Материалы к занятию

Графический модуль Arcade предоставляет вам классы для взаимодействия с пользователем с помощью кнопок, меток и многого другого.

UIWidget являются ядром Arcades GUI. Виджет представляет поведение и графическое представление любого элемента (например, Buttons или Text)

UIWidget имеет следующие свойства

rect - Координаты x и y (внизу слева от виджета), ширина и высота

children - Дочерние виджеты, отображаемые в этом виджете

size_hint - Кортеж из двух элементов. Он определяет сколько родительского пространства он хотел бы занять (диапазон: 0,0-1,0). Для максимального вертикального и горизонтального расширения - size_hint 1

size_hint_min - кортеж из двух целых чисел, определяет минимальный размер виджета.

size_hint_max - кортеж из двух целых чисел, определяет минимальный размер виджета.

Есть три способа обработки событий нажатия кнопки:

1. Создать класс с родительским классом UIFlatButton и реализовать метод с именем on_click.
2. Создать кнопку, а затем установить атрибут on_click этой кнопки равным функции, которую вы хотите вызвать.
3. Создать кнопку. Затем с помощью декоратора указать метод, вызываемый при возникновении on_click события для этой кнопки.

Напишем код в котором используем все три метода.

Для начала создадим графическое окно

```
import arcade


class MyWindow(arcade.Window):
   def __init__(self):
       super().__init__(800, 600, "UIFlatButton Example",
resizable=True)
       arcade.set_background_color(arcade.color.DARK_BLUE_GRAY)

   def on_draw(self):
       self.clear()


window = MyWindow()
arcade.run()
```

Создадим UIManager для обработки пользовательского графического интерфейса.

```
import arcade
import arcade.gui

class MyWindow(arcade.Window):
   def __init__(self):
       super().__init__(800, 600, "UIFlatButton Example",
resizable=True)
       self.manager = arcade.gui.UIManager()
       self.manager.enable()
       arcade.set_background_color(arcade.color.DARK_BLUE_GRAY)

   def on_draw(self):
       self.clear()
       self.manager.draw()


window = MyWindow()
arcade.run()
```

Создадим вертикальный контейнер для группировки кнопок и добавим его в менеджер, выровняв по центру

```
self.v_box = arcade.gui.UIBoxLayout()
self.manager.add(
   arcade.gui.UIAnchorWidget(
       anchor_x="center_x",
       anchor_y="center_y",
       child=self.v_box)
)
```

Теперь добавим 3 кнопки

```
start_button = arcade.gui.UIFlatButton(text="Start Game", width=200)
self.v_box.add(start_button.with_space_around(bottom=20))

settings_button = arcade.gui.UIFlatButton(text="Settings", width=200)
self.v_box.add(settings_button.with_space_around(bottom=20))

quit_button = QuitButton(text="Quit", width=200)
self.v_box.add(quit_button)
```

Для последней кнопки создадим класс(это один из вариантов обработки нажатия мыши)

```
class QuitButton(arcade.gui.UIFlatButton):
   def on_click(self, event: arcade.gui.UIOnClickEvent):
```

```
        arcade.exit()
```

Теперь по нажатию на кнопку выйти, мы выходим из приложения.
Добавим обработку клика вторым методом. Добавим метод в наш основной класс

```
def on_click_start(self, event):
   print("Start:", event)
```

и подключим его к кнопке

```
start_button.on_click = self.on_click_start
```

Ну и последний метод с использованием декоратора.Внутри метода __init__ создадим функцию

```
@settings_button.event("on_click")
def on_click_settings(event):
   print("Settings:", event)
```

Рассмотрим еще один вариант - кнопка может быть в виде изображения.Создадим окно с одной кнопкой

```
import arcade
import arcade.gui


class MyWindow(arcade.Window):
   def __init__(self):
       super().__init__(800, 600, "GUI Widgets Example",
resizable=True)

       self.manager = arcade.gui.UIManager()
       self.manager.enable()

       arcade.set_background_color(arcade.color.DARK_BLUE_GRAY)

       self.v_box = arcade.gui.UIBoxLayout()

       ui_flatbutton = arcade.gui.UIFlatButton(text="Flat
Button", width=200)
       self.v_box.add(ui_flatbutton.with_space_around(bottom=20))

       @ui_flatbutton.event("on_click")
       def on_click_flatbutton(event):
           print("UIFlatButton pressed", event)
```

```
        self.manager.add(
            arcade.gui.UIAnchorWidget(
                anchor_x="center_x",
                anchor_y="center_y",
                child=self.v_box)
        )

    def on_click_start(self, event):
        print("Start:", event)

    def on_draw(self):
        self.clear()
        self.manager.draw()


window = MyWindow()
arcade.run()
```

Теперь загрузим текстуру и используем ее в качестве кнопки, а также повесим обработчик

```
texture =
arcade.load_texture(":resources:onscreen_controls/flat_dark/play.
png")
ui_texture_button = arcade.gui.UITextureButton(texture=texture)

@ui_texture_button.event("on_click")
def on_click_texture_button(event):
    print("UITextureButton pressed", event)

self.v_box.add(ui_texture_button.with_space_around(bottom=20))
```

