label="Tutorial"):
    # create plot
    with dpg.plot(label="Line Series", height=400, width=400):
        # optionally create legend
        dpg.add_plot_legend()

        # REQUIRED: create x and y axes
        dpg.add_plot_axis(dpg.mvXAxis, label="x")
        dpg.add_plot_axis(dpg.mvYAxis, label="y", tag="y_axis")

        # series belong to a y axis
        dpg.add_line_series(sindatax, sindatay, label="0.5 + 0.5 * sin(x)", parent="y_
↪axis")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

3.20.1 Updating Series Data

You can change a series on a plot by

- setting the series value
- deleting that specific series item from they y-axis and adding it again
- deleting all the series items from they y-axis and adding that specific series again

```

import dearpygui.dearpygui as dpg
from math import sin, cos

dpg.create_context()

sindatax = []
sindatay = []
for i in range(0, 500):
    sindatax.append(i / 1000)
    sindatay.append(0.5 + 0.5 * sin(50 * i / 1000))

def update_series():

```

(continues on next page)

(continued from previous page)

```

cosdatax = []
cosdatay = []
for i in range(0, 500):
    cosdatax.append(i / 1000)
    cosdatay.append(0.5 + 0.5 * cos(50 * i / 1000))
dpg.set_value('series_tag', [cosdatax, cosdatay])
dpg.set_item_label('series_tag', "0.5 + 0.5 * cos(x)")

with dpg.window(label="Tutorial", tag="win"):
    dpg.add_button(label="Update Series", callback=update_series)
    # create plot
    with dpg.plot(label="Line Series", height=400, width=400):
        # optionally create legend
        dpg.add_plot_legend()

        # REQUIRED: create x and y axes
        dpg.add_plot_axis(dpg.mvXAxis, label="x")
        dpg.add_plot_axis(dpg.mvYAxis, label="y", tag="y_axis")

        # series belong to a y axis
        dpg.add_line_series(sindatax, sindatay, label="0.5 + 0.5 * sin(x)", parent="y_
↪axis", tag="series_tag")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

3.20.2 Axis Limits

The following commands can be used to control the plot axes limits

- `set_axis_limits(...)`
- `get_axis_limits(...)`
- `set_axis_limits_auto(...)`
- `fit_axis_data(...)`
- `set_axis_limits_constraints(...)`
- `reset_axis_limits_constraints(...)`
- `set_axis_zoom_constraints(...)`
- `reset_axis_zoom_constraints(...)`

An example demonstrating some of this can be found below:

```

import dearpygui.dearpygui as dpg

dpg.create_context()

```

(continues on next page)

(continued from previous page)

```

with dpg.window(label="Tutorial", width=400, height=400):
    with dpg.group(horizontal=True):
        dpg.add_button(label="fit y", callback=lambda: dpg.fit_axis_data("y_axis"))
        dpg.add_button(label="unlock x limits", callback=lambda: dpg.set_axis_limits_
↳ auto("x_axis"))
        dpg.add_button(label="unlock y limits", callback=lambda: dpg.set_axis_limits_
↳ auto("y_axis"))
        dpg.add_button(label="print limits x", callback=lambda: print(dpg.get_axis_
↳ limits("x_axis")))
        dpg.add_button(label="print limits y", callback=lambda: print(dpg.get_axis_
↳ limits("y_axis")))

    with dpg.plot(label="Bar Series", height=-1, width=-1):
        dpg.add_plot_legend()

        # create x axis
        dpg.add_plot_axis(dpg.mvXAxis, label="Student", no_gridlines=True, tag="x_axis")
        dpg.set_axis_limits(dpg.last_item(), 9, 33)
        dpg.set_axis_ticks(dpg.last_item(), (("S1", 11), ("S2", 21), ("S3", 31)))

        # create y axis
        dpg.add_plot_axis(dpg.mvYAxis, label="Score", tag="y_axis")
        dpg.set_axis_limits("y_axis", 0, 110)

        # add series to y axis
        dpg.add_bar_series([10, 20, 30], [100, 75, 90], label="Final Exam", weight=1,
↳ parent="y_axis")
        dpg.add_bar_series([11, 21, 31], [83, 75, 72], label="Midterm Exam", weight=1,
↳ parent="y_axis")
        dpg.add_bar_series([12, 22, 32], [42, 68, 23], label="Course Grade", weight=1,
↳ parent="y_axis")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

3.20.3 Custom Axis Labels

Custom labels can be set per axis using `set_axis_ticks`. They can be reset with `reset_axis_ticks`. An example can be found below

```

import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Tutorial", width=400, height=400):
    with dpg.plot(label="Bar Series", height=-1, width=-1):
        dpg.add_plot_legend()

```

(continues on next page)

(continued from previous page)

```

# create x axis
dpg.add_plot_axis(dpg.mvXAxis, label="Student", no_gridlines=True)
dpg.set_axis_ticks(dpg.last_item(), (("S1", 11), ("S2", 21), ("S3", 31)))

# create y axis
dpg.add_plot_axis(dpg.mvYAxis, label="Score", tag="yaxis_tag")

# add series to y axis
dpg.add_bar_series([10, 20, 30], [100, 75, 90], label="Final Exam", weight=1,
↳parent="yaxis_tag")
dpg.add_bar_series([11, 21, 31], [83, 75, 72], label="Midterm Exam", weight=1,
↳parent="yaxis_tag")
dpg.add_bar_series([12, 22, 32], [42, 68, 23], label="Course Grade", weight=1,
↳parent="yaxis_tag")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

3.20.4 Multiple Y Axes

Plots can contain up to Three Y-axis for different data that needs a different scale.

```

import dearpygui.dearpygui as dpg
from math import sin

dpg.create_context()

sindatax = []
sindatay = []
for i in range(0, 100):
    sindatax.append(i / 100)
    sindatay.append(0.5 + 0.5 * sin(50 * i / 100))

with dpg.window(label="Tutorial", width=400, height=400):
    with dpg.plot(label="Multi Axes Plot", height=400, width=-1):
        dpg.add_plot_legend()

        # create x axis
        dpg.add_plot_axis(dpg.mvXAxis, label="x")

        # create y axis 1
        dpg.add_plot_axis(dpg.mvYAxis, label="y1")
        dpg.add_line_series(sindatax, sindatay, label="y1 lines", parent=dpg.last_item())

        # create y axis 2
        dpg.add_plot_axis(dpg.mvYAxis2, label="y2")
        dpg.add_stem_series(sindatax, sindatay, label="y2 stem", parent=dpg.last_item())

```

(continues on next page)

(continued from previous page)

```

    # create y axis 3
    dpg.add_plot_axis(dpg.mvYAxis3, label="y3 scatter")
    dpg.add_scatter_series(sindatax, sindatay, label="y3", parent=dpg.last_item())

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

3.20.5 Annotations

Annotations can be used to mark locations on a plot.

Annotations are owned by the plot and their coordinates correspond to the 1st y axis.

They are clamped by default.

```

import dearpygui.dearpygui as dpg
from math import sin

dpg.create_context()

sindatax = []
sindatay = []
for i in range(0, 100):
    sindatax.append(i / 100)
    sindatay.append(0.5 + 0.5 * sin(50 * i / 100))

with dpg.window(label="Tutorial", width=400, height=400):
    with dpg.plot(label="Annotations", height=-1, width=-1):
        dpg.add_plot_legend()
        dpg.add_plot_axis(dpg.mvXAxis, label="x")
        dpg.add_plot_axis(dpg.mvYAxis, label="y")
        dpg.add_line_series(sindatax, sindatay, label="0.5 + 0.5 * sin(x)", parent=dpg.
↪last_item())

        # annotations belong to the plot NOT axis
        dpg.add_plot_annotation(label="BL", default_value=(0.25, 0.25), offset=(-15, 15),
↪ color=[255, 255, 0, 255])
        dpg.add_plot_annotation(label="BR", default_value=(0.75, 0.25), offset=(15, 15),
↪ color=[255, 255, 0, 255])
        dpg.add_plot_annotation(label="TR not clamped", default_value=(0.75, 0.75),
↪ offset=(-15, -15),
                                color=[255, 255, 0, 255], clamped=False)
        dpg.add_plot_annotation(label="TL", default_value=(0.25, 0.75), offset=(-15, -
↪15), color=[255, 255, 0, 255])
        dpg.add_plot_annotation(label="Center", default_value=(0.5, 0.5), color=[255,
↪255, 0, 255])

dpg.create_viewport(title='Custom Title', width=800, height=600)

```

(continues on next page)

(continued from previous page)

```
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.20.6 Drag Points and Lines

Drag lines/points are owned by the plot and their coordinates correspond to the 1st y axis. These items can be moved by clicking and dragging.

You can also set a callback to be ran when they are interacted with!

```
import dearpygui.dearpygui as dpg

dpg.create_context()

def print_val(sender):
    print(dpg.get_value(sender))

with dpg.window(label="Tutorial", width=400, height=400):
    with dpg.plot(label="Drag Lines/Points", height=-1, width=-1):
        dpg.add_plot_legend()
        dpg.add_plot_axis(dpg.mvXAxis, label="x")
        dpg.set_axis_limits(dpg.last_item(), -5, 5)
        dpg.add_plot_axis(dpg.mvYAxis, label="y")
        dpg.set_axis_limits(dpg.last_item(), -5, 5)

        # drag lines/points belong to the plot NOT axis
        dpg.add_drag_line(label="dline1", color=[255, 0, 0, 255], default_value=2.0,
↪ callback=print_val)
        dpg.add_drag_line(label="dline2", color=[255, 255, 0, 255], vertical=False,
↪ default_value=-2, callback=print_val)
        dpg.add_drag_point(label="dpoint1", color=[255, 0, 255, 255], default_value=(1.0,
↪ 1.0), callback=print_val)
        dpg.add_drag_point(label="dpoint2", color=[255, 0, 255, 255], default_value=(-1.
↪ 0, 1.0), callback=print_val)

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.20.7 Querying

Querying allows the user to select a region of the plot by **Ctrl + dragging the right mouse button** and dragging five circles of the query rectangle with left mouse button.

Double left click inside a drag rect will remove it (if `min_query_rects` allows). If number of rects exceed `max_query_rects` when create new drag rect, it will replace the last one.

Since DearPyGui 2.0, `query_mod` changes to `query_toggle_mod` for swapping the Ctrl key above. `query_button` is removed, so **dragging the right mouse button** is hardcoded. `min_query_rects`, `max_query_rects` limit the number of drag rects.

Querying requires setting `query` to **True** when creating the plot.

The callback of the plot will run when the plot is being queried (dragging five circles). Or not using plot's callback but drag rect's callback `dpg.add_drag_rect(callback=...)`.

All the query areas are sent through the `app_data` argument as `((x_min, y_min, x_max, y_max), (x_min, y_min, x_max, y_max), ...)`.

It is also possible to poll the plot for the query areas by calling: `get_plot_query_rects` and

Below is an example using the callback

```
import dearpygui.dearpygui as dpg
from math import sin

dpg.create_context()

sindatax = []
sindatay = []
for i in range(0, 100):
    sindatax.append(i / 100)
    sindatay.append(0.5 + 0.5 * sin(50 * i / 100))

with dpg.window(label="Tutorial", width=400, height=600):
    dpg.add_text("Ctrl and drag the right mouse button over the top plot!")

    def query(sender, app_data, user_data):
        # TODO: handle for when app_data is empty - IndexError: tuple index out of range.
        rect_0 = app_data[0]
        # other_rects = app_data[1:]
        dpg.set_axis_limits("xaxis_tag2", rect_0[0], rect_0[2])
        dpg.set_axis_limits("yaxis_tag2", rect_0[1], rect_0[3])

    # plot 1
    with dpg.plot(
        no_title=True, height=200, callback=query, query=True, no_menus=True, width=-1,
        min_query_rects=0, max_query_rects=3,
    ):
        dpg.add_plot_axis(dpg.mvXAxis, label="x")
        dpg.add_plot_axis(dpg.mvYAxis, label="y")
        dpg.add_line_series(sindatax, sindatay, parent=dpg.last_item())

    # plot 2
```

(continues on next page)

(continued from previous page)

```

with dpg.plot(no_title=True, height=200, no_menus=True, width=-1):
    dpg.add_plot_axis(dpg.mvXAxis, label="x1", tag="xaxis_tag2")
    dpg.add_plot_axis(dpg.mvYAxis, label="y1", tag="yaxis_tag2")
    dpg.add_line_series(sindatax, sindatay, parent="yaxis_tag2")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

3.20.8 Custom Context Menus

Plot series are actually containers!

Adding UI Items to a plot series, they will show up when right-clicking the series in the legend.

```

import dearpygui.dearpygui as dpg
from math import sin

dpg.create_context()

sindatax = []
sindatay = []
for i in range(0, 100):
    sindatax.append(i / 100)
    sindatay.append(0.5 + 0.5 * sin(50 * i / 100))

with dpg.window(label="Tutorial", width=400, height=400):
    # create plot
    dpg.add_text("Right click a series in the legend!")
    with dpg.plot(label="Line Series", height=-1, width=-1):
        dpg.add_plot_legend()

        dpg.add_plot_axis(dpg.mvXAxis, label="x")
        dpg.add_plot_axis(dpg.mvYAxis, label="y", tag="yaxis")

        # series 1
        dpg.add_line_series(sindatax, sindatay, label="series 1", parent="yaxis", tag=
↪"series_1")
        dpg.add_button(label="Delete Series 1", parent=dpg.last_item(), callback=lambda:
↪dpg.delete_item("series_1"))

        # series 2
        dpg.add_line_series(sindatax, sindatay, label="series 2", parent="yaxis", tag=
↪"series_2")
        dpg.add_button(label="Delete Series 2", parent=dpg.last_item(), callback=lambda:
↪dpg.delete_item("series_2"))

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()

```

(continues on next page)

```
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.20.9 Custom Series

New in 1.6. Custom series allow you to control the way a series is rendered.

A custom series can currently have between 2 and 5 channels. A channel is an array/list of data. Each channel must be the same length. The first 2 channels and channel count are required arguments. Additional channels can be provided with the `y1`, `y2`, and `y3` keywords. You must also set the “callback” keyword. The second argument will be provided by DPG as a list. The first item being useful information. The following items are the original data sent in but transformed into pixel space. The combination of all this information can be used to create a custom series. See simple example below:

```
import dearpygui.dearpygui as dpg

dpg.create_context()
dpg.create_viewport()
dpg.setup_dearpygui()

x_data = [0.0, 1.0, 2.0, 4.0, 5.0]
y_data = [0.0, 10.0, 20.0, 40.0, 50.0]

def callback(sender, app_data):

    _helper_data = app_data[0]
    transformed_x = app_data[1]
    transformed_y = app_data[2]
    #transformed_y1 = app_data[3] # for channel = 3
    #transformed_y2 = app_data[4] # for channel = 4
    #transformed_y3 = app_data[5] # for channel = 5
    mouse_x_plot_space = _helper_data["MouseX_PlotSpace"] # not used in this example
    mouse_y_plot_space = _helper_data["MouseY_PlotSpace"] # not used in this example
    mouse_x_pixel_space = _helper_data["MouseX_PixelSpace"]
    mouse_y_pixel_space = _helper_data["MouseY_PixelSpace"]
    dpg.delete_item(sender, children_only=True, slot=2)
    dpg.push_container_stack(sender)
    dpg.configure_item("demo_custom_series", tooltip=False)
    for i in range(0, len(transformed_x)):
        dpg.draw_text((transformed_x[i]+15, transformed_y[i]-15), str(i), size=20)
        dpg.draw_circle((transformed_x[i], transformed_y[i]), 15, fill=(50+i*5, 50+i*50,
↪0, 255))
        if mouse_x_pixel_space < transformed_x[i]+15 and mouse_x_pixel_space >
↪transformed_x[i]-15 and mouse_y_pixel_space > transformed_y[i]-15 and mouse_y_pixel_
↪space < transformed_y[i]+15:
            dpg.draw_circle((transformed_x[i], transformed_y[i]), 30)
            dpg.configure_item("demo_custom_series", tooltip=True)
            dpg.set_value("custom_series_text", "Current Point: " + str(i))
    dpg.pop_container_stack()
```

(continues on next page)

(continued from previous page)

```

with dpg.window(label="Tutorial") as win:
    dpg.add_text("Hover an item for a custom tooltip!")
    with dpg.plot(label="Custom Series", height=400, width=-1):
        dpg.add_plot_legend()
        xaxis = dpg.add_plot_axis(dpg.mvXAxis)
        with dpg.plot_axis(dpg.mvYAxis):
            with dpg.custom_series(x_data, y_data, 2, label="Custom Series",
↳callback=callback, tag="demo_custom_series"):
                dpg.add_text("Current Point: ", tag="custom_series_text")
                dpg.fit_axis_data(dpg.top_container_stack())

dpg.set_primary_window(win, True)
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

3.20.10 Colors and Styles

The color and styles of a plot and series can be changed using theme app item

See also:

For more information on item values [Themes](#)

```

import dearpygui.dearpygui as dpg
from math import sin

dpg.create_context()

sindatax = []
sindatay = []
for i in range(0, 100):
    sindatax.append(i / 100)
    sindatay.append(0.5 + 0.5 * sin(50 * i / 100))
sindatay2 = []
for i in range(0, 100):
    sindatay2.append(2 + 0.5 * sin(50 * i / 100))

with dpg.window(label="Tutorial", width=500, height=400):
    # create a theme for the plot
    with dpg.theme(tag="plot_theme"):
        with dpg.theme_component(dpg.mvStemSeries):
            dpg.add_theme_color(dpg.mvPlotCol_Line, (150, 255, 0), category=dpg.
↳mvThemeCat_Plots)
            dpg.add_theme_style(dpg.mvPlotStyleVar_Marker, dpg.mvPlotMarker_Diamond,
↳category=dpg.mvThemeCat_Plots)
            dpg.add_theme_style(dpg.mvPlotStyleVar_MarkerSize, 7, category=dpg.
↳mvThemeCat_Plots)

        with dpg.theme_component(dpg.mvScatterSeries):
            dpg.add_theme_color(dpg.mvPlotCol_Line, (60, 150, 200), category=dpg.
↳mvThemeCat_Plots)

```

(continues on next page)

```
        dpg.add_theme_style(dpg.mvPlotStyleVar_Marker, dpg.mvPlotMarker_Square,
↪ category=dpg.mvThemeCat_Plots)
        dpg.add_theme_style(dpg.mvPlotStyleVar_MarkerSize, 4, category=dpg.
↪ mvThemeCat_Plots)

# create plot
with dpg.plot(tag="plot", label="Line Series", height=-1, width=-1):

    # optionally create legend
    dpg.add_plot_legend()

    # REQUIRED: create x and y axes
    dpg.add_plot_axis(dpg.mvXAxis, label="x")
    dpg.add_plot_axis(dpg.mvYAxis, label="y", tag="yaxis")

    # series belong to a y axis
    dpg.add_stem_series(sindatax, sindatay, label="0.5 + 0.5 * sin(x)", parent="yaxis
↪", tag="series_data")
    dpg.add_scatter_series(sindatax, sindatay2, label="2 + 0.5 * sin(x)", parent=
↪ "yaxis", tag="series_data2")

    # apply theme to series
    dpg.bind_item_theme("series_data", "plot_theme")
    dpg.bind_item_theme("series_data2", "plot_theme")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.20.11 Colormaps

Under construction

3.20.12 Gallery

3.21 Popups

Popups are windows that disappear when clicked outside of the popup's border.

They are typically used as context menus when right-clicking an item or as dialogs.

In DPG, popups are just windows with *popup* set to **True**, *show* set to **False**, and a *clicked_handler* attached to a widget that shows the window when clicked.

Normally when used, a popup will be shown until you click away from it. By default, a right click activates the popup.

Code

```
import dearpygui.dearpygui as dpg

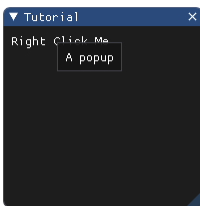
dpg.create_context()

with dpg.window(label="Tutorial"):
    dpg.add_text("Right Click Me")

    with dpg.popup(dpg.last_item()):
        dpg.add_text("A popup")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

Results



3.21.1 Modal Usage

When the modal keyword is set to **True**, the popup will be modal.

This prevents the user from interacting with other windows until the popup is closed. You must hide or delete the popup to remove it.

Code

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Tutorial"):
    dpg.add_text("Left Click Me")

    # check out simple module for details
```

(continues on next page)

(continued from previous page)

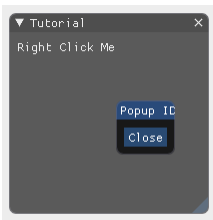
```

    with dpg.popup(dpg.last_item(), mousebutton=dpg.mvMouseButton_Left, modal=True, tag=
↳ "modal_id"):
        dpg.add_button(label="Close", callback=lambda: dpg.configure_item("modal_id",
↳ show=False))

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

Results



Mouse Button options include:

- `_mvMouseButton_Right_`
- `_mvMouseButton_Left_`
- `_mvMouseButton_Middle_`
- `_mvMouseButton_X1_`
- `_mvMouseButton_X2_`

3.21.2 Window as Dialog Popup

This is an example of a window made into a typical dialog.

```

import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Delete Files", modal=True, show=False, tag="modal_id", no_title_
↳ bar=True):
    dpg.add_text("All those beautiful files will be deleted.\nThis operation cannot be
↳ undone!")
    dpg.add_separator()
    dpg.add_checkbox(label="Don't ask me next time")
    with dpg.group(horizontal=True):
        dpg.add_button(label="OK", width=75, callback=lambda: dpg.configure_item("modal_
↳ id", show=False))
        dpg.add_button(label="Cancel", width=75, callback=lambda: dpg.configure_item(
↳ "modal_id", show=False))

with dpg.window(label="Tutorial"):
    dpg.add_button(label="Open Dialog", callback=lambda: dpg.configure_item("modal_id",
↳ show=True))

```

(continues on next page)

(continued from previous page)

```
dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.22 Simple Plots

Simple plots take in a list and plot y-axis data against the number of items in the list. These can be line graphs or histograms and are demonstrated below

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Tutorial", width=500, height=500):
    dpg.add_simple_plot(label="Simpleplot1", default_value=(0.3, 0.9, 0.5, 0.3),
↳height=300)
    dpg.add_simple_plot(label="Simpleplot2", default_value=(0.3, 0.9, 2.5, 8.9), overlay=
↳"Overlying", height=180,
                        histogram=True)

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

You can change the simple plot's data using *set_value*.

Here we are using a mouse move handler and each callback that runs will set the plot data to make it animated!

```
import dearpygui.dearpygui as dpg
from math import sin

dpg.create_context()

def update_plot_data(sender, app_data, plot_data):
    mouse_y = app_data[1]
    if len(plot_data) > 100:
        plot_data.pop(0)
    plot_data.append(sin(mouse_y / 30))
    dpg.set_value("plot", plot_data)

data = []
with dpg.window(label="Tutorial", width=500, height=500):
    dpg.add_simple_plot(label="Simple Plot", min_scale=-1.0, max_scale=1.0, height=300,
↳tag="plot")
```

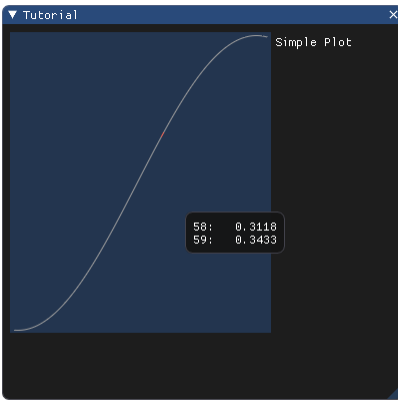
(continues on next page)

(continued from previous page)

```
with dpg.handler_registry():
    dpg.add_mouse_move_handler(callback=update_plot_data, user_data=data)

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

Results



3.23 Staging

The staging system is used to create items or item hierarchies where the parent/root is to be decided at a later time.

Staged items are not submitted for rendering.

Staged items will show up in the item registry.

Items can be moved out of staging by using *move_item*.

The most basic example can be found below:

```
import dearpygui.dearpygui as dpg

dpg.create_context()

def stage_items():
    with dpg.stage(tag="stage1"):
        dpg.add_text("hello, i was added from a stage", tag="text_tag")

def present_stage_items():
    dpg.move_item("text_tag", parent="main_win")

with dpg.window(label="Tutorial", tag="main_win"):
    dpg.add_button(label="stage items", callback=stage_items)
    dpg.add_button(label="present stages items", callback=present_stage_items)
```

(continues on next page)

(continued from previous page)

```
dpg.show_item_registry()

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

Preferred way to “unstage” items is using *unstage*. This will place the items as if they were newly created items according to the standard rules of *Container Stack*.

Also using the *unstage* command will automatically clean up the stage container.

Using *push_container_stack* and *pop_container_stack* is recommended here as it provides better performance when unstaging.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

def stage_items():
    with dpg.stage(tag="stage1"):
        dpg.add_text("hello, i was added from a stage")
        dpg.add_text("hello, i was added from a stage")
        dpg.add_text("hello, i was added from a stage")
        dpg.add_text("hello, i was added from a stage")
        dpg.add_text("hello, i was added from a stage")

def present_stage_items():
    dpg.push_container_stack("main_win")
    dpg.unstage("stage1")
    dpg.pop_container_stack()

with dpg.window(label="Tutorial", tag="main_win", height=400, width=400):
    dpg.add_button(label="stage items", callback=stage_items)
    dpg.add_button(label="present stages items", callback=present_stage_items)

dpg.show_item_registry()

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.23.1 Wrapping Items with Classes

The most obvious benefit of this system is for advanced users who are wrapping DPG items into classes. Rather than having to duplicate the various configurable options as members of the class (to avoid making calls to `configure_item` or `get_item_configuration` before the item is actually created), you can create and stage the item in the constructor of the wrapping class!

Below are 2 examples:

Example 1

```
import dearpygui.dearpygui as dpg

dpg.create_context()

class Button:

    def __init__(self, label):
        with dpg.stage() as self._staging_container_id:
            self._id = dpg.add_button(label=label)

    def set_callback(self, callback):
        dpg.set_item_callback(self._id, callback)

    def get_label(self):
        return dpg.get_item_label(self._id)

    def submit(self, parent):
        dpg.push_container_stack(parent)
        dpg.unstage(self._staging_container_id)
        dpg.pop_container_stack()

my_button = Button("Press me")
my_button.set_callback(lambda: print("I've been pressed!"))

print(my_button.get_label())

with dpg.window(label="Tutorial", tag="main_win"):
    dpg.add_text("hello world")

my_button.submit("main_win")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

Example 2

```
import dearpygui.dearpygui as dpg

dpg.create_context()
```

(continues on next page)

(continued from previous page)

```
class Window:

    def __init__(self, label):
        self._children = []
        with dpg.stage() as stage:
            self.id = dpg.add_window(label=label)
            self.stage = stage

    def add_child(self, child):
        dpg.move_item(child.id, parent=self.id)

    def submit(self):
        dpg.unstage(self.stage)

class Button:

    def __init__(self, label):
        with dpg.stage():
            self.id = dpg.add_button(label=label)

    def set_callback(self, callback):
        dpg.set_item_callback(self.id, callback)

my_button = Button("Press me")
my_button.set_callback(lambda: print("I've been pressed!"))

my_window = Window("Tutorial")
my_window.add_child(my_button)

my_window.submit()

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.24 Tables

The table API is a low level API that can be used to create tables.

It can also be used as a layout mechanism. Tables are composed of multiple components which include columns, rows, cells, and the actual items to be displayed. The best place to learn about the various configuration options for the table is by running the demo!

Below is the minimum example for creating a table

Code

```

import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Tutorial"):
    with dpg.table(header_row=False):

        # use add_table_column to add columns to the table,
        # table columns use child slot 0
        dpg.add_table_column()
        dpg.add_table_column()
        dpg.add_table_column()

        # add_table_next_column will jump to the next row
        # once it reaches the end of the columns
        # table next column use slot 1
        for i in range(0, 4):
            with dpg.table_row():
                for j in range(0, 3):
                    dpg.add_text(f"Row{i} Column{j}")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

Note: The maximum number of columns is 64.

multiple items can go into a single cell by creating a cell as shown

Code

```

import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Tutorial"):

    with dpg.table(header_row=False, resizable=True, policy=dpg.mvTable_
↳SizingStretchProp,
        borders_outerH=True, borders_innerV=True, borders_innerH=True,↳
↳borders_outerV=True):

        dpg.add_table_column(label="Header 1")
        dpg.add_table_column(label="Header 2")
        dpg.add_table_column(label="Header 3")

        # once it reaches the end of the columns
        for i in range(0, 4):
            with dpg.table_row():
                for j in range(0, 3):

```

(continues on next page)

(continued from previous page)

```

        with dpg.table_cell():
            dpg.add_button(label=f"Row{i} Column{j} a")
            dpg.add_button(label=f"Row{i} Column{j} b")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

3.24.1 Borders, Background

You can control the borders of the table using the *borders_innerH*, *borders_innerV*, *borders_outerH*, and *borders_outerV* keywords. You can also turn on alternate row coloring using the *row_background* keyword.

Code

```

import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Tutorial"):

    with dpg.table(header_row=False, row_background=True,
                  borders_innerH=True, borders_outerH=True, borders_innerV=True,
                  borders_outerV=True):

        # use add_table_column to add columns to the table,
        # table columns use child slot 0
        dpg.add_table_column()
        dpg.add_table_column()
        dpg.add_table_column()

        # add_table_next_column will jump to the next row
        # once it reaches the end of the columns
        # table next column use slot 1
        for i in range(0, 4):
            with dpg.table_row():
                for j in range(0, 3):
                    dpg.add_text(f"Row{i} Column{j}")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

3.24.2 Column Headers

Column headers are simply shown by setting *header_row* to **True** and setting the label of the columns.

Code

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Tutorial"):

    with dpg.table(header_row=True):

        # use add_table_column to add columns to the table,
        # table columns use slot 0
        dpg.add_table_column(label="Header 1")
        dpg.add_table_column(label="Header 2")
        dpg.add_table_column(label="Header 3")

        # add_table_next_column will jump to the next row
        # once it reaches the end of the columns
        # table next column use slot 1
        for i in range(0, 4):
            with dpg.table_row():
                for j in range(0, 3):
                    dpg.add_text(f"Row{i} Column{j}")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.24.3 Resizing

In order for a table to have resizable columns, the *resizable* and *borders_innerV* keywords must be set to **True**.

You can also set the sizing policy keyword, *policy*, using the following options

Policy:

- mvTable_SizingFixedFit
- mvTable_SizingFixedSame
- mvTable_SizingStretchProp
- mvTable_SizingStretchSame

3.24.4 Stretch

Below is an example of setting the stretch policy for the entire table

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Tutorial"):

    with dpg.table(header_row=False, resizable=True, policy=dpg.mvTable_
↳SizingStretchProp,
                    borders_outerH=True, borders_innerV=True, borders_outerV=True):

        dpg.add_table_column(label="Header 1")
        dpg.add_table_column(label="Header 2")
        dpg.add_table_column(label="Header 3")

        # add_table_next_column will jump to the next row
        # once it reaches the end of the columns
        # table next column use slot 1
        for i in range(0, 4):
            with dpg.table_row():
                for j in range(0, 3):
                    dpg.add_text(f"Row{i} Column{j}")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

Fixed

Below is an example of setting the fixed fit policy for the entire table

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Tutorial"):

    # Only available if scrollX/scrollY are disabled and stretch columns are not used
    with dpg.table(header_row=False, policy=dpg.mvTable_SizingFixedFit, resizable=True,
↳no_host_extendX=True,
                    borders_innerV=True, borders_outerV=True, borders_outerH=True):

        dpg.add_table_column(label="Header 1")
        dpg.add_table_column(label="Header 2")
        dpg.add_table_column(label="Header 3")
        for i in range(0, 4):
            with dpg.table_row():
                for j in range(0, 3):
                    dpg.add_button(label=f"Row{i} Column{j} a")
```

(continues on next page)

```
dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

Mixed

You can also set columns individually by using the *width_fixed* or *width_stretch* keyword along with the *init_width_or_weight* keyword.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Tutorial"):

    with dpg.table(header_row=True, policy=dpg.mvTable_SizingFixedFit, row_
↳background=True, reorderable=True,
        resizable=True, no_host_extendX=False, hideable=True,
        borders_innerV=True, borders_outerV=True, borders_innerH=True,
        borders_outerH=True):

        dpg.add_table_column(label="AAA", width_fixed=True)
        dpg.add_table_column(label="BBB", width_fixed=True)
        dpg.add_table_column(label="CCC", width_stretch=True, init_width_or_weight=0.0)
        dpg.add_table_column(label="DDD", width_stretch=True, init_width_or_weight=0.0)

        for i in range(0, 5):
            with dpg.table_row():
                for j in range(0, 4):
                    if j == 2 or j == 3:
                        dpg.add_text(f"Stretch {i},{j}")
                    else:
                        dpg.add_text(f"Fixed {i}, {j}")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.24.5 Column Options

There are a large number of options available for table columns which are best learned through running the demo, these include

keyword	default value	description
init_width_or_weight	0.0	sets the starting width (fixed policy) or proportion (stretch) of the column.
default_hide	False	Default as a hidden/disabled column.
default_sort	False	Default as a sorting column.
width_stretch	False	Column will stretch. Preferable with horizontal scrolling disabled (default if table sizing policy is <code>_SizingStretchSame</code> or <code>_SizingStretchProp</code>).
width_fixed	False	Column will not stretch. Preferable with horizontal scrolling enabled (default if table sizing policy is <code>_SizingFixedFit</code> and table is resizable).
no_resize	False	Disable manual resizing.
no_reorder	False	Disable manual reordering this column, this will also prevent other columns from crossing over this column.
no_hide	False	Disable ability to hide/disable this column.
no_clip	False	Disable clipping for this column.
no_sort	False	Disable sorting for this column.
no_sort_ascending	False	Disable ability to sort in the ascending direction.
no_sort_descending	False	Disable ability to sort in the descending direction.
no_header_width	False	Disable header text width contribution to automatic column width.
prefer_sort_ascending	True	Make the initial sort direction Ascending when first sorting on this column (default).
prefer_sort_descending	False	Make the initial sort direction Descending when first sorting on this column.
indent_enabled	False	Use current Indent value when entering cell (default for column 0).
indent_disable	False	Ignore current Indent value when entering cell (default for columns > 0). Indentation changes <code>_within_</code> the cell will still be honored.

3.24.6 Sorting

DPG does not actually do any sorting for you. The table API is a more general purpose API and it is up to the library user to handle sorting. To sort a table based on user interaction, you must assign a callback to the table. This callback will be ran when a user tries to sort a table by clicking on the table's column headers.

```
import dearpygui.dearpygui as dpg

dpg.create_context()
dpg.create_viewport()
dpg.setup_dearpygui()

def sort_callback(sender, sort_specs):

    # sort_specs scenarios:
    # 1. no sorting -> sort_specs == None
    # 2. single sorting -> sort_specs == [[column_id, direction]]
    # 3. multi sorting -> sort_specs == [[column_id, direction], [column_id, direction], ...]
```

(continues on next page)

```
#
# notes:
# 1. direction is ascending if == 1
# 2. direction is ascending if == -1

# no sorting case
if sort_specs is None: return

rows = dpg.get_item_children(sender, 1)

# create a list that can be sorted based on first cell
# value, keeping track of row and value used to sort
sortable_list = []
for row in rows:
    first_cell = dpg.get_item_children(row, 1)[0]
    sortable_list.append([row, dpg.get_value(first_cell)])

def _sorter(e):
    return e[1]

sortable_list.sort(key=_sorter, reverse=sort_specs[0][1] < 0)

# create list of just sorted row ids
new_order = []
for pair in sortable_list:
    new_order.append(pair[0])

dpg.reorder_items(sender, 1, new_order)

with dpg.window(label="Tutorial", width=500):

    with dpg.table(header_row=True, borders_innerH=True, borders_outerH=True,
                  borders_innerV=True, borders_outerV=True, row_background=True,
                  sortable=True, callback=sort_callback):

        dpg.add_table_column(label="One")
        dpg.add_table_column(label="Two", no_sort=True)

        for i in range(25):
            with dpg.table_row():
                dpg.add_input_int(label=" ", step=0, default_value=i)
                dpg.add_text(f"Cell {i}, 1")

# main loop
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.24.7 Scrolling

Under construction

3.24.8 Selecting

You can make rows and/or cells selectable by adding a *selectable* to the table and assigning a callback to it. Use a theme to control the hover style. The *span_columns* option of the *selectable* is used to control whether the row or the cell is selectable.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.theme() as table_theme:
    with dpg.theme_component(dpg.mvTable):
        # dpg.add_theme_color(dpg.mvThemeCol_HeaderHovered, (255, 0, 0, 100),
        ↪category=dpg.mvThemeCat_Core)
        dpg.add_theme_color(dpg.mvThemeCol_HeaderActive, (0, 0, 0, 0), category=dpg.
        ↪mvThemeCat_Core)
        dpg.add_theme_color(dpg.mvThemeCol_Header, (0, 0, 0, 0), category=dpg.mvThemeCat_
        ↪Core)

def clb_selectable(sender, app_data, user_data):
    print(f"Row {user_data}")

with dpg.window(tag="Selectable Tables"):
    with dpg.table(tag="SelectRows", header_row=True) as selectablerows:
        dpg.add_table_column(label="First")
        dpg.add_table_column(label="Second")
        dpg.add_table_column(label="Third")

        for i in range(15):
            with dpg.table_row():
                for j in range(3):
                    dpg.add_selectable(label=f"Row{i} Column{j}", span_columns=True,
                    ↪callback=clb_selectable, user_data=i)
            dpg.bind_item_theme(selectablerows, table_theme)

    with dpg.table(tag="SelectCells", header_row=True) as selectablecells:
        dpg.add_table_column(label="First")
        dpg.add_table_column(label="Second")
        dpg.add_table_column(label="Third")

        for i in range(15):
            with dpg.table_row():
                for j in range(3):
                    dpg.add_selectable(label=f"Row{i} Column{j}", callback=clb_
                    ↪selectable, user_data=(i,j))
            dpg.bind_item_theme(selectablecells, table_theme)

dpg.create_viewport(width=800, height=600)
```

(continues on next page)

```
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.24.9 Clipping

Using a clipper can help performance with large tables.

