Interface graphique Pygame pour Python

Gestion des événements [1]

Pour commencer cette partie sur la gestion des événements, je poserai la question :

Qu'est-ce qu'un événement ?

Un événement peut prendre plusieurs formes, il peut être amené par la pression ou le relâchement d'une touche du clavier, ou encore d'un bouton de la souris, un mouvement de la souris, du joystick, etc...
Mais il peut aussi être un déplacement ou un redimensionnement de la fênetre !

Un événement est donc tout ce que le programme peut "capter", de la part de l'utilisateur.

Nous allons dans cette partie apprendre à capturer tous ces événements et à attribuer à chacun d'eux une action bien précise !
C'est parti !

Evénement de fermeture

Dans le chapitre précédent, nous avions dans notre code une boucle infinie, qui gardait la fenêtre ouverte, mais qui ne recevait aucun événement. Cette boucle ne permettait donc pas de fermer la fenêtre si on le souhaitait.

Nous voulons que quand l'utilisateur clique sur la croix de fermeture de la fenêtre, celle-ci se ferme ! C'est logique non ?

Sans connaître les fonctions permettant cela, auriez-vous une idée d'où il faudrait placer le code pour couper la boucle infinie lors de la réception d'un événement ? Non ? Eh bien dans cette boucle même !

Je vous laisse essayer de comprendre ce code par vous même :

python

```python
continuer = 1

#Boucle infinie
while continuer:
    for event in pygame.event.get():   #On parcours la liste de tous les événements reçus
        if event.type == QUIT:     #Si un de ces événements est de type QUIT
            continuer = 0      #On arrête la boucle

```

Analysons ce code ensemble si vous le voulez bien :

- On lance la boucle infinie, jusqu'ici, pas de changement
- On lance une boucle *for*, qui parcourt tous les événements reçus grâce à la fonction *get()* du module "event" de Pygame. Cette fonction retourne une liste d'objets Event, pour lesquels on peut connaître le type, la touche enfoncée si c'est au clavier, la position du curseur si c'est un clic, etc...
- La condition teste si l'événement est de type QUIT (c'est à dire un Alt+F4 ou un clic sur le bouton de fermeture)
- Si la condition est satisfaite, on demande à la boucle de s'arrêter

Vous pourriez vous demander d'où sort cette variable QUIT, et ce serait légitime... Souvenez vous que lors de l'importation de la bibliothèque Pygame, vous avez inclus les constantes de *python.locals* dans votre script. Et bien ce QUIT est une de ces constantes !
N'hésitez pas à utiliser l'interpréteur pour lire la documentation sur les modules et les fonctions que je vous présente !

Vous pouvez dès maintenant changer le *continuer = int(input())* par cet événement de fermeture !

Nous allons maintenant placer dans cette boucle d'autres conditions, qui permettront de nouvelles actions si l'on envoie d'autres événements à Pygame. Allez, au boulot !

Evénements au clavier

Lorsque vous coderez un jeu, il est probable que vous assignerez des touches clavier aux différentes actions. La touche A peut par exemple donner un coup de poing, alors que la touche Z donne un coup de boule ! (Mmm violence)

Le type d'événement créé lorsque l'on appuie sur une touche est KEYDOWN, (ou KEYUP au relâchement de la touche). Vous penserez donc à créer une condition semblable à la précédente :

python

```python
if event.type == KEYDOWN:

```

Mais attention, cette condition sera vraie quelque soit la touche pressée ! Pour définir une seule touche du clavier, vous devrez utilisez en plus *event.key*, qui détermine la touche pressée, disponible uniquement lors d'un événement clavier.
Cet *event.key* peut prendre les valeurs suivantes :

