Pente : Interface Graphique

D.Moreaux

21 octobre 2024

1 PyGame-CE

1.1 Introduction

La documentation officielle de PyGame-CE se trouve à l'adresse https ://pyga.me/docs/

Important, dans le gestionnaire de packages, installer la version pygame-ce et non la version pygame.

L'exemple de programme de Othelle à http ://cours.endor.be/2IN_Python/Demo_Code/othe permet de se faire une idée de comment on peut gérer un plateau de jeu en PyGame (à noter que pour le moment, les différences entre PyGame et PyGame-CE sont limitées)

2 PyGame-GUI

2.1 Introduction

La documentation officielle de PyGame_GUI se trouve à l'adresse https ://pygame-gui.readthedocs.io/en/latest/

Cette bibliothèque est prévue pour tourner avec la version PyGame-CE de PyGame et le package s'appelle pygame-gui. Attention, l'import dans Python se fait avec `import pygame_gui` avec un underscore et non un tiret.

L'utilisation de cette bibliothèque est assez simple : on crée un ui_manager dans lequel on peut ajouter des contrôles et les retirer (à l'aide de la fonction `kill()`) selon les besoins.

Au niveau de la boucle principale, il faudra ajouter quelques appels à PyGame_GUI (gestion des événements, tracé, mise à jour,. . .) et la bibliothèque se chargera de la gestion des éléments GUI.

2.2 Code de base

Pour permettre à PyGame-GUI de fonctionner, il faut

- Après avoir défini le mode écran et récupéré la surface de ce dernier, instancier un UIManager. Il sera nécessaire de lui fournir les mêmes dimensions qu'au $set_display.Onpourradeplusindiquerunfichier JSONcontenantlethmegra$
- Dans la boucle de gestion des event, appeler la méthode `process_event()` du UIManager avec l'événement en cours de traitement.
- Avant de commencer le tracé, appeler la méthode `update()` du UIManager avec le temps écoulé (dt)
- Après les tracés (et avant le flip), appeler la méthode `draw_ui()` du UIManager en lui passant la surface de l'écran

```
import␣pygame
import␣pygame_gui

pygame.init()
screen␣=␣pygame.display.set_mode((800,␣600))
background␣=␣pygame.Surface((800,␣600))
background.fill(pygame.Color('#000000'))

manager␣=␣pygame_gui.UIManager((800,␣600))␣#␣UIManager

clock␣=␣pygame.time.Clock()
is_running␣=␣True
while␣is_running:
␣␣␣␣time_delta␣=␣clock.tick(60)/1000.0
␣␣␣␣for␣event␣in␣pygame.event.get():
␣␣␣␣␣␣␣␣if␣event.type␣==␣pygame.QUIT:
␣␣␣␣␣␣␣␣␣␣␣␣is_running␣=␣False
␣␣␣␣␣␣␣␣manager.process_events(event)␣#␣Event

␣␣␣␣manager.update(time_delta)␣#␣Update
␣␣␣␣screen.blit(background,␣(0,␣0))
␣␣␣␣manager.draw_ui(window_surface)␣#␣Draw

␣␣␣␣pygame.display.update()
```

2.3 Events

PyGameGUI ajoute une série d'Events qui peuvent être traités dans la boucle. Pour les détecter, on comparera `event.type` au nom de l'événement (de la forme `pygame_gui.UI_...`)

La propriété `ui_element` de l'event vaudra l'élément GUI ayant déclenché l'événement [1]

- UI_BUTTON_PRESSED se déclenche lorsque l'on a cliqué et relaché le bouton de la souris sur l'élément. C'est l'équivalent du onclick du javascript. Par défaut, seul le clic gauche est pris en compte
- UI_TEXT_ENTRY_CHANGED indique que le texte d'une zone d'entrée a été changé. On peut accéder au nouveau texte par la propriété `text` de l'événement
- UI_TEXT_ENTRY_FINISHED indique que l'on a enfoncé la touche Enter alors qu'on était en train d'éditer une zone d'entrée. De nouveau, le texte peut être récupéré dans l'event.
- UI_DROP_DOWN_MENU_CHANGED indique que l'on a changé la valeur d'une liste "drop-down". La propriété `text` de l'objet event permet d'obtenir le nouveau texte choisi et la propriété `selected_option_id` permet de retrouver l'ID de l'option s'il y en a une (sinon, on récupère le texte).
- ...

Lorsque l'on utilise les boîtes de dialogue standard (sélection de fichier, confirmation,...), des événements sont également générés lorsque l'on confirme un choix.

2.4 Elements d'interface

Différents objets peuvent être instanciés à l'aide de `pygame_gui.elements.UI_...(...,manag` où le dernier paramètre indique le UIManager associé.

La valeur retournée peut être sauvée dans une variable pour comparaison ultérieure dans le gestionnaire d'événements ou pour effacer l'élément en appelant sa méthode `kill()`.

On peut aussi activer et désactiver des éléments à l'aide des méthodes `disable()` et `enable()`.

La propriété `relative_rect` est un `pygame.Rect` indiquant la position et la taille de l'élement.

- UILabel permet de mettre une étiquette texte. Le paramètre `text` est une chaîne contenant le texte devant être affiché.

1. Quand on créera un élément dont on désire gérer les événements, on prendra soin de les sauver dans des variables pour permettre la comparaison ultérieurement

— UIButton permet de créer un bouton. Le paramètre `text` contient le texte affiché dans le bouton. On peut utiliser le paramètre `command` pour spécifier une fonction à exécuter lors du clic ou gérer l'événement dans la boucle d'événements.
— UIDropDownMenu permet de créer une liste "drop down". On précisera un paramètre `options_list` qui contiendra une liste de chaînes proposées et `starting_option` qui contiendra l'option de départ (son texte) . Il est également possible de fournir une liste de paire de valeurs (texte et ID) pour les options.
— UITextEntryLine permet de créer une zone d'entrée texte. `initial_text` contiendra le texte initial s'il est non vide. La méthode `get_text()` permet de récupérer le texte entré.

On peut aussi ajouter une propriété `object_id="#..."` à un élément pour faire un lien avec le thème.

La méthode `set_text()` permet de changer le texte d'éléments tels que UILabel, UIButton, UITextEntryLine,...