

Projet Pygame

Ce projet vous présente quelques commandes de Pygame, permettant tout de même de réaliser de nombreux programmes.

NB : les images utilisées sont données en archive.

1 Description du projet

Vous utiliserez le module Pygame pour réaliser un jeu vidéo 2D. Planifiez des objectifs simples et atteignables quitte à être plus ambitieux après. Avec les commandes décrites après, vous pourrez créer de nombreux types de jeux : PacMan, Snake, Super Mario, Doodle Jump, Flappy Bird etc. Attention aux licences des images et des sons que vous utiliserez dans votre jeu, ils doivent être gratuitement utilisables et s'ils ne sont pas libre de droits, veuillez à les créditer.

Avant de commencer votre jeu, il est fortement recommandé de faire les exercices dans cette feuille. Avant de coder votre jeu, il faut que vous ayez des idées claires sur ce que doit être votre jeu : des esquisses de l'interface, les mécanismes du jeu doivent être clairement écrit. Pour ce faire, vous devez créer un cahier des charges décrivant l'intégralité de votre jeu.

Remarques préliminaires :

1. Pour interagir avec Pygame on utilise des événements qui peuvent être un clic de la souris, un appui sur une touche etc.... Ces événements sont stockés dans une file qui s'appelle `pygame.event`. Les événements sont traités dans leur ordre chronologique. On peut avoir le type de l'événement grâce à la commande `event.type`. Avec des instructions conditionnelles, on peut gérer les actions correspondant aux différents événements.
2. Les positions d'un objet sont toujours calculées à partir du coin en haut à gauche du contenant (c'est-à-dire de l'écran pour la fenêtre, de la fenêtre pour les objets qui sont tracés dessus...etc...) et on donne toujours d'abord l'abscisse (de 0 à gauche à la largeur de la fenêtre à droite) puis l'ordonnée (de 0 en haut jusqu'à la hauteur de la fenêtre en bas : l'axe des ordonnées va donc vers le bas..).

2 Création d'une fenêtre

Le programme suivant crée une fenêtre Pygame :

```
1 import pygame #importation de pygame
2 import traceback #module pour recuperer des infos sur les erreurs
3 from pygame.locals import * #on importe les constantes de pygame
4 #on lance pygame
5 #toujours encadrer vos programmes pygame par try et finally ce qui permet de fermer
6 #correctement la fenetre pygame en cas d'erreur
7 try:
8     #creation d'une fenetre
9     fenetre=pygame.display.set_mode((640,480))#fenetre de taille 640*480
10    continuer=1
11    #boucle perpetuelle qui permet de garder la fenetre ouverte
12    while continuer:
13        for event in pygame.event.get(): #pygame prend le premier evenement de la file
14            if event.type==QUIT: #l'evenement QUIT correspond au clic sur la croix
15                continuer=0 #permet de quitter la boucle
16 except:
17     traceback.print_exc()
```

```

18 finally :
19     pygame.quit()
20     exit()

```

Commandes additionnelles :

- Remplir la fenêtre (qui a pour nom `fenetre` dans notre programme) d'une certaine couleur `fenetre.fill((R,G,B))` : R est la quantité de rouge entre 0 et 255, G (ou V) pour le vert et B pour le bleu. Attention il y a une parenthèse pour "fill" et une parenthèse pour la couleur (R,G,B). Pour tester les couleurs vous pouvez aussi utiliser le petit logiciel qui s'appelle la boîte à couleur.
- Mettre un titre à la fenêtre : `pygame.display.set_caption("Mon_titre")`
- Rafraîchir l'écran : indispensable après toute modification : `pygame.display.flip()`

Exercice 1. Créer une fenêtre de taille 300×200 coloriée en bleu.