Lettres:
K_a ... K_z

Nombres:
K_0 ... K_9

Controles:
K_TAB
K_RETURN
K_ESCAPE
K_SCROLLOCK
K_SYSREQ
K_BREAK
K_DELETE
K_BACKSPACE
K_CAPSLOCK
K_CLEAR
K_NUMLOCK

Ponctuation:
K_SPACE
K_PERIOD
K_COMMA
K_QUESTION

K_AMPERSAND
K_ASTERISK
K_AT
K_CARET
K_BACKQUOTE
K_DOLLAR
K_EQUALS
K_EURO
K_EXCLAIM
K_SLASH, K_BACKSLASH
K_COLON, K_SEMICOLON
K_QUOTE, K_QUOTEDBL
K_MINUS, K_PLUS
K_GREATER, K_LESS

Parenthèses:
K_RIGHTBRACKET, K_LEFTBRACKET
K_RIGHTPAREN, K_LEFTPAREN

Touches F:
K_F1 ... K_F15

Touches d'édition:
K_HELP
K_HOME
K_END
K_INSERT
K_PRINT
K_PAGEUP, K_PAGEDOWN
K_FIRST, K_LAST

Clavier numérique:
K_KP0 ... K_KP9
K_KP_DIVIDE
K_KP_ENTER
K_KP_EQUALS
K_KP_MINUS
K_KP_MULTIPLY
K_KP_PERIOD
K_KP_PLUS

SHF,CTL,ALT etc:
K_LALT, K_RALT
K_LCTRL, K_RCTRL
K_LSUPER, K_RSUPER

K_LSHIFT, K_RSHIFT
K_RMETA, K_LMETA

Flèches:
K_LEFT
K_UP
K_RIGHT
K_DOWN

Autres:
K_MENU
K_MODE
K_PAUSE
K_POWER
K_UNDERSCORE
K_HASH

Maintenant vous devez avoir en tête la condition que vous allez créer...
Je voudrais une condition qui écrit "Espace" dans la console lorsqu'on appuie sur la barre d'espace.
Ca y est ?

Voilà la condition à placer à la suite de la condition de fermeture, toujours dans la boucle de lecture des événements *for* :

python
```python
if event.type == KEYDOWN and event.key == K_SPACE:
    print("Espace")

```

Si vous avez plusieurs touches de contrôle dans votre programme, vous préférerez cette syntaxe :

python
```python
if event.type == KEYDOWN:
    if event.key == K_SPACE:
        print("Espace")
    if event.key == K_RETURN:
        print("Entrée")

```

Voilà, vous êtes maintenant capables de capturer tous les événements clavier et d'indiquer une action pour chaqu'un d'eux ! Rien ne vous empêche de créer vos propres fonctions, et de les appeler si la condition est satisfaite !
La liste des valeurs de *event.key* est assez imposante, c'est pourquoi il vaut mieux l'avoir sous la main en cas de besoin plutôt que de l'apprendre par coeur ! Mais en regardant bien, vous vous rendrez compte que les valeurs des touches classiques sont faciles à deviner...

Mouvements d'images

Nous allons maintenant utiliser les événements du clavier pour déplacer des images dans notre fenêtre ! Plutôt intéressant non ?

Je vous propose de récupérer le code permettant d'afficher un fond et un personnage dans une fenêtre
N'oubliez pas, les images dans le même répertoire si vous ne voulez pas changer les chemins !

Et comme on a appris l'événement de fermeture, on peut le rajouter !

Voici le code :

python

```python
import pygame
from pygame.locals import *

pygame.init()

#Ouverture de la fenêtre Pygame
fenetre = pygame.display.set_mode((640, 480))

#Chargement et collage du fond
fond = pygame.image.load("background.jpg").convert()
fenetre.blit(fond, (0,0))

#Chargement et collage du personnage
perso = pygame.image.load("perso.png").convert_alpha()
fenetre.blit(perso, (200,300))

#Rafraîchissement de l'écran
pygame.display.flip()

#BOUCLE INFINIE
continuer = 1
while continuer:
    for event in pygame.event.get():
        if event.type == QUIT:
            continuer = 0

```

C'est bon pour tout le monde ? J'ai juste récupéré le code de la partie précédente et ajouté l'événement de fermeture à celui-ci !