Try using the example below with and with out clipping and see the effect on the framerate listed in metrics.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

def clipper_toggle(sender):
    dpg.configure_item("table_clip", clipper=dpg.get_value(sender))

with dpg.window(label="Tutorial"):
    dpg.add_checkbox(label="clipper", callback=clipper_toggle, default_value=True)

    with dpg.table(header_row=False, tag="table_clip", clipper=True):

        for i in range(5):
            dpg.add_table_column()

        for i in range(30000):
            with dpg.table_row():
                for j in range(5):
                    dpg.add_text(f"Row{i} Column{j}")

dpg.show_metrics()

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.24.10 Filtering

Under construction

3.24.11 Padding

Under construction

3.24.12 Outer Size

Under construction

3.24.13 Column Widths

Under construction

3.24.14 Row Height

Under construction

3.24.15 Search Delay

Under construction

3.25 Textures & Images

DPG uses the Graphics Processing Unit (GPU) to create the graphical user interface(GUI) you see. To display an image, you must first create a texture with the image data that can then be uploaded to the GPU. These textures belong to a texture registry.

We offer 3 types of textures

- Static
- Dynamic
- Raw

These textures are then used in the following App Items

- **mvDrawImage**
- **mvImage**
- **mvImageButton**
- **mvImageSeries**

They are always 1D lists or arrays.

Using the keyword Show on the texture registry will open the texture registry.

3.25.1 Static Textures

Static textures are used for images that do not change often. They are typically loaded at startup. If they need to be updated, you would delete and recreate them. These accept python lists, tuples, numpy arrays, and any type that supports python's buffer protocol with contiguous data. Below is a simple example

```
import dearpygui.dearpygui as dpg

dpg.create_context()

texture_data = []
for i in range(0, 100 * 100):
    texture_data.append(255 / 255)
    texture_data.append(0)
    texture_data.append(255 / 255)
    texture_data.append(255 / 255)

with dpg.texture_registry(show=True):
    dpg.add_static_texture(width=100, height=100, default_value=texture_data, tag=
↳ "texture_tag")

with dpg.window(label="Tutorial"):
    dpg.add_image("texture_tag")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

The texture can be deleted with `dpg.delete_item("texture_tag")`. However, for the tag/alias to be released items that use `"texture_tag"` (such as a plot series) must also be deleted.

3.25.2 Dynamic Textures

Dynamic textures are used for small to medium sized textures that can change per frame. These can be updated with `set_value` but the width and height must be the same as when the texture was first created. These are similar to raw textures except these perform safety checks and conversion. Below is a simple example

```
import dearpygui.dearpygui as dpg

dpg.create_context()

texture_data = []
for i in range(0, 100 * 100):
    texture_data.append(255 / 255)
    texture_data.append(0)
    texture_data.append(255 / 255)
    texture_data.append(255 / 255)

with dpg.texture_registry(show=True):
    dpg.add_dynamic_texture(width=100, height=100, default_value=texture_data, tag=
↳ "texture_tag")
```

(continues on next page)

(continued from previous page)

```

def _update_dynamic_textures(sender, app_data, user_data):
    new_color = dpg.get_value(sender)
    new_color[0] = new_color[0] / 255
    new_color[1] = new_color[1] / 255
    new_color[2] = new_color[2] / 255
    new_color[3] = new_color[3] / 255

    new_texture_data = []
    for i in range(0, 100 * 100):
        new_texture_data.append(new_color[0])
        new_texture_data.append(new_color[1])
        new_texture_data.append(new_color[2])
        new_texture_data.append(new_color[3])

    dpg.set_value("texture_tag", new_texture_data)

with dpg.window(label="Tutorial"):
    dpg.add_image("texture_tag")
    dpg.add_color_picker((255, 0, 255, 255), label="Texture",
                        no_side_preview=True, alpha_bar=True, width=200,
                        callback=_update_dynamic_textures)

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

3.25.3 Raw Textures

Raw textures are used in the same way as dynamic textures. The main differences

- Only accepts arrays (numpy, python, etc.)
- No safety checks are performed.

These textures are used for high performance applications that require updating large textures every frame. Below is a simple example

```

import dearpygui.dearpygui as dpg
import array

dpg.create_context()

texture_data = []
for i in range(0, 100 * 100):
    texture_data.append(255 / 255)

```

(continues on next page)

```

texture_data.append(0)
texture_data.append(255 / 255)
texture_data.append(255 / 255)

raw_data = array.array('f', texture_data)

with dpg.texture_registry(show=True):
    dpg.add_raw_texture(width=100, height=100, default_value=raw_data, format=dpg.
↳mvFormat_Float_rgba, tag="texture_tag")

def update_dynamic_texture(sender, app_data, user_data):
    new_color = dpg.get_value(sender)
    new_color[0] = new_color[0] / 255
    new_color[1] = new_color[1] / 255
    new_color[2] = new_color[2] / 255
    new_color[3] = new_color[3] / 255

    for i in range(0, 100 * 100 * 4):
        raw_data[i] = new_color[i % 4]

with dpg.window(label="Tutorial"):
    dpg.add_image("texture_tag")
    dpg.add_color_picker((255, 0, 255, 255), label="Texture",
        no_side_preview=True, alpha_bar=True, width=200,
        callback=update_dynamic_texture)

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

3.25.4 Formats

The following formats are currently supported

Format	Static Texture	Dynamic Texture	Raw Texture
mvFormat_Float_rgba			
mvFormat_Float_rgb	•	•	*
mvFormat_Int_rgba	•	•	•
mvFormat_Int_rgb	•	•	•

Note:

mvFormat_Float_rgb not currently supported on MacOS
More formats will be added in the future.

3.25.5 Loading Images

DPG provides the function `load_image` for loading image data from a file.

This function returns a tuple where

- 0 -> width
- 1 -> height
- 2 -> channels
- 3 -> data (1D array, mvBuffer)

On failure, returns **None**.

The accepted file types include:

- JPEG (no 12-bit-per-channel JPEG OR JPEG with arithmetic coding)
- PNG
- BMP
- PSD
- GIF
- HDR
- PIC
- PPM
- PGM

A simple example can be found below

```
import dearpygui.dearpygui as dpg

dpg.create_context()

width, height, channels, data = dpg.load_image("Somefile.png")

with dpg.texture_registry(show=True):
    dpg.add_static_texture(width=width, height=height, default_value=data, tag="texture_
↪tag")

with dpg.window(label="Tutorial"):
    dpg.add_image("texture_tag")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.25.6 Saving Images

New in 1.4. DPG provides the function `save_image` for saving image data to a file.

The image is a rectangle of pixels stored from left-to-right, top-to-bottom. Each pixel contains up to 4 components of data interleaved with 8-bits per channel, in the following order: 1=Y, 2=YA, 3=RGB, 4=RGBA. (Y is monochrome color.)

PNG creates output files with the same number of components as the input. The BMP format expands Y to RGB in the file format and does not output alpha.

Additional options will be released with v1.4.1.

The accepted file types include:

- PNG
- JPG (new in v1.4.1)
- BMP (new in v1.4.1)
- TGA (new in v1.4.1)
- HDR (new in v1.4.1)

File type is determined by extension. Must be lowercase (png, jpg, bmp, tga, hdr).

A simple example can be found below

```
import dearpygui.dearpygui as dpg

dpg.create_context()
dpg.create_viewport()
dpg.setup_dearpygui()

width, height = 255, 255

data = []
for i in range(width*height):
    data.append(255)
    data.append(255)
    data.append(0)

with dpg.window(label="Tutorial"):
    dpg.add_button(label="Save Image", callback=lambda:dpg.save_image(file="newImage.png", width=width, height=height, data=data, components=3))

dpg.show_viewport()
while dpg.is_dearpygui_running():
    dpg.render_dearpygui_frame()

dpg.destroy_context()
```

3.26 Themes

Themes are containers which are composed of:

Theme Components: containers within a theme that can specify an item type the theme colors/styles target

Theme Colors: items that are added to a theme component and set colors

Theme Styles: items that are added to a theme component and set styles

The theme can be:

- bound as the default theme. This will have a global effect across all windows and propagate.
- bound to a container. This will propagate to its children if applicable theme components are in the theme.
- bound to an item type if applicable theme components are in the theme.

Theme components must have a specified item type. This can either be *mvAll* for all items or a specific item type.

Style and color items have a named constant and will apply that constant to their components named item type. Style and color items must have a named category. Constants contain their category in the name.

Theme colors and styles fall into the following categories:

mvThemeCat_Plots: Items that are associated with plots. Style/color constants identified by *mvPlotCol_**** or *mvPlotStyle_****

mvThemeCat_Nodes: Items that are associated with Nodes. Style/color constants identified by *mvNodeCol_**** or *mvNodeStyle_****

mvThemeCat_Core: All other items within dearpygui. Style/color constants identified by *mvThemeCol_**** or *mvThemeStyle_****

3.26.1 Default Theme (global)

Default themes will apply the theme globally across all windows and propagate to children.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Tutorial", pos=(20, 50), width=275, height=225) as win1:
    t1 = dpg.add_input_text(default_value="some text")
    t2 = dpg.add_input_text(default_value="some text")
    with dpg.child_window(height=100):
        t3 = dpg.add_input_text(default_value="some text")
        dpg.add_input_int()
    dpg.add_input_text(default_value="some text")

with dpg.window(label="Tutorial", pos=(320, 50), width=275, height=225) as win2:
    dpg.add_input_text(default_value="some text")
    dpg.add_input_int()

with dpg.theme() as global_theme:
    with dpg.theme_component(dpg.mvAll):
        dpg.add_theme_color(dpg.mvThemeCol_FrameBg, (255, 140, 23), category=dpg.
↪mvThemeCat_Core)
```

(continues on next page)

(continued from previous page)

```

    dpg.add_theme_style(dpg.mvStyleVar_FrameRounding, 5, category=dpg.mvThemeCat_
↪Core)

    with dpg.theme_component(dpg.mvInputInt):
        dpg.add_theme_color(dpg.mvThemeCol_FrameBg, (140, 255, 23), category=dpg.
↪mvThemeCat_Core)
        dpg.add_theme_style(dpg.mvStyleVar_FrameRounding, 5, category=dpg.mvThemeCat_
↪Core)

dpg.bind_theme(global_theme)

dpg.show_style_editor()

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

3.26.2 Container Propagation

Applying a theme to a container will propagate the theme to its children:

```

import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Tutorial", pos=(20, 50), width=275, height=225) as win1:
    t1 = dpg.add_input_text(default_value="some text")
    t2 = dpg.add_input_text(default_value="some text")
    with dpg.child_window(height=100):
        t3 = dpg.add_input_text(default_value="some text")
        dpg.add_input_int()
        dpg.add_input_text(default_value="some text")

with dpg.window(label="Tutorial", pos=(320, 50), width=275, height=225) as win2:
    dpg.add_input_text(default_value="some text")
    dpg.add_input_int()

with dpg.theme() as container_theme:

    with dpg.theme_component(dpg.mvAll):
        dpg.add_theme_color(dpg.mvThemeCol_FrameBg, (150, 100, 100), category=dpg.
↪mvThemeCat_Core)
        dpg.add_theme_style(dpg.mvStyleVar_FrameRounding, 5, category=dpg.mvThemeCat_
↪Core)

    with dpg.theme_component(dpg.mvInputInt):
        dpg.add_theme_color(dpg.mvThemeCol_FrameBg, (100, 150, 100), category=dpg.
↪mvThemeCat_Core)
        dpg.add_theme_style(dpg.mvStyleVar_FrameRounding, 5, category=dpg.mvThemeCat_
↪Core)

```

(continues on next page)

(continued from previous page)

```
dpg.bind_item_theme(win1, container_theme)

dpg.show_style_editor()

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.26.3 Item Specific

Applying a theme to an item will override any previous themes on the specified item if the theme contains an applicable component.

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Tutorial", pos=(20, 50), width=275, height=225) as win1:
    t1 = dpg.add_input_text(default_value="some text")
    t2 = dpg.add_input_text(default_value="some text")
    with dpg.child_window(height=100):
        t3 = dpg.add_input_text(default_value="some text")
        dpg.add_input_int()
        dpg.add_input_text(default_value="some text")

with dpg.window(label="Tutorial", pos=(320, 50), width=275, height=225) as win2:
    dpg.add_input_text(default_value="some text")
    dpg.add_input_int()

with dpg.theme() as item_theme:
    with dpg.theme_component(dpg.mvAll):
        dpg.add_theme_color(dpg.mvThemeCol_FrameBg, (200, 200, 100), category=dpg.
↪mvThemeCat_Core)
        dpg.add_theme_style(dpg.mvStyleVar_FrameRounding, 0, category=dpg.mvThemeCat_
↪Core)

dpg.bind_item_theme(t2, item_theme)

dpg.show_style_editor()

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

3.26.4 Priority of Themes

The theme prioritizes the latest applied theme in the order of

1. specific item
2. container inherited
3. global

```
import dearpygui.dearpygui as dpg

dpg.create_context()

with dpg.window(label="Tutorial", pos=(20, 50), width=275, height=225) as win1:
    t1 = dpg.add_input_text(default_value="some text")
    t2 = dpg.add_input_text(default_value="some text")
    with dpg.child_window(height=100):
        t3 = dpg.add_input_text(default_value="some text")
        dpg.add_input_int()
        dpg.add_input_text(default_value="some text")

with dpg.window(label="Tutorial", pos=(320, 50), width=275, height=225) as win2:
    dpg.add_input_text(default_value="some text")
    dpg.add_input_int()

with dpg.theme() as global_theme:
    with dpg.theme_component(dpg.mvAll):
        dpg.add_theme_color(dpg.mvThemeCol_FrameBg, (255, 140, 23), category=dpg.
↪mvThemeCat_Core)
        dpg.add_theme_style(dpg.mvStyleVar_FrameRounding, 5, category=dpg.mvThemeCat_
↪Core)

    with dpg.theme_component(dpg.mvInputInt):
        dpg.add_theme_color(dpg.mvThemeCol_FrameBg, (140, 255, 23), category=dpg.
↪mvThemeCat_Core)
        dpg.add_theme_style(dpg.mvStyleVar_FrameRounding, 5, category=dpg.mvThemeCat_
↪Core)

with dpg.theme() as container_theme:
    with dpg.theme_component(dpg.mvAll):
        dpg.add_theme_color(dpg.mvThemeCol_FrameBg, (150, 100, 100), category=dpg.
↪mvThemeCat_Core)
        dpg.add_theme_style(dpg.mvStyleVar_FrameRounding, 5, category=dpg.mvThemeCat_
↪Core)

    with dpg.theme_component(dpg.mvInputInt):
        dpg.add_theme_color(dpg.mvThemeCol_FrameBg, (100, 150, 100), category=dpg.
↪mvThemeCat_Core)
        dpg.add_theme_style(dpg.mvStyleVar_FrameRounding, 5, category=dpg.mvThemeCat_
↪Core)

with dpg.theme() as item_theme:
    with dpg.theme_component(dpg.mvAll):
        dpg.add_theme_color(dpg.mvThemeCol_FrameBg, (200, 200, 100), category=dpg.
↪mvThemeCat_Core)
```

(continues on next page)

(continued from previous page)

```

        dpg.add_theme_style(dpg.mvStyleVar_FrameRounding, 0, category=dpg.mvThemeCat_
↪Core)

dpg.bind_theme(global_theme)
dpg.bind_item_theme(win1, container_theme)
dpg.bind_item_theme(t2, item_theme)

dpg.show_style_editor()

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

3.26.5 Theme for Disabled Items

Each item has a separate disabled theme that is used when the item is disabled. The disabled theme follows the same propagation rules as the enabled theme. When an item's parameter *enabled* is set to *False* the item will use its disabled theme. If no disabled theme has been set the default theme will be used.

```

import dearpygui.dearpygui as dpg

dpg.create_context()
dpg.create_viewport()
dpg.setup_dearpygui()

with dpg.theme() as disabled_theme:
    with dpg.theme_component(dpg.mvInputFloat, enabled_state=False):
        dpg.add_theme_color(dpg.mvThemeCol_Text, [255, 0, 0])
        dpg.add_theme_color(dpg.mvThemeCol_Button, [255, 0, 0])

    with dpg.theme_component(dpg.mvInputInt, enabled_state=False):
        dpg.add_theme_color(dpg.mvThemeCol_Text, [255, 0, 0])
        dpg.add_theme_color(dpg.mvThemeCol_Button, [255, 0, 0])

dpg.bind_theme(disabled_theme)

with dpg.window(label="tutorial"):
    dpg.add_input_float(label="Input float", enabled=False)
    dpg.add_input_int(label="Input int", enabled=False)

dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()

```

3.26.6 Plot Markers

Plot Markers	
mvPlotMarker_None	mvPlotMarker_Circle
mvPlotMarker_Square	mvPlotMarker_Diamond
mvPlotMarker_Up	mvPlotMarker_Down
mvPlotMarker_Left	mvPlotMarker_Right
mvPlotMarker_Cross	mvPlotMarker_Plus
mvPlotMarker_Asterisk	

3.26.7 Core Colors

Core Colors		
mvThemeCol_Text	mvThemeCol_TabActive	mvThemeCol_SliderGrabActive
mvThemeCol_TextDisabled	mvThemeCol_TabUnfocused	mvThemeCol_Button
mvThemeCol_WindowBg	mvThemeCol_TabUnfocusedActive	mvThemeCol_ButtonHovered
mvThemeCol_ChildBg	mvThemeCol_DockingPreview	mvThemeCol_ButtonActive
mvThemeCol_Border	mvThemeCol_DockingEmptyBg	mvThemeCol_Header
mvThemeCol_PopupBg	mvThemeCol_PlotLines	mvThemeCol_HeaderHovered
mvThemeCol_BorderShadow	mvThemeCol_PlotLinesHovered	mvThemeCol_HeaderActive
mvThemeCol_FrameBg	mvThemeCol_PlotHistogram	mvThemeCol_Separator
mvThemeCol_FrameBgHovered	mvThemeCol_PlotHistogramHovered	mvThemeCol_SeparatorHovered
mvThemeCol_FrameBgActive	mvThemeCol_TableHeaderBg	mvThemeCol_SeparatorActive
mvThemeCol_TitleBg	mvThemeCol_TableBorderStrong	mvThemeCol_ResizeGrip
mvThemeCol_TitleBgActive	mvThemeCol_TableBorderLight	mvThe- meCol_ResizeGripHovered
mvThemeCol_TitleBgCollapsed	mvThemeCol_TableRowBg	mvThemeCol_ResizeGripActive
mvThemeCol_MenuBarBg	mvThemeCol_TableRowBgAlt	mvThemeCol_Tab
mvThemeCol_ScrollbarBg	mvThemeCol_TextSelectedBg	mvThemeCol_TabHovered
mvThemeCol_ScrollbarGrab	mvThemeCol_DragDropTarget	
mvThe- meCol_ScrollbarGrabHovered	mvThemeCol_NavHighlight	
mvThemeCol_ScrollbarGrabActive	mvThe- meCol_NavWindowingHighlight	
mvThemeCol_CheckMark	mvThemeCol_NavWindowingDimBg	
mvThemeCol_SliderGrab	mvThemeCol_ModalWindowDimBg	

3.26.8 Plot Colors

Plot Colors		
mvPlotCol_Line	mvPlotCol_LegendBg	mvPlotCol_AxisBgHovered
mvPlotCol_Fill	mvPlotCol_LegendBorder	mvPlotCol_AxisGrid
mvPlotCol_MarkerOutline	mvPlotCol_LegendText	mvPlotCol_AxisText
mvPlotCol_MarkerFill	mvPlotCol_TitleText	mvPlotCol_Selection
mvPlotCol_ErrorBar	mvPlotCol_InlayText	
mvPlotCol_FrameBg	mvPlotCol_AxisBg	
mvPlotCol_PlotBg	mvPlotCol_AxisBgActive	
mvPlotCol_PlotBorder	mvPlotCol_Crosshairs	

3.26.9 Node Colors

Node Colors		
mvNodeCol_NodeBackground	mvNodeCol_TitleBarSelected	mvNodeCol_BoxSelector
mvNodeCol_NodeBackgroundHovered	mvNodeCol_Link	mvNodeCol_BoxSelectorOutline
mvNodeCol_NodeBackgroundSelected	mvNodeCol_LinkHovered	mvNodeCol_GridBackground
mvNodeCol_NodeOutline	mvNodeCol_LinkSelected	mvNodeCol_GridLine
mvNodeCol_TitleBar	mvNodeCol_Pin	mvNodeCol_PinHovered
mvNodeCol_TitleBarHovered		

3.26.10 Core Styles

Constant	Components
mvStyleVar_Alpha	1
mvStyleVar_DisabledAlpha	1
mvStyleVar_WindowPadding	2
mvStyleVar_WindowRounding	1
mvStyleVar_WindowBorderSize	1
mvStyleVar_WindowMinSize	2
mvStyleVar_WindowTitleAlign	2
mvStyleVar_ChildRounding	1
mvStyleVar_ChildBorderSize	1
mvStyleVar_PopupRounding	1
mvStyleVar_PopupBorderSize	1
mvStyleVar_FramePadding	2
mvStyleVar_FrameRounding	1
mvStyleVar_FrameBorderSize	1
mvStyleVar_ItemSpacing	2
mvStyleVar_ItemInnerSpacing	2
mvStyleVar_IndentSpacing	1
mvStyleVar_CellPadding	2
mvStyleVar_ScrollbarSize	1
mvStyleVar_ScrollbarRounding	1
mvStyleVar_GrabMinSize	1
mvStyleVar_GrabRounding	1
mvStyleVar_TabRounding	1
mvStyleVar_TabBorderSize	1
mvStyleVar_TabBarBorderSize	1
mvStyleVar_TableAngledHeadersAngle	1
mvStyleVar_TableAngledHeadersTextAlign	2
mvStyleVar_ButtonTextAlign	2
mvStyleVar_SelectableTextAlign	2
mvStyleVar_SeparatorTextBorderSize	1
mvStyleVar_SeparatorTextAlign	2
mvStyleVar_SeparatorTextPadding	2

3.26.11 Plot Styles

Constant	Components
mvPlotStyleVar_LineWeight	1
mvPlotStyleVar_Marker	1
mvPlotStyleVar_MarkerSize	1
mvPlotStyleVar_MarkerWeight	1
mvPlotStyleVar_FillAlpha	1
mvPlotStyleVar_ErrorBarSize	1
mvPlotStyleVar_ErrorBarWeight	1
mvPlotStyleVar_DigitalBitHeight	1
mvPlotStyleVar_DigitalBitGap	1
mvPlotStyleVar_PlotBorderSize	1
mvPlotStyleVar_MinorAlpha	1
mvPlotStyleVar_MajorTickLen	2
mvPlotStyleVar_MinorTickLen	2
mvPlotStyleVar_MajorTickSize	2
mvPlotStyleVar_MinorTickSize	2
mvPlotStyleVar_MajorGridSize	2
mvPlotStyleVar_MinorGridSize	2
mvPlotStyleVar_PlotPadding	2
mvPlotStyleVar_LabelPadding	2
mvPlotStyleVar_LegendPadding	2
mvPlotStyleVar_LegendInnerPadding	2
mvPlotStyleVar_LegendSpacing	2
mvPlotStyleVar_MousePosPadding	2
mvPlotStyleVar_AnnotationPadding	2
mvPlotStyleVar_FitPadding	2
mvPlotStyleVar_PlotDefaultSize	2
mvPlotStyleVar_PlotMinSize	2

3.26.12 Node Styles

Constant	Components
mvNodeStyleVar_GridSpacing	1
mvNodeStyleVar_NodeCornerRounding	1
mvNodeStyleVar_NodePaddingHorizontal	1
mvNodeStyleVar_NodePaddingVertical	1
mvNodeStyleVar_NodeBorderThickness	1
mvNodeStyleVar_LinkThickness	1
mvNodeStyleVar_LinkLineSegmentsPerLength	1
mvNodeStyleVar_LinkHoverDistance	1
mvNodeStyleVar_PinCircleRadius	1
mvNodeStyleVar_PinQuadSideLength	1
mvNodeStyleVar_PinTriangleSideLength	1
mvNodeStyleVar_PinLineThickness	1
mvNodeStyleVar_PinHoverRadius	1
mvNodeStyleVar_PinOffset	1

3.27 Tooltips

Tooltips are windows that appear when an item is hovered.

Tooltips are containers that can hold any other UI Item.

