Python DearPyGui 常用控件一

何小有 关注
2020.11.04 20:33:09 字数 1,074 阅读 22

菜单栏

菜单栏是一个 GUI 应用中重要的控件，始终显示在窗口顶部，并具有三个主要部分：

- menu_bar —— 主菜单功能区
- menu —— 下拉菜单或子菜单
- add_menu_item —— 可以运行回调或可以操作的项目

菜单可以根据需要嵌套，而且任何控件都可以添加到菜单中，例如下面栗子中的 “控件列表” 菜单。

```
from dearpygui.core import *
from dearpygui.simple import *

add_additional_font('三极中柔宋.ttf', 18, glyph_ranges='chinese_simplified_common')

def print_me(sender, data):
    log_debug(f"菜单项: {sender}")

show_logger()

with window("Tutorial"):
    with menu_bar("Main Menu Bar"):
        with menu("文件"):
            add_menu_item("保存", callback=print_me)
            add_menu_item("另存为", callback=print_me)

            with menu("设置"):
                add_menu_item("设置项 1", callback=print_me)
                add_menu_item("设置项 2", callback=print_me)

        add_menu_item("帮助", callback=print_me)

        with menu("控件列表"):
            add_checkbox("选择", callback=print_me)
            add_button("点击", callback=print_me)
            add_color_picker4("选择颜色", callback=print_me)

start_dearpygui()
```

何小有 关注
总资产49 (约3.39元)

推荐阅读

Android TV开发之使用Leanback传输控件
阅读 41

12 个实用的前端开发技巧总结
阅读 1,036

Android 11 新特性（含Android Studio 4.0 新特性）
阅读 6,787

CSS也可以这么美之登录页面---第二弹
阅读 979

vue 实现手写电子签名
阅读 652

dearpygui_menubar.png

目录对话框

通过 `select_directory_dialog` 来调用目录对话框，而且必须为其提供回调方法。 回调方法返回的 `data` 参数中将包含 **目录路径** 和 **文件夹路径**。通常，目录对话框是由另一个控件（例如下面栗子中的按钮）调用的。

```
from dearpygui.core import *
from dearpygui.simple import *

add_additional_font('三极中柔宋.ttf', 18, glyph_ranges='chinese_simplified_common')

def directory_picker(sender, data):
    select_directory_dialog(callback=apply_selected_directory)

def apply_selected_directory(sender, data):
    log_debug(data)
    directory = data[0]
    folder = data[1]
    set_value("目录", directory)
    set_value("文件夹", folder)
    set_value("文件夹路径", f"{directory}\\{folder}")

show_logger()

with window("Tutorial"):
    add_button("目录选择器", callback=directory_picker)
    add_text("目录路径: ")
    add_same_line()
    add_label_text("##dir", source="目录", color=[255, 0, 0])
    add_text("文件夹: ")
    add_same_line()
    add_label_text("##folder", source="文件夹", color=[255, 0, 0])
    add_text("文件夹路径: ")
    add_same_line()
    add_label_text("##folderpath", source="文件夹路径", color=[255, 0, 0])

start_dearpygui()
```

dearpygui_selectdirectorydialog.png

文件对话框

写下你的评论... 评论0 赞 ...

文件扩展名的过滤，控制显示哪些后缀名的文件。

```
from dearpygui.core import *
from dearpygui.simple import *

add_additional_font('三极中柔宋.ttf', 18, glyph_ranges='chinese_simplified_common')

def file_picker(sender, data):
    open_file_dialog(callback=apply_selected_file, extensions=".*,.py")

def apply_selected_file(sender, data):
    log_debug(data)
    directory = data[0]
    file = data[1]
    set_value("目录", directory)
    set_value("文件", file)
    set_value("文件路径", f"{directory}\\{file}")

show_logger()

with window("Tutorial"):
    add_button("文件选择器", callback=file_picker)
    add_text("目录路径: ")
    add_same_line()
    add_label_text("##filedir", source="目录", color=[255, 0, 0])
    add_text("文件: ")
    add_same_line()
    add_label_text("##file", source="文件", color=[255, 0, 0])
    add_text("文件路径: ")
    add_same_line()
    add_label_text("##filepath", source="文件路径", color=[255, 0, 0])

start_dearpygui()
```


