Projet – Une animation avec Pygame

Cette séquence s'appuie sur :

- https://pixees.fr/informatiquelycee/n_site/nsi_term_projet_3.html

1 Comprendre pygame

Quelques mots sur Pygame :
Pygame est une bibliothèque libre multiplate-forme qui facilite le développement de jeux vidéo temps réel avec le langage de programmation Python. Elle permet de programmer la partie multimédia (graphismes, son et entrées au clavier, à la souris ou au joystick), sans se heurter aux difficultés des langages de bas niveau. Pygame est distribuée selon les termes de la licence GNU LGPL. (d'après Wikipédia)
Avant de pouvoir s'attaquer à la création du morpion proprement dit, il est nécessaire de travailler sur les bases de l'utilisation de Pygame :

A faire vous même 1.

Saisissez, analysez et testez ce code

```
import pygame

surf = pygame.display.set_mode((800,600))
pygame.quit()
```

Quelques explications sur le code ci-dessus :

- la première ligne permet d'importer la bibliothèque `pygame`
- `surf = pygame.display.set_mode((800, 600))` permet de créer la "surface" `pygame`, cette "surface" aura pour dimension 800 pixels de large et 600 pixels de haut
- la dernière ligne permet de quitter "proprement" votre programme `pygame` (si vous omettez cette ligne vous risquez de vous retrouver bloqué avec une fenêtre impossible à fermer)

Vous avez peut-être remarqué qu'une fenêtre s'ouvre et se referme quasi immédiatement après. Pourquoi ?
L'interpréteur Python exécute les instructions ligne après ligne, une fois la dernière ligne exécutée, le programme est terminé et la fenêtre se ferme.
Il faut donc empêcher la fenêtre de se refermer, il faut donc empêcher le programme de se terminer. Pour cela nous allons employer une boucle. On appelle souvent cette boucle une "boucle de jeu"

A faire vous même 2.

Saisissez, analysez et testez ce code

```
import pygame

surf = pygame.display.set_mode((800,600))
run = True
while run :
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
pygame.quit()
```

Nous avons bien notre boucle (`while run` avec au départ `run = True`).
Cette boucle ne peut pas continuer indéfiniment, il faut laisser à l'utilisateur la possibilité de sortir du programme. Pour cela nous utilisons le gestionnaire d'événements de `pygame` :
`pygame` "surveille" tous les événements qui pourraient survenir (principalement une action de l'utilisateur sur la souris ou sur le clavier). `pygame.event.get()` renvoie un tableau avec tous les événements en cours, la boucle `for` permet de parcourir tous ces événements. On retrouve ces événements dans l'objet `event`. Un des événements possibles est "le clic de souris sur la croix en haut à gauche de la fenêtre" traduit par `pygame.QUIT`. Si l'utilisateur clique sur "la croix en haut à gauche de la fenêtre", on "entre" dans le `if` et la variable `run` devient `False` : la "boucle de jeu" se termine, la ligne `pygame.quit()` est exécutée, et le programme se termine.

Il faut bien comprendre que la liste des événements en cours est réactualisée en permanence grâce à la "boucle de jeu" : les instructions contenues dans la boucle sont exécutées des dizaines de fois par seconde, on a donc `pygame.event.get()` qui est exécutée plusieurs dizaines de fois par seconde, la liste des événements est donc mise à jour plusieurs dizaines de fois par seconde ! On peut donc dire que les événements (clavier et souris) sont "surveillés" en "permanence" (même si on devrait plutôt dire que les événements sont "controlés" plusieurs dizaines de fois par seconde).
`pygame` permet de dessiner des formes diverses :

A faire vous même 3.

Saisissez, analysez et testez ce code

```
import pygame

surf = pygame.display.set_mode((800,600))
run = True
while run :
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      run = False
  pygame.draw.line(surf,(255,255,255),(10,20),(150,200),2)
  pygame.display.flip()
pygame.quit()
```

La ligne `pygame.draw.line(...)` permet d'afficher une ligne, la méthode `line` prend en paramètres :

- la surface sur laquelle nous allons dessiner (`surf`)
- la couleur de la ligne au format (r,v,b) (tuple de 3 nombres compris entre 0 et 255), ici on a une ligne blanche avec (255,255,255)
- coordonnées du point de départ : tuple (x,y). Ici notre point de départ a pour coordonnées (10,20)

- coordonnées du point d'arrivée' : tuple (x,y). Ici notre point d'arrivée a pour coordonnées (150,200)
- épaisseur de la ligne. Ici nous avons une épaisseur de 2 pixels

ATTENTION : le point de coordonnées (0,0) est en haut et à gauche de la fenêtre
La ligne `pygame.display.flip()` indique à `pygame` qu'il faut afficher tout ce qui doit être affiché (cette ligne est nécessaire à partir du moment où vous cherchez à dessiner quelque chose)
Il est aussi possible de dessiner des cercles :

A faire vous même 4.