Tooltips must have the parent argument. This is the *tag* of the parent what will show the tooltip.

```
import dearpygui.dearpygui as dpg

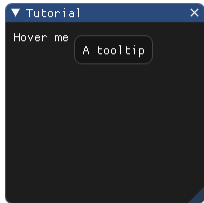
dpg.create_context()

with dpg.window(label="Tutorial"):
    dpg.add_text("Hover me", tag="tooltip_parent")

    with dpg.tooltip("tooltip_parent"):
        dpg.add_text("A tooltip")

dpg.create_viewport(title='Custom Title', width=800, height=600)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

Results



3.28 dearpygui.dearpygui

`dearpygui.dearpygui.add_2d_histogram_series(x, y, **kwargs)`

Adds a 2d histogram series.

Parameters

- **x** (*Any*) –
- **y** (*Any*) –
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)

- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **show** (*bool, optional*) – Attempt to render widget.
- **xbins** (*int, optional*) –
- **ybins** (*int, optional*) –
- **xmin_range** (*float, optional*) – set the min x range value, the values under this min will be ignored
- **xmax_range** (*float, optional*) – set the max x range value, the values over this max will be ignored
- **ymin_range** (*float, optional*) – set the min y range value, the values under this min will be ignored
- **ymax_range** (*float, optional*) – set the max y range value, the values over this max will be ignored. If all xmin, xmax, ymin and ymax are 0.0, then the values will be the min and max values of the series
- **density** (*bool, optional*) – counts will be normalized, i.e. the PDF will be visualized
- **outliers** (*bool, optional*) – exclude values outside the specified histogram range from the count used for normalizing
- **col_major** (*bool, optional*) – data will be read in column major order
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_3d_slider(**kwargs)`

Adds a 3D box slider.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.

- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*Union[List[float], Tuple[float, ...]], optional*) –
- **max_x** (*float, optional*) – Applies upper limit to slider.
- **max_y** (*float, optional*) – Applies upper limit to slider.
- **max_z** (*float, optional*) – Applies upper limit to slider.
- **min_x** (*float, optional*) – Applies lower limit to slider.
- **min_y** (*float, optional*) – Applies lower limit to slider.
- **min_z** (*float, optional*) – Applies lower limit to slider.
- **scale** (*float, optional*) – Size of the widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_activated_handler(parent, **kwargs)`
 deprecated function

`dearpygui.dearpygui.add_active_handler(parent, **kwargs)`
 deprecated function

`dearpygui.dearpygui.add_alias(alias, item)`
 Adds an alias.

Parameters

- **alias** (*str*) –
- **item** (*Union[int, str]*) –

Returns None

`dearpygui.dearpygui.add_area_series(x, y, **kwargs)`
 Adds an area series to a plot.

Parameters

- **x** (*Any*) –
- **y** (*Any*) –
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).

- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **show** (*bool, optional*) – Attempt to render widget.
- **fill** (*Union[List[int], Tuple[int, ...]], optional*) –
- **contribute_to_bounds** (*bool, optional*) –
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_axis_tag(**kwargs)`

Adds custom labels to axes.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **show** (*bool, optional*) – Attempt to render widget.
- **default_value** (*float, optional*) –
- **color** (*Union[List[int], Tuple[int, ...]], optional*) –
- **auto_rounding** (*bool, optional*) – When enabled, the value displayed on the tag will be automatically rounded to the precision of other values displayed at axis’ ticks. Only makes sense when label is not set, i.e. when the tag displays its location on the axis.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_bar_group_series(values, label_ids, group_size, **kwargs)`

Adds a bar groups series to a plot.

Parameters

- **values** (*Any*) –
- **label_ids** (*Union[List[str], Tuple[str, ...]]*) – Label of each bar in a group
- **group_size** (*int*) – Number of bars in a group
- **label** (*str, optional*) – Overrides ‘name’ as label.

- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **show** (*bool, optional*) – Attempt to render widget.
- **group_width** (*float, optional*) – Size of bar groups
- **shift** (*int, optional*) – The position on the x axis where to start plotting bar groups
- **horizontal** (*bool, optional*) – bar groups will be rendered horizontally on the current y-axis
- **stacked** (*bool, optional*) – items in a group will be stacked on top of each other
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_bar_series(x, y, **kwargs)`

Adds a bar series to a plot.

Parameters

- **x** (*Any*) –
- **y** (*Any*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **show** (*bool, optional*) – Attempt to render widget.
- **weight** (*float, optional*) –
- **horizontal** (*bool, optional*) – bars will be rendered horizontally on the current y-axis
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_bool_value(**kwargs)`

Adds a bool value.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **default_value** (*bool, optional*) –
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_button(**kwargs)`

Adds a button.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.

- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **small** (*bool, optional*) – Shrinks the size of the button to the text of the label it contains. Useful for embedding in text.
- **arrow** (*bool, optional*) – Displays an arrow in place of the text string. This requires the direction keyword.
- **direction** (*int, optional*) – Sets the cardinal direction for the arrow by using constants `mvDir_Left`, `mvDir_Up`, `mvDir_Down`, `mvDir_Right`, `mvDir_None`. Arrow keyword must be set to `True`.
- **repeat** (*bool, optional*) – Hold to continuously repeat the click.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_candle_series(dates, opens, closes, lows, highs, **kwargs)`

Adds a candle series to a plot.

Parameters

- **dates** (*Any*) –
- **opens** (*Any*) –
- **closes** (*Any*) –
- **lows** (*Any*) –
- **highs** (*Any*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **show** (*bool, optional*) – Attempt to render widget.
- **bull_color** (*Union[List[int], Tuple[int, ...]], optional*) –
- **bear_color** (*Union[List[int], Tuple[int, ...]], optional*) –
- **weight** (*float, optional*) –
- **tooltip** (*bool, optional*) –
- **time_unit** (*int, optional*) – `mvTimeUnit_*` constants. Default `mvTimeUnit_Day`.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_char_remap(source, target, **kwargs)`

Remaps a character.

Parameters

- **source** (*int*) –
- **target** (*int*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns `Union[int, str]`

`dearpygui.dearpygui.add_checkbox(**kwargs)`

Adds a checkbox.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.

- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*bool, optional*) – Sets the default value of the checkmark
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_child(**kwargs)`

(deprecated function) Adds an embedded child window. Will show scrollbars when items do not fit.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **delay_search** (*bool, optional*) – Delays searching container for specified items until the end of the app. Possible optimization when a container has many children that are not accessed often.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **border** (*bool, optional*) – Shows/Hides the border around the sides.
- **autosize_x** (*bool, optional*) – Autosize the window to its parents size in x.
- **autosize_y** (*bool, optional*) – Autosize the window to its parents size in y.
- **no_scrollbar** (*bool, optional*) – Disable scrollbars (window can still scroll with mouse or programmatically).
- **horizontal_scrollbar** (*bool, optional*) – Allow horizontal scrollbar to appear (off by default).

- **menubar** (*bool, optional*) – Shows/Hides the menubar at the top.

Returns Union[int, str]

`dearpygui.dearpygui.add_child_window(**kwargs)`

Adds an embedded child window. Will show scrollbars when items do not fit. About using `auto_resize/resizable` flags: size measurement for a given axis is only performed when the child window is within visible boundaries, or is just appearing and it won't update its auto-size while clipped. While not perfect, it is a better default behavior as the always-on performance gain is more valuable than the occasional 'resizing after becoming visible again' glitch. You may also use `always_auto_resize` to force an update even when child window is not in view. However doing so will degrade performance. Remember that combining both `auto_resize_x` and `auto_resize_y` defeats purpose of a scrolling region and is NOT recommended.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **border** (*bool, optional*) – Shows/Hides the border around the sides.
- **autosize_x** (*bool, optional*) – Autosize the window to its parents size in x.
- **autosize_y** (*bool, optional*) – Autosize the window to its parents size in y.
- **no_scrollbar** (*bool, optional*) – Disable scrollbars (window can still scroll with mouse or programmatically).
- **horizontal_scrollbar** (*bool, optional*) – Allow horizontal scrollbar to appear (off by default).
- **menubar** (*bool, optional*) – Shows/Hides the menubar at the top.

- **no_scroll_with_mouse** (*bool, optional*) – Disable user vertically scrolling with mouse wheel.
- **flattened_navigation** (*bool, optional*) – Allow gamepad/keyboard navigation to cross over parent border to this child (only use on child that have no scrolling!)
- **always_use_window_padding** (*bool, optional*) – Pad with `style.WindowPadding` even if no border are drawn (no padding by default for non-bordered child windows because it makes more sense)
- **resizable_x** (*bool, optional*) – Allow resize from right border (layout direction). Enable `.ini` saving.
- **resizable_y** (*bool, optional*) – Allow resize from bottom border (layout direction).
- **always_auto_resize** (*bool, optional*) – Combined with `auto_resize_x/auto_resize_y`. Always measure size even when child is hidden and always disable clipping optimization! NOT RECOMMENDED.
- **frame_style** (*bool, optional*) – Style the child window like a framed item: use `FrameBg`, `FrameRounding`, `FrameBorderSize`, `FramePadding` instead of `ChildBg`, `ChildRounding`, `ChildBorderSize`, `WindowPadding`.
- **auto_resize_x** (*bool, optional*) – Enable auto-resizing width based on child content. Read ‘IMPORTANT: Size measurement’ details above.
- **auto_resize_y** (*bool, optional*) – Enable auto-resizing height based on child content. Read ‘IMPORTANT: Size measurement’ details above.
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Returns `Union[int, str]`

`dearpygui.dearpygui.add_clicked_handler(parent, button=-1, **kwargs)`
 deprecated function

`dearpygui.dearpygui.add_clipper(**kwargs)`

Helper to manually clip large list of items. Increases performance by not searching or drawing widgets outside of the clipped region.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.

- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Returns Union[int, str]

`dearpygui.dearpygui.add_collapsing_header(**kwargs)`

Adds a collapsing header to add items to. Must be closed with the end command.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **closable** (*bool, optional*) – Adds the ability to hide this widget by pressing the (x) in the top right of widget.
- **default_open** (*bool, optional*) – Sets the collapseable header open by default.
- **open_on_double_click** (*bool, optional*) – Need double-click to open node.
- **open_on_arrow** (*bool, optional*) – Only open when clicking on the arrow part.
- **leaf** (*bool, optional*) – No collapsing, no arrow (use as a convenience for leaf nodes).
- **bullet** (*bool, optional*) – Display a bullet instead of arrow.
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Returns Union[int, str]

`dearpygui.dearpygui.add_color_button(default_value=(0, 0, 0, 255), **kwargs)`

Adds a color button.

Parameters

- **default_value** (Union[List[int], Tuple[int, ...]], optional) –
- **label** (str, optional) – Overrides ‘name’ as label.
- **user_data** (Any, optional) – User data for callbacks
- **use_internal_label** (bool, optional) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (Union[int, str], optional) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (int, optional) – Width of the item.
- **height** (int, optional) – Height of the item.
- **indent** (int, optional) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (Union[int, str], optional) – Parent to add this item to. (runtime adding)
- **before** (Union[int, str], optional) – This item will be displayed before the specified item in the parent.
- **payload_type** (str, optional) – Sender string type must be the same as the target for the target to run the payload_callback.
- **callback** (Callable, optional) – Registers a callback.
- **drag_callback** (Callable, optional) – Registers a drag callback for drag and drop.
- **drop_callback** (Callable, optional) – Registers a drop callback for drag and drop.
- **show** (bool, optional) – Attempt to render widget.
- **enabled** (bool, optional) – Turns off functionality of widget and applies the disabled theme.
- **pos** (Union[List[int], Tuple[int, ...]], optional) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (str, optional) – Used by filter widget.
- **tracked** (bool, optional) – Scroll tracking
- **track_offset** (float, optional) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **no_alpha** (bool, optional) – Removes the displayed slider that can change alpha channel.
- **no_border** (bool, optional) – Disable border around the image.
- **no_drag_drop** (bool, optional) – Disable ability to drag and drop small preview (color square) to apply colors to other items.
- **alpha_preview** (int, optional) – mvColorEdit_AlphaPreviewNone, mvColorEdit_AlphaPreview, or mvColorEdit_AlphaPreviewHalf
- **no_tooltip** (bool, optional) – Disable tooltip when hovering the button.
- **id** (Union[int, str], optional) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_color_edit(default_value=(0, 0, 0, 255), **kwargs)`

Adds an RGBA color editor. Left clicking the small color preview will provide a color picker. Click and dragging the small color preview will copy the color to be applied on any other color widget.

Parameters

- **default_value** (Union[List[int], Tuple[int, ...]], optional) –
- **label** (str, optional) – Overrides ‘name’ as label.
- **user_data** (Any, optional) – User data for callbacks
- **use_internal_label** (bool, optional) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (Union[int, str], optional) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (int, optional) – Width of the item.
- **height** (int, optional) – Height of the item.
- **indent** (int, optional) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (Union[int, str], optional) – Parent to add this item to. (runtime adding)
- **before** (Union[int, str], optional) – This item will be displayed before the specified item in the parent.
- **source** (Union[int, str], optional) – Overrides ‘id’ as value storage key.
- **payload_type** (str, optional) – Sender string type must be the same as the target for the target to run the payload_callback.
- **callback** (Callable, optional) – Registers a callback.
- **drag_callback** (Callable, optional) – Registers a drag callback for drag and drop.
- **drop_callback** (Callable, optional) – Registers a drop callback for drag and drop.
- **show** (bool, optional) – Attempt to render widget.
- **enabled** (bool, optional) – Turns off functionality of widget and applies the disabled theme.
- **pos** (Union[List[int], Tuple[int, ...]], optional) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (str, optional) – Used by filter widget.
- **tracked** (bool, optional) – Scroll tracking
- **track_offset** (float, optional) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **no_alpha** (bool, optional) – Removes the displayed slider that can change alpha channel.
- **no_picker** (bool, optional) – Disable picker popup when color square is clicked.
- **no_options** (bool, optional) – Disable toggling options menu when right-clicking on inputs/small preview.
- **no_small_preview** (bool, optional) – Disable colored square preview next to the inputs. (e.g. to show only the inputs). This only displays if the side preview is not shown.

- **no_inputs** (*bool, optional*) – Disable inputs sliders/text widgets. (e.g. to show only the small preview colored square)
- **no_tooltip** (*bool, optional*) – Disable tooltip when hovering the preview.
- **no_label** (*bool, optional*) – Disable display of inline text label.
- **no_drag_drop** (*bool, optional*) – Disable ability to drag and drop small preview (color square) to apply colors to other items.
- **alpha_bar** (*bool, optional*) – Show vertical alpha bar/gradient in picker.
- **alpha_preview** (*int, optional*) – `mvColorEdit_AlphaPreviewNone`, `mvColorEdit_AlphaPreview`, or `mvColorEdit_AlphaPreviewHalf`
- **display_mode** (*int, optional*) – `mvColorEdit_rgb`, `mvColorEdit_hsv`, or `mvColorEdit_hex`
- **display_type** (*int, optional*) – `mvColorEdit_uint8` or `mvColorEdit_float`
- **input_mode** (*int, optional*) – `mvColorEdit_input_*` values
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_color_picker(default_value=(0, 0, 0, 255), **kwargs)`

Adds an RGB color picker. Right click the color picker for options. Click and drag the color preview to copy the color and drop on any other color widget to apply. Right Click allows the style of the color picker to be changed.

Parameters

- **default_value** (*Union[List[int], Tuple[int, ...]], optional*) –
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.

- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **no_alpha** (*bool, optional*) – Removes the displayed slider that can change alpha channel.
- **no_side_preview** (*bool, optional*) – Disable bigger color preview on right side of the picker, use small colored square preview instead, unless small preview is also hidden.
- **no_small_preview** (*bool, optional*) – Disable colored square preview next to the inputs. (e.g. to show only the inputs). This only displays if the side preview is not shown.
- **no_inputs** (*bool, optional*) – Disable inputs sliders/text widgets. (e.g. to show only the small preview colored square)
- **no_tooltip** (*bool, optional*) – Disable tooltip when hovering the preview.
- **no_label** (*bool, optional*) – Disable display of inline text label.
- **alpha_bar** (*bool, optional*) – Show vertical alpha bar/gradient in picker.
- **display_rgb** (*bool, optional*) – Override `_display_` type among RGB/HSV/Hex.
- **display_hsv** (*bool, optional*) – Override `_display_` type among RGB/HSV/Hex.
- **display_hex** (*bool, optional*) – Override `_display_` type among RGB/HSV/Hex.
- **picker_mode** (*int, optional*) – `mvColorPicker_bar` or `mvColorPicker_wheel`
- **alpha_preview** (*int, optional*) – `mvColorEdit_AlphaPreviewNone`, `mvColorEdit_AlphaPreview`, or `mvColorEdit_AlphaPreviewHalf`
- **display_type** (*int, optional*) – `mvColorEdit_uint8` or `mvColorEdit_float`
- **input_mode** (*int, optional*) – `mvColorEdit_input_*` values.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_color_value(**kwargs)`

Adds a color value.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **default_value** (*Union[List[float], Tuple[float, ...]], optional*) –

- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_colormap(colors, qualitative, **kwargs)`

Adds a legend that pairs colors with normalized value 0.0->1.0. Each color will be This is typically used with a heat series. (ex. `[[0, 0, 0, 255], [255, 255, 255, 255]]`) will be mapped to a soft transition from 0.0-1.0)

Parameters

- **colors** (*Any*) – colors that will be mapped to the normalized value 0.0->1.0
- **qualitative** (*bool*) – Qualitative will create hard transitions for color boundries across the value range when enabled.
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **show** (*bool, optional*) – Attempt to render widget.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_colormap_button(default_value=(0, 0, 0, 255), **kwargs)`

Adds a button that a color map can be bound to.

Parameters

- **default_value** (*Union[List[int], Tuple[int, ...]], optional*) –
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **callback** (*Callable, optional*) – Registers a callback.

- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_colormap_registry(**kwargs)`

Adds a colormap registry.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_colormap_scale(**kwargs)`

Adds a legend that pairs values with colors. This is typically used with a heat series.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)

- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **colormap** (*Union[int, str], optional*) – `mvPlotColormap_*` constants or `mvColorMap` uuid from a color map registry
- **min_scale** (*float, optional*) – Sets the min number of the color scale. Typically is the same as the min scale from the heat series.
- **max_scale** (*float, optional*) – Sets the max number of the color scale. Typically is the same as the max scale from the heat series.
- **format** (*str, optional*) – Formatting used for the labels.
- **reverse_dir** (*bool, optional*) – invert the colormap bar and axis scale (this only affects rendering; if you only want to reverse the scale mapping, make `scale_min > scale_max`)
- **mirror** (*bool, optional*) – render the colormap label and tick labels on the opposite side
- **id** (*Union[int, str], optional*) – (deprecated)
- **drag_callback** (*Callable, optional*) – (deprecated)

Returns `Union[int, str]`

`dearpygui.dearpygui.add_colormap_slider(**kwargs)`

Adds a color slider that a color map can be bound to.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.

- **callback** (*Callable, optional*) – Registers a callback.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*float, optional*) –
- **id** (*Union[int, str], optional*) – (deprecated)
- **drag_callback** (*Callable, optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_combo(items=(), **kwargs)`

Adds a combo dropdown that allows a user to select a single option from a drop down window. All items will be shown as selectable on the dropdown.

Parameters

- **items** (*Union[List[str], Tuple[str, ...]], optional*) – A tuple of items to be shown in the drop down window. Can consist of any combination of types but will convert all items to strings to be shown.
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.

- **pos** (*Union[List[int], Tuple[int, ...]]*, *optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str*, *optional*) – Used by filter widget.
- **tracked** (*bool*, *optional*) – Scroll tracking
- **track_offset** (*float*, *optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*str*, *optional*) – Sets a selected item from the drop down by specifying the string value.
- **popup_align_left** (*bool*, *optional*) – Align the contents on the popup toward the left.
- **no_arrow_button** (*bool*, *optional*) – Display the preview box without the square arrow button indicating dropdown activity.
- **no_preview** (*bool*, *optional*) – Display only the square arrow button and not the selected value.
- **fit_width** (*bool*, *optional*) – Fit the available width.
- **height_mode** (*int*, *optional*) – Controls the number of items shown in the dropdown by the constants `mvComboHeight_Small`, `mvComboHeight_Regular`, `mvComboHeight_Large`, `mvComboHeight_Largest`
- **id** (*Union[int, str]*, *optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_custom_series(x, y, channel_count, **kwargs)`

Adds a custom series to a plot. New in 1.6.

Parameters

- **x** (*Any*) –
- **y** (*Any*) –
- **channel_count** (*int*) –
- **label** (*str*, *optional*) – Overrides 'name' as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str]*, *optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str]*, *optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str]*, *optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str]*, *optional*) – Overrides 'id' as value storage key.
- **callback** (*Callable*, *optional*) – Registers a callback.
- **show** (*bool*, *optional*) – Attempt to render widget.
- **y1** (*Any*, *optional*) –
- **y2** (*Any*, *optional*) –
- **y3** (*Any*, *optional*) –

- **tooltip** (*bool, optional*) – Show tooltip when plot is hovered.
- **no_fit** (*bool, optional*) – the item won't be considered for plot fits
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_date_picker(**kwargs)`

Adds a date picker.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*dict, optional*) –
- **level** (*int, optional*) – Use available constants. `mvDatePickerLevel_Day`, `mvDatePickerLevel_Month`, `mvDatePickerLevel_Year`
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_deactivated_after_edit_handler(parent, **kwargs)`

deprecated function

`dearpygui.dearpygui.add_deactivated_handler(parent, **kwargs)`

deprecated function

`dearpygui.dearpygui.add_digital_series(x, y, **kwargs)`

Adds a digital series to a plot. Digital plots do not respond to y drag or zoom, and are always referenced to the bottom of the plot.

Parameters

- **x** (*Any*) –
- **y** (*Any*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns `Union[int, str]`

`dearpygui.dearpygui.add_double4_value(**kwargs)`

Adds a double value.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **default_value** (*Any, optional*) –
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns `Union[int, str]`

`dearpygui.dearpygui.add_double_value(**kwargs)`

Adds a double value.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks

- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **default_value** (*float, optional*) –
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_drag_double(**kwargs)`

Adds drag for a single double value. Useful when drag float is not accurate enough. Directly entry can be done with double click or CTRL+Click. Min and Max alone are a soft limit for the drag. Use clamped keyword to also apply limits to the direct entry modes.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom

- **default_value** (*float, optional*) –
- **format** (*str, optional*) – Determines the format the float will be displayed as use python string formatting.
- **speed** (*float, optional*) – Sets the sensitivity the float will be modified while dragging.
- **min_value** (*float, optional*) – Applies a limit only to dragging entry only.
- **max_value** (*float, optional*) – Applies a limit only to dragging entry only.
- **no_input** (*bool, optional*) – Disable direct entry methods or Enter key allowing to input text directly into the widget.
- **clamped** (*bool, optional*) – Applies the min and max limits to direct entry methods also such as double click and CTRL+Click.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_drag_doublex(**kwargs)`

Adds drag input for a set of double values up to 4. Useful when drag float is not accurate enough. Directly entry can be done with double click or CTRL+Click. Min and Max alone are a soft limit for the drag. Use `clamped` keyword to also apply limits to the direct entry modes.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.

- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*Any, optional*) –
- **size** (*int, optional*) – Number of doubles to be displayed.
- **format** (*str, optional*) – Determines the format the float will be displayed as use python string formatting.
- **speed** (*float, optional*) – Sets the sensitivity the float will be modified while dragging.
- **min_value** (*float, optional*) – Applies a limit only to dragging entry only.
- **max_value** (*float, optional*) – Applies a limit only to dragging entry only.
- **no_input** (*bool, optional*) – Disable direct entry methods or Enter key allowing to input text directly into the widget.
- **clamped** (*bool, optional*) – Applies the min and max limits to direct entry methods also such as double click and CTRL+Click.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_drag_float(**kwargs)`

Adds drag for a single float value. Directly entry can be done with double click or CTRL+Click. Min and Max alone are a soft limit for the drag. Use clamped keyword to also apply limits to the direct entry modes.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.

- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*float, optional*) –
- **format** (*str, optional*) – Determines the format the float will be displayed as use python string formatting.
- **speed** (*float, optional*) – Sets the sensitivity the float will be modified while dragging.
- **min_value** (*float, optional*) – Applies a limit only to dragging entry only.
- **max_value** (*float, optional*) – Applies a limit only to dragging entry only.
- **no_input** (*bool, optional*) – Disable direct entry methods or Enter key allowing to input text directly into the widget.
- **clamped** (*bool, optional*) – Applies the min and max limits to direct entry methods also such as double click and CTRL+Click.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_drag_floatx(**kwargs)`

Adds drag input for a set of float values up to 4. Directly entry can be done with double click or CTRL+Click. Min and Max alone are a soft limit for the drag. Use `clamped` keyword to also apply limits to the direct entry modes.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **callback** (*Callable, optional*) – Registers a callback.

- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*Union[List[float], Tuple[float, ...]], optional*) –
- **size** (*int, optional*) – Number of floats to be displayed.
- **format** (*str, optional*) – Determines the format the float will be displayed as use python string formatting.
- **speed** (*float, optional*) – Sets the sensitivity the float will be modified while dragging.
- **min_value** (*float, optional*) – Applies a limit only to dragging entry only.
- **max_value** (*float, optional*) – Applies a limit only to dragging entry only.
- **no_input** (*bool, optional*) – Disable direct entry methods or Enter key allowing to input text directly into the widget.
- **clamped** (*bool, optional*) – Applies the min and max limits to direct entry methods also such as double click and CTRL+Click.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_drag_int(**kwargs)`

Adds drag for a single int value. Directly entry can be done with double click or CTRL+Click. Min and Max alone are a soft limit for the drag. Use `clamped` keyword to also apply limits to the direct entry modes.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.

- **payload_type** (*str*, *optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **callback** (*Callable*, *optional*) – Registers a callback.
- **drag_callback** (*Callable*, *optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable*, *optional*) – Registers a drop callback for drag and drop.
- **show** (*bool*, *optional*) – Attempt to render widget.
- **enabled** (*bool*, *optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]]*, *optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str*, *optional*) – Used by filter widget.
- **tracked** (*bool*, *optional*) – Scroll tracking
- **track_offset** (*float*, *optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*int*, *optional*) –
- **format** (*str*, *optional*) – Determines the format the float will be displayed as use python string formatting.
- **speed** (*float*, *optional*) – Sets the sensitivity the float will be modified while dragging.
- **min_value** (*int*, *optional*) – Applies a limit only to dragging entry only.
- **max_value** (*int*, *optional*) – Applies a limit only to dragging entry only.
- **no_input** (*bool*, *optional*) – Disable direct entry methods or Enter key allowing to input text directly into the widget.
- **clamped** (*bool*, *optional*) – Applies the min and max limits to direct entry methods also such as double click and CTRL+Click.
- **id** (*Union[int, str]*, *optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_drag_intx(**kwargs)`

Adds drag input for a set of int values up to 4. Directly entry can be done with double click or CTRL+Click. Min and Max alone are a soft limit for the drag. Use `clamped` keyword to also apply limits to the direct entry modes.

Parameters

- **label** (*str*, *optional*) – Overrides 'name' as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str]*, *optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int*, *optional*) – Width of the item.
- **indent** (*int*, *optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str]*, *optional*) – Parent to add this item to. (runtime adding)

- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*Union[List[int], Tuple[int, ...]], optional*) –
- **size** (*int, optional*) – Number of ints to be displayed.
- **format** (*str, optional*) – Determines the format the int will be displayed as use python string formatting.
- **speed** (*float, optional*) – Sets the sensitivity the float will be modified while dragging.
- **min_value** (*int, optional*) – Applies a limit only to dragging entry only.
- **max_value** (*int, optional*) – Applies a limit only to dragging entry only.
- **no_input** (*bool, optional*) – Disable direct entry methods or Enter key allowing to input text directly into the widget.
- **clamped** (*bool, optional*) – Applies the min and max limits to direct entry methods also such as double click and CTRL+Click.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_drag_line(**kwargs)`

Adds a drag line to a plot.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)

- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **default_value** (*float, optional*) –
- **color** (*Union[List[int], Tuple[int, ...]], optional*) –
- **thickness** (*float, optional*) –
- **show_label** (*bool, optional*) –
- **vertical** (*bool, optional*) –
- **delayed** (*bool, optional*) – tool rendering will be delayed one frame; useful when applying position-constraints
- **no_cursor** (*bool, optional*) – drag tools won’t change cursor icons when hovered or held
- **no_fit** (*bool, optional*) – the drag tool won’t be considered for plot fits
- **no_inputs** (*bool, optional*) – lock the tool from user inputs
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_drag_payload(**kwargs)`

User data payload for drag and drop operations.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **show** (*bool, optional*) – Attempt to render widget.
- **drag_data** (*Any, optional*) – Drag data
- **drop_data** (*Any, optional*) – Drop data
- **payload_type** (*str, optional*) –
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_drag_point(**kwargs)`

Adds a drag point to a plot.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.

- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **default_value** (*Any, optional*) –
- **color** (*Union[List[int], Tuple[int, ...]], optional*) –
- **thickness** (*float, optional*) –
- **show_label** (*bool, optional*) –
- **offset** (*Union[List[float], Tuple[float, ...]], optional*) – Offset of the label, in pixels, relative to the drag point itself
- **clamped** (*bool, optional*) – Keep the label within the visible area of the plot even if the drag point itself goes outside of the visible area
- **delayed** (*bool, optional*) – tool rendering will be delayed one frame; useful when applying position-constraints
- **no_cursor** (*bool, optional*) – drag tools won’t change cursor icons when hovered or held
- **no_fit** (*bool, optional*) – the drag tool won’t be considered for plot fits
- **no_inputs** (*bool, optional*) – lock the tool from user inputs
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_drag_rect(**kwargs)`

Adds a drag rectangle to a plot.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.

- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **default_value** (*Any, optional*) – The coordinates are specified in a sequence of: (xmin, ymin, xmax, ymax)
- **color** (*Union[List[int], Tuple[int, ...]], optional*) –
- **delayed** (*bool, optional*) – tool rendering will be delayed one frame; useful when applying position-constraints
- **no_cursor** (*bool, optional*) – drag tools won't change cursor icons when hovered or held
- **no_fit** (*bool, optional*) – the drag tool won't be considered for plot fits
- **no_inputs** (*bool, optional*) – lock the tool from user inputs
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_draw_layer(**kwargs)`

New in 1.1. Creates a layer useful for grouping drawlist items.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **perspective_divide** (*bool, optional*) – New in 1.1. apply perspective divide
- **depth_clipping** (*bool, optional*) – New in 1.1. apply depth clipping
- **cull_mode** (*int, optional*) – New in 1.1. culling mode, `mvCullMode_*` constants. Only works with triangles currently.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_draw_node(**kwargs)`

New in 1.1. Creates a drawing node to associate a transformation matrix. Child node matrices will concatenate.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).

- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_drawlist(width, height, **kwargs)`

Adds a drawing canvas.

Parameters

- **width** (*int*) –
- **height** (*int*) –
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Returns Union[int, str]

`dearpygui.dearpygui.add_dummy(**kwargs)`

(deprecated function) Adds a spacer or 'dummy' object.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks.

- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int]], optional*) – Places the item relative to window coordinates, [0,0] is top left.

Returns Union[int, str]

`dearpygui.dearpygui.add_dynamic_texture`(*width, height, default_value, **kwargs*)
Adds a dynamic texture.

Parameters

- **width** (*int*) –
- **height** (*int*) –
- **default_value** (*Union[List[float], Tuple[float, ...]]*) –
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_edited_handler`(*parent, **kwargs*)
deprecated function

`dearpygui.dearpygui.add_error_series`(*x, y, negative, positive, **kwargs*)
Adds an error series to a plot.

Parameters

- **x** (*Any*) –
- **y** (*Any*) –
- **negative** (*Any*) –
- **positive** (*Any*) –

- **label** (*str*, *optional*) – Overrides ‘name’ as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str]*, *optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str]*, *optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str]*, *optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str]*, *optional*) – Overrides ‘id’ as value storage key.
- **show** (*bool*, *optional*) – Attempt to render widget.
- **contribute_to_bounds** (*bool*, *optional*) –
- **horizontal** (*bool*, *optional*) –
- **id** (*Union[int, str]*, *optional*) – (deprecated)

Returns `Union[int, str]`

`dearpygui.dearpygui.add_file_dialog(**kwargs)`

Displays a file or directory selector depending on keywords. Displays a file dialog by default. Callback will be ran when the file or directory picker is closed. The `app_data` argument will be populated with information related to the file and directory as a dictionary.

Parameters

- **label** (*str*, *optional*) – Overrides ‘name’ as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str]*, *optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int*, *optional*) – Width of the item.
- **height** (*int*, *optional*) – Height of the item.
- **callback** (*Callable*, *optional*) – Registers a callback.
- **show** (*bool*, *optional*) – Attempt to render widget.
- **default_path** (*str*, *optional*) – Path that the file dialog will default to when opened.
- **default_filename** (*str*, *optional*) – Default name that will show in the file name input.
- **file_count** (*int*, *optional*) – Number of visible files in the dialog.
- **modal** (*bool*, *optional*) – Forces user interaction with the file selector.
- **directory_selector** (*bool*, *optional*) – Shows only directory/paths as options. Allows selection of directory/paths only.
- **min_size** (*Union[List[int], Tuple[int, ...]]*, *optional*) – Minimum window size.
- **max_size** (*Union[List[int], Tuple[int, ...]]*, *optional*) – Maximum window size.

- **cancel_callback** (*Callable, optional*) – Callback called when cancel button is clicked.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_file_extension(extension, **kwargs)`

Creates a file extension filter option in the file dialog.

Parameters

- **extension** (*str*) – Extension that will show as an when the parent is a file dialog.
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **custom_text** (*str, optional*) – Replaces the displayed text in the drop down for this extension.
- **color** (*Union[List[int], Tuple[int, ...]], optional*) – Color for the text that will be shown with specified extensions.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_filter_set(**kwargs)`

Helper to parse and apply text filters (e.g. aaaaa[, bbbbb][, ccccc])

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.

- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Returns Union[int, str]

`dearpygui.dearpygui.add_float4_value(**kwargs)`

Adds a float4 value.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **default_value** (*Union[List[float], Tuple[float, ...]], optional*) –
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_float_value(**kwargs)`

Adds a float value.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **default_value** (*float, optional*) –
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_float_vect_value(**kwargs)`

Adds a float vect value.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks

- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **default_value** (*Union[List[float], Tuple[float, ...]], optional*) –
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_focus_handler(parent, **kwargs)`
 deprecated function

`dearpygui.dearpygui.add_font(file, size, **kwargs)`
 Adds font to a font registry.

Parameters

- **file** (*str*) –
- **size** (*int*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **pixel_snapH** (*bool, optional*) – Align every glyph to pixel boundary in horizontal direction. Useful if you are rendering text piece-by-piece (e.g. for coloring).
- **pixel_snapV** (*bool, optional*) – Align scaled GlyphOffset.y to pixel boundaries in ImGui.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)
- **default_font** (*bool, optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_font_chars(chars: Union[List[int], Tuple[int, ...]], *, label: str = None, user_data: Any = None, use_internal_label: bool = True, tag: Union[int, str] = 0, parent: Union[int, str] = 0, **kwargs) → Union[int, str]`

(deprecated function) Adds specific font characters to a font.

Parameters

- **chars** (*Union[List[int], Tuple[int, ...]]*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).

- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_font_range`(*first_char: int, last_char: int, *, label: str = None, user_data: Any = None, use_internal_label: bool = True, tag: Union[int, str] = 0, parent: Union[int, str] = 0, **kwargs*) → Union[int, str]

(deprecated function) Adds a range of font characters to a font.

Parameters

- **first_char** (*int*) –
- **last_char** (*int*) –
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_font_range_hint`(*hint: int, *, label: str = None, user_data: Any = None, use_internal_label: bool = True, tag: Union[int, str] = 0, parent: Union[int, str] = 0, **kwargs*) → Union[int, str]

(deprecated function) Adds a range of font characters (**mvFontRangeHint_** constants).

Parameters

- **hint** (*int*) –
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_font_registry`(***kwargs*)

Adds a font registry.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.

- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_group(**kwargs)`

Creates a group that other widgets can belong to. The group allows item commands to be issued for all of its members. Enable property acts in a special way enabling/disabling everything inside the group. (Use `mvStyleVar_DisabledAlpha` to edit colors within the disabled group.)

Args: `label` (str, optional): Overrides 'name' as label. `user_data` (*Any, optional*): User data for callbacks `use_internal_label` (*bool, optional*): Use generated internal label instead of user specified (appends ### uuid). `tag` (*Union[int, str], optional*): Unique id used to programmatically refer to the item.If label is unused this will be the label. `width` (*int, optional*): Width of the item. `height` (*int, optional*): Height of the item. `indent` (*int, optional*): Offsets the widget to the right the specified number multiplied by the indent style. `parent` (*Union[int, str], optional*): Parent to add this item to. (`runtime adding`) `before` (*Union[int, str], optional*): This item will be displayed before the specified item in the parent. `payload_type` (str, optional): Sender string type must be the same as the target for the target to run the `payload_callback`. `drag_callback` (*Callable, optional*): Registers a drag callback for drag and drop. `drop_callback` (*Callable, optional*): Registers a drop callback for drag and drop. `show` (*bool, optional*): Attempt to render widget. `enabled` (*bool, optional*): Turns off functionality of widget and applies the disabled theme. `pos` (*Union[List[int], Tuple[int, ...]], optional*): Places the item relative to window coordinates, [0,0] is top left. `filter_key` (str, optional): Used by filter widget. `tracked` (*bool, optional*): Scroll tracking `track_offset` (float, optional): 0.0f:top, 0.5f:center, 1.0f:bottom `horizontal` (*bool, optional*): Forces child widgets to be added in a horizontal layout. `horizontal_spacing` (float, optional): Spacing for the horizontal layout. `xoffset` (float, optional): Offset from containing window x item location within group. `id` (*Union[int, str], optional*): (deprecated) `delay_search` (*bool, optional*): (deprecated)This was used as an optimization hint but is not relevant anymore.

Returns: Union[int, str]

`dearpygui.dearpygui.add_handler_registry(**kwargs)`

Adds a handler registry.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_heat_series(x, rows, cols, **kwargs)`

Adds a heat series to a plot.

Parameters

- **x** (*Any*) –
- **rows** (*int*) –
- **cols** (*int*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **show** (*bool, optional*) – Attempt to render widget.
- **scale_min** (*float, optional*) – Sets the color scale min. Typically paired with the color scale widget `scale_min`.
- **scale_max** (*float, optional*) – Sets the color scale max. Typically paired with the color scale widget `scale_max`.
- **bounds_min** (*Any, optional*) –
- **bounds_max** (*Any, optional*) –
- **format** (*str, optional*) –
- **contribute_to_bounds** (*bool, optional*) –
- **col_major** (*bool, optional*) – data will be read in column major order
- **id** (*Union[int, str], optional*) – (deprecated)

Returns `Union[int, str]`

`dearpygui.dearpygui.add_histogram_series(x, **kwargs)`

Adds a histogram series to a plot.

Parameters

- **x** (*Any*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)

- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **show** (*bool, optional*) – Attempt to render widget.
- **bins** (*int, optional*) –
- **bar_scale** (*float, optional*) –
- **min_range** (*float, optional*) – set the min range value, the values under this min will be ignored
- **max_range** (*float, optional*) – set the max range value, the values over this max will be ignored. If both min and max are 0.0, then the values will be the min and max values of the series
- **cumulative** (*bool, optional*) – each bin will contain its count plus the counts of all previous bins
- **density** (*bool, optional*) – counts will be normalized, i.e. the PDF will be visualized, or the CDF will be visualized if Cumulative is also set
- **outliers** (*bool, optional*) – exclude values outside the specified histogram range from the count toward normalizing and cumulative counts
- **horizontal** (*bool, optional*) – histogram bars will be rendered horizontally
- **contribute_to_bounds** (*bool, optional*) –
- **id** (*Union[int, str], optional*) – (deprecated)
- **cumlative** (*bool, optional*) – (deprecated)Deprecated because of typo

Returns Union[int, str]

`dearpygui.dearpygui.add_hline_series(x, **kwargs)`

(deprecated function) Adds an infinite horizontal line series to a plot.

Parameters

- **x** (*Any*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_hover_handler(parent, **kwargs)`
deprecated function

`dearpygui.dearpygui.add_image(texture_tag, **kwargs)`

Adds an image from a specified texture. `uv_min` and `uv_max` represent the normalized texture coordinates of the original image that will be shown. Using range (0.0,0.0)->(1.0,1.0) for texture coordinates will generally display the entire texture.

Parameters

- **texture_tag** (*Union[int, str]*) – The texture_tag should come from a texture that was added to a texture registry.
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **tint_color** (*Union[List[float], Tuple[float, ...]], optional*) – Applies a color tint to the entire texture.
- **border_color** (*Union[List[float], Tuple[float, ...]], optional*) – Displays a border of the specified color around the texture.
- **uv_min** (*Union[List[float], Tuple[float, ...]], optional*) – Normalized texture coordinates min point.
- **uv_max** (*Union[List[float], Tuple[float, ...]], optional*) – Normalized texture coordinates max point.

- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_image_button(texture_tag, **kwargs)`

Adds an button with a texture. `uv_min` and `uv_max` represent the normalized texture coordinates of the original image that will be shown. Using range (0.0,0.0)->(1.0,1.0) texture coordinates will generally display the entire texture

Parameters

- **texture_tag** (*Union[int, str]*) – The texture_tag should come from a texture that was added to a texture registry.
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **tint_color** (*Union[List[float], Tuple[float, ...]], optional*) – Applies a color tint to the entire texture.
- **background_color** (*Union[List[float], Tuple[float, ...]], optional*) – Displays a border of the specified color around the texture.

- **uv_min** (*Union[List[float], Tuple[float, ...]], optional*) – Normalized texture coordinates min point.
- **uv_max** (*Union[List[float], Tuple[float, ...]], optional*) – Normalized texture coordinates max point.
- **id** (*Union[int, str], optional*) – (deprecated)
- **frame_padding** (*int, optional*) – (deprecated) Empty space around the outside of the texture. Button will show around the texture.

Returns Union[int, str]

`dearpygui.dearpygui.add_image_series(texture_tag, bounds_min, bounds_max, **kwargs)`

Adds an image series to a plot.

Parameters

- **texture_tag** (*Union[int, str]*) –
- **bounds_min** (*Any*) –
- **bounds_max** (*Any*) –
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **show** (*bool, optional*) – Attempt to render widget.
- **uv_min** (*Union[List[float], Tuple[float, ...]], optional*) – normalized texture coordinates
- **uv_max** (*Union[List[float], Tuple[float, ...]], optional*) – normalized texture coordinates
- **tint_color** (*Union[List[int], Tuple[int, ...]], optional*) –
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_inf_line_series(x, **kwargs)`

Adds an infinite line series to a plot.

Parameters

- **x** (*Any*) –
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).

- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **show** (*bool, optional*) – Attempt to render widget.
- **horizontal** (*bool, optional*) –
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_input_double(**kwargs)`

Adds input for an double. Useful when input float is not accurate enough. +/- buttons can be activated by setting the value of step.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom

- **default_value** (*float, optional*) –
- **format** (*str, optional*) – Determines the format the float will be displayed as use python string formatting.
- **min_value** (*float, optional*) – Value for lower limit of input. By default this limits the step buttons. Use `min_clamped` to limit manual input.
- **max_value** (*float, optional*) – Value for upper limit of input. By default this limits the step buttons. Use `max_clamped` to limit manual input.
- **step** (*float, optional*) – Increment to change value by when the step buttons are pressed. Setting this and `step_fast` to a value of 0 or less will turn off step buttons.
- **step_fast** (*float, optional*) – Increment to change value by when `ctrl + step` buttons are pressed. Setting this and `step` to a value of 0 or less will turn off step buttons.
- **min_clamped** (*bool, optional*) – Activates and deactivates the enforcement of `min_value`.
- **max_clamped** (*bool, optional*) – Activates and deactivates the enforcement of `max_value`.
- **on_enter** (*bool, optional*) – Only runs callback on enter key press.
- **readonly** (*bool, optional*) – Activates read only mode where no text can be input but text can still be highlighted.
- **accept_empty_input** (*bool, optional*) – Treat empty input as a zero value.
- **display_empty_value** (*bool, optional*) – If True, display an empty input if the value is zero. Generally used together with `accept_empty_input`.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_input_doublex(**kwargs)`

Adds multi double input for up to 4 double values. Useful when input float mulit is not accurate enough.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **callback** (*Callable, optional*) – Registers a callback.

- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*Any, optional*) –
- **format** (*str, optional*) – Determines the format the float will be displayed as use python string formatting.
- **min_value** (*float, optional*) – Value for lower limit of input for each cell. Use `min_clamped` to turn on.
- **max_value** (*float, optional*) – Value for upper limit of input for each cell. Use `max_clamped` to turn on.
- **size** (*int, optional*) – Number of components displayed for input.
- **min_clamped** (*bool, optional*) – Activates and deactivates the enforcement of `min_value`.
- **max_clamped** (*bool, optional*) – Activates and deactivates the enforcement of `max_value`.
- **on_enter** (*bool, optional*) – Only runs callback on enter key press.
- **readonly** (*bool, optional*) – Activates read only mode where no text can be input but text can still be highlighted.
- **accept_empty_input** (*bool, optional*) – Treat empty input as a zero value.
- **display_empty_value** (*bool, optional*) – If True, display an empty input if the value is zero. Generally used together with `accept_empty_input`.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_input_float(**kwargs)`

Adds input for an float. +/- buttons can be activated by setting the value of step.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.

- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*float, optional*) –
- **format** (*str, optional*) – Determines the format the float will be displayed as use python string formatting.
- **min_value** (*float, optional*) – Value for lower limit of input. By default this limits the step buttons. Use min_clamped to limit manual input.
- **max_value** (*float, optional*) – Value for upper limit of input. By default this limits the step buttons. Use max_clamped to limit manual input.
- **step** (*float, optional*) – Increment to change value by when the step buttons are pressed. Setting this and step_fast to a value of 0 or less will turn off step buttons.
- **step_fast** (*float, optional*) – Increment to change value by when ctrl + step buttons are pressed. Setting this and step to a value of 0 or less will turn off step buttons.
- **min_clamped** (*bool, optional*) – Activates and deactivates the enforcement of min_value.
- **max_clamped** (*bool, optional*) – Activates and deactivates the enforcement of max_value.
- **on_enter** (*bool, optional*) – Only runs callback on enter key press.
- **readonly** (*bool, optional*) – Activates read only mode where no text can be input but text can still be highlighted.
- **accept_empty_input** (*bool, optional*) – Treat empty input as a zero value.
- **display_empty_value** (*bool, optional*) – If True, display an empty input if the value is zero. Generally used together with accept_empty_input.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_input_floatx(**kwargs)`

Adds multi float input for up to 4 float values.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*Union[List[float], Tuple[float, ...]], optional*) –
- **format** (*str, optional*) – Determines the format the float will be displayed as use python string formatting.
- **min_value** (*float, optional*) – Value for lower limit of input for each cell. Use `min_clamped` to turn on.
- **max_value** (*float, optional*) – Value for upper limit of input for each cell. Use `max_clamped` to turn on.
- **size** (*int, optional*) – Number of components displayed for input.
- **min_clamped** (*bool, optional*) – Activates and deactivates the enforcement of `min_value`.

- **max_clamped** (*bool, optional*) – Activates and deactivates the enforcement of `max_value`.
- **on_enter** (*bool, optional*) – Only runs callback on enter key press.
- **readonly** (*bool, optional*) – Activates read only mode where no text can be input but text can still be highlighted.
- **accept_empty_input** (*bool, optional*) – Treat empty input as a zero value.
- **display_empty_value** (*bool, optional*) – If True, display an empty input if the value is zero. Generally used together with `accept_empty_input`.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_input_int(**kwargs)`

Adds input for an int. +/- buttons can be activated by setting the value of `step`.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*int, optional*) –

- **min_value** (*int, optional*) – Value for lower limit of input. By default this limits the step buttons. Use `min_clamped` to limit manual input.
- **max_value** (*int, optional*) – Value for upper limit of input. By default this limits the step buttons. Use `max_clamped` to limit manual input.
- **step** (*int, optional*) – Increment to change value by when the step buttons are pressed. Setting this and `step_fast` to a value of 0 or less will turn off step buttons.
- **step_fast** (*int, optional*) – Increment to change value by when `ctrl + step` buttons are pressed. Setting this and `step` to a value of 0 or less will turn off step buttons.
- **min_clamped** (*bool, optional*) – Activates and deactivates the enforcement of `min_value`.
- **max_clamped** (*bool, optional*) – Activates and deactivates the enforcement of `max_value`.
- **on_enter** (*bool, optional*) – Only runs callback on enter key press.
- **readonly** (*bool, optional*) – Activates read only mode where no text can be input but text can still be highlighted.
- **accept_empty_input** (*bool, optional*) – Treat empty input as a zero value.
- **display_empty_value** (*bool, optional*) – If True, display an empty input if the value is zero. Generally used together with `accept_empty_input`.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_input_intx(**kwargs)`

Adds multi int input for up to 4 integer values.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.

- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*Union[List[int], Tuple[int, ...]], optional*) –
- **min_value** (*int, optional*) – Value for lower limit of input for each cell. Use `min_clamped` to turn on.
- **max_value** (*int, optional*) – Value for upper limit of input for each cell. Use `max_clamped` to turn on.
- **size** (*int, optional*) – Number of components displayed for input.
- **min_clamped** (*bool, optional*) – Activates and deactivates the enforcement of `min_value`.
- **max_clamped** (*bool, optional*) – Activates and deactivates the enforcement of `max_value`.
- **on_enter** (*bool, optional*) – Only runs callback on enter.
- **readonly** (*bool, optional*) – Activates read only mode where no text can be input but text can still be highlighted.
- **accept_empty_input** (*bool, optional*) – Treat empty input as a zero value.
- **display_empty_value** (*bool, optional*) – If True, display an empty input if the value is zero. Generally used together with `accept_empty_input`.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_input_text(**kwargs)`
Adds input for text.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)

- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*str, optional*) –
- **hint** (*str, optional*) – Displayed only when value is an empty string. Will reappear if input value is set to empty string. Will not show if default value is anything other than default empty string.
- **multiline** (*bool, optional*) – Allows for multiline text input.
- **no_spaces** (*bool, optional*) – Filter out spaces and tabs.
- **uppercase** (*bool, optional*) – Automatically make all inputs uppercase.
- **tab_input** (*bool, optional*) – Allows tabs to be input into the string value instead of changing item focus.
- **decimal** (*bool, optional*) – Only allow characters 0123456789.+*/
- **hexadecimal** (*bool, optional*) – Only allow characters 0123456789ABCDEFabcdef
- **readonly** (*bool, optional*) – Activates read only mode where no text can be input but text can still be highlighted.
- **password** (*bool, optional*) – Display all input characters as ‘*’.
- **scientific** (*bool, optional*) – Only allow characters 0123456789.+*/eE (Scientific notation input)
- **on_enter** (*bool, optional*) – Only runs callback on enter key press.
- **auto_select_all** (*bool, optional*) – Select entire text when first taking mouse focus
- **ctrl_enter_for_new_line** (*bool, optional*) – In multi-line mode, unfocus with Enter, add new line with Ctrl+Enter (default is opposite: unfocus with Ctrl+Enter, add line with Enter).
- **no_horizontal_scroll** (*bool, optional*) – Disable following the cursor horizontally
- **always_overwrite** (*bool, optional*) – Overwrite mode
- **no_undo_redo** (*bool, optional*) – Disable undo/redo.

- **escape_clears_all** (*bool, optional*) – Escape key clears content if not empty, and deactivate otherwise (contrast to default behavior of Escape to revert)
- **elide_left** (*bool, optional*) – When text doesn't fit an inactive input field, clip it on the left side and ensure the right side stays visible. Useful for path/filenames. Single-line inputs only.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_int4_value(**kwargs)`

Adds a int4 value.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **default_value** (*Union[List[int], Tuple[int, ...]], optional*) –
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_int_value(**kwargs)`

Adds a int value.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **default_value** (*int, optional*) –
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_item_activated_handler(**kwargs)`

Adds a activated handler.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.

- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_item_active_handler(**kwargs)`

Adds a active handler.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_item_clicked_handler(button=-1, **kwargs)`

Adds a clicked handler.

Parameters

- **button** (*int, optional*) – Submits callback for all mouse buttons
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_item_deactivated_after_edit_handler(**kwargs)`

Adds a deactivated after edit handler.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_item_deactivated_handler(**kwargs)`

Adds a deactivated handler.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_item_double_clicked_handler(button=-1, **kwargs)`

Adds a double click handler.

Parameters

- **button** (*int, optional*) – Submits callback for all mouse buttons
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)

- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_item_edited_handler(**kwargs)`

Adds an edited handler.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_item_focus_handler(**kwargs)`

Adds a focus handler.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **event_type** (*int, optional*) – What kind of events to track: just got focus (`mvEventType_Enter`), currently having focus (`mvEventType_On`), lost focus (`mvEventType_Leave`). Can be a combination of these flags. Defaults to `mvEventType_On`.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_item_handler_registry(**kwargs)`

Adds an item handler registry.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.

- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_item_hover_handler(**kwargs)`

Adds a hover handler.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **event_type** (*int, optional*) – What kind of events to track: mouse-in (`mvEventType_Enter`), mouse-over (`mvEventType_On`), mouse-out (`mvEventType_Leave`). Can be a combination of these flags. Defaults to mouse-over.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_item_resize_handler(**kwargs)`

Adds a resize handler.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_item_scroll_handler(**kwargs)`

Adds a scroll handler.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_item_toggled_open_handler(**kwargs)`

Adds a toggged open handler.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **two_way** (*bool, optional*) – Trigger on both ‘opened’ and ‘closed’ events, i.e. when the ‘opened’ state is toggled between the two values. If False, some containers will trigger it only on the ‘opened’ event.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_item_visible_handler(**kwargs)`

Adds a visible handler.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.

- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_key_down_handler(key=0, **kwargs)`

Adds a key down handler.

Parameters

- **key** (*int, optional*) – Submits callback for all keys
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_key_press_handler(key=0, **kwargs)`

Adds a key press handler.

Parameters

- **key** (*int, optional*) – Submits callback for all keys
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_key_release_handler(key=0, **kwargs)`

Adds a key release handler.

Parameters

- **key** (*int, optional*) – Submits callback for all keys
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_knob_float(**kwargs)`

Adds a knob that rotates based on change in x mouse position.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.

- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*float, optional*) –
- **min_value** (*float, optional*) – Applies lower limit to value.
- **max_value** (*float, optional*) – Applies upper limit to value.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_line_series(x, y, **kwargs)`

Adds a line series to a plot.

Parameters

- **x** (*Any*) –
- **y** (*Any*) –
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **show** (*bool, optional*) – Attempt to render widget.
- **segments** (*bool, optional*) – a line segment will be rendered from every two consecutive points
- **loop** (*bool, optional*) – the last and first point will be connected to form a closed loop
- **skip_nan** (*bool, optional*) – NaNs values will be skipped instead of rendered as missing data
- **no_clip** (*bool, optional*) – markers (if displayed) on the edge of a plot will not be clipped
- **shaded** (*bool, optional*) – a filled region between the line and horizontal origin will be rendered; use `add_shade_series` for more advanced cases
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_listbox(items=(), **kwargs)`

Adds a listbox. If height is not large enough to show all items a scroll bar will appear.

Parameters

- **items** (*Union[List[str], Tuple[str, ...]], optional*) – A tuple of items to be shown in the listbox. Can consist of any combination of types. All items will be displayed as strings.
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*str, optional*) – String value of the item that will be selected by default.
- **num_items** (*int, optional*) – Expands the height of the listbox to show specified number of items.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_loading_indicator(**kwargs)`

Adds a rotating animated loading symbol.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks

- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **style** (*int, optional*) – mvLoadInd_DottedCircle is rotating dots style, mvLoadInd_Ring is rotating bar style.
- **circle_count** (*int, optional*) – DottedCircle style: number of dots to show.
- **speed** (*float, optional*) – Speed with which the animation will rotate.
- **radius** (*float, optional*) – Scale factor for the loading indicator radius. The size of the indicator is determined by font size and this scale factor.
- **thickness** (*float, optional*) – Ring style: scale factor of line thickness; thickness=1 corresponds to line width being 1/8 of the ring diameter.
- **color** (*Union[List[int], Tuple[int, ...]], optional*) – Main color of the indicator. If omitted, the color for mvThemeCol_Button will be used.
- **secondary_color** (*Union[List[int], Tuple[int, ...]], optional*) – Dotted-Circle style: color of ‘inactive’ dots. If omitted, the color for mvThemeCol_ButtonHovered will be used.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

dearpygui.dearpygui.add_menu(**kwargs)
Adds a menu to an existing menu bar.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.

- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Returns Union[int, str]

`dearpygui.dearpygui.add_menu_bar(**kwargs)`

Adds a menu bar to a window.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Returns Union[int, str]

`dearpygui.dearpygui.add_menu_item(**kwargs)`

Adds a menu item to an existing menu. Menu items act similar to selectables and has a bool value. When placed in a menu the checkmark will reflect its value.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.

- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **callback** (*Callable, optional*) – Registers a callback.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*bool, optional*) – This value also controls the checkmark when shown.
- **shortcut** (*str, optional*) – Displays text on the menu item. Typically used to show a shortcut key command.
- **check** (*bool, optional*) – Displays a checkmark on the menu item when it is selected and placed in a menu.
- **id** (*Union[int, str], optional*) – (deprecated)
- **drag_callback** (*Callable, optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_mouse_click_handler(button=-1, **kwargs)`

Adds a mouse click handler.

Parameters

- **button** (*int, optional*) – Submits callback for all mouse buttons
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.

- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_mouse_double_click_handler(button=- 1, **kwargs)`

Adds a mouse double click handler.

Parameters

- **button** (*int, optional*) – Submits callback for all mouse buttons
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_mouse_down_handler(button=- 1, **kwargs)`

Adds a mouse down handler.

Parameters

- **button** (*int, optional*) – Submits callback for all mouse buttons
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_mouse_drag_handler(button=- 1, threshold=10.0, **kwargs)`

Adds a mouse drag handler.

Parameters

- **button** (*int, optional*) – Submits callback for all mouse buttons
- **threshold** (*float, optional*) – The threshold the mouse must be dragged before the callback is ran

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_mouse_move_handler(**kwargs)`

Adds a mouse move handler.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_mouse_release_handler(button=- 1, **kwargs)`

Adds a mouse release handler.

Parameters

- **button** (*int, optional*) – Submits callback for all mouse buttons
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_mouse_wheel_handler(**kwargs)`

Adds a mouse wheel handler.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_node(**kwargs)`

Adds a node to a node editor.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **draggable** (*bool, optional*) – Allow node to be draggable.
- **id** (*Union[int, str], optional*) – (deprecated)

- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Returns Union[int, str]

`dearpygui.dearpygui.add_node_attribute(**kwargs)`

Adds a node attribute to a node.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **attribute_type** (*int, optional*) – `mvNode_Attr_Input`, `mvNode_Attr_Output`, or `mvNode_Attr_Static`.
- **shape** (*int, optional*) – Pin shape.
- **category** (*str, optional*) – Category
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_node_editor(**kwargs)`

Adds a node editor.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)

- **before** (*Union[int, str]*, *optional*) – This item will be displayed before the specified item in the parent.
- **callback** (*Callable*, *optional*) – Registers a callback.
- **show** (*bool*, *optional*) – Attempt to render widget.
- **filter_key** (*str*, *optional*) – Used by filter widget.
- **tracked** (*bool*, *optional*) – Scroll tracking
- **track_offset** (*float*, *optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **delink_callback** (*Callable*, *optional*) – Callback ran when a link is detached.
- **menubar** (*bool*, *optional*) – Shows or hides the menubar.
- **minimap** (*bool*, *optional*) – Shows or hides the Minimap. New in 1.6.
- **minimap_location** (*int*, *optional*) – `mvNodeMiniMap_Location_*` constants. New in 1.6.
- **id** (*Union[int, str]*, *optional*) – (deprecated)
- **delay_search** (*bool*, *optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Returns `Union[int, str]`

`dearpygui.dearpygui.add_node_link(attr_1, attr_2, **kwargs)`

Adds a node link between 2 node attributes.

Parameters

- **attr_1** (*Union[int, str]*) –
- **attr_2** (*Union[int, str]*) –
- **label** (*str*, *optional*) – Overrides 'name' as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str]*, *optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str]*, *optional*) – Parent to add this item to. (runtime adding)
- **show** (*bool*, *optional*) – Attempt to render widget.
- **id** (*Union[int, str]*, *optional*) – (deprecated)

Returns `Union[int, str]`

`dearpygui.dearpygui.add_pie_series(x, y, radius, values, labels, **kwargs)`

Adds an pie series to a plot.

Parameters

- **x** (*float*) –
- **y** (*float*) –
- **radius** (*float*) –
- **values** (*Any*) –

- **labels** (*Union[List[str], Tuple[str, ...]]*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **show** (*bool, optional*) – Attempt to render widget.
- **format** (*str, optional*) –
- **angle** (*float, optional*) –
- **normalize** (*bool, optional*) – force normalization of pie chart values (i.e. always make a full circle if sum < 0)
- **ignore_hidden** (*bool, optional*) – ignore hidden slices when drawing the pie chart (as if they were not there)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_plot(**kwargs)`

Adds a plot which is used to hold series, and can be drawn to with draw commands. For all `_mod` parameters use `mvKey_ModX` enums, or `mvKey_ModDisabled` to disable the modifier.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **callback** (*Callable, optional*) – Registers a callback.

- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **no_title** (*bool, optional*) – the plot title will not be displayed
- **no_menus** (*bool, optional*) – the user will not be able to open context menus with right-click
- **no_box_select** (*bool, optional*) – the user will not be able to box-select with right-click drag
- **no_mouse_pos** (*bool, optional*) – the text of mouse position, in plot coordinates, will not be displayed inside of the plot
- **query** (*bool, optional*) – the user will be able to draw query rects with CTRL + right-click drag
- **query_color** (*Union[List[float], Tuple[float, ...]], optional*) – Color of the query rectangles.
- **min_query_rects** (*int, optional*) – The minimum number of query rects that can be in the plot. If there are less rects than this value, it won't be possible to delete them.
- **max_query_rects** (*int, optional*) – The maximum number of query rects that can be in the plot. If the number is reached any rect added will replace the latest one. (0 means unlimited)
- **crosshairs** (*bool, optional*) – the default mouse cursor will be replaced with a crosshair when hovered
- **equal_aspects** (*bool, optional*) – primary x and y axes will be constrained to have the same units/pixel (does not apply to auxiliary y-axes)
- **no_inputs** (*bool, optional*) – the user will not be able to interact with the plot
- **no_frame** (*bool, optional*) – the ImGui frame will not be rendered
- **use_local_time** (*bool, optional*) – axis labels will be formatted for your timezone when
- **use_ISO8601** (*bool, optional*) – dates will be formatted according to ISO 8601 where applicable (e.g. YYYY-MM-DD, YYYY-MM, -MM-DD, etc.)
- **use_24hour_clock** (*bool, optional*) – times will be formatted using a 24 hour clock
- **pan_button** (*int, optional*) – mouse button that enables panning when held
- **pan_mod** (*int, optional*) – optional modifier that must be held for panning
- **context_menu_button** (*int, optional*) – opens context menus (if enabled) when clicked
- **fit_button** (*int, optional*) – fits visible data when double clicked

- **box_select_button** (*int*, *optional*) – begins box selection when pressed and confirms selection when released
- **box_select_mod** (*int*, *optional*) – begins box selection when pressed and confirms selection when released
- **box_select_cancel_button** (*int*, *optional*) – cancels active box selection when pressed
- **query_toggle_mod** (*int*, *optional*) – when held, active box selections turn into queries
- **horizontal_mod** (*int*, *optional*) – expands active box selection/query horizontally to plot edge when held
- **vertical_mod** (*int*, *optional*) – expands active box selection/query vertically to plot edge when held
- **override_mod** (*int*, *optional*) – when held, all input is ignored; used to enable axis/plots as DND sources
- **zoom_mod** (*int*, *optional*) – optional modifier that must be held for scroll wheel zooming
- **zoom_rate** (*int*, *optional*) – zoom rate for scroll (e.g. 0.1f = 10% plot range every scroll click); make negative to invert
- **id** (*Union[int, str]*, *optional*) – (deprecated)
- **delay_search** (*bool*, *optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.
- **no_highlight** (*bool*, *optional*) – (deprecated) Removed because not supported from the backend anymore. To control the highlighting of series use the same argument in *add_plot_legend*
- **no_child** (*bool*, *optional*) – (deprecated) a child window region will not be used to capture mouse scroll (can boost performance for single ImGui window applications)
- **anti_aliased** (*bool*, *optional*) – (deprecated) This feature was deprecated in ImPlot. To enable/disable anti_aliasing use *dpg.configure_app()* with the *anti_aliasing* parameters.
- **query_button** (*int*, *optional*) – (deprecated) This refers to the old way of querying of ImPlot, now replaced with *DragRect()*
- **query_mod** (*int*, *optional*) – (deprecated) This refers to the old way of querying of ImPlot, now replaced with *DragRect()*

Returns Union[int, str]

`dearpygui.dearpygui.add_plot_annotation(**kwargs)`

Adds an annotation to a plot.

Parameters

- **label** (*str*, *optional*) – Overrides 'name' as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str]*, *optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str]*, *optional*) – Parent to add this item to. (runtime adding)

- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **show** (*bool, optional*) – Attempt to render widget.
- **default_value** (*Any, optional*) –
- **offset** (*Union[List[float], Tuple[float, ...]], optional*) –
- **color** (*Union[List[int], Tuple[int, ...]], optional*) –
- **clamped** (*bool, optional*) –
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_plot_axis(axis, **kwargs)`

Adds an axis to a plot.

Parameters

- **axis** (*int*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **no_label** (*bool, optional*) – the axis label will not be displayed
- **no_gridlines** (*bool, optional*) – no grid lines will be displayed
- **no_tick_marks** (*bool, optional*) – no tick marks will be displayed
- **no_tick_labels** (*bool, optional*) – no text labels will be displayed
- **no_initial_fit** (*bool, optional*) – axis will not be initially fit to data extents on the first rendered frame
- **no_menus** (*bool, optional*) – the user will not be able to open context menus with right-click
- **no_side_switch** (*bool, optional*) – the user will not be able to switch the axis side by dragging it
- **no_highlight** (*bool, optional*) – the axis will not have its background highlighted when hovered or held
- **opposite** (*bool, optional*) – axis ticks and labels will be rendered on the conventionally opposite side (i.e, right or top)

- **foreground_grid** (*bool, optional*) – grid lines will be displayed in the foreground (i.e. on top of data) instead of the background
- **tick_format** (*str, optional*) – Sets a custom tick label formatter
- **scale** (*int, optional*) – Sets the axis' scale. Can have only **mvPlotScale_** values
- **invert** (*bool, optional*) – the axis values will be inverted (i.e. growing from right to left)
- **auto_fit** (*bool, optional*) – axis will be auto-fitting to data extents
- **range_fit** (*bool, optional*) – axis will only fit points if the point is in the visible range of the **orthogonal** axis
- **pan_stretch** (*bool, optional*) – panning in a locked or constrained state will cause the axis to stretch if possible
- **lock_min** (*bool, optional*) – the axis minimum value will be locked when panning/zooming
- **lock_max** (*bool, optional*) – the axis maximum value will be locked when panning/zooming
- **id** (*Union[int, str], optional*) – (deprecated)
- **log_scale** (*bool, optional*) – (deprecated)Old way to set log scale in the axis. Use 'scale' argument instead.
- **time** (*bool, optional*) – (deprecated)Old way to set time scale in the axis. Use 'scale' argument instead.

Returns Union[int, str]

`dearpygui.dearpygui.add_plot_legend(**kwargs)`

Adds a plot legend to a plot.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **location** (*int, optional*) – location, `mvPlot_Location_*`
- **horizontal** (*bool, optional*) – legend entries will be displayed horizontally
- **sort** (*bool, optional*) – legend entries will be displayed in alphabetical order
- **outside** (*bool, optional*) – legend will be rendered outside of the plot area

- **no_highlight_item** (*bool, optional*) – plot items will not be highlighted when their legend entry is hovered
- **no_highlight_axis** (*bool, optional*) – axes will not be highlighted when legend entries are hovered (only relevant if x/y-axis count > 1)
- **no_menus** (*bool, optional*) – the user will not be able to open context menus with right-click
- **no_buttons** (*bool, optional*) – legend icons will not function as hide/show buttons
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_progress_bar(**kwargs)`

Adds a progress bar.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **overlay** (*str, optional*) – Overlaid text onto the bar that typically used to display the value of the progress.
- **default_value** (*float, optional*) – Normalized value to fill the bar from 0.0 to 1.0. Put a negative value to show an indeterminate progress bar.

- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_radio_button(items=(), **kwargs)`

Adds a set of radio buttons. If items keyword is empty, nothing will be shown.

Parameters

- **items** (*Union[List[str], Tuple[str, ...]], optional*) – A tuple of items to be shown as radio options. Can consist of any combination of types. All types will be shown as strings.
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*str, optional*) – Default selected radio option. Set by using the string value of the item.
- **horizontal** (*bool, optional*) – Displays the radio options horizontally.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_raw_texture(width, height, default_value, **kwargs)`

Adds a raw texture.

Parameters

- **width** (*int*) –
- **height** (*int*) –
- **default_value** (*Union[List[float], Tuple[float, ...]]*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **format** (*int, optional*) – Data format.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns *Union[int, str]*

`dearpygui.dearpygui.add_resize_handler(parent, **kwargs)`
 deprecated function

`dearpygui.dearpygui.add_same_line(**kwargs)`
 deprecated function

`dearpygui.dearpygui.add_scatter_series(x, y, **kwargs)`
 Adds a scatter series to a plot.

Parameters

- **x** (*Any*) –
- **y** (*Any*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **show** (*bool, optional*) – Attempt to render widget.
- **no_clip** (*bool, optional*) – markers on the edge of a plot will not be clipped
- **id** (*Union[int, str], optional*) – (deprecated)

Returns *Union[int, str]*

`dearpygui.dearpygui.add_selectable(**kwargs)`
 Adds a selectable. Similar to a button but can indicate its selected state.

Parameters

- **label** (*str*, *optional*) – Overrides ‘name’ as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str]*, *optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int*, *optional*) – Width of the item.
- **height** (*int*, *optional*) – Height of the item.
- **indent** (*int*, *optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str]*, *optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str]*, *optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str]*, *optional*) – Overrides ‘id’ as value storage key.