dearpygui_openfiledialog.png

绘制图表

Dear PyGui 具有 `simple_plot`（简单绘图）和 `plot`（绘图）两个绘图方式，两者都是动态的。`simple_plot`（简单绘图）接受列表参数，并基于列表中的数据数据绘制 y轴 数据，可以是折线图或直方图。

```
from dearpygui.core import *
from dearpygui.simple import *

with window("Tutorial"):
    add_simple_plot("Simpleplot1", value=[0.3, 0.9, 0.5, 0.3], height=300)
```

dearpygui_addsimpleplot.png

而 `plot`（绘图）则具有更多的功能，绘图同时使用 x轴 和 y轴 坐标，使用 `add_plot` 方法创建，然后可以将数据作为线形图或散布图添加，`plot`（绘图）的特点有：

- 单击 & 拖动 —— 平移绘图
- 单击 & 拖动轴 —— 在一个方向上平移绘图
- 双击 —— 将绘图缩放并移动到数据区域
- 右键单击 & 拖动 —— 缩放区域
- 双右键单击 —— 打开设置
- Shift + 右键单击 & 拖动 —— 缩放并填充当前轴的区域
- 滚动鼠标滚轮 —— 缩放

```python
from dearpygui.core import *
from dearpygui.simple import *
from math import cos, sin

def plot_callback(sender, data):
    clear_plot("Plot")
    data1 = []
    for i in range(0, 100):
        data1.append([3.14 * i / 180, cos(3 * 3.14 * i / 180)])
    data2 = []
    for i in range(0, 100):
        data2.append([3.14 * i / 180, sin(2 * 3.14 * i / 180)])
    add_line_series("Plot", "Cos", data1, weight=2, color=[0, 0, 255, 100])
    add_shade_series("Plot", "Cos", data1, weight=2, fill=[255, 0, 0, 100])
    add_scatter_series("Plot", "Sin", data2, color=[0, 255, 0, 100])

with window("Tutorial"):
    add_button("Plot data", callback=plot_callback)
    add_plot("Plot", height=-1)

start_dearpygui()
```


dearpygui_addplot.png

通过 `set_value` 方法可以更改绘图调用的值，使 `simple_plot` （简单绘图）实现动态实时绘制。

```python
from dearpygui.core import *
from dearpygui.simple import *
from math import sin

def on_render(sender, data):
    frame_count = get_data("frame_count")
    frame_count += 1
    add_data("frame_count", frame_count)
    plot_data = get_value("plot_data")
    if len(plot_data) > 100:
        plot_data.pop(0)
    plot_data.append(sin(frame_count / 30))
    set_value("plot_data", plot_data)

with window("Tutorial"):
    add_simple_plot("Simple Plot", source="plot_data", minscale=-1.0, maxscale=1.0, height=300
    add_data("frame_count", 0)
    set_render_callback(on_render)

start_dearpygui()
```

dearpygui_simpleplotsetvalue.png

同样的，`plot`（绘图）也可以动态实时绘制，举个栗子，我们使用`set_render_callback`设置一个渲染回调实现动态绘制。

```python
from dearpygui.core import *
from dearpygui.simple import *
from math import cos

def plot_callback(sender, data):
    # 跟踪每一帧
    frame_count = get_data("frame_count")
    frame_count += 1
    add_data("frame_count", frame_count)
    # 更新 plot_data
    plot_data = get_data("plot_data")
    if len(plot_data) > 2000:
        frame_count = 0
        plot_data.clear()
    plot_data.append([3.14 * frame_count / 180, cos(3 * 3.14 * frame_count / 180)])
    add_data("plot_data", plot_data)
    # 绘制新数据
    clear_plot("Plot")
    add_line_series("Plot", "Cos", plot_data, weight=2)

with window("Tutorial"):
    add_plot("Plot", height=-1)
    add_data("plot_data", [])
    add_data("frame_count", 0)
    set_render_callback(plot_callback)

start_dearpygui()
```