3 Les images

Pour afficher une image :

- Il faut charger l'image (après avoir lancé Pygame) :
`image1=pygame.image.load("image.jpg").convert()`, entre les guillemets on met l'adresse de l'image par rapport au dossier où se trouve le programme. On peut utiliser toutes sortes de formats d'images (jpg, png ...). `convert` sert à convertir l'image dans un format dont l'affichage est plus rapide.
- Il faut "coller", on dit "blitter" l'image sur l'écran : `fenetre.blit(image1,(x,y))` où x,y sont les coordonnées du coin en haut à gauche de l'image
- Il faut rafraîchir l'affichage : `pygame.display.flip()`

Commandes complémentaires : On peut (entre autres) :

- Rendre le fond de l'image transparent de deux manières : si c'est une image au format png dont le fond est transparent, remplacer `convert` par `convert_alpha`. Si c'est une image autre dont le fond a une couleur (R,G,B), on utilise la commande : `image1.set_colorkey((R,G,B))`
- Redimensionner une image : `image1=pygame.transform.scale(image1,(30,30))` pour avoir une image 30×30

Exercice 2. Afficher une image de votre choix avec un fond transparent dans la fenêtre créée précédemment. L'image devra être de taille 20×20 et être dans le coin en bas à droite.

4 Les événements

On reprend le programme du début et on insère après le bloc `if event.type==QUIT` d'autres blocs, par exemple :

```

1 if event.type==KEYDOWN: #appui sur une touche
2 if event.key==K_UP: #la touche est la fleche vers le haut / déplacer le personnage d'
   une case vers le haut...
3 if event.key==K_DOWN: #déplacer le personnage d'une case vers le bas

```

Les différents types d'événements sont :

- `event.type==QUIT` (c'est l'une des "constantes importées de `pygame.locals`) : c'est le fait d'appuyer sur la croix : vous pouvez décider d'associer ceci à la fermeture de la fenêtre (c'est ce qui est fait en général) ou autre chose !
- `event.type == KEYDOWN` correspond à l'enfoncement d'une touche (il y a aussi `KEYUP` pour le relâchement d'une touche). On peut alors récupérer la touche appuyée avec la commande : `event.key` qui peut prendre différentes valeurs. La liste des différentes touches est disponible à cette adresse : <http://thepythongamebook.com/fr:glossaire:p:pygame:keycodes>. Si on maintient la touche enfoncée on peut répéter l'événement,

avec la commande `pygame.key.set_repeat(400,30)`, où 400 est le délai (en millisecondes) avant que la répétition ne commence et à partir de là, un nouvel événement est généré toutes les 30 ms. Si votre touche provoque un déplacement et que vous appuyez 550 ms sur la touche, le déplacement va se faire une première fois au début, il y aura 400 ms d'attente, puis il y aura un déplacement à 400 ms, à 430 ms, à 460 ms....Le délai est là pour qu'il n'y ait pas de répétitions indésirables.

- `event.type==MOUSEBUTTONDOWN` quand on clique avec la souris. Pour récupérer les coordonnées x et y de la position de la souris on écrit `(x,y)=event.pos`
- `event.type==MOUSEMOTION` quand on déplace la souris, par exemple si on veut déplacer un personnage avec la souris (on récupère la position comme ci-dessus). Attention ce type d'événements peut rapidement remplir la file des événements....

Exercice 3. Créer une fenêtre de taille 600×400 et écrire un programme qui affiche l'image précédente à l'endroit où on a cliqué avec la souris. Comment faire pour qu'un seul personnage soit affiché à chaque fois ?

5 Les mouvements

Il est recommandé d'utiliser des rectangles qui contiennent les objets, ce qui permet de les déplacer facilement et de mieux gérer l'affichage, les effets de bord et les collisions.

- pour récupérer le rectangle d'un objet appelé objet :

```
1 objet_rect=objet.get_rect()
2 #objet_rect vaut alors (x,y,largeur,hauteur) ou (x,y) est la position de l'
  objet
3 #et largeur et hauteur sont les dimensions du rectangle qui entoure l'image.
```

- pour créer un rectangle :

```
1 mon_rect=pygame.Rect(x,y,largeur, hauteur) #ou x,y correspond a la position du
  rectangle
```

- pour déplacer un rectangle (et l'objet qui est dedans) :

```
1 mon_rect.move(x,y) #(x est le déplacement horizontal, y vertical)
```

Attention, il faut blitter l'objet et remettre le "fond" à la place précédente de l'objet `fenetre.blit(objet,mon_rect)` et pour le fond cela dépend des situations (voir l'exemple qui suit) et ne pas oublier de rafraîchir la fenêtre `fenetre.display.flip()`

Voici les différents attributs d'un rect appelé r :

- `r.left` (ou `right`, `bottom`, `top`)
- `r.center`
- `r.centerx` (ou `centery`)
- `r.size`
- `r.width` (ou `height`)
- `r.topleft` (ou `topright`)
- `r.midtop` (ou `midbottom`, `midright`, `midleft`)

N'hésitez pas à consulter la documentation pour plus de détails sur ces attributs.