- **payload_type** (*str*, *optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **callback** (*Callable*, *optional*) – Registers a callback.
- **drag_callback** (*Callable*, *optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable*, *optional*) – Registers a drop callback for drag and drop.
- **show** (*bool*, *optional*) – Attempt to render widget.
- **enabled** (*bool*, *optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]]*, *optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str*, *optional*) – Used by filter widget.
- **tracked** (*bool*, *optional*) – Scroll tracking
- **track_offset** (*float*, *optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*bool*, *optional*) –
- **span_columns** (*bool*, *optional*) – Forces the selectable to span the width of all columns if placed in a table.
- **disable_popup_close** (*bool*, *optional*) – Disable closing a modal or popup window.
- **select_on_nav** (*bool*, *optional*) – Auto-select when moved into with keyboard navigation, unless Ctrl is held.
- **id** (*Union[int, str]*, *optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_separator(**kwargs)`

Adds a horizontal line separator. Use ‘label’ parameter to add text and `mvStyleVar_SeparatorText*` elements to style it.

Parameters

- **label** (*str*, *optional*) – Overrides ‘name’ as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str]*, *optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **indent** (*int*, *optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str]*, *optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str]*, *optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool*, *optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]]*, *optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **id** (*Union[int, str]*, *optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_series_value(**kwargs)`

Adds a plot series value.

Parameters

- **label** (*str*, *optional*) – Overrides ‘name’ as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str]*, *optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **source** (*Union[int, str]*, *optional*) – Overrides ‘id’ as value storage key.
- **default_value** (*Any*, *optional*) –
- **parent** (*Union[int, str]*, *optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str]*, *optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_shade_series(x, y1, **kwargs)`

Adds a shade series to a plot.

Parameters

- **x** (*Any*) –
- **y1** (*Any*) –
- **label** (*str*, *optional*) – Overrides ‘name’ as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).

- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **show** (*bool, optional*) – Attempt to render widget.
- **y2** (*Any, optional*) –
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_simple_plot(**kwargs)`

Adds a simple plot for visualization of a 1 dimensional set of values.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*Union[List[float], Tuple[float, ...]], optional*) –
- **overlay** (*str, optional*) – overlays text (similar to a plot title)
- **histogram** (*bool, optional*) –
- **autosize** (*bool, optional*) –

- **min_scale** (*float, optional*) –
- **max_scale** (*float, optional*) –
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_slider_double(**kwargs)`

Adds slider for a single double value. Useful when slider float is not accurate enough. Directly entry can be done with double click or CTRL+Click. Min and Max alone are a soft limit for the slider. Use clamped keyword to also apply limits to the direct entry modes.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*float, optional*) –
- **vertical** (*bool, optional*) – Sets orientation of the sidebar and slider to vertical.
- **no_input** (*bool, optional*) – Disable direct entry methods double-click or ctrl+click or Enter key allowing to input text directly into the item.

- **clamped** (*bool, optional*) – Applies the min and max limits to direct entry methods also such as double click and CTRL+Click.
- **min_value** (*float, optional*) – Applies a limit only to sliding entry only.
- **max_value** (*float, optional*) – Applies a limit only to sliding entry only.
- **format** (*str, optional*) – Determines the format the float will be displayed as use python string formatting.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_slider_doublex(**kwargs)`

Adds multi slider for up to 4 double values. Useful for when multi slide float is not accurate enough. Directly entry can be done with double click or CTRL+Click. Min and Max alone are a soft limit for the slider. Use `clamped` keyword to also apply limits to the direct entry modes.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*Any, optional*) –

- **size** (*int, optional*) – Number of doubles to be displayed.
- **no_input** (*bool, optional*) – Disable direct entry methods double-click or ctrl+click or Enter key allowing to input text directly into the item.
- **clamped** (*bool, optional*) – Applies the min and max limits to direct entry methods also such as double click and CTRL+Click.
- **min_value** (*float, optional*) – Applies a limit only to sliding entry only.
- **max_value** (*float, optional*) – Applies a limit only to sliding entry only.
- **format** (*str, optional*) – Determines the format the int will be displayed as use python string formatting.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_slider_float(**kwargs)`

Adds slider for a single float value. Directly entry can be done with double click or CTRL+Click. Min and Max alone are a soft limit for the slider. Use `clamped` keyword to also apply limits to the direct entry modes.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.

- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*float, optional*) –
- **vertical** (*bool, optional*) – Sets orientation of the sidebar and slider to vertical.
- **no_input** (*bool, optional*) – Disable direct entry methods double-click or ctrl+click or Enter key allowing to input text directly into the item.
- **clamped** (*bool, optional*) – Applies the min and max limits to direct entry methods also such as double click and CTRL+Click.
- **min_value** (*float, optional*) – Applies a limit only to sliding entry only.
- **max_value** (*float, optional*) – Applies a limit only to sliding entry only.
- **format** (*str, optional*) – Determines the format the float will be displayed as use python string formatting.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_slider_floatx(**kwargs)`

Adds multi slider for up to 4 float values. Directly entry can be done with double click or CTRL+Click. Min and Max alone are a soft limit for the slider. Use clamped keyword to also apply limits to the direct entry modes.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.

- **pos** (*Union[List[int], Tuple[int, ...]]*, *optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str*, *optional*) – Used by filter widget.
- **tracked** (*bool*, *optional*) – Scroll tracking
- **track_offset** (*float*, *optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*Union[List[float], Tuple[float, ...]]*, *optional*) –
- **size** (*int*, *optional*) – Number of floats to be displayed.
- **no_input** (*bool*, *optional*) – Disable direct entry methods double-click or ctrl+click or Enter key allowing to input text directly into the item.
- **clamped** (*bool*, *optional*) – Applies the min and max limits to direct entry methods also such as double click and CTRL+Click.
- **min_value** (*float*, *optional*) – Applies a limit only to sliding entry only.
- **max_value** (*float*, *optional*) – Applies a limit only to sliding entry only.
- **format** (*str*, *optional*) – Determines the format the int will be displayed as use python string formatting.
- **id** (*Union[int, str]*, *optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_slider_int(**kwargs)`

Adds slider for a single int value. Directly entry can be done with double click or CTRL+Click. Min and Max alone are a soft limit for the slider. Use `clamped` keyword to also apply limits to the direct entry modes.

Parameters

- **label** (*str*, *optional*) – Overrides 'name' as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str]*, *optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int*, *optional*) – Width of the item.
- **height** (*int*, *optional*) – Height of the item.
- **indent** (*int*, *optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str]*, *optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str]*, *optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str]*, *optional*) – Overrides 'id' as value storage key.
- **payload_type** (*str*, *optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **callback** (*Callable*, *optional*) – Registers a callback.
- **drag_callback** (*Callable*, *optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable*, *optional*) – Registers a drop callback for drag and drop.

- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*int, optional*) –
- **vertical** (*bool, optional*) – Sets orientation of the sidebar and slider to vertical.
- **no_input** (*bool, optional*) – Disable direct entry methods double-click or ctrl+click or Enter key allowing to input text directly into the item.
- **clamped** (*bool, optional*) – Applies the min and max limits to direct entry methods also such as double click and CTRL+Click.
- **min_value** (*int, optional*) – Applies a limit only to sliding entry only.
- **max_value** (*int, optional*) – Applies a limit only to sliding entry only.
- **format** (*str, optional*) – Determines the format the int will be displayed as use python string formatting.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_slider_intx(**kwargs)`

Adds multi slider for up to 4 int values. Directly entry can be done with double click or CTRL+Click. Min and Max alone are a soft limit for the slider. Use `clamped` keyword to also apply limits to the direct entry modes.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **callback** (*Callable, optional*) – Registers a callback.

- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*Union[List[int], Tuple[int, ...]], optional*) –
- **size** (*int, optional*) – Number of ints to be displayed.
- **no_input** (*bool, optional*) – Disable direct entry methods double-click or ctrl+click or Enter key allowing to input text directly into the item.
- **clamped** (*bool, optional*) – Applies the min and max limits to direct entry methods also such as double click and CTRL+Click.
- **min_value** (*int, optional*) – Applies a limit only to sliding entry only.
- **max_value** (*int, optional*) – Applies a limit only to sliding entry only.
- **format** (*str, optional*) – Determines the format the int will be displayed as use python string formatting.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_spacer(**kwargs)`

Adds a spacer item that can be used to help with layouts or can be used as a placeholder item.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.

- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_spacing(**kwargs)`
(deprecated function) Adds vertical spacing.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks.
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **count** (*int, optional*) – Number of spacings to add the size is dependant on the curret style.

Returns Union[int, str]

`dearpygui.dearpygui.add_stage(**kwargs)`
Adds a stage.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_staging_container(**kwargs)`
deprecated function

`dearpygui.dearpygui.add_stair_series(x, y, **kwargs)`
Adds a stair series to a plot.

Parameters

- **x** (*Any*) –
- **y** (*Any*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **show** (*bool, optional*) – Attempt to render widget.
- **pre_step** (*bool, optional*) – the y value is continued constantly to the left from every x position, i.e. the interval (x[i-1], x[i]] has the value y[i]
- **shaded** (*bool, optional*) – a filled region between the line and horizontal origin will be rendered; use `add_shade_series` for more advanced cases
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_static_texture`(*width, height, default_value, **kwargs*)
Adds a static texture.

Parameters

- **width** (*int*) –
- **height** (*int*) –
- **default_value** (*Union[List[float], Tuple[float, ...]]*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_stem_series`(*x, y, **kwargs*)
Adds a stem series to a plot.

Parameters

- **x** (*Any*) –
- **y** (*Any*) –

- **label** (*str*, *optional*) – Overrides ‘name’ as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str]*, *optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **indent** (*int*, *optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str]*, *optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str]*, *optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str]*, *optional*) – Overrides ‘id’ as value storage key.
- **show** (*bool*, *optional*) – Attempt to render widget.
- **horizontal** (*bool*, *optional*) – stems will be rendered horizontally on the current y-axis
- **id** (*Union[int, str]*, *optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_string_value(**kwargs)`

Adds a string value.

Parameters

- **label** (*str*, *optional*) – Overrides ‘name’ as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str]*, *optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **source** (*Union[int, str]*, *optional*) – Overrides ‘id’ as value storage key.
- **default_value** (*str*, *optional*) –
- **parent** (*Union[int, str]*, *optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str]*, *optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_subplots(rows, columns, **kwargs)`

Adds a collection of plots.

Parameters

- **rows** (*int*) –
- **columns** (*int*) –
- **label** (*str*, *optional*) – Overrides ‘name’ as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).

- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **row_ratios** (*Union[List[float], Tuple[float, ...]], optional*) –
- **column_ratios** (*Union[List[float], Tuple[float, ...]], optional*) –
- **no_title** (*bool, optional*) – the subplot title will not be displayed
- **no_menus** (*bool, optional*) – the user will not be able to open context menus with right-click
- **no_resize** (*bool, optional*) – resize splitters between subplot cells will not be provided
- **no_align** (*bool, optional*) – subplot edges will not be aligned vertically or horizontally
- **share_series** (*bool, optional*) – when set to True, series from all sub-plots will be shared to some extent, using a single common color set and showing them in a single legend in the subplots item. Otherwise each plot will be independent from others and will have its own legend
- **link_rows** (*bool, optional*) – link the y-axis limits of all plots in each row (does not apply auxiliary y-axes)
- **link_columns** (*bool, optional*) – link the x-axis limits of all plots in each column
- **link_all_x** (*bool, optional*) – link the x-axis limits in every plot in the subplot
- **link_all_y** (*bool, optional*) – link the y-axis limits in every plot in the subplot (does not apply to auxiliary y-axes)
- **column_major** (*bool, optional*) – subplots are added in column major order instead of the default row major order
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Returns Union[int, str]

`dearpygui.dearpygui.add_synced_tables(**kwargs)`

Links all tables that are immediate children of this container so that they share their state (mostly column sizes). Other children are rendered as is. This is an experimental feature, use with caution.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **filter_key** (*str, optional*) – Used by filter widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_tab(**kwargs)`

Adds a tab to a tab bar.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **closable** (*bool, optional*) – Creates a button on the tab that can hide the tab.
- **no_tooltip** (*bool, optional*) – Disable tooltip for the given tab.

- **order_mode** (*int, optional*) – set using a constant: `mvTabOrder_Reorderable`: allows reordering, `mvTabOrder_Fixed`: fixed ordering, `mvTabOrder_Leading`: adds tab to front, `mvTabOrder_Trailing`: adds tab to back
- **unsaved_document** (*bool, optional*) – Display a dot next to the title.
- **no_close_with_middle_click** (*bool, optional*) – Disable closing this tab (if `closable==True`) by clicking with middle mouse button.
- **no_reorder** (*bool, optional*) – Disable reordering this tab or having another tab cross over this tab.
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Returns Union[int, str]

`dearpygui.dearpygui.add_tab_bar(**kwargs)`

Adds a tab bar.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **reorderable** (*bool, optional*) – Allows for the user to change the order of the tabs.
- **tab_list_popup_button** (*bool, optional*) – Show a button to select active tab from a dropdown list.
- **no_close_with_middle_click** (*bool, optional*) – Disable closing tabs (that have `closable=True`) by clicking with middle mouse button.
- **no_scrolling_buttons** (*bool, optional*) – Disable left/right scrolling buttons when tab buttons don't fit the container width.
- **no_tooltip** (*bool, optional*) – Disable tooltips when hovering a tab with a long name.

- **draw_selected_overline** (*bool, optional*) – Draw selected overline markers over selected tab.
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Returns Union[int, str]

`dearpygui.dearpygui.add_tab_button(**kwargs)`

Adds a tab button to a tab bar.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **no_reorder** (*bool, optional*) – Disable reordering this tab or having another tab cross over this tab. Fixes the position of this tab in relation to the order of neighboring tabs at start.
- **leading** (*bool, optional*) – Enforce the tab position to the left of the tab bar (after the tab list popup button).
- **trailing** (*bool, optional*) – Enforce the tab position to the right of the tab bar (before the scrolling buttons).
- **no_tooltip** (*bool, optional*) – Disable tooltip for the given tab.
- **unsaved_document** (*bool, optional*) – Display a dot next to the title.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_table(**kwargs)`

Adds a table.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **header_row** (*bool, optional*) – show headers at the top of the columns
- **clipper** (*bool, optional*) – Use clipper (rows must be same height).
- **inner_width** (*int, optional*) –
- **policy** (*int, optional*) –
- **freeze_rows** (*int, optional*) –
- **freeze_columns** (*int, optional*) –
- **sort_multi** (*bool, optional*) – Hold shift when clicking headers to sort on multiple column.
- **sort_tristate** (*bool, optional*) – Allow no sorting, disable default sorting.
- **resizable** (*bool, optional*) – Enable resizing columns
- **reorderable** (*bool, optional*) – Enable reordering columns in header row (need calling `TableSetupColumn()` + `TableHeadersRow()` to display headers)
- **hideable** (*bool, optional*) – Enable hiding/disabling columns in context menu.
- **sortable** (*bool, optional*) – Enable sorting. Call `TableGetSortSpecs()` to obtain sort specs. Also see `ImGuiTableFlags_SortMulti` and `ImGuiTableFlags_SortTristate`.
- **context_menu_in_body** (*bool, optional*) – Right-click on columns body/contents will display table context menu. By default it is available in `TableHeadersRow()`.

- **row_background** (*bool, optional*) – Set each RowBg color with `ImGuiCol_TableRowBg` or `ImGuiCol_TableRowBgAlt` (equivalent of calling `TableSetBgColor` with `ImGuiTableBgFlags_RowBg0` on each row manually)
- **borders_innerH** (*bool, optional*) – Draw horizontal borders between rows.
- **borders_outerH** (*bool, optional*) – Draw horizontal borders at the top and bottom.
- **borders_innerV** (*bool, optional*) – Draw vertical borders between columns.
- **borders_outerV** (*bool, optional*) – Draw vertical borders on the left and right sides.
- **no_host_extendX** (*bool, optional*) – Make outer width auto-fit to columns, overriding `outer_size.x` value. Only available when `ScrollX/ScrollY` are disabled and `Stretch` columns are not used.
- **no_host_extendY** (*bool, optional*) – Make outer height stop exactly at `outer_size.y` (prevent auto-extending table past the limit). Only available when `ScrollX/ScrollY` are disabled. Data below the limit will be clipped and not visible.
- **no_keep_columns_visible** (*bool, optional*) – Disable keeping column always minimally visible when `ScrollX` is off and table gets too small. Not recommended if columns are resizable.
- **precise_widths** (*bool, optional*) – Disable distributing remainder width to stretched columns (width allocation on a 100-wide table with 3 columns: Without this flag: 33,33,34. With this flag: 33,33,33). With larger number of columns, resizing will appear to be less smooth.
- **no_clip** (*bool, optional*) – Disable clipping rectangle for every individual columns.
- **pad_outerX** (*bool, optional*) – Default if `BordersOuterV` is on. Enable outer-most padding. Generally desirable if you have headers.
- **no_pad_outerX** (*bool, optional*) – Default if `BordersOuterV` is off. Disable outer-most padding.
- **no_pad_innerX** (*bool, optional*) – Disable inner padding between columns (double inner padding if `BordersOuterV` is on, single inner padding if `BordersOuterV` is off).
- **scrollX** (*bool, optional*) – Enable horizontal scrolling. Require ‘outer_size’ parameter of `BeginTable()` to specify the container size. Changes default sizing policy. Because this create a child window, `ScrollY` is currently generally recommended when using `ScrollX`.
- **scrollY** (*bool, optional*) – Enable vertical scrolling.
- **no_saved_settings** (*bool, optional*) – Never load/save settings in .ini file.
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Returns `Union[int, str]`

`dearpygui.dearpygui.add_table_cell(**kwargs)`

Adds a table.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks

- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **height** (*int, optional*) – Height of the item.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **filter_key** (*str, optional*) – Used by filter widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_table_column(**kwargs)`

Adds a table column.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **enabled** (*bool, optional*) – Turns off functionality of widget and applies the disabled theme.
- **init_width_or_weight** (*float, optional*) –
- **default_hide** (*bool, optional*) – Default as a hidden/disabled column.
- **default_sort** (*bool, optional*) – Default as a sorting column.
- **width_stretch** (*bool, optional*) – Column will stretch. Preferable with horizontal scrolling disabled (default if table sizing policy is `_SizingStretchSame` or `_SizingStretchProp`).
- **width_fixed** (*bool, optional*) – Column will not stretch. Preferable with horizontal scrolling enabled (default if table sizing policy is `_SizingFixedFit` and table is resizable).
- **no_resize** (*bool, optional*) – Disable manual resizing.
- **no_reorder** (*bool, optional*) – Disable manual reordering this column, this will also prevent other columns from crossing over this column.
- **no_hide** (*bool, optional*) – Disable ability to hide/disable this column.

- **no_clip** (*bool, optional*) – Disable clipping for this column (all NoClip columns will render in a same draw command).
- **no_sort** (*bool, optional*) – Disable ability to sort on this field (even if ImGuiTableFlags_Sortable is set on the table).
- **no_sort_ascending** (*bool, optional*) – Disable ability to sort in the ascending direction.
- **no_sort_descending** (*bool, optional*) – Disable ability to sort in the descending direction.
- **no_header_width** (*bool, optional*) – Disable header text width contribution to automatic column width.
- **prefer_sort_ascending** (*bool, optional*) – Make the initial sort direction Ascending when first sorting on this column (default).
- **prefer_sort_descending** (*bool, optional*) – Make the initial sort direction Descending when first sorting on this column.
- **indent_enable** (*bool, optional*) – Use current Indent value when entering cell (default for column 0).
- **indent_disable** (*bool, optional*) – Ignore current Indent value when entering cell (default for columns > 0). Indentation changes `_within_` the cell will still be honored.
- **angled_header** (*bool, optional*) – Set this parameter to True to display the header text for this column in an angled (diagonal) orientation. This will add an additional row to accommodate the angled text.
- **no_header_label** (*bool, optional*) – Disable horizontal label for this column. Name will still appear in context menu or in angled headers.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_table_next_column(**kwargs)`
 deprecated function

`dearpygui.dearpygui.add_table_row(**kwargs)`
 Adds a table row.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **height** (*int, optional*) – Height of the item.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **filter_key** (*str, optional*) – Used by filter widget.

- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_template_registry(**kwargs)`

Adds a template registry.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_text(default_value="", **kwargs)`

Adds text. Text can have an optional label that will display to the right of the text.

Parameters

- **default_value** (*str, optional*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom

- **wrap** (*int*, *optional*) – Number of pixels from the start of the item until wrapping starts.
- **bullet** (*bool*, *optional*) – Places a bullet to the left of the text.
- **color** (*Union[List[int], Tuple[int, ...]]*, *optional*) – Color of the text (rgba).
- **show_label** (*bool*, *optional*) – Displays the label to the right of the text.
- **id** (*Union[int, str]*, *optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_text_point(x, y, **kwargs)`

Adds a label series to a plot. x and y can only have one elements each.

Parameters

- **x** (*float*) –
- **y** (*float*) –
- **label** (*str*, *optional*) – Overrides 'name' as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str]*, *optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str]*, *optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str]*, *optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str]*, *optional*) – Overrides 'id' as value storage key.
- **show** (*bool*, *optional*) – Attempt to render widget.
- **offset** (*Union[List[float], Tuple[float, ...]]*, *optional*) – Offset of the label, in pixels, relative to the coordinates.
- **vertical** (*bool*, *optional*) –
- **id** (*Union[int, str]*, *optional*) – (deprecated)
- **x_offset** (*int*, *optional*) – (deprecated) Old way to set x offset of the label. Use *offset* argument instead.
- **y_offset** (*int*, *optional*) – (deprecated) Old way to set y offset of the label. Use *offset* argument instead.

Returns Union[int, str]

`dearpygui.dearpygui.add_texture_registry(**kwargs)`

Adds a dynamic texture.

Parameters

- **label** (*str*, *optional*) – Overrides 'name' as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).

- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_theme(**kwargs)`

Adds a theme.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **id** (*Union[int, str], optional*) – (deprecated)
- **default_theme** (*bool, optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_theme_color(target=0, value=(0, 0, 0, 255), **kwargs)`

Adds a theme color.

Parameters

- **target** (*int, optional*) –
- **value** (*Union[List[int], Tuple[int, ...]], optional*) –
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **category** (*int, optional*) – Options include mvThemeCat_Core, mvThemeCat_Plots, mvThemeCat_Nodes.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_theme_component(item_type=0, **kwargs)`

Adds a theme component.

Parameters

- **item_type** (*int, optional*) –
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks

- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **enabled_state** (*bool, optional*) –
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_theme_style(target=0, x=1.0, y=-1.0, **kwargs)`

Adds a theme style.

Parameters

- **target** (*int, optional*) –
- **x** (*float, optional*) –
- **y** (*float, optional*) –
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **category** (*int, optional*) – Options include `mvThemeCat_Core`, `mvThemeCat_Plots`, `mvThemeCat_Nodes`.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_time_picker(**kwargs)`

Adds a time picker.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)

- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_value** (*dict, optional*) –
- **hour24** (*bool, optional*) – Show 24 hour clock instead of 12 hour.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_toggled_open_handler(parent, **kwargs)`
 deprecated function

`dearpygui.dearpygui.add_tooltip(parent, **kwargs)`
 Adds a tooltip window.

Parameters

- **parent** (*Union[int, str]*) –
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **show** (*bool, optional*) – Attempt to render widget.
- **delay** (*float, optional*) – Activation delay: time, in seconds, during which the mouse should stay still in order to display the tooltip. May be zero for instant activation.
- **hide_on_activity** (*bool, optional*) – Hide the tooltip if the user has moved the mouse. If False, the tooltip will follow mouse pointer.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_tree_node(**kwargs)`
 Adds a tree node to add items to.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_open** (*bool, optional*) – Sets the tree node open by default.
- **open_on_double_click** (*bool, optional*) – Need double-click to open node.
- **open_on_arrow** (*bool, optional*) – Only open when clicking on the arrow part.
- **leaf** (*bool, optional*) – No collapsing, no arrow (use as a convenience for leaf nodes).
- **bullet** (*bool, optional*) – Display a bullet instead of arrow.
- **selectable** (*bool, optional*) – Makes the tree selectable.
- **span_text_width** (*bool, optional*) – Makes hitbox and highlight only cover the label.
- **span_full_width** (*bool, optional*) – Extend hit box to the left-most and right-most edges (cover the indent area).
- **catch_nav_left** (*bool, optional*) – Keyboard navigation: left arrow within this node’s children, if unhandled, moves focus to this node. When setting it to True on a node, better set it on all children nodes in the subtree as well, otherwise it might give unexpected navigation jumps.
- **lines** (*int, optional*) – Experimental. Draw lines connecting tree_node hierarchy. One of dpg.mvTreeLines constants. To work correctly, must be configured the same way in every tree node.
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Returns Union[int, str]

`dearpygui.dearpygui.add_value_registry(**kwargs)`

Adds a value registry.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_viewport_drawlist(**kwargs)`

A container that is used to present draw items or layers directly to the viewport. By default this will draw to the back of the viewport. Layers and draw items should be added to this widget as children.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **show** (*bool, optional*) – Attempt to render widget.
- **filter_key** (*str, optional*) – Used by filter widget.
- **front** (*bool, optional*) – Draws to the front of the view port instead of the back.
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated)This was used as an optimization hint but is not relevant anymore.

Returns Union[int, str]

`dearpygui.dearpygui.add_viewport_menu_bar(**kwargs)`

Adds a menubar to the viewport.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.

- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Returns Union[int, str]

`dearpygui.dearpygui.add_visible_handler(parent, **kwargs)`
deprecated function

`dearpygui.dearpygui.add_vline_series(x, **kwargs)`
(deprecated function) Adds an infinite vertical line series to a plot.

Parameters

- **x** (*Any*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides ‘id’ as value storage key.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.add_window(**kwargs)`
Creates a new window for following items to be added to.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **show** (*bool, optional*) – Attempt to render widget.

- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **min_size** (*Union[List[int], Tuple[int, ...]], optional*) – Minimum window size.
- **max_size** (*Union[List[int], Tuple[int, ...]], optional*) – Maximum window size.
- **menubar** (*bool, optional*) – Shows or hides the menubar.
- **collapsed** (*bool, optional*) – Collapse the window.
- **autosize** (*bool, optional*) – Autosized the window to fit it's items.
- **no_resize** (*bool, optional*) – Allows for the window size to be changed or fixed.
- **unsaved_document** (*bool, optional*) – Show a special marker if the document is not saved.
- **no_title_bar** (*bool, optional*) – Title name for the title bar of the window.
- **no_move** (*bool, optional*) – Allows for the window's position to be changed or fixed.
- **no_scrollbar** (*bool, optional*) – Disable scrollbars. (window can still scroll with mouse or programmatically)
- **no_collapse** (*bool, optional*) – Disable user collapsing window by double-clicking on it.
- **horizontal_scrollbar** (*bool, optional*) – Allow horizontal scrollbar to appear. (off by default)
- **no_focus_on_appearing** (*bool, optional*) – Disable taking focus when transitioning from hidden to visible state.
- **no_bring_to_front_on_focus** (*bool, optional*) – Disable bringing window to front when taking focus. (e.g. clicking on it or programmatically giving it focus)
- **no_close** (*bool, optional*) – Disable user closing the window by removing the close button.
- **no_background** (*bool, optional*) – Sets Background and border alpha to transparent.
- **modal** (*bool, optional*) – Fills area behind window according to the theme and disables user ability to interact with anything except the window.
- **popup** (*bool, optional*) – Fills area behind window according to the theme, removes title bar, collapse and close. Window can be closed by selecting area in the background behind the window.
- **no_saved_settings** (*bool, optional*) – Never load/save settings in .ini file.
- **no_open_over_existing_popup** (*bool, optional*) – Don't open if there's already a popup
- **no_scroll_with_mouse** (*bool, optional*) – Disable user vertically scrolling with mouse wheel.
- **no_docking** (*bool, optional*) – Disable docking of this window
- **copy_contents_shortcut** (*bool, optional*) – Experimental. If True, window contents can be copied to clipboard by pressing Ctrl+C. Might be useful for message boxes.
- **on_close** (*Callable, optional*) – Callback ran when window is closed.

- **id** (*Union[int, str]*, *optional*) – (deprecated)
- **delay_search** (*bool*, *optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Returns Union[int, str]

`dearpygui.dearpygui.apply_transform(item, transform)`

New in 1.1. Applies a transformation matrix to a layer.

Parameters

- **item** (*Union[int, str]*) – Drawing node to apply transform to.
- **transform** (*Any*) – Transformation matrix.

Returns None

`dearpygui.dearpygui.bind_colormap(item, source)`

Sets the color map for widgets that accept it.

Parameters

- **item** (*Union[int, str]*) – item that the color map will be applied to
- **source** (*Union[int, str]*) – The colormap tag. This should come from a colormap that was added to a colormap registry. Built in color maps are accessible through their corresponding constants `mvPlotColormap_Twilight`, `mvPlotColormap_***`

Returns None

`dearpygui.dearpygui.bind_font(font)`

Binds a global font.

Parameters **font** (*Union[int, str]*) –

Returns Union[int, str]

`dearpygui.dearpygui.bind_item_font(item, font)`

Sets an item's font.

Parameters

- **item** (*Union[int, str]*) –
- **font** (*Union[int, str]*) –

Returns None

`dearpygui.dearpygui.bind_item_handler_registry(item, handler_registry)`

Binds an item handler registry to an item.

Parameters

- **item** (*Union[int, str]*) –
- **handler_registry** (*Union[int, str]*) –

Returns None

`dearpygui.dearpygui.bind_item_theme(item, theme)`

Binds a theme to an item.

Parameters

- **item** (*Union[int, str]*) –
- **theme** (*Union[int, str]*) –

Returns None

`dearpygui.dearpygui.bind_theme(theme)`

Binds a global theme.

Parameters `theme` (*Union[int, str]*) –

Returns None

`dearpygui.dearpygui.capture_next_item(callback, **kwargs)`

Captures the next item.

Parameters

- **callback** (*Callable*) –
- **user_data** (*Any, optional*) – New in 1.3. Optional user data to send to the callback

Returns None

`dearpygui.dearpygui.child(**kwargs)`

(deprecated function) Adds an embedded child window. Will show scrollbars when items do not fit.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **delay_search** (*bool, optional*) – Delays searching container for specified items until the end of the app. Possible optimization when a container has many children that are not accessed often.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **border** (*bool, optional*) – Shows/Hides the border around the sides.

- **autosize_x** (*bool, optional*) – Autosize the window to its parents size in x.
- **autosize_y** (*bool, optional*) – Autosize the window to its parents size in y.
- **no_scrollbar** (*bool, optional*) – Disable scrollbars (window can still scroll with mouse or programmatically).
- **horizontal_scrollbar** (*bool, optional*) – Allow horizontal scrollbar to appear (off by default).
- **menubar** (*bool, optional*) – Shows/Hides the menubar at the top.

Yields Union[int, str]

`dearpygui.dearpygui.child_window(**kwargs)`

Adds an embedded child window. Will show scrollbars when items do not fit. About using `auto_resize/resizable` flags: size measurement for a given axis is only performed when the child window is within visible boundaries, or is just appearing and it won't update its auto-size while clipped. While not perfect, it is a better default behavior as the always-on performance gain is more valuable than the occasional 'resizing after becoming visible again' glitch. You may also use `always_auto_resize` to force an update even when child window is not in view. However doing so will degrade performance. Remember that combining both `auto_resize_x` and `auto_resize_y` defeats purpose of a scrolling region and is NOT recommended.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **border** (*bool, optional*) – Shows/Hides the border around the sides.
- **autosize_x** (*bool, optional*) – Autosize the window to its parents size in x.

- **autosize_y** (*bool, optional*) – Autosize the window to its parents size in y.
- **no_scrollbar** (*bool, optional*) – Disable scrollbars (window can still scroll with mouse or programmatically).
- **horizontal_scrollbar** (*bool, optional*) – Allow horizontal scrollbar to appear (off by default).
- **menubar** (*bool, optional*) – Shows/Hides the menubar at the top.
- **no_scroll_with_mouse** (*bool, optional*) – Disable user vertically scrolling with mouse wheel.
- **flattened_navigation** (*bool, optional*) – Allow gamepad/keyboard navigation to cross over parent border to this child (only use on child that have no scrolling!)
- **always_use_window_padding** (*bool, optional*) – Pad with `style.WindowPadding` even if no border are drawn (no padding by default for non-bordered child windows because it makes more sense)
- **resizable_x** (*bool, optional*) – Allow resize from right border (layout direction). Enable `.ini` saving.
- **resizable_y** (*bool, optional*) – Allow resize from bottom border (layout direction).
- **always_auto_resize** (*bool, optional*) – Combined with `auto_resize_x/auto_resize_y`. Always measure size even when child is hidden and always disable clipping optimization! NOT RECOMMENDED.
- **frame_style** (*bool, optional*) – Style the child window like a framed item: use `FrameBg`, `FrameRounding`, `FrameBorderSize`, `FramePadding` instead of `ChildBg`, `ChildRounding`, `ChildBorderSize`, `WindowPadding`.
- **auto_resize_x** (*bool, optional*) – Enable auto-resizing width based on child content. Read ‘IMPORTANT: Size measurement’ details above.
- **auto_resize_y** (*bool, optional*) – Enable auto-resizing height based on child content. Read ‘IMPORTANT: Size measurement’ details above.
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Yields `Union[int, str]`

`dearpygui.dearpygui.cleanup_dearpygui()`
 deprecated function

`dearpygui.dearpygui.clear_selected_links(node_editor)`
 Clears a node editor’s selected links.

Parameters `node_editor` (*Union[int, str]*) –

Returns `None`

`dearpygui.dearpygui.clear_selected_nodes(node_editor)`
 Clears a node editor’s selected nodes.

Parameters `node_editor` (*Union[int, str]*) –

Returns `None`

`dearpygui.dearpygui.clipper(**kwargs)`

Helper to manually clip large list of items. Increases performance by not searching or drawing widgets outside of the clipped region.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated)This was used as an optimization hint but is not relevant anymore.

Yields `Union[int, str]`

`dearpygui.dearpygui.collapsing_header(**kwargs)`

Adds a collapsing header to add items to. Must be closed with the end command.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.

- **pos** (*Union[List[int], Tuple[int, ...]]*, *optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str*, *optional*) – Used by filter widget.
- **tracked** (*bool*, *optional*) – Scroll tracking
- **track_offset** (*float*, *optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **closable** (*bool*, *optional*) – Adds the ability to hide this widget by pressing the (x) in the top right of widget.
- **default_open** (*bool*, *optional*) – Sets the collapseable header open by default.
- **open_on_double_click** (*bool*, *optional*) – Need double-click to open node.
- **open_on_arrow** (*bool*, *optional*) – Only open when clicking on the arrow part.
- **leaf** (*bool*, *optional*) – No collapsing, no arrow (use as a convenience for leaf nodes).
- **bullet** (*bool*, *optional*) – Display a bullet instead of arrow.
- **id** (*Union[int, str]*, *optional*) – (deprecated)
- **delay_search** (*bool*, *optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Yields Union[int, str]

`dearpygui.dearpygui.colormap_registry(**kwargs)`

Adds a colormap registry.

Parameters

- **label** (*str*, *optional*) – Overrides ‘name’ as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str]*, *optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **show** (*bool*, *optional*) – Attempt to render widget.
- **id** (*Union[int, str]*, *optional*) – (deprecated)