Nous allons essayer de faire bouger le personnage avec les flèches du clavier !
Pour cela nous utiliserons un nouvel objet, l'objet *Rect*, qui permet de manipuler des surfaces rectangulaires. Comme toutes nos images sont rectangles, il facilitera leur déplacement et leur manipulation, vous verrez !

Rect stocke en fait les positions d'une surface. Pour créer un Rect, nous utilisons la méthode de Surface *get_rect()*.
Pour obtenir le Rect *position_perso* à partir de *perso* :

python

```python
position_perso = perso.get_rect()
```

```
2
```

Les valeurs par défauts pour l'abscisse et l'ordonnée sont 0, donc si utilisez le Rect de base pour établir la position, le personnage sera dans l'angle en haut à gauche :

python

```
1 fenetre.blit(perso, position_perso)
2
```

Nous possédons maintenant une image *perso* donc les coordonnées sont gérés par le Rect *position_perso*.

Le code de chargement complet de l'image est celui ci :

python

```
1 #Chargement et collage du personnage
2 perso = pygame.image.load("perso.png").convert_alpha()
3 position_perso = perso.get_rect()
4 fenetre.blit(perso, position_perso)
5
```

Nous voulons maintenant que quand on appuie sur la flèche du bas, le personnage descende de 3 pixels.

Pour déplacer un Rect, nous utilisons ce code :

python

```
1 nom_du_rect.move(déplacement_x, déplacement_y)
2
```

Les déplacements peuvent être positifs ou négatifs.

Nous allons donc procéder comme ceci :

- Attente de l'événement "flèche bas"
- Modification de la position du Rect
- **Re-**collage du fond pour recouvrir la fenêtre et repartir à zéro
- Collage du personnage à sa nouvelle position
- Rafraîchissement de l'écran

Vous devriez avoir une idée du code nécessaire au déplacement maintenant. 
Essayez, essayez, n'oubliez pas que le mouvement doit être effectué à chaque fois qu'on appuie (on me dit "boucle" ? ) !
Et voici ma correction :

python

```
1 import pygame
2 from pygame.locals import *
3
4 pygame.init()
5
```

```
#Ouverture de la fenêtre Pygame
fenetre = pygame.display.set_mode((640, 480))

#Chargement et collage du fond
fond = pygame.image.load("background.jpg").convert()
fenetre.blit(fond, (0,0))

#Chargement et collage du personnage
perso = pygame.image.load("perso.png").convert_alpha()
position_perso = perso.get_rect()
fenetre.blit(perso, position_perso)

#Rafraîchissement de l'écran
pygame.display.flip()

#BOUCLE INFINIE
continuer = 1
while continuer:
    for event in pygame.event.get():    #Attente des événements
        if event.type == QUIT:
            continuer = 0
        if event.type == KEYDOWN:
            if event.key == K_DOWN: #Si "flèche bas"
                #On descend le perso
                position_perso = position_perso.move(0,3)

    #Re-collage
    fenetre.blit(fond, (0,0))
    fenetre.blit(perso, position_perso)
    #Rafraichissement
    pygame.display.flip()

```

Ce code mérite une petite explication. 

- Avant la boucle, je crée un Rect pour gérer la position de mon personnage. Et je colle le personnage à sa position initiale.
- Je lance la boucle.
- Dans la boucle, j'attend un événement "flèche bas".
- Si je reçois l'événement, je déplace le Rect de 3px vers le bas.
- Toujours dans la boucle, puisque ça doit arriver à chaque fois qu'on appuie sur la flèche, je recouvre la fenêtre avec le fond, je place mon personnage à sa nouvelle position et je rafraîchis l'écran !
- J'attend un autre événement...etc...