Ну и давайте используем label для указания какого либо текста

```
ui_text_label = arcade.gui.UITextArea(text="Пример",
                                      width=450,
                                      height=40,
                                      font_size=24,
                                      font_name="Kenney Future")
self.v_box.add(ui_text_label.with_space_around(bottom=0))

text = "Этот текст мы установили внутри нашего виджета, Это очень
просто! "

ui_text_label = arcade.gui.UITextArea(text=text,
                                      width=450,
                                      height=60,
                                      font_size=12,
                                      font_name="Arial")
self.v_box.add(ui_text_label.with_space_around(bottom=0))
```

Для начала создадим окно

```
import arcade
import arcade.gui


class MyWindow(arcade.Window):

   def __init__(self):
       super().__init__(800, 600, "OKMessageBox Example",
resizable=True)
       arcade.set_background_color(arcade.color.COOL_GREY)

   def on_draw(self):
       self.clear()


window = MyWindow()
arcade.run()
```

Добавим менеджер, контейнер и кнопку

```
import arcade
import arcade.gui


class MyWindow(arcade.Window):

   def __init__(self):
       super().__init__(800, 600, "OKMessageBox Example",
resizable=True)
       arcade.set_background_color(arcade.color.COOL_GREY)

       self.manager = arcade.gui.UIManager()
       self.manager.enable()

       self.v_box = arcade.gui.UIBoxLayout()

       open_message_box_button =
arcade.gui.UIFlatButton(text="Open", width=200)
       self.v_box.add(open_message_box_button)

       self.manager.add(
           arcade.gui.UIAnchorWidget(
               anchor_x="center_x",
               anchor_y="center_y",
               child=self.v_box)
       )
   def on_draw(self):
       self.clear()
```

```
        self.manager.draw()

window = MyWindow()
arcade.run()
```

Добавим обработчик нажатия на кнопку, в котором мы создадим виджет сообщения.

```
def on_click_open(self, event):
   message_box = arcade.gui.UIMessageBox(
        width=300,
        height=200,
        message_text=(
            "Это сообщение "
            "Нажмите Ок или Cancel"
        ),
        callback=self.on_message_box_close,
        buttons=["Ok", "Cancel"]
   )

   self.manager.add(message_box)
```

и подключим его к кнопке

```
open_message_box_button.on_click = self.on_click_open
```

Последнее что мы добавим обработчик нажатия кнопки отмены

```
def on_message_box_close(self, button_text):
   print(f"User pressed {button_text}.")
```

Отлично все работает.

Напоследок рассмотрим как создать прокручиваемое сообщение.

Создадим окно с менеджером

```
import arcade
from arcade.gui import UIManager


class MyWindow(arcade.Window):
```

```
   def __init__(self):
       super().__init__(800, 600, "Scrollable Text",
resizable=True)
       self.manager = UIManager()
       self.manager.enable()
       arcade.set_background_color(arcade.color.DARK_BLUE_GRAY)

   def on_draw(self):
       self.clear()
       self.manager.draw()


window = MyWindow()
arcade.run()
```

Создадим константу со случайно сгенерированным текстом

```
LOREM_IPSUM = (
   "Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Praesent eget pellentesque velit. "
   "Nam eu rhoncus nulla. Fusce ornare libero eget ex vulputate,
vitae mattis orci eleifend. "
   "Donec quis volutpat arcu. Proin lacinia velit id imperdiet
ultrices. Fusce porta magna leo, "
   "non maximus justo facilisis vel. Duis pretium sem ut eros
scelerisque, a dignissim ante "
   "pellentesque. Cras rutrum aliquam fermentum. Donec id mollis
mi.\n"
   "\n"
   "Nullam vitae nunc aliquet, lobortis purus eget, porttitor
purus. Curabitur feugiat purus sit "
   "amet finibus accumsan. Proin varius, enim in pretium
pulvinar, augue erat pellentesque ipsum, "
   "sit amet varius leo risus quis tellus. Donec posuere ligula
risus, et scelerisque nibh cursus "
   "ac. Mauris feugiat tortor turpis, vitae imperdiet mi euismod
aliquam. Fusce vel ligula volutpat, "
   "finibus sapien in, lacinia lorem. Proin tincidunt gravida
nisl in pellentesque. Aenean sed "
   "arcu ipsum. Vivamus quam arcu, elementum nec auctor non,
convallis non elit. Maecenas id "
   "scelerisque lectus. Vivamus eget sem tristique, dictum lorem
eget, maximus leo. Mauris lorem "
   "tellus, molestie eu orci ut, porta aliquam est. Nullam
lobortis tempor magna, egestas lacinia lectus.\n"
)
```

Подгрузим фоновую текстуру

```
bg_tex =
load_texture(":resources:gui_basic_assets/window/grey_panel.png")
```

Добавим текст с фоном и без

```
self.manager.add(
   UITexturePane(
       UIInputText(x=340, y=200, width=200, height=50,
text="Hello"),
       tex=bg_tex,
       padding=(10, 10, 10, 10)
   ))
self.manager.add(
   UIInputText(x=340, y=110, width=200, height=50, text="Hello"),
)
```

А теперь добавим текст со скролом

```
text_area = UITextArea(x=100,
                       y=200,
                       width=200,
                       height=300,
                       text=LOREM_IPSUM,
                       text_color=(0, 0, 0, 255))
self.manager.add(
   UITexturePane(
       text_area.with_space_around(right=20),
       tex=bg_tex,
       padding=(10, 10, 10, 10)
   )
)
```