Yields Union[int, str]

`dearpygui.dearpygui.configure_app(**kwargs) → None`

Configures an item after creation.

`dearpygui.dearpygui.configure_item(item: Union[int, str], **kwargs) → None`

Configures an item after creation.

`dearpygui.dearpygui.configure_viewport(item: Union[int, str], **kwargs) → None`

Configures a viewport after creation.

`dearpygui.dearpygui.create_context()`

Creates the Dear PyGui context.

Args: :returns: None

`dearpygui.dearpygui.create_fps_matrix(eye, pitch, yaw)`

New in 1.1. Create a ‘first person shooter’ matrix.

Parameters

- **eye** (*Union[List[float], Tuple[float, ...]]*) – eye position
- **pitch** (*float*) – pitch (in radians)
- **yaw** (*float*) – yaw (in radians)

Returns Any

`dearpygui.dearpygui.create_lookat_matrix(eye, target, up)`

New in 1.1. Creates a ‘Look at matrix’.

Parameters

- **eye** (*Union[List[float], Tuple[float, ...]]*) – eye position
- **target** (*Union[List[float], Tuple[float, ...]]*) – target position
- **up** (*Union[List[float], Tuple[float, ...]]*) – up vector

Returns Any

`dearpygui.dearpygui.create_orthographic_matrix(left, right, bottom, top, zNear, zFar)`

New in 1.1. Creates an orthographic matrix.

Parameters

- **left** (*float*) – left plane
- **right** (*float*) – right plane
- **bottom** (*float*) – bottom plane
- **top** (*float*) – top plane
- **zNear** (*float*) – Near clipping plane.
- **zFar** (*float*) – Far clipping plane.

Returns Any

`dearpygui.dearpygui.create_perspective_matrix(fov, aspect, zNear, zFar)`

New in 1.1. Creates a perspective matrix.

Parameters

- **fov** (*float*) – Field of view (in radians)
- **aspect** (*float*) – Aspect ratio (width/height)
- **zNear** (*float*) – Near clipping plane.
- **zFar** (*float*) – Far clipping plane.

Returns Any

`dearpygui.dearpygui.create_rotation_matrix(angle, axis)`

New in 1.1. Applies a transformation matrix to a layer.

Parameters

- **angle** (*float*) – angle to rotate
- **axis** (*Union[List[float], Tuple[float, ...]]*) – axis to rotate around

Returns Any

`dearpygui.dearpygui.create_scale_matrix(scales)`

New in 1.1. Applies a transformation matrix to a layer.

Parameters **scales** (*Union[List[float], Tuple[float, ...]]*) – scale values per axis

Returns Any

`dearpygui.dearpygui.create_translation_matrix(translation)`

New in 1.1. Creates a translation matrix.

Parameters `translation` (`Union[List[float], Tuple[float, ...]]`) – translation vector

Returns Any

`dearpygui.dearpygui.create_viewport(**kwargs)`

Creates a viewport. Viewports are required.

Parameters

- **title** (`str`, *optional*) – Sets the title of the viewport.
- **small_icon** (`str`, *optional*) – Sets the small icon that is found in the viewport’s decorator bar. Must be ***.ico on windows and either *.ico or ***.png on mac**.
- **large_icon** (`str`, *optional*) – Sets the large icon that is found in the task bar while the app is running. Must be ***.ico on windows and either *.ico or ***.png on mac**.
- **width** (`int`, *optional*) – Sets the width of the drawable space on the viewport. Does not include the border.
- **height** (`int`, *optional*) – Sets the height of the drawable space on the viewport. Does not include the border or decorator bar.
- **x_pos** (`int`, *optional*) – Sets x position the viewport will be drawn in screen coordinates.
- **y_pos** (`int`, *optional*) – Sets y position the viewport will be drawn in screen coordinates.
- **min_width** (`int`, *optional*) – Applies a minimum limit to the width of the viewport.
- **max_width** (`int`, *optional*) – Applies a maximum limit to the width of the viewport.
- **min_height** (`int`, *optional*) – Applies a minimum limit to the height of the viewport.
- **max_height** (`int`, *optional*) – Applies a maximum limit to the height of the viewport.
- **resizable** (`bool`, *optional*) – Enables and Disables user ability to resize the viewport.
- **vsync** (`bool`, *optional*) – Enables and Disables the renderloop vsync limit. vsync frame value is set by refresh rate of display.
- **always_on_top** (`bool`, *optional*) – Forces the viewport to always be drawn on top of all other viewports.
- **decorated** (`bool`, *optional*) – Enabled and disabled the decorator bar at the top of the viewport.
- **clear_color** (`Union[List[float], Tuple[float, ...]]`, *optional*) – Sets the color of the back of the viewport.
- **disable_close** (`bool`, *optional*) – Disables the viewport close button. can be used with `set_exit_callback`

Returns None

`dearpygui.dearpygui.custom_series(x, y, channel_count, **kwargs)`

Adds a custom series to a plot. New in 1.6.

Parameters

- **x** (*Any*) –
- **y** (*Any*) –

- **channel_count** (*int*) –
- **label** (*str*, *optional*) – Overrides ‘name’ as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str]*, *optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str]*, *optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str]*, *optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str]*, *optional*) – Overrides ‘id’ as value storage key.
- **callback** (*Callable*, *optional*) – Registers a callback.
- **show** (*bool*, *optional*) – Attempt to render widget.
- **y1** (*Any*, *optional*) –
- **y2** (*Any*, *optional*) –
- **y3** (*Any*, *optional*) –
- **tooltip** (*bool*, *optional*) – Show tooltip when plot is hovered.
- **no_fit** (*bool*, *optional*) – the item won’t be considered for plot fits
- **id** (*Union[int, str]*, *optional*) – (deprecated)

Yields *Union[int, str]*

`dearpygui.dearpygui.delete_item(item, **kwargs)`
Deletes an item..

Parameters

- **item** (*Union[int, str]*) –
- **children_only** (*bool*, *optional*) –
- **slot** (*int*, *optional*) –

Returns *None*

`dearpygui.dearpygui.deprecated(reason)`

`dearpygui.dearpygui.destroy_context()`
Destroys the Dear PyGui context.

Args: :returns: *None*

`dearpygui.dearpygui.disable_item(item: Union[int, str])`
Disables the item.

Parameters ****item** – Item to disable.

Returns *None*

`dearpygui.dearpygui.does_alias_exist(alias)`
Checks if an alias exist.

Parameters **alias** (*str*) –

Returns bool

`dearpygui.dearpygui.does_item_exist(item)`

Checks if an item exist..

Parameters `item` (*Union[int, str]*) –

Returns bool

`dearpygui.dearpygui.drag_payload(**kwargs)`

User data payload for drag and drop operations.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **show** (*bool, optional*) – Attempt to render widget.
- **drag_data** (*Any, optional*) – Drag data
- **drop_data** (*Any, optional*) – Drop data
- **payload_type** (*str, optional*) –
- **id** (*Union[int, str], optional*) – (deprecated)

Yields *Union[int, str]*

`dearpygui.dearpygui.draw_arrow(p1, p2, **kwargs)`

Adds an arrow.

Parameters

- **p1** (*Union[List[float], Tuple[float, ...]]*) – Arrow tip.
- **p2** (*Union[List[float], Tuple[float, ...]]*) – Arrow tail.
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **color** (*Union[List[int], Tuple[int, ...]], optional*) –
- **thickness** (*float, optional*) –
- **size** (*int, optional*) –

- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.draw_bezier_cubic(p1, p2, p3, p4, **kwargs)`

Adds a cubic bezier curve.

Parameters

- **p1** (*Union[List[float], Tuple[float, ...]]*) – First point in curve.
- **p2** (*Union[List[float], Tuple[float, ...]]*) – Second point in curve.
- **p3** (*Union[List[float], Tuple[float, ...]]*) – Third point in curve.
- **p4** (*Union[List[float], Tuple[float, ...]]*) – Fourth point in curve.
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **color** (*Union[List[int], Tuple[int, ...]], optional*) –
- **thickness** (*float, optional*) –
- **segments** (*int, optional*) – Number of segments to approximate bezier curve.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.draw_bezier_quadratic(p1, p2, p3, **kwargs)`

Adds a quadratic bezier curve.

Parameters

- **p1** (*Union[List[float], Tuple[float, ...]]*) – First point in curve.
- **p2** (*Union[List[float], Tuple[float, ...]]*) – Second point in curve.
- **p3** (*Union[List[float], Tuple[float, ...]]*) – Third point in curve.
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.

- **show** (*bool, optional*) – Attempt to render widget.
- **color** (*Union[List[int], Tuple[int, ...]], optional*) –
- **thickness** (*float, optional*) –
- **segments** (*int, optional*) – Number of segments to approximate bezier curve.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.draw_circle(center, radius, **kwargs)`

Adds a circle

Parameters

- **center** (*Union[List[float], Tuple[float, ...]]*) –
- **radius** (*float*) –
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **color** (*Union[List[int], Tuple[int, ...]], optional*) –
- **fill** (*Union[List[int], Tuple[int, ...]], optional*) –
- **thickness** (*float, optional*) –
- **segments** (*int, optional*) – Number of segments to approximate circle.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.draw_ellipse(pmin, pmax, **kwargs)`

Adds an ellipse.

Parameters

- **pmin** (*Union[List[float], Tuple[float, ...]]*) – Min point of bounding rectangle.
- **pmax** (*Union[List[float], Tuple[float, ...]]*) – Max point of bounding rectangle.
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.

- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **color** (*Union[List[int], Tuple[int, ...]], optional*) –
- **fill** (*Union[List[int], Tuple[int, ...]], optional*) –
- **thickness** (*float, optional*) –
- **segments** (*int, optional*) – Number of segments to approximate bezier curve.
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.draw_image(texture_tag, pmin, pmax, **kwargs)`

Adds an image (for a drawing).

Parameters

- **texture_tag** (*Union[int, str]*) –
- **pmin** (*Union[List[float], Tuple[float, ...]]*) – Point of to start drawing texture.
- **pmax** (*Union[List[float], Tuple[float, ...]]*) – Point to complete drawing texture.
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **uv_min** (*Union[List[float], Tuple[float, ...]], optional*) – Normalized coordinates on texture that will be drawn.
- **uv_max** (*Union[List[float], Tuple[float, ...]], optional*) – Normalized coordinates on texture that will be drawn.
- **color** (*Union[List[int], Tuple[int, ...]], optional*) –
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.draw_image_quad(texture_tag, p1, p2, p3, p4, **kwargs)`

Adds an image (for a drawing).

Parameters

- **texture_tag** (*Union[int, str]*) –
- **p1** (*Union[List[float], Tuple[float, ...]]*) –

- **p2** (*Union[List[float], Tuple[float, ...]]*) –
- **p3** (*Union[List[float], Tuple[float, ...]]*) –
- **p4** (*Union[List[float], Tuple[float, ...]]*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **uv1** (*Union[List[float], Tuple[float, ...]], optional*) – Normalized coordinates on texture that will be drawn.
- **uv2** (*Union[List[float], Tuple[float, ...]], optional*) – Normalized coordinates on texture that will be drawn.
- **uv3** (*Union[List[float], Tuple[float, ...]], optional*) – Normalized coordinates on texture that will be drawn.
- **uv4** (*Union[List[float], Tuple[float, ...]], optional*) – Normalized coordinates on texture that will be drawn.
- **color** (*Union[List[int], Tuple[int, ...]], optional*) –
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

dearpygui.dearpygui.**draw_layer**(**kwargs)

New in 1.1. Creates a layer useful for grouping drawlist items.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **perspective_divide** (*bool, optional*) – New in 1.1. apply perspective divide
- **depth_clipping** (*bool, optional*) – New in 1.1. apply depth clipping

- **cull_mode** (*int, optional*) – New in 1.1. culling mode, `mvCullMode_*` constants. Only works with triangles currently.
- **id** (*Union[int, str], optional*) – (deprecated)

Yields Union[int, str]

`dearpygui.dearpygui.draw_line(p1, p2, **kwargs)`

Adds a line.

Parameters

- **p1** (*Union[List[float], Tuple[float, ...]]*) – Start of line.
- **p2** (*Union[List[float], Tuple[float, ...]]*) – End of line.
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **color** (*Union[List[int], Tuple[int, ...]], optional*) –
- **thickness** (*float, optional*) –
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.draw_node(**kwargs)`

New in 1.1. Creates a drawing node to associate a transformation matrix. Child node matrices will concatenate.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Yields Union[int, str]

`dearpygui.dearpygui.draw_polygon(points, **kwargs)`

Adds a polygon.

Parameters

- **points** (*List[List[float]]*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **color** (*Union[List[int], Tuple[int, ...]], optional*) –
- **fill** (*Union[List[int], Tuple[int, ...]], optional*) –
- **thickness** (*float, optional*) –
- **id** (*Union[int, str], optional*) – (deprecated)

Returns *Union[int, str]*

`dearpygui.dearpygui.draw_polyline(points, **kwargs)`

Adds a polyline.

Parameters

- **points** (*List[List[float]]*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **closed** (*bool, optional*) – Will close the polyline by returning to the first point.
- **color** (*Union[List[int], Tuple[int, ...]], optional*) –
- **thickness** (*float, optional*) –
- **id** (*Union[int, str], optional*) – (deprecated)

Returns *Union[int, str]*

`dearpygui.dearpygui.draw_quad(p1, p2, p3, p4, **kwargs)`

Adds a quad.

Parameters

- **p1** (*Union[List[float], Tuple[float, ...]]*) –
- **p2** (*Union[List[float], Tuple[float, ...]]*) –
- **p3** (*Union[List[float], Tuple[float, ...]]*) –
- **p4** (*Union[List[float], Tuple[float, ...]]*) –
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **color** (*Union[List[int], Tuple[int, ...]], optional*) –
- **fill** (*Union[List[int], Tuple[int, ...]], optional*) –
- **thickness** (*float, optional*) –
- **id** (*Union[int, str], optional*) – (deprecated)

Returns `Union[int, str]`

`dearpygui.dearpygui.draw_rectangle(pmin, pmax, **kwargs)`

Adds a rectangle.

Parameters

- **pmin** (*Union[List[float], Tuple[float, ...]]*) – Min point of bounding rectangle.
- **pmax** (*Union[List[float], Tuple[float, ...]]*) – Max point of bounding rectangle.
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **color** (*Union[List[int], Tuple[int, ...]], optional*) –

- **fill** (*Union[List[int], Tuple[int, ...]], optional*) –
- **multicolor** (*bool, optional*) –
- **rounding** (*float, optional*) – Number of pixels of the radius that will round the corners of the rectangle. Note: doesn't work with multicolor
- **thickness** (*float, optional*) –
- **corner_colors** (*Any, optional*) – Corner colors in a list, starting with upper-left and going clockwise: (upper-left, upper-right, bottom-right, bottom-left). 'multicolor' must be set to 'True'.
- **id** (*Union[int, str], optional*) – (deprecated)
- **color_upper_left** (*Union[List[int], Tuple[int, ...]], optional*) – (deprecated) Use `corner_colors` instead
- **color_upper_right** (*Union[List[int], Tuple[int, ...]], optional*) – (deprecated) Use `corner_colors` instead
- **color_bottom_right** (*Union[List[int], Tuple[int, ...]], optional*) – (deprecated) Use `corner_colors` instead
- **color_bottom_left** (*Union[List[int], Tuple[int, ...]], optional*) – (deprecated) Use `corner_colors` instead

Returns Union[int, str]

`dearpygui.dearpygui.draw_text(pos, text, **kwargs)`

Adds text (drawlist).

Parameters

- **pos** (*Union[List[float], Tuple[float, ...]]*) – Top left point of bounding text rectangle.
- **text** (*str*) – Text to draw.
- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **color** (*Union[List[int], Tuple[int, ...]], optional*) –
- **size** (*float, optional*) –
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.draw_triangle(p1, p2, p3, **kwargs)`

Adds a triangle.

Parameters

- **p1** (*Union[List[float], Tuple[float, ...]]*) –
- **p2** (*Union[List[float], Tuple[float, ...]]*) –
- **p3** (*Union[List[float], Tuple[float, ...]]*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **color** (*Union[List[int], Tuple[int, ...]], optional*) –
- **fill** (*Union[List[int], Tuple[int, ...]], optional*) –
- **thickness** (*float, optional*) –
- **id** (*Union[int, str], optional*) – (deprecated)

Returns Union[int, str]

`dearpygui.dearpygui.drawlist(width, height, **kwargs)`

Adds a drawing canvas.

Parameters

- **width** (*int*) –
- **height** (*int*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking

- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Yields Union[int, str]

`dearpygui.dearpygui.empty_container_stack()`

Empties the container stack.

Args: :returns: None

`dearpygui.dearpygui.enable_item(item: Union[int, str])`

Enables the item.

Parameters ****item** – Item to enable.

Returns None

`dearpygui.dearpygui.file_dialog(**kwargs)`

Displays a file or directory selector depending on keywords. Displays a file dialog by default. Callback will be ran when the file or directory picker is closed. The `app_data` arguemnt will be populated with information related to the file and directory as a dictionary.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **default_path** (*str, optional*) – Path that the file dialog will default to when opened.
- **default_filename** (*str, optional*) – Default name that will show in the file name input.
- **file_count** (*int, optional*) – Number of visible files in the dialog.
- **modal** (*bool, optional*) – Forces user interaction with the file selector.
- **directory_selector** (*bool, optional*) – Shows only directory/paths as options. Allows selection of directory/paths only.
- **min_size** (*Union[List[int], Tuple[int, ...]], optional*) – Minimum window size.
- **max_size** (*Union[List[int], Tuple[int, ...]], optional*) – Maximum window size.
- **cancel_callback** (*Callable, optional*) – Callback called when cancel button is clicked.
- **id** (*Union[int, str], optional*) – (deprecated)

Yields Union[int, str]

`dearpygui.dearpygui.filter_set(**kwargs)`

Helper to parse and apply text filters (e.g. aaaaa[, bbbbb][, ccccc])

Parameters

- **label** (*str*, *optional*) – Overrides ‘name’ as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str]*, *optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **width** (*int*, *optional*) – Width of the item.
- **indent** (*int*, *optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str]*, *optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str]*, *optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool*, *optional*) – Attempt to render widget.
- **id** (*Union[int, str]*, *optional*) – (deprecated)
- **delay_search** (*bool*, *optional*) – (deprecated)This was used as an optimization hint but is not relevant anymore.

Yields Union[int, str]

`dearpygui.dearpygui.fit_axis_data(axis)`

Sets the axis boundaries max/min in the data series currently on the plot.

Parameters *axis* (*Union[int, str]*) –

Returns None

`dearpygui.dearpygui.focus_item(item)`

Focuses an item.

Parameters *item* (*Union[int, str]*) –

Returns None

`dearpygui.dearpygui.font(file, size, **kwargs)`

Adds font to a font registry.

Parameters

- **file** (*str*) –
- **size** (*int*) –
- **label** (*str*, *optional*) – Overrides ‘name’ as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str]*, *optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.

- **pixel_snapH** (*bool, optional*) – Align every glyph to pixel boundary in horizontal direction. Useful if you are rendering text piece-by-piece (e.g. for coloring).
- **pixel_snapV** (*bool, optional*) – Align scaled GlyphOffset.y to pixel boundaries in ImGui.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **id** (*Union[int, str], optional*) – (deprecated)
- **default_font** (*bool, optional*) – (deprecated)

Yields Union[int, str]

`dearpygui.dearpygui.font_registry(**kwargs)`

Adds a font registry.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Yields Union[int, str]

`dearpygui.dearpygui.generate_uuid()`

Generate a new UUID.

Args: :returns: Union[int, str]

`dearpygui.dearpygui.get_active_window()`

Returns the active window.

Args: :returns: Union[int, str]

`dearpygui.dearpygui.get_alias_id(alias)`

Returns the ID associated with an alias.

Parameters **alias** (*str*) –

Returns Union[int, str]

`dearpygui.dearpygui.get_aliases()`

Returns all aliases.

Args: :returns: Union[List[str], Tuple[str, ...]]

`dearpygui.dearpygui.get_all_items()`

Returns all items.

Args: :returns: Union[List[int], Tuple[int, ...]]

`dearpygui.dearpygui.get_app_configuration()`

Returns app configuration.

Args: :returns: dict

`dearpygui.dearpygui.get_available_content_region(item: Union[int, str]) → List[int]`

Returns item's available content region.

Returns position

`dearpygui.dearpygui.get_axis_limits(axis)`

Get the specified axis limits.

Parameters `axis` (*Union[int, str]*) –

Returns Union[List[float], Tuple[float, ...]]

`dearpygui.dearpygui.get_callback_queue()`

New in 1.2. Returns and clears callback queue.

Args: :returns: Any

`dearpygui.dearpygui.get_clipboard_text()`

New in 1.3. Gets the clipboard text.

Args: :returns: str

`dearpygui.dearpygui.get_colormap_color(colormap, index)`

Returns a color from a colormap given an index ≥ 0 . (ex. 0 will be the first color in the color list of the colormap) Modulo will be performed against the number of items in the color list.

Parameters

- **colormap** (*Union[int, str]*) – The colormap tag. This should come from a colormap that was added to a colormap registry. Built in color maps are accessible through their corresponding constants `mvPlotColormap_Twilight`, `mvPlotColormap_***`
- **index** (*int*) – Desired position of the color in the colors list value of the colormap being queried

Returns Union[List[int], Tuple[int, ...]]

`dearpygui.dearpygui.get_dearpygui_version()`

return Dear PyGui Version

`dearpygui.dearpygui.get_delta_time()`

Returns time since last frame.

Args: :returns: float

`dearpygui.dearpygui.get_drawing_mouse_pos()`

Returns mouse position in drawing.

Args: :returns: Union[List[int], Tuple[int, ...]]

`dearpygui.dearpygui.get_file_dialog_info(file_dialog)`

Returns information related to the file dialog. Typically used while the file dialog is in use to query data about the state or info related to the file dialog.

Parameters `file_dialog` (*Union[int, str]*) –

Returns dict

`dearpygui.dearpygui.get_focused_item()`

Returns the item currently having focus.

Args: :returns: Union[int, str]

`dearpygui.dearpygui.get_frame_count()`

Returns frame count.

Args: :returns: int

`dearpygui.dearpygui.get_frame_rate()`
Returns the average frame rate across 120 frames.

Args: :returns: float

`dearpygui.dearpygui.get_global_font_scale()`
Returns global font scale.

Args: :returns: float

`dearpygui.dearpygui.get_item_alias(item)`
Returns an item's alias.

Parameters `item` (*Union[int, str]*) –

Returns str

`dearpygui.dearpygui.get_item_callback(item: Union[int, str])` → Optional[Callable]
Gets the item's callback.

Returns callback as a callable or None

`dearpygui.dearpygui.get_item_children(item: Union[int, str], slot: int = -1)` → Optional[Union[dict, List[int]]]

Provides access to the item's children slots.

Returns A 2-D tuple of children slots ex. ((child_slot_1),(child_slot_2),(child_slot_3),...) or a single slot if slot is used.

`dearpygui.dearpygui.get_item_configuration(item)`
Returns an item's configuration.

Parameters `item` (*Union[int, str]*) –

Returns dict

`dearpygui.dearpygui.get_item_disabled_theme(item: Union[int, str])` → int
Gets the item's disabled theme.

Returns theme's uuid

`dearpygui.dearpygui.get_item_drag_callback(item: Union[int, str])` → Optional[Callable]
Gets the item's drag callback.

Returns callback as a callable or None

`dearpygui.dearpygui.get_item_drop_callback(item: Union[int, str])` → Optional[Callable]
Gets the item's drop callback.

Returns callback as a callable or None

`dearpygui.dearpygui.get_item_filter_key(item: Union[int, str])` → Optional[str]
Gets the item's filter key.

Returns filter key as a string or None

`dearpygui.dearpygui.get_item_font(item: Union[int, str])` → int
Gets the item's font.

Returns font's uuid

`dearpygui.dearpygui.get_item_height(item: Union[int, str])` → Optional[int]
Gets the item's height.

Returns height as a int or None

`dearpygui.dearpygui.get_item_indent(item: Union[int, str]) → Optional[int]`
Gets the item's indent.

Returns indent as a int or None

`dearpygui.dearpygui.get_item_info(item)`
Returns an item's information.

Parameters `item` (`Union[int, str]`) –

Returns dict

`dearpygui.dearpygui.get_item_label(item: Union[int, str]) → Optional[str]`
Gets the item's label.

Returns label as a string or None

`dearpygui.dearpygui.get_item_parent(item: Union[int, str]) → Optional[int]`
Gets the item's parent.

Returns parent as a int or None

`dearpygui.dearpygui.get_item_pos(item: Union[int, str]) → List[int]`
Returns item's position.

Returns position

`dearpygui.dearpygui.get_item_rect_max(item: Union[int, str]) → List[int]`
Returns item's maximum content region.

Returns position

`dearpygui.dearpygui.get_item_rect_min(item: Union[int, str]) → List[int]`
Returns item's minimum content region.

Returns position

`dearpygui.dearpygui.get_item_rect_size(item: Union[int, str]) → List[int]`
Returns item's available content region.

Returns position

`dearpygui.dearpygui.get_item_slot(item: Union[int, str]) → Optional[int]`
Returns an item's target slot.

Returns slot as a int

`dearpygui.dearpygui.get_item_source(item: Union[int, str]) → Optional[str]`
Gets the item's source.

Returns source as a string or None

`dearpygui.dearpygui.get_item_state(item)`
Returns an item's state.

Parameters `item` (`Union[int, str]`) –

Returns dict

`dearpygui.dearpygui.get_item_theme(item: Union[int, str]) → int`
Gets the item's theme.

Returns theme's uuid

`dearpygui.dearpygui.get_item_track_offset(item: Union[int, str]) → Optional[float]`
Gets the item's track offset.

Returns track offset as a int or None

`dearpygui.dearpygui.get_item_type(item: Union[int, str]) → str`
Gets the item's type.

Returns type as a string or None

`dearpygui.dearpygui.get_item_types()`
Returns an item types.

Args: :returns: dict

`dearpygui.dearpygui.get_item_user_data(item: Union[int, str]) → Optional[Any]`
Gets the item's callback data.

Returns callback data as a python object or None

`dearpygui.dearpygui.get_item_width(item: Union[int, str]) → Optional[int]`
Gets the item's width.

Returns width as a int or None

`dearpygui.dearpygui.get_major_version()`
return Dear PyGui Major Version

`dearpygui.dearpygui.get_minor_version()`
return Dear PyGui Minor Version

`dearpygui.dearpygui.get_mouse_drag_delta()`
Returns mouse drag delta.

Args: :returns: float

`dearpygui.dearpygui.get_mouse_pos(**kwargs)`
Returns mouse position.

Parameters `local (bool, optional) –`

Returns Union[List[int], Tuple[int, ...]]

`dearpygui.dearpygui.get_platform()`
New in 1.6. Returns platform constant.

Args: :returns: int

`dearpygui.dearpygui.get_plot_mouse_pos()`
Returns mouse position in plot.

Args: :returns: Union[List[int], Tuple[int, ...]]

`dearpygui.dearpygui.get_plot_query_area(plot: Union[int, str], **kwargs)`
(deprecated function) Returns the last/current query area of the plot. If no area is available [0, 0, 0, 0] will be returned.

Parameters `plot (Union[int, str]) –`

Returns Union[List[float], Tuple[float, ...]]

`dearpygui.dearpygui.get_plot_query_rects(plot)`

Returns the query rects of the plot. Returns an array of array containing the top-left coordinates and bottom-right coordinates of the plot area.

Parameters `plot (Union[int, str]) –`

Returns List[List[float]]

`dearpygui.dearpygui.get_selected_links(node_editor)`

Returns a node editor's selected links.

Parameters `node_editor` (*Union[int, str]*) –

Returns `List[List[str]]`

`dearpygui.dearpygui.get_selected_nodes(node_editor)`

Returns a node editor's selected nodes.

Parameters `node_editor` (*Union[int, str]*) –

Returns `Union[List[int], Tuple[int, ...]]`

`dearpygui.dearpygui.get_text_size(text, **kwargs)`

Returns width/height of text with specified font (must occur after 1st frame).

Parameters

- **text** (*str*) –
- **wrap_width** (*float, optional*) – Wrap width to use (-1.0 turns wrap off).
- **font** (*Union[int, str], optional*) – Font to use.

Returns `Union[List[float], Tuple[float, ...]]`

`dearpygui.dearpygui.get_total_time()`

Returns total time since Dear PyGui has started.

Args: :returns: float

`dearpygui.dearpygui.get_value(item)`

Returns an item's value.

Parameters `item` (*Union[int, str]*) –

Returns Any

`dearpygui.dearpygui.get_values(items)`

Returns values of a list of items.

Parameters `items` (*Union[List[int], Tuple[int, ...]]*) –

Returns Any

`dearpygui.dearpygui.get_viewport_clear_color() → List[int]`

Gets the viewport's clear color.

Returns `List[int]`

`dearpygui.dearpygui.get_viewport_client_height() → int`

Gets the viewport's client height.

Returns viewport width

`dearpygui.dearpygui.get_viewport_client_width() → int`

Gets the viewport's client width.

Returns viewport width

`dearpygui.dearpygui.get_viewport_configuration(item)`

Returns a viewport's configuration.

Parameters `item` (*Union[int, str]*) –

Returns dict

`dearpygui.dearpygui.get_viewport_height()` → int
Gets the viewport's height.

Returns int

`dearpygui.dearpygui.get_viewport_max_height()` → int
Gets the viewport's max width.

Returns int

`dearpygui.dearpygui.get_viewport_max_width()` → int
Gets the viewport's max width.

Returns int

`dearpygui.dearpygui.get_viewport_min_height()` → int
Gets the viewport's minimum height.

Returns int

`dearpygui.dearpygui.get_viewport_min_width()` → int
Gets the viewport's minimum width.

Returns int

`dearpygui.dearpygui.get_viewport_pos()` → List[float]
Gets the viewport's position.

Returns viewport position.

`dearpygui.dearpygui.get_viewport_title()` → str
Gets the viewport's title.

Returns str

`dearpygui.dearpygui.get_viewport_width()` → int
Gets the viewport's width.

Returns viewport width

`dearpygui.dearpygui.get_windows()`
Returns all windows.

Args: :returns: Union[List[int], Tuple[int, ...]]

`dearpygui.dearpygui.get_x_scroll(item)`
Undocumented

Parameters *item* (Union[int, str]) –

Returns float

`dearpygui.dearpygui.get_x_scroll_max(item)`
Undocumented

Parameters *item* (Union[int, str]) –

Returns float

`dearpygui.dearpygui.get_y_scroll(item)`
Undocumented

Parameters *item* (Union[int, str]) –

Returns float

`dearpygui.dearpygui.get_y_scroll_max(item)`

Undocumented

Parameters `item` (`Union[int, str]`) –

Returns float

`dearpygui.dearpygui.group(**kwargs)`

Creates a group that other widgets can belong to. The group allows item commands to be issued for all of its members. Enable property acts in a special way enabling/disabling everything inside the group. (Use `mvStyle-Var_DisabledAlpha` to edit colors within the disabled group.)

Args: `label` (`str`, optional): Overrides ‘name’ as label. `user_data` (`Any`, optional): User data for callbacks `use_internal_label` (`bool`, optional): Use generated internal label instead of user specified (appends `### uuid`). `tag` (`Union[int, str]`, optional): Unique id used to programmatically refer to the item.If label is unused this will be the label. `width` (`int`, optional): Width of the item. `height` (`int`, optional): Height of the item. `indent` (`int`, optional): Offsets the widget to the right the specified number multiplied by the indent style. `parent` (`Union[int, str]`, optional): Parent to add this item to. (`runtime adding`) `before` (`Union[int, str]`, optional): This item will be displayed before the specified item in the parent. `payload_type` (`str`, optional): Sender string type must be the same as the target for the target to run the `payload_callback`. `drag_callback` (`Callable`, optional): Registers a drag callback for drag and drop. `drop_callback` (`Callable`, optional): Registers a drop callback for drag and drop. `show` (`bool`, optional): Attempt to render widget. `enabled` (`bool`, optional): Turns off functionality of widget and applies the disabled theme. `pos` (`Union[List[int], Tuple[int, ...]]`, optional): Places the item relative to window coordinates, [0,0] is top left. `filter_key` (`str`, optional): Used by filter widget. `tracked` (`bool`, optional): Scroll tracking `track_offset` (`float`, optional): 0.0f:top, 0.5f:center, 1.0f:bottom `horizontal` (`bool`, optional): Forces child widgets to be added in a horizontal layout. `horizontal_spacing` (`float`, optional): Spacing for the horizontal layout. `xoffset` (`float`, optional): Offset from containing window x item location within group. `id` (`Union[int, str]`, optional): (deprecated) `delay_search` (`bool`, optional): (deprecated)This was used as an optimization hint but is not relevant anymore.

Yields: `Union[int, str]`

`dearpygui.dearpygui.handler_registry(**kwargs)`

Adds a handler registry.

Parameters

- **label** (`str`, optional) – Overrides ‘name’ as label.
- **user_data** (`Any`, optional) – User data for callbacks
- **use_internal_label** (`bool`, optional) – Use generated internal label instead of user specified (appends `### uuid`).
- **tag** (`Union[int, str]`, optional) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **show** (`bool`, optional) – Attempt to render widget.
- **id** (`Union[int, str]`, optional) – (deprecated)

Yields `Union[int, str]`

`dearpygui.dearpygui.hide_item(item: Union[int, str], *, children_only: bool = False)`

Hides the item.

Parameters `**item` – Item to hide.

Returns None

`dearpygui.dearpygui.highlight_table_cell(table, row, column, color)`
Highlight specified table cell.

Parameters

- **table** (*Union[int, str]*) –
- **row** (*int*) –
- **column** (*int*) –
- **color** (*Union[List[int], Tuple[int, ...]]*) –

Returns None

`dearpygui.dearpygui.highlight_table_column(table, column, color)`
Highlight specified table column.

Parameters

- **table** (*Union[int, str]*) –
- **column** (*int*) –
- **color** (*Union[List[int], Tuple[int, ...]]*) –

Returns None

`dearpygui.dearpygui.highlight_table_row(table, row, color)`
Highlight specified table row.

Parameters

- **table** (*Union[int, str]*) –
- **row** (*int*) –
- **color** (*Union[List[int], Tuple[int, ...]]*) –

Returns None

`dearpygui.dearpygui.is_dearpygui_running()`
Checks if Dear PyGui is running

Args: :returns: bool

`dearpygui.dearpygui.is_item_activated(item: Union[int, str]) → Optional[bool]`
Checks if item is activated.

Returns status as a bool

`dearpygui.dearpygui.is_item_active(item: Union[int, str]) → Optional[bool]`
Checks if item is active.

Returns status as a bool

`dearpygui.dearpygui.is_item_clicked(item: Union[int, str]) → Optional[bool]`
Checks if item is clicked.

Returns status as a bool

`dearpygui.dearpygui.is_item_container(item: Union[int, str]) → Optional[bool]`
Checks if item is a container.

Returns status as a bool

`dearpygui.dearpygui.is_item_deactivated(item: Union[int, str]) → Optional[bool]`
Checks if item is deactivated.

Returns status as a bool

`dearpygui.dearpygui.is_item_deactivated_after_edit(item: Union[int, str]) → Optional[bool]`
Checks if item is `deactivated_after_edit`.

Returns status as a bool

`dearpygui.dearpygui.is_item_edited(item: Union[int, str]) → Optional[bool]`
Checks if item is edited.

Returns status as a bool

`dearpygui.dearpygui.is_item_enabled(item: Union[int, str]) → Optional[bool]`
Checks if item is enabled.

Returns status as a bool

`dearpygui.dearpygui.is_item_focused(item: Union[int, str]) → Optional[bool]`
Checks if item is focused.

Returns status as a bool

`dearpygui.dearpygui.is_item_hovered(item: Union[int, str]) → Optional[bool]`
Checks if item is hovered.

Returns status as a bool

`dearpygui.dearpygui.is_item_left_clicked(item: Union[int, str]) → Optional[bool]`
Checks if item is left clicked.

Returns status as a bool

`dearpygui.dearpygui.is_item_middle_clicked(item: Union[int, str]) → Optional[bool]`
Checks if item is middle clicked.

Returns status as a bool

`dearpygui.dearpygui.is_item_ok(item: Union[int, str]) → Optional[bool]`
Checks if item is ok and can be used.

Returns status as a bool

`dearpygui.dearpygui.is_item_right_clicked(item: Union[int, str]) → Optional[bool]`
Checks if item is right clicked.

Returns status as a bool

`dearpygui.dearpygui.is_item_search_delayed(item: Union[int, str]) → Optional[bool]`
Checks if item is search delayed.

Returns tracked as a bool or None

`dearpygui.dearpygui.is_item_shown(item: Union[int, str]) → Optional[bool]`
Checks if item is shown.

Returns status as a bool

`dearpygui.dearpygui.is_item_toggled_open(item: Union[int, str]) → Optional[bool]`
Checks if item is `toggled_open`.

Returns status as a bool

`dearpygui.dearpygui.is_item_tracked(item: Union[int, str]) → Optional[bool]`
Checks if item is tracked.

Returns tracked as a bool or None

`dearpygui.dearpygui.is_item_visible(item: Union[int, str]) → Optional[bool]`
Checks if item is visible.

Returns status as a bool

`dearpygui.dearpygui.is_key_down(key)`
Checks if key is down.

Parameters `key (int)` –

Returns bool

`dearpygui.dearpygui.is_key_pressed(key)`
Checks if key is pressed.

Parameters `key (int)` –

Returns bool

`dearpygui.dearpygui.is_key_released(key)`
Checks if key is released.

Parameters `key (int)` –

Returns bool

`dearpygui.dearpygui.is_mouse_button_clicked(button)`
Checks if mouse button is clicked.

Parameters `button (int)` –

Returns bool

`dearpygui.dearpygui.is_mouse_button_double_clicked(button)`
Checks if mouse button is double clicked.

Parameters `button (int)` –

Returns bool

`dearpygui.dearpygui.is_mouse_button_down(button)`
Checks if mouse button is down.

Parameters `button (int)` –

Returns bool

`dearpygui.dearpygui.is_mouse_button_dragging(button, threshold)`
Checks if mouse button is down and dragging.

Parameters

- `button (int)` –
- `threshold (float)` –

Returns bool

`dearpygui.dearpygui.is_mouse_button_released(button)`
Checks if mouse button is released.

Parameters `button (int)` –

Returns bool

`dearpygui.dearpygui.is_plot_queried(plot: Union[int, str], **kwargs)`
(deprecated function) Returns true if the plot is currently being queried.

Parameters `plot(Union[int, str])` –

Returns bool

`dearpygui.dearpygui.is_table_cell_highlighted(table, row, column)`

Checks if a table cell is highlighted.

Parameters

- **table** (`Union[int, str]`) –
- **row** (`int`) –
- **column** (`int`) –

Returns bool

`dearpygui.dearpygui.is_table_column_highlighted(table, column)`

Checks if a table column is highlighted.

Parameters

- **table** (`Union[int, str]`) –
- **column** (`int`) –

Returns bool

`dearpygui.dearpygui.is_table_row_highlighted(table, row)`

Checks if a table row is highlighted.

Parameters

- **table** (`Union[int, str]`) –
- **row** (`int`) –

Returns bool

`dearpygui.dearpygui.is_viewport_always_top()` → bool

Checks the viewport always on top flag.

Returns bool

`dearpygui.dearpygui.is_viewport_created()`

deprecated function

`dearpygui.dearpygui.is_viewport_decorated()` → bool

Checks if the viewport is decorated.

Returns bool

`dearpygui.dearpygui.is_viewport_ok()`

Checks if a viewport has been created and shown.

Args: :returns: bool

`dearpygui.dearpygui.is_viewport_resizable()` → bool

Checks the viewport resizable flag.

Returns bool

`dearpygui.dearpygui.is_viewport_vsync_on()` → bool

Checks the viewport vsync flag.

Returns bool

`dearpygui.dearpygui.item_handler_registry(**kwargs)`

Adds an item handler registry.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Yields Union[int, str]

`dearpygui.dearpygui.last_container()`

Returns the last container item added.

Args: :returns: Union[int, str]

`dearpygui.dearpygui.last_item()`

Returns the last item added.

Args: :returns: Union[int, str]

`dearpygui.dearpygui.last_root()`

Returns the last root added (registry or window).

Args: :returns: Union[int, str]

`dearpygui.dearpygui.load_image(file, **kwargs)`

Loads an image. Returns width, height, channels, mvBuffer

Parameters

- **file** (*str*) –
- **gamma** (*float, optional*) – Gamma correction factor. (default is 1.0 to avoid automatic gamma correction on loading).
- **gamma_scale_factor** (*float, optional*) – Gamma scale factor.

Returns Any

`dearpygui.dearpygui.load_init_file(file)`

deprecated function

`dearpygui.dearpygui.lock_mutex()`

Locks render thread mutex.

Args: :returns: None

`dearpygui.dearpygui.maximize_viewport()`

Maximizes the viewport.

Args: :returns: None

`dearpygui.dearpygui.menu(**kwargs)`

Adds a menu to an existing menu bar.

Parameters

- **label** (*str*, *optional*) – Overrides ‘name’ as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str]*, *optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **indent** (*int*, *optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str]*, *optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str]*, *optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str*, *optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **drop_callback** (*Callable*, *optional*) – Registers a drop callback for drag and drop.
- **show** (*bool*, *optional*) – Attempt to render widget.
- **enabled** (*bool*, *optional*) – Turns off functionality of widget and applies the disabled theme.
- **filter_key** (*str*, *optional*) – Used by filter widget.
- **tracked** (*bool*, *optional*) – Scroll tracking
- **track_offset** (*float*, *optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **id** (*Union[int, str]*, *optional*) – (deprecated)
- **delay_search** (*bool*, *optional*) – (deprecated)This was used as an optimization hint but is not relevant anymore.

Yields `Union[int, str]`

`dearpygui.dearpygui.menu_bar(**kwargs)`

Adds a menu bar to a window.

Parameters

- **label** (*str*, *optional*) – Overrides ‘name’ as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str]*, *optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **indent** (*int*, *optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str]*, *optional*) – Parent to add this item to. (runtime adding)
- **show** (*bool*, *optional*) – Attempt to render widget.
- **id** (*Union[int, str]*, *optional*) – (deprecated)
- **delay_search** (*bool*, *optional*) – (deprecated)This was used as an optimization hint but is not relevant anymore.

Yields Union[int, str]

`dearpygui.dearpygui.minimize_viewport()`

Minimizes a viewport.

Args: :returns: None

`dearpygui.dearpygui.move_item(item, **kwargs)`

Moves an item to a new location.

Parameters

- **item** (Union[int, str]) –
- **parent** (Union[int, str], optional) –
- **before** (Union[int, str], optional) –

Returns None

`dearpygui.dearpygui.move_item_down(item)`

Moves an item down.

Parameters **item** (Union[int, str]) –

Returns None

`dearpygui.dearpygui.move_item_up(item)`

Moves an item up.

Parameters **item** (Union[int, str]) –

Returns None

`dearpygui.dearpygui.mutex()`

Handles locking/unlocking render thread mutex.

`dearpygui.dearpygui.node(**kwargs)`

Adds a node to a node editor.

Parameters

- **label** (str, optional) – Overrides 'name' as label.
- **user_data** (Any, optional) – User data for callbacks