Piouf, c'était un petit peu plus long que ce qu'on avait l'habitude de voir.  Essayez de bien comprendre ce code, pour que celui-ci soit bien clair dans votre esprit...

Je vous laisse le soin de coder les conditions pour les autres événements (gauche, droite, haut), et vous aurez un programme qui permet de déplacer un personnage avec les flèches du clavier !

Passionnant non ?
Je peux vous dire que la première fois que j'ai réussi ça, j'étais très fier !

Si vous souhaitez pouvoir effectuer le déplacement plusieurs fois en laissant la touche enfoncée, vous devez utiliser la fonction *pygame.key.set_repeat()*, qui prend en paramètres :

- Le délai avant de continuer les déplacements quand la touche reste enfoncée (en millisecondes)
- Le temps entre chaque déplacement. (en millisecondes)

Placez la avant la boucle principale.
Par exemple :

```python
pygame.key.set_repeat(400, 30)

```

Ca y est ! Vous savez déplacer les images de votre programme grâce aux événements clavier !
Notre jeu peut gérer le clavier, si on passait à la souris ?

Evénements à la souris

Evénement au clic

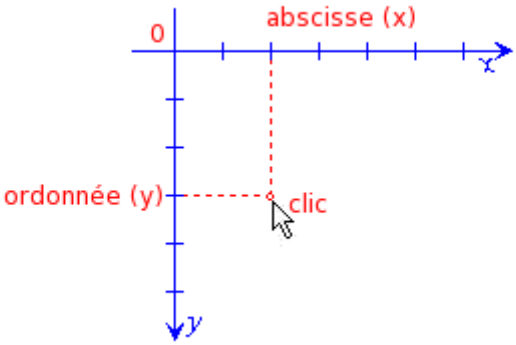
Les événements au clic peuvent être utiles pour créer des menus cliquables, ou encore pour coder un clone d'Age of Empire et permettre le contrôle à la souris !

Le type d'événement crée lors d'un clic est MOUSEBUTTONDOWN, (ou MOUSEBUTTONUP au relâchement du bouton).
Un événement de souris possède deux attributs : le bouton *(event.button)* et la position du clic dans la fenêtre *(event.pos)*.

event.button peut prendre les valeurs suivantes :

- 1 (bouton gauche)
- 2 (bouton milieu ou gauche+droite)
- 3 (bouton droite)
- 4 (molette haut)
- 5 (molette bas)

event.pos renvoie un tuple contenant l'abscisse et l'ordonnée à partir de l'angle haut-gauche, comme sur cette image :



On peut donc dire que *event.pos[0] = abscisse_clic* et *event.pos[1] = ordonnee_clic*

Je voudrais maintenant que vous me donniez la condition permettant d'afficher "Zone dangereuse" lorsque l'on clique avec le bouton droit dans la bande de 100px de hauteur d'en haut. (j'aime les exercices aux consignes compliquées)

C'est bon ? Voici la correction !

python

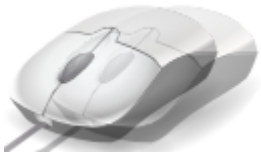
```python
if event.type == MOUSEBUTTONDOWN and event.button == 3 and event.pos[1] < 100:
    print("Zone dangereuse")

```

Pour délimiter la bande de 100px d'en haut, on doit dire que le message doit s'afficher si l'ordonnée du clic est inférieure à 100 ! Avec un peu de logique...

Vous pouvez aussi utiliser MOUSEBUTTONUP, qui peut être pratique pour éviter les clics trop rapide, et laisser l'utilisateur retirer son pointeur avant de relâcher

Evénement au mouvement de souris



L'événement au mouvement de la souris peut par exemple vous servir à créer un écran de veille qui s'éteint quand on agite la souris !