Exercice 4. Afficher l'image précédente, récupérer son rect et l'afficher dans la console. Recommencer mais en choisissant vous même le rect pour que le personnage soit au centre de l'écran.

Le script suivant gère le déplacement d'un personnage :

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

```

6 try:
7     fenetre=pygame.display.set_mode((640,480))
8     fenetre.fill((255,255,255)) #on remplit la fenetre de blanc
9     image1=pygame.image.load("image.png").convert_alpha()
10    image1_rect=image1.get_rect() #on cree un rectangle entourant l'image
11    fenetre.blit(image1,image1_rect) #on blitte l'image dans ce rectangle
12    pygame.key.set_repeat(400,30) #on active la repetition des touches pygame.display.
    flip()
13    continuer=1
14    while continuer:
15        for event in pygame.event.get():
16            #attention, toujours prévoir un moyen de sortir de la boucle
17            if event.type==QUIT:
18                continuer=0
19            elif event.type==KEYDOWN:
20                if event.key==K_LEFT:
21                    image1_rect=image1_rect.move(-5,0) #on deplace le rectangle de l'image
    de 5 pixel vers la gauche
22                    if image1_rect.left<0: #si le bord gauche de l'image sort du cadre, on
    remet l'image a droite
23                        image1_rect.left=610
24                    if event.key==K_RIGHT:
25                        image1_rect=image1_rect.move(5,0)
26                        if image1_rect.right>640:
27                            image1_rect.right=30
28                        #si on ne re-remplit pas le fond, on verra l'objet aux deux positions
29                        fenetre.fill((255,255,255))
30                        fenetre.blit(image1,image1_rect)
31                        pygame.display.flip()
32    except:
33        traceback.print_exc()
34    finally:
35        pygame.quit()
36        exit()

```

Exercice 5. Compléter le programme ci-dessus en ajoutant des déplacements vers le haut et vers le bas. Ajouter des instructions de telle sorte que le personnage soit remis au centre quand on appuie sur Espace. *Veillez à bien modifier le chemin de l'image ligne 9.*

6 Les textes

Les commandes suivantes permettent d'afficher une chaîne de caractères :

```

1 chaine="ma chaine sur une seule ligne" #ne pas hesiter a utiliser la methode format
2 font=pygame.font.SysFont("broadway",24,bold=False,italic=False) #Police : Broadway,
    taille 24, pas de gras, pas d'italique
3 text=font.render(chaine,1,(R,G,B)) #pour creer l'objet texte
4 fenetre.blit(text,(30,30)) #ou (30,30) correspond a la position du texte
5 fenetre.display.flip() #pour rafraichir l'affichage

```

Exercice 6. Compléter le programme précédent pour que l'appui sur la touche Echap provoque l'affichage du texte suivant : « Je me suis déplacé n fois » si le personnage a effectué n mouvements .

7 Les dessins

On peut dessiner des formes dans Pygame :

- un rectangle : `mon_rectangle=pygame.draw.rect(surface,color,rect,epaisseur)` avec `surface` est la surface sur laquelle on veut dessiner le rectangle (la fenêtre ou autre chose), `color` c'est un triplet (R,G,B), `rect` correspond à la position : ce peut être un `pygame.Rect` ou

- (positionx, positiony, largeur, hauteur) , epaisseur est l'épaisseur du trait (0 donne un rectangle plein).
- un cercle : `mon_cercle=pygame.draw.circle(surface,color,pos_centre,rayon,epaisseur)`
où `pos_centre` est du type (x,y)
- une ligne
`ma_ligne=pygame.draw(surface,color,position_depart,position_arrivee,epaisseur)`

Inutile de blitter, mais il faut rafraichir l'écran.

On peut créer des surfaces (par exemple pour avoir deux parties dans une fenêtre) :

`ma_surface=pygame.Surface((34,34))` crée un rectangle noir de taille 34×34 On peut le remplir entièrement ou partiellement `surf.fill((R,G,B))` le remplira entièrement ou `surf.fill((R,G,B), rect)` remplira le rect seulement, où `rect` est un `pygame.Rect` ou bien `(x,y,largeur,hauteur)`

Ceci s'applique aussi au remplissage de la fenêtre.