- **use_internal_label** (bool, optional) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (Union[int, str], optional) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (Union[int, str], optional) – Parent to add this item to. (runtime adding)
- **before** (Union[int, str], optional) – This item will be displayed before the specified item in the parent.
- **payload_type** (str, optional) – Sender string type must be the same as the target for the target to run the payload_callback.
- **drag_callback** (Callable, optional) – Registers a drag callback for drag and drop.
- **drop_callback** (Callable, optional) – Registers a drop callback for drag and drop.
- **show** (bool, optional) – Attempt to render widget.
- **pos** (Union[List[int], Tuple[int, ...]], optional) – Places the item relative to window coordinates, [0,0] is top left.

- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **draggable** (*bool, optional*) – Allow node to be draggable.
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Yields Union[int, str]

`dearpygui.dearpygui.node_attribute(**kwargs)`

Adds a node attribute to a node.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **attribute_type** (*int, optional*) – mvNode_Attr_Input, mvNode_Attr_Output, or mvNode_Attr_Static.
- **shape** (*int, optional*) – Pin shape.
- **category** (*str, optional*) – Category
- **id** (*Union[int, str], optional*) – (deprecated)

Yields Union[int, str]

`dearpygui.dearpygui.node_editor(**kwargs)`

Adds a node editor.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).

- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **delink_callback** (*Callable, optional*) – Callback ran when a link is detached.
- **menubar** (*bool, optional*) – Shows or hides the menubar.
- **minimap** (*bool, optional*) – Shows or hides the Minimap. New in 1.6.
- **minimap_location** (*int, optional*) – `mvNodeMiniMap_Location_*` constants. New in 1.6.
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Yields `Union[int, str]`

`dearpygui.dearpygui.output_frame_buffer(file="", **kwargs)`

Outputs frame buffer as a png if file is specified or through the second argument of a callback if specified. Render loop must have been started.

Parameters

- **file** (*str, optional*) –
- **callback** (*Callable, optional*) – Callback will return framebuffer as an array through the second arg.

Returns `Any`

`dearpygui.dearpygui.plot(**kwargs)`

Adds a plot which is used to hold series, and can be drawn to with draw commands. For all `_mod` parameters use `mvKey_ModX` enums, or `mvKey_ModDisabled` to disable the modifier.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.

- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **callback** (*Callable, optional*) – Registers a callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **no_title** (*bool, optional*) – the plot title will not be displayed
- **no_menus** (*bool, optional*) – the user will not be able to open context menus with right-click
- **no_box_select** (*bool, optional*) – the user will not be able to box-select with right-click drag
- **no_mouse_pos** (*bool, optional*) – the text of mouse position, in plot coordinates, will not be displayed inside of the plot
- **query** (*bool, optional*) – the user will be able to draw query rects with CTRL + right-click drag
- **query_color** (*Union[List[float], Tuple[float, ...]], optional*) – Color of the query rectangles.
- **min_query_rects** (*int, optional*) – The minimum number of query rects that can be in the plot. If there are less rects than this value, it won't be possible to delete them.
- **max_query_rects** (*int, optional*) – The maximum number of query rects that can be in the plot. If the number is reached any rect added will replace the latest one. (0 means unlimited)
- **crosshairs** (*bool, optional*) – the default mouse cursor will be replaced with a crosshair when hovered
- **equal_aspects** (*bool, optional*) – primary x and y axes will be constrained to have the same units/pixel (does not apply to auxiliary y-axes)
- **no_inputs** (*bool, optional*) – the user will not be able to interact with the plot
- **no_frame** (*bool, optional*) – the ImGui frame will not be rendered
- **use_local_time** (*bool, optional*) – axis labels will be formatted for your timezone when

- **use_ISO8601** (*bool, optional*) – dates will be formatted according to ISO 8601 where applicable (e.g. YYYY-MM-DD, YYYY-MM, –MM-DD, etc.)
- **use_24hour_clock** (*bool, optional*) – times will be formatted using a 24 hour clock
- **pan_button** (*int, optional*) – mouse button that enables panning when held
- **pan_mod** (*int, optional*) – optional modifier that must be held for panning
- **context_menu_button** (*int, optional*) – opens context menus (if enabled) when clicked
- **fit_button** (*int, optional*) – fits visible data when double clicked
- **box_select_button** (*int, optional*) – begins box selection when pressed and confirms selection when released
- **box_select_mod** (*int, optional*) – begins box selection when pressed and confirms selection when released
- **box_select_cancel_button** (*int, optional*) – cancels active box selection when pressed
- **query_toggle_mod** (*int, optional*) – when held, active box selections turn into queries
- **horizontal_mod** (*int, optional*) – expands active box selection/query horizontally to plot edge when held
- **vertical_mod** (*int, optional*) – expands active box selection/query vertically to plot edge when held
- **override_mod** (*int, optional*) – when held, all input is ignored; used to enable axis/plots as DND sources
- **zoom_mod** (*int, optional*) – optional modifier that must be held for scroll wheel zooming
- **zoom_rate** (*int, optional*) – zoom rate for scroll (e.g. 0.1f = 10% plot range every scroll click); make negative to invert
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.
- **no_highlight** (*bool, optional*) – (deprecated) Removed because not supported from the backend anymore. To control the highlighting of series use the same argument in *add_plot_legend*
- **no_child** (*bool, optional*) – (deprecated) a child window region will not be used to capture mouse scroll (can boost performance for single ImGui window applications)
- **anti_aliased** (*bool, optional*) – (deprecated) This feature was deprecated in ImPlot. To enable/disable anti_aliasing use *dpg.configure_app()* with the *anti_aliasing* parameters.
- **query_button** (*int, optional*) – (deprecated) This refers to the old way of querying of ImPlot, now replaced with *DragRect()*
- **query_mod** (*int, optional*) – (deprecated) This refers to the old way of querying of ImPlot, now replaced with *DragRect()*

Yields Union[int, str]

`dearpygui.dearpygui.plot_axis(axis, **kwargs)`

Adds an axis to a plot.

Parameters

- **axis** (*int*) –
- **label** (*str*, *optional*) – Overrides ‘name’ as label.
- **user_data** (*Any*, *optional*) – User data for callbacks
- **use_internal_label** (*bool*, *optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str]*, *optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str]*, *optional*) – Parent to add this item to. (runtime adding)
- **payload_type** (*str*, *optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **drop_callback** (*Callable*, *optional*) – Registers a drop callback for drag and drop.
- **show** (*bool*, *optional*) – Attempt to render widget.
- **no_label** (*bool*, *optional*) – the axis label will not be displayed
- **no_gridlines** (*bool*, *optional*) – no grid lines will be displayed
- **no_tick_marks** (*bool*, *optional*) – no tick marks will be displayed
- **no_tick_labels** (*bool*, *optional*) – no text labels will be displayed
- **no_initial_fit** (*bool*, *optional*) – axis will not be initially fit to data extents on the first rendered frame
- **no_menus** (*bool*, *optional*) – the user will not be able to open context menus with right-click
- **no_side_switch** (*bool*, *optional*) – the user will not be able to switch the axis side by dragging it
- **no_highlight** (*bool*, *optional*) – the axis will not have its background highlighted when hovered or held
- **opposite** (*bool*, *optional*) – axis ticks and labels will be rendered on the conventionally opposite side (i.e. right or top)
- **foreground_grid** (*bool*, *optional*) – grid lines will be displayed in the foreground (i.e. on top of data) instead of the background
- **tick_format** (*str*, *optional*) – Sets a custom tick label formatter
- **scale** (*int*, *optional*) – Sets the axis’ scale. Can have only `mvPlotScale_` values
- **invert** (*bool*, *optional*) – the axis values will be inverted (i.e. growing from right to left)
- **auto_fit** (*bool*, *optional*) – axis will be auto-fitting to data extents
- **range_fit** (*bool*, *optional*) – axis will only fit points if the point is in the visible range of the **orthogonal** axis
- **pan_stretch** (*bool*, *optional*) – panning in a locked or constrained state will cause the axis to stretch if possible
- **lock_min** (*bool*, *optional*) – the axis minimum value will be locked when panning/zooming
- **lock_max** (*bool*, *optional*) – the axis maximum value will be locked when panning/zooming

- **id** (*Union[int, str], optional*) – (deprecated)
- **log_scale** (*bool, optional*) – (deprecated) Old way to set log scale in the axis. Use ‘scale’ argument instead.
- **time** (*bool, optional*) – (deprecated) Old way to set time scale in the axis. Use ‘scale’ argument instead.

Yields Union[int, str]

`dearpygui.dearpygui.pop_container_stack()`

Pops the top item off the parent stack and return its ID.

Args: :returns: Union[int, str]

`dearpygui.dearpygui.popup` (*parent: Union[int, str], mousebutton: int = 0, modal: bool = False, tag: Union[int, str] = 0, min_size: Union[List[int], Tuple[int, ...]] = [100, 100], max_size: Union[List[int], Tuple[int, ...]] = [30000, 30000], no_move: bool = False, no_background: bool = False*) → int

A window that will be displayed when a parent item is hovered and the corresponding mouse button has been clicked. By default a popup will shrink fit the items it contains. This is useful for context windows, and simple modal window popups. When popups are used a modal they have more available settings (i.e. title, resize, width, height) These can be set by using configure item. This is a light wrapper over window. For more control over a modal|popup window use a normal window with the modal|popup keyword and set the item handler and mouse events manually.

Parameters

- **parent** – The UI item that will need to be hovered.
- ****mousebutton** – The mouse button that will trigger the window to popup.
- ****modal** – Will force the user to interact with the popup.
- ****min_size** – New in 1.4. Minimum window size.
- ****max_size** – New in 1.4. Maximum window size.
- ****no_move** – New in 1.4. Prevents the window from moving based on user input.
- ****no_background** – New in 1.4. Sets Background and border alpha to transparent.

Returns item’s uuid

`dearpygui.dearpygui.push_container_stack(item)`

Pushes an item onto the container stack.

Parameters *item* (*Union[int, str]*) –

Returns bool

`dearpygui.dearpygui.remove_alias(alias)`

Removes an alias.

Parameters *alias* (*str*) –

Returns None

`dearpygui.dearpygui.render_dearpygui_frame()`

Render a single Dear PyGui frame.

Args: :returns: None

`dearpygui.dearpygui.reorder_items(container, slot, new_order)`

Reorders an item’s children.

Parameters

- **container** (*Union[int, str]*) –
- **slot** (*int*) –
- **new_order** (*Union[List[int], Tuple[int, ...]]*) –

Returns None`dearpygui.dearpygui.reset_axis_limits_constraints(axis)`

Remove an axis' limits constraints

Parameters **axis** (*Union[int, str]*) –**Returns** None`dearpygui.dearpygui.reset_axis_ticks(axis)`

Removes the manually set axis ticks and applies the default axis ticks

Parameters **axis** (*Union[int, str]*) –**Returns** None`dearpygui.dearpygui.reset_axis_zoom_constraints(axis)`

Remove an axis' zoom constraints

Parameters **axis** (*Union[int, str]*) –**Returns** None`dearpygui.dearpygui.reset_default_theme(item, source)`

deprecated function

`dearpygui.dearpygui.reset_pos(item)`

Resets an item's position after using 'set_item_pos'.

Parameters **item** (*Union[int, str]*) –**Returns** None`dearpygui.dearpygui.run_callbacks(jobs)`

New in 1.2. Runs callbacks from the callback queue and checks arguments.

`dearpygui.dearpygui.sample_colormap(colormap, t)`

Returns a color from a colormap given t between 0.0-1.0.

Parameters

- **colormap** (*Union[int, str]*) – The colormap tag. This should come from a colormap that was added to a colormap registry. Built in color maps are accessible through their corresponding constants `mvPlotColormap_Twilight`, `mvPlotColormap_***`
- **t** (*float*) – Value of the colormap to sample between 0.0-1.0

Returns *Union[List[int], Tuple[int, ...]]*`dearpygui.dearpygui.save_image(file, width, height, data, **kwargs)`

Saves an image. Possible formats: png, bmp, tga, hdr, jpg.

Parameters

- **file** (*str*) –
- **width** (*int*) –
- **height** (*int*) –

- **data** (*Any*) –
- **components** (*int*, *optional*) – Number of components (1-4). Default of 4.
- **quality** (*int*, *optional*) – Stride in bytes (only used for jpg).

Returns None

`dearpygui.dearpygui.save_init_file(file)`

Save dpg.ini file.

Parameters `file` (*str*) –

Returns None

`dearpygui.dearpygui.set_axis_limits(axis, ymin, ymax)`

Sets limits on the axis for pan and zoom.

Parameters

- **axis** (*Union[int, str]*) –
- **ymin** (*float*) –
- **ymax** (*float*) –

Returns None

`dearpygui.dearpygui.set_axis_limits_auto(axis)`

Removes all limits on specified axis.

Parameters `axis` (*Union[int, str]*) –

Returns None

`dearpygui.dearpygui.set_axis_limits_constraints(axis, vmin, vmax)`

Sets an axis' limits constraints so that users can't pan beyond a min or max value

Parameters

- **axis** (*Union[int, str]*) –
- **vmin** (*float*) –
- **vmax** (*float*) –

Returns None

`dearpygui.dearpygui.set_axis_ticks(axis, label_pairs)`

Replaces axis ticks with 'label_pairs' argument.

Parameters

- **axis** (*Union[int, str]*) –
- **label_pairs** (*Any*) – Tuples of label and value in the form '(label, axis_value), (label, axis_value), ...'

Returns None

`dearpygui.dearpygui.set_axis_zoom_constraints(axis, vmin, vmax)`

Sets an axis' zoom constraints so that users can't zoom beyond a min or max value

Parameters

- **axis** (*Union[int, str]*) –
- **vmin** (*float*) –

- **vmax** (*float*) –

Returns None

`dearpygui.dearpygui.set_clip_space(item, top_left_x, top_left_y, width, height, min_depth, max_depth)`

New in 1.1. Set the clip space for depth clipping and ‘viewport’ transformation.

Parameters

- **item** (*Union[int, str]*) – draw layer to set clip space
- **top_left_x** (*float*) – angle to rotate
- **top_left_y** (*float*) – angle to rotate
- **width** (*float*) – angle to rotate
- **height** (*float*) – angle to rotate
- **min_depth** (*float*) – angle to rotate
- **max_depth** (*float*) – angle to rotate

Returns None

`dearpygui.dearpygui.set_clipboard_text(text)`

New in 1.3. Sets the clipboard text.

Parameters **text** (*str*) –

Returns None

`dearpygui.dearpygui.set_colormap(item, source)`

deprecated function

`dearpygui.dearpygui.set_exit_callback(callback, **kwargs)`

Sets a callback to run on last frame.

Parameters

- **callback** (*Callable*) –
- **user_data** (*Any, optional*) – New in 1.3. Optional user data to send to the callback

Returns str

`dearpygui.dearpygui.set_frame_callback(frame, callback, **kwargs)`

Sets a callback to run on first frame.

Parameters

- **frame** (*int*) –
- **callback** (*Callable*) –
- **user_data** (*Any, optional*) – New in 1.3. Optional user data to send to the callback

Returns str

`dearpygui.dearpygui.set_global_font_scale(scale)`

Sets global font scale.

Parameters **scale** (*float*) –

Returns None

`dearpygui.dearpygui.set_init_file(file='dpg.ini')`

deprecated function

`dearpygui.dearpygui.set_item_alias(item, alias)`

Sets an item's alias.

Parameters

- **item** (*Union[int, str]*) –
- **alias** (*str*) –

Returns None

`dearpygui.dearpygui.set_item_callback(item: Union[int, str], callback: Callable)`

Sets the item's callack.

Parameters

- **item** – Item the callback will be applied to.
- **callback** – Callback to be applied.

Returns None

`dearpygui.dearpygui.set_item_children(item, source, slot)`

Sets an item's children.

Parameters

- **item** (*Union[int, str]*) –
- **source** (*Union[int, str]*) –
- **slot** (*int*) –

Returns None

`dearpygui.dearpygui.set_item_drag_callback(item: Union[int, str], callback: Callable)`

Sets the item's drag callack.

Parameters

- **item** – Item the callback will be applied to.
- **callback** – Callback to be applied.

Returns None

`dearpygui.dearpygui.set_item_drop_callback(item: Union[int, str], callback: Callable)`

Sets the item's drop callack.

Parameters

- **item** – Item the callback will be applied to.
- **callback** – Callback to be applied.

Returns None

`dearpygui.dearpygui.set_item_font(item, font)`

deprecated function

`dearpygui.dearpygui.set_item_height(item: Union[int, str], height: int)`

Sets the item's height.

Parameters

- **item** – Item the Height will be applied to.
- **height** – Height to be applied.

Returns None

`dearpygui.dearpygui.set_item_indent(item: Union[int, str], indent: int)`
Sets the item's indent.

Parameters

- **item** – Item the Height will be applied to.
- **height** – Height to be applied.

Returns None

`dearpygui.dearpygui.set_item_label(item: Union[int, str], label: str)`
Sets the item's displayed label, anything after the characters “##” in the name will not be shown.

Parameters

- **item** – Item label will be applied to.
- **label** – Displayed name to be applied.

Returns None

`dearpygui.dearpygui.set_item_payload_type(item: Union[int, str], payload_type: str)`
Sets the item's payload type.

Parameters

- **item** – Item the Height will be applied to.
- **height** – Height to be applied.

Returns None

`dearpygui.dearpygui.set_item_pos(item: Union[int, str], pos: List[float])`
Sets the item's position.

Parameters

- **item** – Item the absolute position will be applied to.
- **pos** – X and Y positions relative to parent of the item.

Returns None

`dearpygui.dearpygui.set_item_source(item: Union[int, str], source: Union[int, str])`
Sets the item's value, to the source's value. Widget's value will now be “linked” to source's value.

Parameters

- **item** – Item to me linked.
- **source** – Source to link to.

Returns None

`dearpygui.dearpygui.set_item_theme(item, theme)`
deprecated function

`dearpygui.dearpygui.set_item_track_offset(item: Union[int, str], offset: float)`
Sets the item's track offset.

Parameters

- **item** – Item the Height will be applied to.
- **height** – Height to be applied.

Returns None

`dearpygui.dearpygui.set_item_type_disabled_theme(item, theme)`
 deprecated function

`dearpygui.dearpygui.set_item_type_theme(item, theme)`
 deprecated function

`dearpygui.dearpygui.set_item_user_data(item: Union[int, str], user_data: Any)`
 Sets the item's callack_data to any python object.

Parameters

- **item** – Item the callback will be applied to.
- **user_data** – Callback_data to be applied.

Returns None

`dearpygui.dearpygui.set_item_width(item: Union[int, str], width: int)`
 Sets the item's width.

Parameters

- **item** – Item the Width will be applied to.
- **width** – Width to be applied.

Returns None

`dearpygui.dearpygui.set_primary_window(window, value=True)`
 Sets the primary window.

Parameters

- **window** (*Union[int, str]*) –
- **value** (*bool, optional*) – True to select this window as primary, False to make it a regular window (without any primary window). Only one window at a time can be primary.

Returns None

`dearpygui.dearpygui.set_staging_mode(mode)`
 deprecated function

`dearpygui.dearpygui.set_start_callback(callback)`
 deprecated function

`dearpygui.dearpygui.set_table_row_color(table, row, color)`
 Set table row color.

Parameters

- **table** (*Union[int, str]*) –
- **row** (*int*) –
- **color** (*Union[List[int], Tuple[int, ...]]*) –

Returns None

`dearpygui.dearpygui.set_value(item, value)`
 Set's an item's value.

Parameters

- **item** (*Union[int, str]*) –

- **value** (*Any*) –

Returns None

`dearpygui.dearpygui.set_viewport_always_top(value: bool)`

Sets the viewport always on top.

Returns None

`dearpygui.dearpygui.set_viewport_clear_color(color: List[int])`

Sets the viewport's clear color.

Returns None

`dearpygui.dearpygui.set_viewport_decorated(value: bool)`

Sets the viewport to be decorated.

Returns None

`dearpygui.dearpygui.set_viewport_height(height: int)`

Sets the viewport's height.

Returns None

`dearpygui.dearpygui.set_viewport_large_icon(icon: str)`

Sets the viewport's large icon. Must be ico for windows.

Returns None

`dearpygui.dearpygui.set_viewport_max_height(height: int)`

Sets the viewport's max width.

Returns None

`dearpygui.dearpygui.set_viewport_max_width(width: int)`

Sets the viewport's max width.

Returns None

`dearpygui.dearpygui.set_viewport_min_height(height: int)`

Sets the viewport's minimum height.

Returns None

`dearpygui.dearpygui.set_viewport_min_width(width: int)`

Sets the viewport's minimum width.

Returns None

`dearpygui.dearpygui.set_viewport_pos(pos: List[float])`

Sets the viewport's position.

Returns None

`dearpygui.dearpygui.set_viewport_resizable(value: bool)`

Sets the viewport resizable.

Returns None

`dearpygui.dearpygui.set_viewport_resize_callback(callback, **kwargs)`

Sets a callback to run on viewport resize.

Parameters

- **callback** (*Callable*) –
- **user_data** (*Any, optional*) – New in 1.3. Optional user data to send to the callback

Returns str

`dearpygui.dearpygui.set_viewport_small_icon(icon: str)`
Sets the viewport's small icon. Must be ico for windows.

Returns None

`dearpygui.dearpygui.set_viewport_title(title: str)`
Sets the viewport's title.

Returns None

`dearpygui.dearpygui.set_viewport_vsync(value: bool)`
Sets the viewport vsync.

Returns None

`dearpygui.dearpygui.set_viewport_width(width: int)`
Sets the viewport's width.

Returns None

`dearpygui.dearpygui.set_x_scroll(item, value, **kwargs)`
Sets horizontal scroll position.

Parameters

- **item** (*Union[int, str]*) –
- **value** (*float*) – Scroll position
- **when** (*int, optional*) – Specifies whether the scroll position will be set in the nearest frame (`mvSetScrollFlags_Now`) or with a 1-frame delay (`mvSetScrollFlags_Delayed`). The former prevents flickering, the latter works better if contents change in the same frame as when `set_x_scroll` called. `mvSetScrollFlags_Both` can also be used to set the position twice.

Returns None

`dearpygui.dearpygui.set_y_scroll(item, value, **kwargs)`
Sets vertical scroll position.

Parameters

- **item** (*Union[int, str]*) –
- **value** (*float*) – Scroll position
- **when** (*int, optional*) – Specifies whether the scroll position will be set in the nearest frame (`mvSetScrollFlags_Now`) or with a 1-frame delay (`mvSetScrollFlags_Delayed`). The former prevents flickering, the latter works better if contents change in the same frame as when `set_x_scroll` called. `mvSetScrollFlags_Both` can also be used to set the position twice.

Returns None

`dearpygui.dearpygui.setup_dearpygui()`
Sets up Dear PyGui

Parameters `viewport` (*Union[int, str], optional*) – (deprecated)

Returns None

`dearpygui.dearpygui.setup_registries()` → None
Adds default registries for fonts, handlers, textures, colormaps, and values.

`dearpygui.dearpygui.setup_viewport()`
deprecated function

`dearpygui.dearpygui.show_about()` → None
Shows the standard about window

Returns None

`dearpygui.dearpygui.show_debug()` → None
Shows the standard debug window

Returns None

`dearpygui.dearpygui.show_documentation()` → None
Shows the standard documentation window

Returns None

`dearpygui.dearpygui.show_font_manager()` → None
Shows a debug tool for the font manager

Returns None

`dearpygui.dearpygui.show_imgui_demo()`
Shows the imgui demo.

Args: :returns: None

`dearpygui.dearpygui.show_implot_demo()`
Shows the implot demo.

Args: :returns: None

`dearpygui.dearpygui.show_item(item: Union[int, str])`
Shows the item.

Parameters `item` – Item to show.

Returns None

`dearpygui.dearpygui.show_item_debug(item)`
Shows an item's debug window

Parameters `item` (`Union[int, str]`) –

Returns None

`dearpygui.dearpygui.show_item_registry()` → None
Shows the item hierarchy of your application

Returns None

`dearpygui.dearpygui.show_metrics()` → None
Shows the standard metrics window

Returns None

`dearpygui.dearpygui.show_style_editor()` → None
Shows the standard style editor window

Returns None

`dearpygui.dearpygui.show_tool(tool)`
Shows a built in tool.

Parameters `tool` (`Union[int, str]`) –

Returns str

`dearpygui.dearpygui.show_viewport(**kwargs)`

Shows the main viewport.

Parameters

- **minimized** (*bool, optional*) – Sets the state of the viewport to minimized
- **maximized** (*bool, optional*) – Sets the state of the viewport to maximized
- **viewport** (*Union[int, str], optional*) – (deprecated)

Returns None

`dearpygui.dearpygui.split_frame()`

Waits one frame.

Parameters **delay** (*int, optional*) – (deprecated) Do not use it anymore, it has no effect.

Returns None

`dearpygui.dearpygui.stage(**kwargs)`

Adds a stage.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **id** (*Union[int, str], optional*) – (deprecated)

Yields Union[int, str]

`dearpygui.dearpygui.staging_container(**kwargs)`

deprecated function :param **label: Overrides 'name' as label. :type **label: str :param **user_data: User data for callbacks. :type **user_data: Any :param **use_internal_label: Use generated internal label instead of user specified (appends ### uuid). :type **use_internal_label: bool :param **id: Unique id used to programmatically refer to the item. If label is unused this will be the label. :type **id: Union[int, str]

Yields Union[int, str]

`dearpygui.dearpygui.start_dearpygui()`

Prepares viewport (if not done already). sets up, cleans up, and runs main event loop.

Returns None

`dearpygui.dearpygui.stop_dearpygui()`

Stops Dear PyGui

Args: :returns: None

`dearpygui.dearpygui.subplots(rows, columns, **kwargs)`

Adds a collection of plots.

Parameters

- **rows** (*int*) –
- **columns** (*int*) –
- **label** (*str, optional*) – Overrides 'name' as label.

- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **row_ratios** (*Union[List[float], Tuple[float, ...]], optional*) –
- **column_ratios** (*Union[List[float], Tuple[float, ...]], optional*) –
- **no_title** (*bool, optional*) – the subplot title will not be displayed
- **no_menus** (*bool, optional*) – the user will not be able to open context menus with right-click
- **no_resize** (*bool, optional*) – resize splitters between subplot cells will not be provided
- **no_align** (*bool, optional*) – subplot edges will not be aligned vertically or horizontally
- **share_series** (*bool, optional*) – when set to True, series from all sub-plots will be shared to some extent, using a single common color set and showing them in a single legend in the subplots item. Otherwise each plot will be independent from others and will have its own legend
- **link_rows** (*bool, optional*) – link the y-axis limits of all plots in each row (does not apply auxiliary y-axes)
- **link_columns** (*bool, optional*) – link the x-axis limits of all plots in each column
- **link_all_x** (*bool, optional*) – link the x-axis limits in every plot in the subplot
- **link_all_y** (*bool, optional*) – link the y-axis limits in every plot in the subplot (does not apply to auxiliary y-axes)
- **column_major** (*bool, optional*) – subplots are added in column major order instead of the default row major order
- **id** (*Union[int, str], optional*) – (deprecated)

- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Yields Union[int, str]

`dearpygui.dearpygui.synced_tables(**kwargs)`

Links all tables that are immediate children of this container so that they share their state (mostly column sizes). Other children are rendered as is. This is an experimental feature, use with caution.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **filter_key** (*str, optional*) – Used by filter widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Yields Union[int, str]

`dearpygui.dearpygui.tab(**kwargs)`

Adds a tab to a tab bar.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the `payload_callback`.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking

- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **closable** (*bool, optional*) – Creates a button on the tab that can hide the tab.
- **no_tooltip** (*bool, optional*) – Disable tooltip for the given tab.
- **order_mode** (*int, optional*) – set using a constant: `mvTabOrder_Reorderable`: allows reordering, `mvTabOrder_Fixed`: fixed ordering, `mvTabOrder_Leading`: adds tab to front, `mvTabOrder_Trailing`: adds tab to back
- **unsaved_document** (*bool, optional*) – Display a dot next to the title.
- **no_close_with_middle_click** (*bool, optional*) – Disable closing this tab (if `closable==True`) by clicking with middle mouse button.
- **no_reorder** (*bool, optional*) – Disable reordering this tab or having another tab cross over this tab.
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Yields Union[int, str]

`dearpygui.dearpygui.tab_bar(**kwargs)`

Adds a tab bar.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **reorderable** (*bool, optional*) – Allows for the user to change the order of the tabs.
- **tab_list_popup_button** (*bool, optional*) – Show a button to select active tab from a dropdown list.

- **no_close_with_middle_click** (*bool, optional*) – Disable closing tabs (that have `closable=True`) by clicking with middle mouse button.
- **no_scrolling_buttons** (*bool, optional*) – Disable left/right scrolling buttons when tab buttons don't fit the container width.
- **no_tooltip** (*bool, optional*) – Disable tooltips when hovering a tab with a long name.
- **draw_selected_overline** (*bool, optional*) – Draw selected overline markers over selected tab.
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Yields Union[int, str]

`dearpygui.dearpygui.table(**kwargs)`

Adds a table.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **source** (*Union[int, str], optional*) – Overrides 'id' as value storage key.
- **callback** (*Callable, optional*) – Registers a callback.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **header_row** (*bool, optional*) – show headers at the top of the columns
- **clipper** (*bool, optional*) – Use clipper (rows must be same height).
- **inner_width** (*int, optional*) –
- **policy** (*int, optional*) –
- **freeze_rows** (*int, optional*) –
- **freeze_columns** (*int, optional*) –

- **sort_multi** (*bool, optional*) – Hold shift when clicking headers to sort on multiple column.
- **sort_tristate** (*bool, optional*) – Allow no sorting, disable default sorting.
- **resizable** (*bool, optional*) – Enable resizing columns
- **reorderable** (*bool, optional*) – Enable reordering columns in header row (need calling `TableSetupColumn()` + `TableHeadersRow()` to display headers)
- **hideable** (*bool, optional*) – Enable hiding/disabling columns in context menu.
- **sortable** (*bool, optional*) – Enable sorting. Call `TableGetSortSpecs()` to obtain sort specs. Also see `ImGuiTableFlags_SortMulti` and `ImGuiTableFlags_SortTristate`.
- **context_menu_in_body** (*bool, optional*) – Right-click on columns body/contents will display table context menu. By default it is available in `TableHeadersRow()`.
- **row_background** (*bool, optional*) – Set each `RowBg` color with `ImGuiCol_TableRowBg` or `ImGuiCol_TableRowBgAlt` (equivalent of calling `TableSetBgColor` with `ImGuiTableBgFlags_RowBg0` on each row manually)
- **borders_innerH** (*bool, optional*) – Draw horizontal borders between rows.
- **borders_outerH** (*bool, optional*) – Draw horizontal borders at the top and bottom.
- **borders_innerV** (*bool, optional*) – Draw vertical borders between columns.
- **borders_outerV** (*bool, optional*) – Draw vertical borders on the left and right sides.
- **no_host_extendX** (*bool, optional*) – Make outer width auto-fit to columns, overriding `outer_size.x` value. Only available when `ScrollX/ScrollY` are disabled and `Stretch` columns are not used.
- **no_host_extendY** (*bool, optional*) – Make outer height stop exactly at `outer_size.y` (prevent auto-extending table past the limit). Only available when `ScrollX/ScrollY` are disabled. Data below the limit will be clipped and not visible.
- **no_keep_columns_visible** (*bool, optional*) – Disable keeping column always minimally visible when `ScrollX` is off and table gets too small. Not recommended if columns are resizable.
- **precise_widths** (*bool, optional*) – Disable distributing remainder width to stretched columns (width allocation on a 100-wide table with 3 columns: Without this flag: 33,33,34. With this flag: 33,33,33). With larger number of columns, resizing will appear to be less smooth.
- **no_clip** (*bool, optional*) – Disable clipping rectangle for every individual columns.
- **pad_outerX** (*bool, optional*) – Default if `BordersOuterV` is on. Enable outer-most padding. Generally desirable if you have headers.
- **no_pad_outerX** (*bool, optional*) – Default if `BordersOuterV` is off. Disable outer-most padding.
- **no_pad_innerX** (*bool, optional*) – Disable inner padding between columns (double inner padding if `BordersOuterV` is on, single inner padding if `BordersOuterV` is off).
- **scrollX** (*bool, optional*) – Enable horizontal scrolling. Require ‘`outer_size`’ parameter of `BeginTable()` to specify the container size. Changes default sizing policy. Because this create a child window, `ScrollY` is currently generally recommended when using `ScrollX`.
- **scrollY** (*bool, optional*) – Enable vertical scrolling.
- **no_saved_settings** (*bool, optional*) – Never load/save settings in `.ini` file.

- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Yields Union[int, str]

`dearpygui.dearpygui.table_cell(**kwargs)`

Adds a table.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **height** (*int, optional*) – Height of the item.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **filter_key** (*str, optional*) – Used by filter widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Yields Union[int, str]

`dearpygui.dearpygui.table_row(**kwargs)`

Adds a table row.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **height** (*int, optional*) – Height of the item.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **show** (*bool, optional*) – Attempt to render widget.
- **filter_key** (*str, optional*) – Used by filter widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Yields Union[int, str]

`dearpygui.dearpygui.template_registry(**kwargs)`

Adds a template registry.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **id** (*Union[int, str], optional*) – (deprecated)

Yields Union[int, str]

`dearpygui.dearpygui.texture_registry(**kwargs)`

Adds a dynamic texture.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)

Yields Union[int, str]

`dearpygui.dearpygui.theme(**kwargs)`

Adds a theme.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **id** (*Union[int, str], optional*) – (deprecated)
- **default_theme** (*bool, optional*) – (deprecated)

Yields Union[int, str]

`dearpygui.dearpygui.theme_component(item_type=0, **kwargs)`

Adds a theme component.

Parameters

- **item_type** (*int, optional*) –

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **enabled_state** (*bool, optional*) –
- **id** (*Union[int, str], optional*) – (deprecated)

Yields Union[int, str]

`dearpygui.dearpygui.toggle_viewport_fullscreen()`

Toggle viewport fullscreen mode..

Args: :returns: None

`dearpygui.dearpygui.tooltip(parent, **kwargs)`

Adds a tooltip window.

Parameters

- **parent** (*Union[int, str]*) –
- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **show** (*bool, optional*) – Attempt to render widget.
- **delay** (*float, optional*) – Activation delay: time, in seconds, during which the mouse should stay still in order to display the tooltip. May be zero for instant activation.
- **hide_on_activity** (*bool, optional*) – Hide the tooltip if the user has moved the mouse. If False, the tooltip will follow mouse pointer.
- **id** (*Union[int, str], optional*) – (deprecated)

Yields Union[int, str]

`dearpygui.dearpygui.top_container_stack()`

Returns the item on the top of the container stack.

Args: :returns: Union[int, str]

`dearpygui.dearpygui.track_item(item: Union[int, str])`

Track item in scroll region.

Parameters

- **item** – Item the callback will be applied to.
- **callback** – Callback to be applied.

Returns None

`dearpygui.dearpygui.tree_node(**kwargs)`

Adds a tree node to add items to.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **before** (*Union[int, str], optional*) – This item will be displayed before the specified item in the parent.
- **payload_type** (*str, optional*) – Sender string type must be the same as the target for the target to run the payload_callback.
- **drag_callback** (*Callable, optional*) – Registers a drag callback for drag and drop.
- **drop_callback** (*Callable, optional*) – Registers a drop callback for drag and drop.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **filter_key** (*str, optional*) – Used by filter widget.
- **tracked** (*bool, optional*) – Scroll tracking
- **track_offset** (*float, optional*) – 0.0f:top, 0.5f:center, 1.0f:bottom
- **default_open** (*bool, optional*) – Sets the tree node open by default.
- **open_on_double_click** (*bool, optional*) – Need double-click to open node.
- **open_on_arrow** (*bool, optional*) – Only open when clicking on the arrow part.
- **leaf** (*bool, optional*) – No collapsing, no arrow (use as a convenience for leaf nodes).
- **bullet** (*bool, optional*) – Display a bullet instead of arrow.
- **selectable** (*bool, optional*) – Makes the tree selectable.
- **span_text_width** (*bool, optional*) – Makes hitbox and highlight only cover the label.
- **span_full_width** (*bool, optional*) – Extend hit box to the left-most and right-most edges (cover the indent area).
- **catch_nav_left** (*bool, optional*) – Keyboard navigation: left arrow within this node’s children, if unhandled, moves focus to this node. When setting it to True on a node, better set it on all children nodes in the subtree as well, otherwise it might give unexpected navigation jumps.

- **lines** (*int*, *optional*) – Experimental. Draw lines connecting `tree_node` hierarchy. One of `dpg.mvTreeLines` constants. To work correctly, must be configured the same way in every tree node.
- **id** (*Union[int, str]*, *optional*) – (deprecated)
- **delay_search** (*bool*, *optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Yields `Union[int, str]`

`dearpygui.dearpygui.unhighlight_table_cell(table, row, column)`

Unhighlight specified table cell.

Parameters

- **table** (*Union[int, str]*) –
- **row** (*int*) –
- **column** (*int*) –

Returns `None`

`dearpygui.dearpygui.unhighlight_table_column(table, column)`

Unhighlight specified table column.

Parameters

- **table** (*Union[int, str]*) –
- **column** (*int*) –

Returns `None`

`dearpygui.dearpygui.unhighlight_table_row(table, row)`

Unhighlight specified table row.

Parameters

- **table** (*Union[int, str]*) –
- **row** (*int*) –

Returns `None`

`dearpygui.dearpygui.unlock_mutex()`

Unlocks render thread mutex

Args: :returns: `None`

`dearpygui.dearpygui.unset_table_row_color(table, row)`

Remove user set table row color.

Parameters

- **table** (*Union[int, str]*) –
- **row** (*int*) –

Returns `None`

`dearpygui.dearpygui.unstage(item)`

Unstages an item.

Parameters **item** (*Union[int, str]*) –

Returns `None`

`dearpygui.dearpygui.untrack_item(item: Union[int, str])`

Track item in scroll region.

Parameters

- **item** – Item the callback will be applied to.
- **callback** – Callback to be applied.

Returns None

`dearpygui.dearpygui.value_registry(**kwargs)`

Adds a value registry.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **id** (*Union[int, str], optional*) – (deprecated)

Yields Union[int, str]

`dearpygui.dearpygui.viewport_drawlist(**kwargs)`

A container that is used to present draw items or layers directly to the viewport. By default this will draw to the back of the viewport. Layers and draw items should be added to this widget as children.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item.If label is unused this will be the label.
- **show** (*bool, optional*) – Attempt to render widget.
- **filter_key** (*str, optional*) – Used by filter widget.
- **front** (*bool, optional*) – Draws to the front of the view port instead of the back.
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated)This was used as an optimization hint but is not relevant anymore.

Yields Union[int, str]

`dearpygui.dearpygui.viewport_menu_bar(**kwargs)`

Adds a menubar to the viewport.

Parameters

- **label** (*str, optional*) – Overrides ‘name’ as label.
- **user_data** (*Any, optional*) – User data for callbacks

- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **parent** (*Union[int, str], optional*) – Parent to add this item to. (runtime adding)
- **show** (*bool, optional*) – Attempt to render widget.
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Yields Union[int, str]

`dearpygui.dearpygui.window(**kwargs)`

Creates a new window for following items to be added to.

Parameters

- **label** (*str, optional*) – Overrides 'name' as label.
- **user_data** (*Any, optional*) – User data for callbacks
- **use_internal_label** (*bool, optional*) – Use generated internal label instead of user specified (appends ### uuid).
- **tag** (*Union[int, str], optional*) – Unique id used to programmatically refer to the item. If label is unused this will be the label.
- **width** (*int, optional*) – Width of the item.
- **height** (*int, optional*) – Height of the item.
- **indent** (*int, optional*) – Offsets the widget to the right the specified number multiplied by the indent style.
- **show** (*bool, optional*) – Attempt to render widget.
- **pos** (*Union[List[int], Tuple[int, ...]], optional*) – Places the item relative to window coordinates, [0,0] is top left.
- **min_size** (*Union[List[int], Tuple[int, ...]], optional*) – Minimum window size.
- **max_size** (*Union[List[int], Tuple[int, ...]], optional*) – Maximum window size.
- **menubar** (*bool, optional*) – Shows or hides the menubar.
- **collapsed** (*bool, optional*) – Collapse the window.
- **autosize** (*bool, optional*) – Autosized the window to fit it's items.
- **no_resize** (*bool, optional*) – Allows for the window size to be changed or fixed.
- **unsaved_document** (*bool, optional*) – Show a special marker if the document is not saved.