Le type d'événement crée lors du mouvement est MOUSEMOTION.
Un événement de mouvement de souris possède 3 attributs :

- la nouvelle position (*pos* (nouvelle à chaque pixel de mouvement)
- le déplacement relatif (*rel*, le nombre de pixel de déplacement depuis la dernière position)
- les boutons pressés pendant le mouvement *(buttons)*

Les données sont renvoyés sous forme de tuples :

- coordonnées x et y pour *pos*

- coordonnées x et y pour *rel*
- (gauche, milieu, droit) prenant pour valeur 0 (non pressé) ou 1 (pressé) pour *buttons*

Je vous propose de chercher la condition qui fermera la fenêtre lorsqu'on agite la souris en appuyant sur le bouton de gauche ! 
C'est bon ? Correction !

python

```python
if event.type == MOUSEMOTION and event.buttons[0] == 1:
    continuer = 0

```

Et voilà, vous connaissez tous les événements que vous pouvez récupérer à partir de la souris ! 

Mouvements d'images

Même principe que pour le clavier, on va essayer de faire bouger l'image en fonction des événements de la souris ! 

J'aimerais qu'à chaque fois que l'on clique, l'image se retrouve à la position du clic ! Facile non ? Pour cet exercice, nous n'utiliserons pas les Rect, mais deux variables de position, une pour les abscisses, et l'autre pour les ordonnées 

Voilà le code !

python

```python
import pygame
from pygame.locals import *

pygame.init()

#Ouverture de la fenêtre Pygame
fenetre = pygame.display.set_mode((640, 480))

#Chargement et collage du fond
fond = pygame.image.load("background.jpg").convert()
fenetre.blit(fond, (0,0))

#Chargement et collage du personnage
perso = pygame.image.load("perso.png").convert_alpha()
perso_x = 0
perso_y = 0
fenetre.blit(perso, (perso_x, perso_y))

#Rafraîchissement de l'écran
pygame.display.flip()

#BOUCLE INFINIE
continuer = 1
while continuer:
    for event in pygame.event.get():    #Attente des événements
        if event.type == QUIT:
            continuer = 0
        if event.type == MOUSEBUTTONDOWN:
```

```
            if event.button == 1:   #Si clic gauche
                #On change les coordonnées du perso
                perso_x = event.pos[0]
                perso_y = event.pos[1]

    #Re-collage
    fenetre.blit(fond, (0,0))
    fenetre.blit(perso, (perso_x, perso_y))
    #Rafraichissement
    pygame.display.flip()

```

Et voilà ! Le personnage prend la position de chaque clic gauche ! 

Je voudrais maintenant que l'image suive le curseur de la souris, et ça grâce aux événements de mouvement. Vous pourriez le faire ? 
A chaque mouvement, on met à jour les coordonnées de l'image !

Il suffit de remplacer la condition précédente par :

python

```python
if event.type == MOUSEMOTION: #Si mouvement de souris
    #On change les coordonnées du perso
    perso_x = event.pos[0]
    perso_y = event.pos[1]

```

Et voilà ! Vous connaissez maintenant toutes les possibilités de mouvements d'images grâce aux événements de la souris !

N'hésitez pas à pratiquer pour bien faire rentrer tout ces attributs et ces constantes dans votre tête. Ca deviendra naturel au bout de quelques essais !

Nous en avons fini avec les événements, j'espère que vous vous rendez compte de l'importance de ceux-ci dans tous les jeux vidéos, et des possibilités que ceux-ci vous donnent !

J'espère que tout cela ne vous a pas fait fuir, car maintenant, on va s'amuser un peu ! 
La prochaine partie, c'est TP (ouuuuuuuuuais !). Nous allons voir comment créer un petit jeu grâce à notre ami, le singe le plus connu du jeu vidéo, Donkey Kong ! 

☐ J'AI TERMINÉ CE CHAPITRE ET JE PASSE AU SUIVANT

‹ PREMIÈRES FENÊTRES

TP - DK LABYRINTHE ! ›