Les différents objets peuvent alors être blittés sur cette surface : la position s'entend alors par rapport au coin supérieur gauche de `ma_surface`. Il faut évidemment blitter `ma_surface` et rafraîchir l'écran.

Exercice 7. Écrire un programme qui trace les segments entre les points où on aura cliqué successivement avec la souris

8 Le temps

Les différentes instructions suivantes permettent de gérer le temps :

- `pygame.time.delay(100)` crée une attente de 100 ms
- `pygame.time.get_ticks()` compte le temps en ms depuis `pygame.init()`
- limiter le nombre d'images à 60 images par seconde :

```
1 clock=pygame.time.Clock
2 clock.tick(60)
```

- Si on veut répéter une action à intervalles de temps réguliers on peut suivre un schéma analogue au suivant en utilisant un timer :

```
1 import pygame
2
3 import os
4 import traceback
5 from pygame.locals import *
6
7 pygame.init()
8 try:
9     fenetre=pygame.display.set_mode((640,480))
10    fenetre.fill((255,255,255))
11    image1=pygame.image.load("image.png").convert_alpha()
12    image1_rect=image1.get_rect()
13    fenetre.blit(image1,image1_rect)
14    pygame.display.flip()
15
16    #on cree un evenement qui n'est pas un evenement predefini par le systeme : il
17    #faut lui donner un numeero pour que pygame le gere
18
19    depla=USEREVENT+1 #on donne un numero a l'evenement entre USEREVENT et
20    NUMEVENTS (qui sont des constantes de Pygame)
21    pygame.time.set_timer(depla,150) #l'evenement va se mettre dans la file des
22    evenements toutes les 150 ms
23    continuer=1
24    while continuer:
25        for event in pygame.event.get():
26            if event.type==QUIT:
27                continuer=0
28            elif event.type==depla: #gestion de l'evenement repetitif
```

```

26         #on fait ce que l'on veut, ici on deplace le personnage en
    diagonale
27         image1_rect=image1_rect.move(3,3)
28         fenetre.fill((255,255,255))
29         fenetre.blit(image1,image1_rect)
30         pygame.display.flip()
31 except :
32     traceback.print_exc()
33 finally:
34     pygame.quit()
35     exit()

```

9 Les animations

Pour animer un objet le principe est simple et est le même que dans les dessins animés : on affiche successivement des images de l'objet dans différentes positions après avoir effacé le précédent.

```

1 import sys,pygame
2 from pygame.locals import *
3 import traceback
4 blue=145,197,235
5 clock=pygame.time.Clock()
6 #voir a la fin, on veut choisir le nombre d'images par secondes
7 def image(chaine):
8     """fonctions qui charge les sprites et rend le fond transparent etc.."""
9     im=pygame.image.load(chaine)
10    im=im.convert_alpha(im)
11    return im
12 try:
13     screen=pygame.display.set_mode((400,151))
14
15     b=[0,0,0,0]
16     for i in range(4):
17         b[i]=image("course{}".format(i)) #voir la methode "format" sur les chaines de
            caracteres
18     #on fait une liste des images du personnage dans differentes positions
19     continuer=1
20     i=0 #index de l'image qu'on va afficher
21     while continuer:
22         for event in pygame.event.get():
23             if event.type in (QUIT, KEYDOWN):#pour quitter
24                 continuer=0
25             screen.fill(blue)
26             #on va afficher successivement les images de la liste en revenant au dzbut
27             #quand on est a la fin (avec i%4)
28             screen.blit(b[i%4],(180,54)) #(180,54) est la position ou on blitte
29             i=i+1 #pour afficher ensuite l'image suivante
30             clock.tick(10) #on limite le nombre d'images a 10 images par seconde.
31 #sinon on ne voit rien, c'est trop rapide
32     pygame.display.flip()
33 except :
34     traceback.print_exc()
35 finally:
36     pygame.quit()

```

Beaucoup d'autres instructions existent, il vous revient d'explorer la documentation officiel de Pygame : <http://www.pygame.org/docs/>.