- **no_title_bar** (*bool, optional*) – Title name for the title bar of the window.
- **no_move** (*bool, optional*) – Allows for the window's position to be changed or fixed.

- **no_scrollbar** (*bool, optional*) – Disable scrollbars. (window can still scroll with mouse or programmatically)
- **no_collapse** (*bool, optional*) – Disable user collapsing window by double-clicking on it.
- **horizontal_scrollbar** (*bool, optional*) – Allow horizontal scrollbar to appear. (off by default)
- **no_focus_on_appearing** (*bool, optional*) – Disable taking focus when transitioning from hidden to visible state.
- **no_bring_to_front_on_focus** (*bool, optional*) – Disable bringing window to front when taking focus. (e.g. clicking on it or programmatically giving it focus)
- **no_close** (*bool, optional*) – Disable user closing the window by removing the close button.
- **no_background** (*bool, optional*) – Sets Background and border alpha to transparent.
- **modal** (*bool, optional*) – Fills area behind window according to the theme and disables user ability to interact with anything except the window.
- **popup** (*bool, optional*) – Fills area behind window according to the theme, removes title bar, collapse and close. Window can be closed by selecting area in the background behind the window.
- **no_saved_settings** (*bool, optional*) – Never load/save settings in .ini file.
- **no_open_over_existing_popup** (*bool, optional*) – Don't open if there's already a popup
- **no_scroll_with_mouse** (*bool, optional*) – Disable user vertically scrolling with mouse wheel.
- **no_docking** (*bool, optional*) – Disable docking of this window
- **copy_contents_shortcut** (*bool, optional*) – Experimental. If True, window contents can be copied to clipboard by pressing Ctrl+C. Might be useful for message boxes.
- **on_close** (*Callable, optional*) – Callback ran when window is closed.
- **id** (*Union[int, str], optional*) – (deprecated)
- **delay_search** (*bool, optional*) – (deprecated) This was used as an optimization hint but is not relevant anymore.

Yields Union[int, str]

- *Showcase*
- *Video Tutorials*

4.1 Showcase

The following apps have been developed with Dear PyGui by various developers.

4.1.1 Tetris

Tetris is a remake of the original Tetris tile-matching game as adopted by IBM PC. Even though Dear PyGui is not a game engine, it can easily handle graphical animations such as these.

The source code is available in the [Tetris Github repository](#).

4.1.2 Snake

Snake is a simple game with customisable settings for changing the speed and colours and fixing the snake length. Entirely made with Dear PyGui.

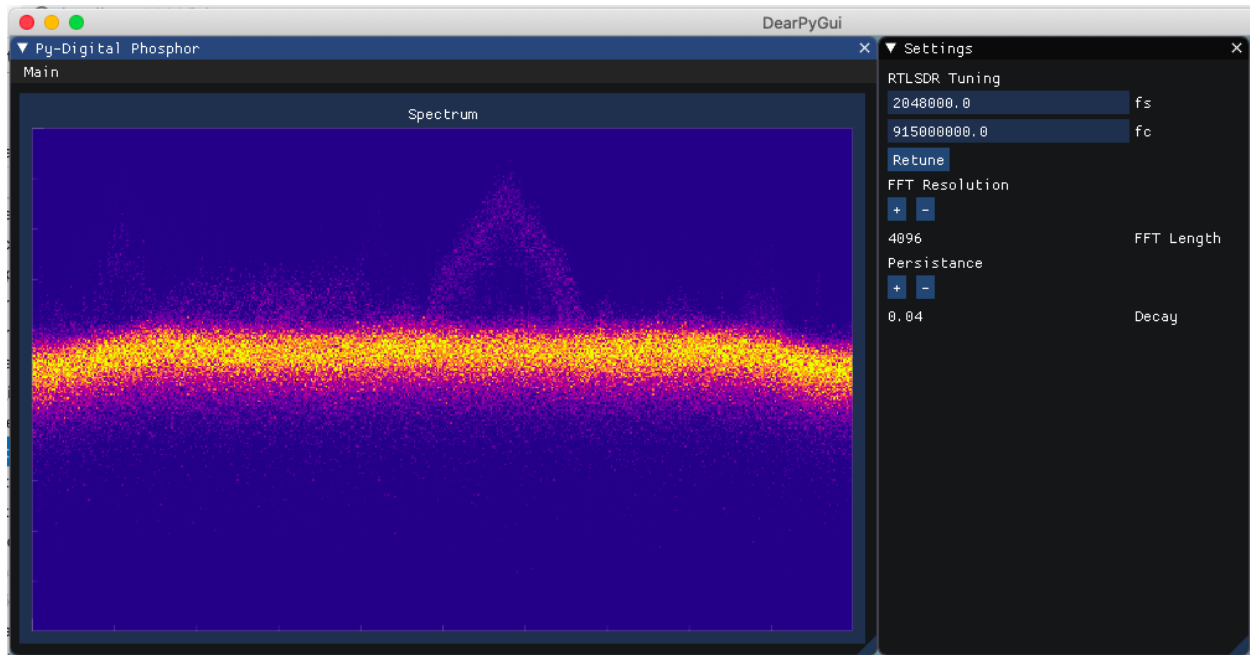
The source code is available in the [Snake Github repository](#).

4.1.3 Digital Phosphor Display with RTLSDR

[Digital Phosphor Display Video](#)

This video demonstrates an Intensity graded FFT or Python Digital Phosphor Display by Thomas Schucker. The accompanying [blog post](#) shows how to create a this dynamic graph.

The source code is available in the [Digital Phosphor Github repository](#).



4.2 Video Tutorials

The following video tutorials explain various aspects of Dear PyGui, which are outlined per video below. The first section contains all videos by the core developers. The second section lists a great video by the community.

View all [video tutorials on Dear PyGui](#) by the core developers on YouTube.

4.2.1 Introduction to Dear PyGui

Dear PyGui is an extended Python wrapper around Dear ImGui. Dear ImGui is an Immediate Mode GUI (ImGui) for real-time applications

While Dear ImGui uses the fast Immediate Mode in the background, it simulates a traditional retained mode GUI to the Python developer, which makes it easy to work with.

Requirements for installation: Python 3.6 and up (64-bit only)

IDE: Setting up PyCharm, only pip install DPG is required

How to run the built-in demo.

Changing the default font type (OTF or TTF font)

Creating a basic window

Using a primary window so that the window matches the area of the viewport

4.2.2 Basics of callbacks

Callbacks are functions that are run when any action is taken on a certain widget.

If a widget supports it, there will be a keyword called `callback` and that can be any callable.

When using a callback, the keyword should not end with round brackets, e.g. `()`. The correct way to use a callback is: `add_button("Press me", callback=func_name)`. The function `func_name` is called without `()`.

The callback always transmits two *variables* to the callback function, e.g. `sender` and `data`.

`Sender` is the name of the widget. `Data` is `None` by default, but additional data can be sent using the following: `callback_data='extra data'`.

The tutorial shows the use of a callback for a float slider.

Throughout the tutorial, the use of the built-in logger and documentation is demonstrated.

The callback can be changed during runtime using `set_item_callback`.

4.2.3 ID system and debug tool

The first argument of a widget is the ID.

The label defaults to the ID if no label is provided.

Widgets need unique IDs, but can share the same label. There are two ways to accomplish this.

Method 1: `add_input_float('Float1', label='float1')` and `add_input_float('Float2', label='float1')`

Method 2: `add_input_float('Float1##1')` and `add_input_float('Float1##2')`

ID's are used to retrieve data from widgets.

Start the debug tool by typing `show_debug()`

The debug tool lists all available commands under tab `Commands`

You can execute code at runtime using the debug tool.

Track the float values by `log_debug(get_value('Float1##1'))`

4.2.4 Parent stack system

The parent stack is a collection of all containers in a GUI.

A window is a root item, meaning that it can't have a parent and doesn't need to look at the parent stack. A window is also a container. Because it is a container, a window gets added to the parent stack.

When an item is not a root item, it requires a parent. Every tab bar is added to the parent stack and to a container. A tab bar is a container itself as well. A tab is a child of tab bar, but it is also a container.

When adding a second item of a parent, it is necessary to remove the first item from the parent stack, e.g. `pop` it, so that the second item becomes part of the containing parent and not its sibling.

The `end()` command in the `core` module pops an item off of the parent stack.

A checkbox or (radio) button is part of a container, but not a container itself.

The `simple` module adds the context managers (e.g. `'with window'`). The *with* statements of the context managers automate the application of the `end()` statement, making the code easier to read.

4.2.5 Value Storage System

In many GUI's the widget's value is stored inside the widget.

In Dear PyGui, a key-value pair for each widget is stored in the value storage system. A key-value pair tracks the type of the value and the value itself. A widget's value can be retrieved and changed through the widget (by the user) and by the program.

Every widget has a keyword source, which by default is equal to the widget's name. If you specify the source, the widget will use that key instead to look up and change values in the value storage system. This allows several widgets to manipulate a single value.

If multiple widgets refer to the same keyword, the type and size have to be the same.

Pre-add a value with `add_value` if you are using multiple widgets of different types or sizes on a single key-value pair.

A code example is given to demonstrate the value storage system and its types and sizes.

4.2.6 Widget basics

This tutorial shows how to use a number of widget types. Widget types include `button`, `checkbox`, `label_text`, `input_int`, `drag_int`, `radio_button`, `combo`, `listbox` and `progress_bar` widgets.

The use of the `callback` keyword of a widget is shown. For example, `add_button('Press me', callback=callback_function)`.

The `callback_function` is called whenever that button is pressed. The callback always sends two arguments to the `callback_function`: `sender` and `data`. `Sender` is the name of the widget. The `'data'` argument is often empty unless the widget has data to send or it is specified in the code. Nonetheless, the argument `'data'` is always included.

The use of a number of widget specific keywords are discussed.

It is demonstrated how a progress bar widget can be controlled via a `drag_int` slider using `set_value(...)` and `configure_item(...)`

Many widgets have multi-component versions as well.

More complex use of widgets and multi-component widgets will be shown in future videos.

4.2.7 Tab bar, tabs, and tab button basics

Create a tab bar with the context manager from the `simple` module, e.g. `with tab_bar('tb1') -> with tab('t1') -> add_button('b1')`.

You can add a callback to a `tab_bar` using `with tab_bar('tb1', callback=callback)`.

You can add a button to a `tab_bar` using `add_tab_button('+')`.

Tabs in a tab bar can be made reorderable by using the keyword `reorderable=True` on the `tab_bar`.

4.2.8 Simple Plot & Tooltip

Simple Plots is for plotting simple data. This is not to be confused with the more powerful and complex *Plots*.

Create a basic histogram using `add_simple_plot("Plot 1", value=[1, 4.3, 8, 9, 3], histogram=True)`. There are several keywords to customise the plot.

`add_text("Hover me", tip="A simple tooltip")`. This simple tooltip is only for text. The *Tooltips* is more powerful.

The tooltip widget is a container, i.e. context manager, just like 'with window' and 'with group'. The widget basically acts as another window, so that it can contain any other widget, such as a graph. The example in the video shows how to embed a simple plot in a tooltip in two lines of code.

Note that the user cannot interact with the tooltip widget.

4.2.9 Popups

Popups require a parent. That may change in future versions of Dear PyGui.

A popup is a container, so it has a context manager (with popup:).

Popup is the only widget where the name is not the first argument.

By default, popups are set on the right-click. To change to left-click, add the keyword `mousebutton=mvMouseButtonLeft`.

Popups are a container and can contain any other widget, i.e. plots.

The modal keyword greys everything else out to draw attention to the popup.

To close the modal popup, it is necessary to add a button with a callback `close_popup("popup1")`.

4.2.10 Experimental Windows Docking

The docking feature enables the user to dock windows to each other and the viewport.

The docking feature is not documented yet (as of September 2025).

Setting the *docking* argument to *True* in `configure_app()` will enable experimental docking.

By default, the user needs to hold the shift key to enable docking.

Setting `docking_shift_only = False` enables docking without holding the shift key.

Setting `docking_space = True` enables docking windows to the viewport.

The docking feature is experimental because you cannot programmatically set up the docking positions.

When the feature comes out of experimental, it can also function as a layout tool, but it still requires lots of work to be released as non-experimental.

4.2.11 Smart tables

This is an elaborate tutorial on creating a smart, interactive table.

The table is created using `managed_columns`.

The widgets used in the table are `add_text`, `add_input_text` and `add_input_float`

After creating a working example, the code is refactored into a `SmartTable` class with `header`, `row` and `get_cell_data` methods.

A widget's label can be hidden by using `##` at the beginning of a label's name, e.g. `add_input_text('##input_text_1')` where `input_text_1` is not shown in the GUI.

Using `add_separator()` to change the horizontal spacing of the widgets.

Using the built-in Dear PyGui debugger and logger for solving an coding issue.

4.2.12 Community Videos

[Creating a complete Python app with Dear PyGui](#)

Learn how to create a fully-functional Python app step by step! In this project, we will build a graphic user interface with the brand new Dear PyGui library! We will connect this interface to a Simple SMS Spam Filter, which we've built together in a previous project. We will learn how to display images, text, user input, buttons, and separators, as well as hiding widgets and "click" event callback functions.

4.3 Glossary

- `alias` - A string that takes the place of the regular `int` ID. Aliases can be used anywhere UUID's can be used.
- `item` - Everything in **Dear PyGui** created with a context manager or a `add_` command.
- `root` - An item which has no parent (i.e. window, registries, etc.)
- `window` - A **Dear ImGui** window created with `add_window(...)`.
- `viewport` - The operating system window.

PYTHON MODULE INDEX

d

`dearpygui.dearpygui`, 103

A

- add_2d_histogram_series() (in module *dearpygui.dearpygui*), 103
 add_3d_slider() (in module *dearpygui.dearpygui*), 104
 add_activated_handler() (in module *dearpygui.dearpygui*), 105
 add_active_handler() (in module *dearpygui.dearpygui*), 105
 add_alias() (in module *dearpygui.dearpygui*), 105
 add_area_series() (in module *dearpygui.dearpygui*), 105
 add_axis_tag() (in module *dearpygui.dearpygui*), 106
 add_bar_group_series() (in module *dearpygui.dearpygui*), 106
 add_bar_series() (in module *dearpygui.dearpygui*), 107
 add_bool_value() (in module *dearpygui.dearpygui*), 107
 add_button() (in module *dearpygui.dearpygui*), 108
 add_candle_series() (in module *dearpygui.dearpygui*), 109
 add_char_remap() (in module *dearpygui.dearpygui*), 109
 add_checkbox() (in module *dearpygui.dearpygui*), 110
 add_child() (in module *dearpygui.dearpygui*), 111
 add_child_window() (in module *dearpygui.dearpygui*), 112
 add_clicked_handler() (in module *dearpygui.dearpygui*), 113
 add_clipper() (in module *dearpygui.dearpygui*), 113
 add_collapsing_header() (in module *dearpygui.dearpygui*), 114
 add_color_button() (in module *dearpygui.dearpygui*), 115
 add_color_edit() (in module *dearpygui.dearpygui*), 116
 add_color_picker() (in module *dearpygui.dearpygui*), 117
 add_color_value() (in module *dearpygui.dearpygui*), 118
 add_colormap() (in module *dearpygui.dearpygui*), 119
 add_colormap_button() (in module *dearpygui.dearpygui*), 119
 add_colormap_registry() (in module *dearpygui.dearpygui*), 120
 add_colormap_scale() (in module *dearpygui.dearpygui*), 120
 add_colormap_slider() (in module *dearpygui.dearpygui*), 121
 add_combo() (in module *dearpygui.dearpygui*), 122
 add_custom_series() (in module *dearpygui.dearpygui*), 123
 add_date_picker() (in module *dearpygui.dearpygui*), 124
 add_deactivated_after_edit_handler() (in module *dearpygui.dearpygui*), 124
 add_deactivated_handler() (in module *dearpygui.dearpygui*), 124
 add_digital_series() (in module *dearpygui.dearpygui*), 124
 add_double4_value() (in module *dearpygui.dearpygui*), 125
 add_double_value() (in module *dearpygui.dearpygui*), 125
 add_drag_double() (in module *dearpygui.dearpygui*), 126
 add_drag_doublex() (in module *dearpygui.dearpygui*), 127
 add_drag_float() (in module *dearpygui.dearpygui*), 128
 add_drag_floatx() (in module *dearpygui.dearpygui*), 129
 add_drag_int() (in module *dearpygui.dearpygui*), 130
 add_drag_intx() (in module *dearpygui.dearpygui*), 131
 add_drag_line() (in module *dearpygui.dearpygui*), 132
 add_drag_payload() (in module *dearpygui.dearpygui*), 133
 add_drag_point() (in module *dearpygui.dearpygui*), 133
 add_drag_rect() (in module *dearpygui.dearpygui*), 134

add_draw_layer() (in module *dearpygui.dearpygui*), 135
 add_draw_node() (in module *dearpygui.dearpygui*), 135
 add_drawlist() (in module *dearpygui.dearpygui*), 136
 add_dummy() (in module *dearpygui.dearpygui*), 136
 add_dynamic_texture() (in module *dearpygui.dearpygui*), 137
 add_edited_handler() (in module *dearpygui.dearpygui*), 137
 add_error_series() (in module *dearpygui.dearpygui*), 137
 add_file_dialog() (in module *dearpygui.dearpygui*), 138
 add_file_extension() (in module *dearpygui.dearpygui*), 139
 add_filter_set() (in module *dearpygui.dearpygui*), 139
 add_float4_value() (in module *dearpygui.dearpygui*), 140
 add_float_value() (in module *dearpygui.dearpygui*), 140
 add_float_vect_value() (in module *dearpygui.dearpygui*), 140
 add_focus_handler() (in module *dearpygui.dearpygui*), 141
 add_font() (in module *dearpygui.dearpygui*), 141
 add_font_chars() (in module *dearpygui.dearpygui*), 141
 add_font_range() (in module *dearpygui.dearpygui*), 142
 add_font_range_hint() (in module *dearpygui.dearpygui*), 142
 add_font_registry() (in module *dearpygui.dearpygui*), 142
 add_group() (in module *dearpygui.dearpygui*), 143
 add_handler_registry() (in module *dearpygui.dearpygui*), 143
 add_heat_series() (in module *dearpygui.dearpygui*), 143
 add_histogram_series() (in module *dearpygui.dearpygui*), 144
 add_hline_series() (in module *dearpygui.dearpygui*), 145
 add_hover_handler() (in module *dearpygui.dearpygui*), 145
 add_image() (in module *dearpygui.dearpygui*), 146
 add_image_button() (in module *dearpygui.dearpygui*), 147
 add_image_series() (in module *dearpygui.dearpygui*), 148
 add_inf_line_series() (in module *dearpygui.dearpygui*), 148
 add_input_double() (in module *dearpygui.dearpygui*), 149
 add_input_doublex() (in module *dearpygui.dearpygui*), 150
 add_input_float() (in module *dearpygui.dearpygui*), 151
 add_input_floatx() (in module *dearpygui.dearpygui*), 153
 add_input_int() (in module *dearpygui.dearpygui*), 154
 add_input_intx() (in module *dearpygui.dearpygui*), 155
 add_input_text() (in module *dearpygui.dearpygui*), 156
 add_int4_value() (in module *dearpygui.dearpygui*), 158
 add_int_value() (in module *dearpygui.dearpygui*), 158
 add_item_activated_handler() (in module *dearpygui.dearpygui*), 158
 add_item_active_handler() (in module *dearpygui.dearpygui*), 159
 add_item_clicked_handler() (in module *dearpygui.dearpygui*), 159
 add_item_deactivated_after_edit_handler() (in module *dearpygui.dearpygui*), 159
 add_item_deactivated_handler() (in module *dearpygui.dearpygui*), 160
 add_item_double_clicked_handler() (in module *dearpygui.dearpygui*), 160
 add_item_edited_handler() (in module *dearpygui.dearpygui*), 161
 add_item_focus_handler() (in module *dearpygui.dearpygui*), 161
 add_item_handler_registry() (in module *dearpygui.dearpygui*), 161
 add_item_hover_handler() (in module *dearpygui.dearpygui*), 162
 add_item_resize_handler() (in module *dearpygui.dearpygui*), 162
 add_item_scroll_handler() (in module *dearpygui.dearpygui*), 162
 add_item_toggled_open_handler() (in module *dearpygui.dearpygui*), 163
 add_item_visible_handler() (in module *dearpygui.dearpygui*), 163
 add_key_down_handler() (in module *dearpygui.dearpygui*), 164
 add_key_press_handler() (in module *dearpygui.dearpygui*), 164
 add_key_release_handler() (in module *dearpygui.dearpygui*), 164
 add_knob_float() (in module *dearpygui.dearpygui*), 165
 add_line_series() (in module *dearpygui.dearpygui*),

- 166
- `add_listbox()` (in module `dearpygui.dearpygui`), 166
- `add_loading_indicator()` (in module `dearpygui.dearpygui`), 167
- `add_menu()` (in module `dearpygui.dearpygui`), 168
- `add_menu_bar()` (in module `dearpygui.dearpygui`), 169
- `add_menu_item()` (in module `dearpygui.dearpygui`), 169
- `add_mouse_click_handler()` (in module `dearpygui.dearpygui`), 170
- `add_mouse_double_click_handler()` (in module `dearpygui.dearpygui`), 171
- `add_mouse_down_handler()` (in module `dearpygui.dearpygui`), 171
- `add_mouse_drag_handler()` (in module `dearpygui.dearpygui`), 171
- `add_mouse_move_handler()` (in module `dearpygui.dearpygui`), 172
- `add_mouse_release_handler()` (in module `dearpygui.dearpygui`), 172
- `add_mouse_wheel_handler()` (in module `dearpygui.dearpygui`), 173
- `add_node()` (in module `dearpygui.dearpygui`), 173
- `add_node_attribute()` (in module `dearpygui.dearpygui`), 174
- `add_node_editor()` (in module `dearpygui.dearpygui`), 174
- `add_node_link()` (in module `dearpygui.dearpygui`), 175
- `add_pie_series()` (in module `dearpygui.dearpygui`), 175
- `add_plot()` (in module `dearpygui.dearpygui`), 176
- `add_plot_annotation()` (in module `dearpygui.dearpygui`), 178
- `add_plot_axis()` (in module `dearpygui.dearpygui`), 179
- `add_plot_legend()` (in module `dearpygui.dearpygui`), 180
- `add_progress_bar()` (in module `dearpygui.dearpygui`), 181
- `add_radio_button()` (in module `dearpygui.dearpygui`), 182
- `add_raw_texture()` (in module `dearpygui.dearpygui`), 182
- `add_resize_handler()` (in module `dearpygui.dearpygui`), 183
- `add_same_line()` (in module `dearpygui.dearpygui`), 183
- `add_scatter_series()` (in module `dearpygui.dearpygui`), 183
- `add_selectable()` (in module `dearpygui.dearpygui`), 183
- `add_separator()` (in module `dearpygui.dearpygui`), 184
- `add_series_value()` (in module `dearpygui.dearpygui`), 185
- `add_shade_series()` (in module `dearpygui.dearpygui`), 185
- `add_simple_plot()` (in module `dearpygui.dearpygui`), 186
- `add_slider_double()` (in module `dearpygui.dearpygui`), 187
- `add_slider_doublex()` (in module `dearpygui.dearpygui`), 188
- `add_slider_float()` (in module `dearpygui.dearpygui`), 189
- `add_slider_floatx()` (in module `dearpygui.dearpygui`), 190
- `add_slider_int()` (in module `dearpygui.dearpygui`), 191
- `add_slider_intx()` (in module `dearpygui.dearpygui`), 192
- `add_spacer()` (in module `dearpygui.dearpygui`), 193
- `add_spacing()` (in module `dearpygui.dearpygui`), 194
- `add_stage()` (in module `dearpygui.dearpygui`), 194
- `add_staging_container()` (in module `dearpygui.dearpygui`), 194
- `add_stair_series()` (in module `dearpygui.dearpygui`), 194
- `add_static_texture()` (in module `dearpygui.dearpygui`), 195
- `add_stem_series()` (in module `dearpygui.dearpygui`), 195
- `add_string_value()` (in module `dearpygui.dearpygui`), 196
- `add_subplots()` (in module `dearpygui.dearpygui`), 196
- `add_synced_tables()` (in module `dearpygui.dearpygui`), 197
- `add_tab()` (in module `dearpygui.dearpygui`), 198
- `add_tab_bar()` (in module `dearpygui.dearpygui`), 199
- `add_tab_button()` (in module `dearpygui.dearpygui`), 200
- `add_table()` (in module `dearpygui.dearpygui`), 200
- `add_table_cell()` (in module `dearpygui.dearpygui`), 202
- `add_table_column()` (in module `dearpygui.dearpygui`), 203
- `add_table_next_column()` (in module `dearpygui.dearpygui`), 204
- `add_table_row()` (in module `dearpygui.dearpygui`), 204
- `add_template_registry()` (in module `dearpygui.dearpygui`), 205
- `add_text()` (in module `dearpygui.dearpygui`), 205
- `add_text_point()` (in module `dearpygui.dearpygui`), 206
- `add_texture_registry()` (in module `dearpygui.dearpygui`), 206

- add_theme() (in module *dearpygui.dearpygui*), 207
 add_theme_color() (in module *dearpygui.dearpygui*), 207
 add_theme_component() (in module *dearpygui.dearpygui*), 207
 add_theme_style() (in module *dearpygui.dearpygui*), 208
 add_time_picker() (in module *dearpygui.dearpygui*), 208
 add_toggled_open_handler() (in module *dearpygui.dearpygui*), 209
 add_tooltip() (in module *dearpygui.dearpygui*), 209
 add_tree_node() (in module *dearpygui.dearpygui*), 209
 add_value_registry() (in module *dearpygui.dearpygui*), 211
 add_viewport_drawlist() (in module *dearpygui.dearpygui*), 211
 add_viewport_menu_bar() (in module *dearpygui.dearpygui*), 211
 add_visible_handler() (in module *dearpygui.dearpygui*), 212
 add_vline_series() (in module *dearpygui.dearpygui*), 212
 add_window() (in module *dearpygui.dearpygui*), 212
 apply_transform() (in module *dearpygui.dearpygui*), 214
- ## B
- bind_colormap() (in module *dearpygui.dearpygui*), 214
 bind_font() (in module *dearpygui.dearpygui*), 214
 bind_item_font() (in module *dearpygui.dearpygui*), 214
 bind_item_handler_registry() (in module *dearpygui.dearpygui*), 214
 bind_item_theme() (in module *dearpygui.dearpygui*), 214
 bind_theme() (in module *dearpygui.dearpygui*), 215
- ## C
- capture_next_item() (in module *dearpygui.dearpygui*), 215
 child() (in module *dearpygui.dearpygui*), 215
 child_window() (in module *dearpygui.dearpygui*), 216
 cleanup_dearpygui() (in module *dearpygui.dearpygui*), 217
 clear_selected_links() (in module *dearpygui.dearpygui*), 217
 clear_selected_nodes() (in module *dearpygui.dearpygui*), 217
 clipper() (in module *dearpygui.dearpygui*), 217
 collapsing_header() (in module *dearpygui.dearpygui*), 218
- colormap_registry() (in module *dearpygui.dearpygui*), 219
 configure_app() (in module *dearpygui.dearpygui*), 219
 configure_item() (in module *dearpygui.dearpygui*), 219
 configure_viewport() (in module *dearpygui.dearpygui*), 219
 create_context() (in module *dearpygui.dearpygui*), 219
 create_fps_matrix() (in module *dearpygui.dearpygui*), 219
 create_lookat_matrix() (in module *dearpygui.dearpygui*), 220
 create_orthographic_matrix() (in module *dearpygui.dearpygui*), 220
 create_perspective_matrix() (in module *dearpygui.dearpygui*), 220
 create_rotation_matrix() (in module *dearpygui.dearpygui*), 220
 create_scale_matrix() (in module *dearpygui.dearpygui*), 220
 create_translation_matrix() (in module *dearpygui.dearpygui*), 221
 create_viewport() (in module *dearpygui.dearpygui*), 221
 custom_series() (in module *dearpygui.dearpygui*), 221
- ## D
- dearpygui.dearpygui module, 103
 delete_item() (in module *dearpygui.dearpygui*), 222
 deprecated() (in module *dearpygui.dearpygui*), 222
 destroy_context() (in module *dearpygui.dearpygui*), 222
 disable_item() (in module *dearpygui.dearpygui*), 222
 does_alias_exist() (in module *dearpygui.dearpygui*), 222
 does_item_exist() (in module *dearpygui.dearpygui*), 223
 drag_payload() (in module *dearpygui.dearpygui*), 223
 draw_arrow() (in module *dearpygui.dearpygui*), 223
 draw_bezier_cubic() (in module *dearpygui.dearpygui*), 224
 draw_bezier_quadratic() (in module *dearpygui.dearpygui*), 224
 draw_circle() (in module *dearpygui.dearpygui*), 225
 draw_ellipse() (in module *dearpygui.dearpygui*), 225
 draw_image() (in module *dearpygui.dearpygui*), 226
 draw_image_quad() (in module *dearpygui.dearpygui*), 226
 draw_layer() (in module *dearpygui.dearpygui*), 227
 draw_line() (in module *dearpygui.dearpygui*), 228

- draw_node() (in module *dearpygui.dearpygui*), 228
- draw_polygon() (in module *dearpygui.dearpygui*), 228
- draw_polyline() (in module *dearpygui.dearpygui*), 229
- draw_quad() (in module *dearpygui.dearpygui*), 229
- draw_rectangle() (in module *dearpygui.dearpygui*), 230
- draw_text() (in module *dearpygui.dearpygui*), 231
- draw_triangle() (in module *dearpygui.dearpygui*), 231
- drawlist() (in module *dearpygui.dearpygui*), 232
- ## E
- empty_container_stack() (in module *dearpygui.dearpygui*), 233
- enable_item() (in module *dearpygui.dearpygui*), 233
- ## F
- file_dialog() (in module *dearpygui.dearpygui*), 233
- filter_set() (in module *dearpygui.dearpygui*), 234
- fit_axis_data() (in module *dearpygui.dearpygui*), 234
- focus_item() (in module *dearpygui.dearpygui*), 234
- font() (in module *dearpygui.dearpygui*), 234
- font_registry() (in module *dearpygui.dearpygui*), 235
- ## G
- generate_uuid() (in module *dearpygui.dearpygui*), 235
- get_active_window() (in module *dearpygui.dearpygui*), 235
- get_alias_id() (in module *dearpygui.dearpygui*), 235
- get_aliases() (in module *dearpygui.dearpygui*), 235
- get_all_items() (in module *dearpygui.dearpygui*), 235
- get_app_configuration() (in module *dearpygui.dearpygui*), 235
- get_available_content_region() (in module *dearpygui.dearpygui*), 235
- get_axis_limits() (in module *dearpygui.dearpygui*), 236
- get_callback_queue() (in module *dearpygui.dearpygui*), 236
- get_clipboard_text() (in module *dearpygui.dearpygui*), 236
- get_colormap_color() (in module *dearpygui.dearpygui*), 236
- get_dearpygui_version() (in module *dearpygui.dearpygui*), 236
- get_delta_time() (in module *dearpygui.dearpygui*), 236
- get_drawing_mouse_pos() (in module *dearpygui.dearpygui*), 236
- get_file_dialog_info() (in module *dearpygui.dearpygui*), 236
- get_focused_item() (in module *dearpygui.dearpygui*), 236
- get_frame_count() (in module *dearpygui.dearpygui*), 236
- get_frame_rate() (in module *dearpygui.dearpygui*), 237
- get_global_font_scale() (in module *dearpygui.dearpygui*), 237
- get_item_alias() (in module *dearpygui.dearpygui*), 237
- get_item_callback() (in module *dearpygui.dearpygui*), 237
- get_item_children() (in module *dearpygui.dearpygui*), 237
- get_item_configuration() (in module *dearpygui.dearpygui*), 237
- get_item_disabled_theme() (in module *dearpygui.dearpygui*), 237
- get_item_drag_callback() (in module *dearpygui.dearpygui*), 237
- get_item_drop_callback() (in module *dearpygui.dearpygui*), 237
- get_item_filter_key() (in module *dearpygui.dearpygui*), 237
- get_item_font() (in module *dearpygui.dearpygui*), 237
- get_item_height() (in module *dearpygui.dearpygui*), 237
- get_item_indent() (in module *dearpygui.dearpygui*), 237
- get_item_info() (in module *dearpygui.dearpygui*), 238
- get_item_label() (in module *dearpygui.dearpygui*), 238
- get_item_parent() (in module *dearpygui.dearpygui*), 238
- get_item_pos() (in module *dearpygui.dearpygui*), 238
- get_item_rect_max() (in module *dearpygui.dearpygui*), 238
- get_item_rect_min() (in module *dearpygui.dearpygui*), 238
- get_item_rect_size() (in module *dearpygui.dearpygui*), 238
- get_item_slot() (in module *dearpygui.dearpygui*), 238
- get_item_source() (in module *dearpygui.dearpygui*), 238
- get_item_state() (in module *dearpygui.dearpygui*), 238
- get_item_theme() (in module *dearpygui.dearpygui*), 238
- get_item_track_offset() (in module *dearpygui.dearpygui*), 238

- dearpygui.dearpygui*), 238
- `get_item_type()` (in module *dearpygui.dearpygui*), 239
- `get_item_types()` (in module *dearpygui.dearpygui*), 239
- `get_item_user_data()` (in module *dearpygui.dearpygui*), 239
- `get_item_width()` (in module *dearpygui.dearpygui*), 239
- `get_major_version()` (in module *dearpygui.dearpygui*), 239
- `get_minor_version()` (in module *dearpygui.dearpygui*), 239
- `get_mouse_drag_delta()` (in module *dearpygui.dearpygui*), 239
- `get_mouse_pos()` (in module *dearpygui.dearpygui*), 239
- `get_platform()` (in module *dearpygui.dearpygui*), 239
- `get_plot_mouse_pos()` (in module *dearpygui.dearpygui*), 239
- `get_plot_query_area()` (in module *dearpygui.dearpygui*), 239
- `get_plot_query_rects()` (in module *dearpygui.dearpygui*), 239
- `get_selected_links()` (in module *dearpygui.dearpygui*), 239
- `get_selected_nodes()` (in module *dearpygui.dearpygui*), 240
- `get_text_size()` (in module *dearpygui.dearpygui*), 240
- `get_total_time()` (in module *dearpygui.dearpygui*), 240
- `get_value()` (in module *dearpygui.dearpygui*), 240
- `get_values()` (in module *dearpygui.dearpygui*), 240
- `get_viewport_clear_color()` (in module *dearpygui.dearpygui*), 240
- `get_viewport_client_height()` (in module *dearpygui.dearpygui*), 240
- `get_viewport_client_width()` (in module *dearpygui.dearpygui*), 240
- `get_viewport_configuration()` (in module *dearpygui.dearpygui*), 240
- `get_viewport_height()` (in module *dearpygui.dearpygui*), 240
- `get_viewport_max_height()` (in module *dearpygui.dearpygui*), 241
- `get_viewport_max_width()` (in module *dearpygui.dearpygui*), 241
- `get_viewport_min_height()` (in module *dearpygui.dearpygui*), 241
- `get_viewport_min_width()` (in module *dearpygui.dearpygui*), 241
- `get_viewport_pos()` (in module *dearpygui.dearpygui*), 241
- `get_viewport_title()` (in module *dearpygui.dearpygui*), 241
- `get_viewport_width()` (in module *dearpygui.dearpygui*), 241
- `get_windows()` (in module *dearpygui.dearpygui*), 241
- `get_x_scroll()` (in module *dearpygui.dearpygui*), 241
- `get_x_scroll_max()` (in module *dearpygui.dearpygui*), 241
- `get_y_scroll()` (in module *dearpygui.dearpygui*), 241
- `get_y_scroll_max()` (in module *dearpygui.dearpygui*), 241
- `group()` (in module *dearpygui.dearpygui*), 242
- ## H
- `handler_registry()` (in module *dearpygui.dearpygui*), 242
- `hide_item()` (in module *dearpygui.dearpygui*), 242
- `highlight_table_cell()` (in module *dearpygui.dearpygui*), 242
- `highlight_table_column()` (in module *dearpygui.dearpygui*), 243
- `highlight_table_row()` (in module *dearpygui.dearpygui*), 243
- ## I
- `is_dearpygui_running()` (in module *dearpygui.dearpygui*), 243
- `is_item_activated()` (in module *dearpygui.dearpygui*), 243
- `is_item_active()` (in module *dearpygui.dearpygui*), 243
- `is_item_clicked()` (in module *dearpygui.dearpygui*), 243
- `is_item_container()` (in module *dearpygui.dearpygui*), 243
- `is_item_deactivated()` (in module *dearpygui.dearpygui*), 243
- `is_item_deactivated_after_edit()` (in module *dearpygui.dearpygui*), 244
- `is_item_edited()` (in module *dearpygui.dearpygui*), 244
- `is_item_enabled()` (in module *dearpygui.dearpygui*), 244
- `is_item_focused()` (in module *dearpygui.dearpygui*), 244
- `is_item_hovered()` (in module *dearpygui.dearpygui*), 244
- `is_item_left_clicked()` (in module *dearpygui.dearpygui*), 244
- `is_item_middle_clicked()` (in module *dearpygui.dearpygui*), 244
- `is_item_ok()` (in module *dearpygui.dearpygui*), 244
- `is_item_right_clicked()` (in module *dearpygui.dearpygui*), 244

- [is_item_search_delayed\(\)](#) (in module `dearpygui.dearpygui`), 244
[is_item_shown\(\)](#) (in module `dearpygui.dearpygui`), 244
[is_item_toggled_open\(\)](#) (in module `dearpygui.dearpygui`), 244
[is_item_tracked\(\)](#) (in module `dearpygui.dearpygui`), 244
[is_item_visible\(\)](#) (in module `dearpygui.dearpygui`), 244
[is_key_down\(\)](#) (in module `dearpygui.dearpygui`), 245
[is_key_pressed\(\)](#) (in module `dearpygui.dearpygui`), 245
[is_key_released\(\)](#) (in module `dearpygui.dearpygui`), 245
[is_mouse_button_clicked\(\)](#) (in module `dearpygui.dearpygui`), 245
[is_mouse_button_double_clicked\(\)](#) (in module `dearpygui.dearpygui`), 245
[is_mouse_button_down\(\)](#) (in module `dearpygui.dearpygui`), 245
[is_mouse_button_dragging\(\)](#) (in module `dearpygui.dearpygui`), 245
[is_mouse_button_released\(\)](#) (in module `dearpygui.dearpygui`), 245
[is_plot_queried\(\)](#) (in module `dearpygui.dearpygui`), 245
[is_table_cell_highlighted\(\)](#) (in module `dearpygui.dearpygui`), 246
[is_table_column_highlighted\(\)](#) (in module `dearpygui.dearpygui`), 246
[is_table_row_highlighted\(\)](#) (in module `dearpygui.dearpygui`), 246
[is_viewport_always_top\(\)](#) (in module `dearpygui.dearpygui`), 246
[is_viewport_created\(\)](#) (in module `dearpygui.dearpygui`), 246
[is_viewport_decorated\(\)](#) (in module `dearpygui.dearpygui`), 246
[is_viewport_ok\(\)](#) (in module `dearpygui.dearpygui`), 246
[is_viewport_resizable\(\)](#) (in module `dearpygui.dearpygui`), 246
[is_viewport_vsync_on\(\)](#) (in module `dearpygui.dearpygui`), 246
[item_handler_registry\(\)](#) (in module `dearpygui.dearpygui`), 246
- L**
- [last_container\(\)](#) (in module `dearpygui.dearpygui`), 247
[last_item\(\)](#) (in module `dearpygui.dearpygui`), 247
[last_root\(\)](#) (in module `dearpygui.dearpygui`), 247
[load_image\(\)](#) (in module `dearpygui.dearpygui`), 247
[load_init_file\(\)](#) (in module `dearpygui.dearpygui`), 247
[lock_mutex\(\)](#) (in module `dearpygui.dearpygui`), 247
- M**
- [maximize_viewport\(\)](#) (in module `dearpygui.dearpygui`), 247
[menu\(\)](#) (in module `dearpygui.dearpygui`), 247
[menu_bar\(\)](#) (in module `dearpygui.dearpygui`), 248
[minimize_viewport\(\)](#) (in module `dearpygui.dearpygui`), 249
 module
 [dearpygui.dearpygui](#), 103
[move_item\(\)](#) (in module `dearpygui.dearpygui`), 249
[move_item_down\(\)](#) (in module `dearpygui.dearpygui`), 249
[move_item_up\(\)](#) (in module `dearpygui.dearpygui`), 249
[mutex\(\)](#) (in module `dearpygui.dearpygui`), 249
- N**
- [node\(\)](#) (in module `dearpygui.dearpygui`), 249
[node_attribute\(\)](#) (in module `dearpygui.dearpygui`), 250
[node_editor\(\)](#) (in module `dearpygui.dearpygui`), 250
- O**
- [output_frame_buffer\(\)](#) (in module `dearpygui.dearpygui`), 251
- P**
- [plot\(\)](#) (in module `dearpygui.dearpygui`), 251
[plot_axis\(\)](#) (in module `dearpygui.dearpygui`), 253
[pop_container_stack\(\)](#) (in module `dearpygui.dearpygui`), 255
[popup\(\)](#) (in module `dearpygui.dearpygui`), 255
[push_container_stack\(\)](#) (in module `dearpygui.dearpygui`), 255
- R**
- [remove_alias\(\)](#) (in module `dearpygui.dearpygui`), 255
[render_dearpygui_frame\(\)](#) (in module `dearpygui.dearpygui`), 255
[reorder_items\(\)](#) (in module `dearpygui.dearpygui`), 255
[reset_axis_limits_constraints\(\)](#) (in module `dearpygui.dearpygui`), 256
[reset_axis_ticks\(\)](#) (in module `dearpygui.dearpygui`), 256
[reset_axis_zoom_constraints\(\)](#) (in module `dearpygui.dearpygui`), 256
[reset_default_theme\(\)](#) (in module `dearpygui.dearpygui`), 256
[reset_pos\(\)](#) (in module `dearpygui.dearpygui`), 256

- run_callbacks() (in module *dearpygui.dearpygui*), 256
- ## S
- sample_colormap() (in module *dearpygui.dearpygui*), 256
- save_image() (in module *dearpygui.dearpygui*), 256
- save_init_file() (in module *dearpygui.dearpygui*), 257
- set_axis_limits() (in module *dearpygui.dearpygui*), 257
- set_axis_limits_auto() (in module *dearpygui.dearpygui*), 257
- set_axis_limits_constraints() (in module *dearpygui.dearpygui*), 257
- set_axis_ticks() (in module *dearpygui.dearpygui*), 257
- set_axis_zoom_constraints() (in module *dearpygui.dearpygui*), 257
- set_clip_space() (in module *dearpygui.dearpygui*), 258
- set_clipboard_text() (in module *dearpygui.dearpygui*), 258
- set_colormap() (in module *dearpygui.dearpygui*), 258
- set_exit_callback() (in module *dearpygui.dearpygui*), 258
- set_frame_callback() (in module *dearpygui.dearpygui*), 258
- set_global_font_scale() (in module *dearpygui.dearpygui*), 258
- set_init_file() (in module *dearpygui.dearpygui*), 258
- set_item_alias() (in module *dearpygui.dearpygui*), 258
- set_item_callback() (in module *dearpygui.dearpygui*), 259
- set_item_children() (in module *dearpygui.dearpygui*), 259
- set_item_drag_callback() (in module *dearpygui.dearpygui*), 259
- set_item_drop_callback() (in module *dearpygui.dearpygui*), 259
- set_item_font() (in module *dearpygui.dearpygui*), 259
- set_item_height() (in module *dearpygui.dearpygui*), 259
- set_item_indent() (in module *dearpygui.dearpygui*), 260
- set_item_label() (in module *dearpygui.dearpygui*), 260
- set_item_payload_type() (in module *dearpygui.dearpygui*), 260
- set_item_pos() (in module *dearpygui.dearpygui*), 260
- set_item_source() (in module *dearpygui.dearpygui*), 260
- set_item_theme() (in module *dearpygui.dearpygui*), 260
- set_item_track_offset() (in module *dearpygui.dearpygui*), 260
- set_item_type_disabled_theme() (in module *dearpygui.dearpygui*), 261
- set_item_type_theme() (in module *dearpygui.dearpygui*), 261
- set_item_user_data() (in module *dearpygui.dearpygui*), 261
- set_item_width() (in module *dearpygui.dearpygui*), 261
- set_primary_window() (in module *dearpygui.dearpygui*), 261
- set_staging_mode() (in module *dearpygui.dearpygui*), 261
- set_start_callback() (in module *dearpygui.dearpygui*), 261
- set_table_row_color() (in module *dearpygui.dearpygui*), 261
- set_value() (in module *dearpygui.dearpygui*), 261
- set_viewport_always_top() (in module *dearpygui.dearpygui*), 262
- set_viewport_clear_color() (in module *dearpygui.dearpygui*), 262
- set_viewport_decorated() (in module *dearpygui.dearpygui*), 262
- set_viewport_height() (in module *dearpygui.dearpygui*), 262
- set_viewport_large_icon() (in module *dearpygui.dearpygui*), 262
- set_viewport_max_height() (in module *dearpygui.dearpygui*), 262
- set_viewport_max_width() (in module *dearpygui.dearpygui*), 262
- set_viewport_min_height() (in module *dearpygui.dearpygui*), 262
- set_viewport_min_width() (in module *dearpygui.dearpygui*), 262
- set_viewport_pos() (in module *dearpygui.dearpygui*), 262
- set_viewport_resizable() (in module *dearpygui.dearpygui*), 262
- set_viewport_resize_callback() (in module *dearpygui.dearpygui*), 262
- set_viewport_small_icon() (in module *dearpygui.dearpygui*), 263
- set_viewport_title() (in module *dearpygui.dearpygui*), 263
- set_viewport_vsync() (in module *dearpygui.dearpygui*), 263
- set_viewport_width() (in module *dearpygui.dearpygui*), 263

dearpygui.dearpygui), 263
 set_x_scroll() (in module *dearpygui.dearpygui*), 263
 set_y_scroll() (in module *dearpygui.dearpygui*), 263
 setup_dearpygui() (in module *dearpygui.dearpygui*),
 263
 setup_registries() (in module *dearpygui.dearpygui*),
 263
 setup_viewport() (in module *dearpygui.dearpygui*),
 263
 show_about() (in module *dearpygui.dearpygui*), 263
 show_debug() (in module *dearpygui.dearpygui*), 264
 show_documentation() (in module
dearpygui.dearpygui), 264
 show_font_manager() (in module
dearpygui.dearpygui), 264
 show_imgui_demo() (in module *dearpygui.dearpygui*),
 264
 show_implot_demo() (in module *dearpygui.dearpygui*),
 264
 show_item() (in module *dearpygui.dearpygui*), 264
 show_item_debug() (in module *dearpygui.dearpygui*),
 264
 show_item_registry() (in module
dearpygui.dearpygui), 264
 show_metrics() (in module *dearpygui.dearpygui*), 264
 show_style_editor() (in module
dearpygui.dearpygui), 264
 show_tool() (in module *dearpygui.dearpygui*), 264
 show_viewport() (in module *dearpygui.dearpygui*),
 264
 split_frame() (in module *dearpygui.dearpygui*), 265
 stage() (in module *dearpygui.dearpygui*), 265
 staging_container() (in module
dearpygui.dearpygui), 265
 start_dearpygui() (in module *dearpygui.dearpygui*),
 265
 stop_dearpygui() (in module *dearpygui.dearpygui*),
 265
 subplots() (in module *dearpygui.dearpygui*), 265
 synced_tables() (in module *dearpygui.dearpygui*),
 267

T

tab() (in module *dearpygui.dearpygui*), 267
 tab_bar() (in module *dearpygui.dearpygui*), 268
 table() (in module *dearpygui.dearpygui*), 269
 table_cell() (in module *dearpygui.dearpygui*), 271
 table_row() (in module *dearpygui.dearpygui*), 271
 template_registry() (in module
dearpygui.dearpygui), 271
 texture_registry() (in module *dearpygui.dearpygui*),
 272
 theme() (in module *dearpygui.dearpygui*), 272

theme_component() (in module *dearpygui.dearpygui*),
 272
 toggle_viewport_fullscreen() (in module
dearpygui.dearpygui), 273
 tooltip() (in module *dearpygui.dearpygui*), 273
 top_container_stack() (in module
dearpygui.dearpygui), 273
 track_item() (in module *dearpygui.dearpygui*), 273
 tree_node() (in module *dearpygui.dearpygui*), 274

U

unhighlight_table_cell() (in module
dearpygui.dearpygui), 275
 unhighlight_table_column() (in module
dearpygui.dearpygui), 275
 unhighlight_table_row() (in module
dearpygui.dearpygui), 275
 unlock_mutex() (in module *dearpygui.dearpygui*), 275
 unset_table_row_color() (in module
dearpygui.dearpygui), 275
 unstage() (in module *dearpygui.dearpygui*), 275
 untrack_item() (in module *dearpygui.dearpygui*), 275

V

value_registry() (in module *dearpygui.dearpygui*),
 276
 viewport_drawlist() (in module
dearpygui.dearpygui), 276
 viewport_menu_bar() (in module
dearpygui.dearpygui), 276

W

window() (in module *dearpygui.dearpygui*), 277