dearpygui_plotsetvalue.png

绘画与画布

Dear PyGui 有一个低级绘图 API，可以用来原始绘画、自定义控件甚至动态绘画。先通过调用 `add_drawing` 方法开始绘画，再通过调用各种绘画方法来添加笔画。需要注意的是，画布的原点位于左下角。

```
from dearpygui.core import *
from dearpygui.simple import *

with window("Tutorial"):
    add_drawing("Drawing_1", width=120, height=120)

draw_line("Drawing_1", [10, 10], [100, 100], [255, 0, 0, 255], 1)
draw_text("Drawing_1", [16, 16], "Origin", color=[250, 250, 250, 255], size=15)
draw_arrow("Drawing_1", [100, 65], [50, 70], [0, 200, 255], 1, 10)

start_dearpygui()
```


dearpygui_adddrawing.png

```python
from dearpygui.core import *
from dearpygui.simple import *

with window("Tutorial"):
    add_drawing("Drawing_1", width=240, height=240)

draw_line("Drawing_1", [10, 10], [100, 100], [255, 0, 0, 255], 1)
draw_text("Drawing_1", [16, 16], "Origin", color=[250, 250, 250, 255], size=15)
draw_arrow("Drawing_1", [100, 65], [50, 70], [0, 200, 255], 1, 10)

set_drawing_origin("Drawing_1", 15, 15)
set_drawing_scale("Drawing_1", 2, 2)
set_drawing_size("Drawing_1", 250, 250)

start_dearpygui()
```


dearpygui_setdrawingorigin.png

绘画（`drawing`）可以显示的图像类型有 .png、.jpg、.bmp，使用时需掉用 `draw_image` 以绘制图像。通过 `pmin` 和 `pmax` 参数，我们可以将图像绘制到画布上矩形的左上和右下区域，图像会缩放自动缩放以适应指定区域。

使用 `uv_min` 和 `uv_max` 参数，我们可以控制图像要绘制到哪个区域的 标量（scalar），默认情况下，`uv_min = [0,0]` 和 `uv_max = [1,1]` 将显示整个图像，而 `uv_min = [0,0]` 和 `uv_max = [0.5,0.5]` 则仅显示图形的一部分。

```python
from dearpygui.core import *
from dearpygui.simple import *

with window("Tutorial"):
    add_drawing("Drawing_1", width=700, height=700)
```

```
start_dearpygui()
```


dearpygui_pokemonpng.png

尽管我们可以通过清除和重绘整个图来实现图形的动态化，但是 DearPyGui 还提供了一种更有效的方法，要使绘画（`drawing`）动态化，应该使用 `tag` 参数标记要重绘的控件，然后，只要使用相同的标签去调用。这样，我们就能实现仅清除该控件，并将其重新绘制。

```
from dearpygui.core import *
from dearpygui.simple import *

def on_render(sender, data):
    counter = get_data("counter")
    counter += 1
    modifier = get_data("modifier")
    if counter < 300:
        modifier += 1
    elif counter < 600:
        modifier -= 1
    else:
        counter = 0
        modifier = 2
    xpos = 15 + modifier * 1.25
    ypos = 15 + modifier * 1.25
    color1 = 255 - modifier * .8
    color3 = 255 - modifier * .3
    color2 = 255 - modifier * .8
    radius = 15 + modifier / 2
    segments = round(35 - modifier / 10)
    draw_circle("Drawing_1", [xpos, ypos], radius, [color1, color3, color2, 255], segments=seg
                tag="circle##dynamic")
    add_data("counter", counter)
    add_data("modifier", modifier)

```

```
33 set_render_callback(on_render)
34
35 start_dearpygui()
```


dearpygui_drawcircletag.png