Saisissez, analysez et testez ce code

```
import pygame

surf = pygame.display.set_mode((800,600))
run = True
while run :
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      run = False
  pygame.draw.circle(surf, (255,0,0), (400, 300), 30, 2)
  pygame.display.flip()
pygame.quit()
```

La ligne `pygame.draw.circle(...)` permet d'afficher un cercle, la méthode `circle` prend en paramètres :

- la surface sur laquelle nous allons dessiner (`surf`)
- la couleur de la ligne au format (r,v,b) (tuple de 3 nombres compris entre 0 et 255), ici on a un cercle rouge avec (255,0,0)
- coordonnées du centre du cercle : tuple (x,y). Ici le centre du cercle a pour coordonnées (400,300)
- rayon du cercle. Ici notre rayon est de 30 pixels
- épaisseur de la ligne. Ici nous avons une épaisseur de 2 pixels

Il est possible de dessiner d'autres formes : rectangle, polygone, ellipse... Pour en savoir plus, n'hésitez pas à consulter la documentation de Pygame

Il est possible de créer des animations dans `pygame` :
Nous allons déplacer notre cercle en modifiant ces coordonnées à chaque tour de boucle :

A faire vous même 5.

Saisissez, analysez et testez ce code

```
import pygame

surf = pygame.display.set_mode((800,600))
run = True
posX = 50
while run :
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      run = False
  pygame.draw.circle(surf, (255,0,0), (posX, 300), 30, 2)
  posX = posX + 1
  pygame.display.flip()
pygame.quit()
```

Comme vous pouvez le constater, nous avons un problème : les cercles précédents restent affichés. Il faut donc effacer l'image précédente avant de pouvoir en afficher une nouvelle.

A faire vous même 6.

Saisissez, analysez et testez ce code

```
import pygame

surf = pygame.display.set_mode((800,600))
run = True
posX = 50
while run :
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      run = False
  surf.fill((0,0,0))
  pygame.draw.circle(surf, (255,0,0), (posX, 300), 30, 2)
  posX = posX + 1
  pygame.display.flip()
pygame.quit()
```

la ligne `surf.fill((0,0,0))` permet d'effacer l'écran avant de réafficher le cercle à une position différente. `fill` prend en paramètre un tuple qui permet de définir la couleur de fond de la surface (ici avec (0,0,0) nous avons du noir en arrière-plan).
Il est important de bien comprendre que même si vous n'avez pas d'animation à gérer, il est important d'effacer la surface avant d'afficher une nouvelle image. Nous utiliserons donc systématiquement le `fill`.
Il est relativement simple d'avoir une balle qui rebondit sur le bord de l'écran :

A faire vous même 7.

Saisissez, analysez et testez ce code

```
import pygame
surf = pygame.display.set_mode((800,600))
run = True
posX = 50
vx = 1
while run :
```

```
    for event in pygame.event.get():
      if event.type == pygame.QUIT:
        run = False
    surf.fill((0,0,0))
    pygame.draw.circle(surf, (255,0,0), (posX, 300), 30, 2)
    if posX>770 or posX<30 :
      vx=-vx
    posX = posX + vx
    pygame.display.flip()
pygame.quit()
```

Le système essaye d'exécuter le plus grand nombre de fois possible la "boucle de jeu". Sachant que plus votre microprocesseur est rapide et plus le nombre d'exécutions de la "boucle de jeu" par seconde sera grand, vous risquez d'avoir un jeu qui ne tournera pas du tout de la même façon sur 2 machines différentes. Pour éviter cet inconvénient, il est possible de limiter le nombre d'exécutions de la "boucle de jeu" par seconde :

A faire vous même 8.

Saisissez, analysez et testez ce code

```
import pygame

surf = pygame.display.set_mode((800,600))
run = True
posX = 50
vx = 1
clock=pygame.time.Clock()
while run :
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      run = False
  clock.tick(60)
  surf.fill((0,0,0))
  pygame.draw.circle(surf, (255,0,0), (posX, 300), 30, 2)
  if posX>770 or posX<30 :
    vx=-vx
  posX = posX + vx
  pygame.display.flip()
pygame.quit()
```

Nous avons ajouté 2 lignes afin de contrôler le nombre d'exécutions de la "boucle de jeu" par seconde :

- `clock=pygame.time.Clock()` permet de définir un système d'horloge
- `clock.tick(60)` permet de limiter le nombre d'exécutions de la "boucle de jeu" à 60 par seconde (nous aurons donc une fréquence d'affichage de 60 images par seconde)

Ce système de limitation est surtout important si vous avez à gérer des animations.
`pygame` permet d'afficher des images relativement simplement :
Commencez par télécharger cette image. Placez-là dans votre répertoire courant (là où vous avez placé vos programmes Pygame)

A faire vous même 9.

Saisissez, analysez et testez ce code

```
import pygame

surf = pygame.display.set_mode((800,600))
run = True
img = pygame.image.load("pyg.png")
while run :
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      run = False
  surf.fill((0,0,0))
  surf.blit(img,(200,200))
  pygame.display.flip()
pygame.quit()
```

L'affichage de l'image se fait en 2 étapes :

- création d'un objet de type image (`img` dans notre exemple) à l'aide de la méthode `load`. cette méthode `load` prend un seul paramètre : l'url de l'image
- utilisation de la méthode `blit` pour afficher l'image (la méthode `blit` doit être utilisée dans la "boucle de jeu"). La méthode `blit` prend 2 paramètres : l'objet image à afficher (dans notre cas `img`) et un tuple qui correspond aux coordonnées du coin haut-gauche de l'image ((200,200) dans notre exemple)

Comme déjà dit plus haut, il est possible de gérer les événements "clavier" et "souris" (utilisation du clavier et de la souris par l'utilisateur). Nous allons ici uniquement nous intéresser aux événements "souris". Si vous avez besoin d'utiliser les événements "clavier" n'hésitez pas à consulter la documentation de `pygame`.
Il existe un événement `pygame.MOUSEBUTTONDOWN` qui correspond à un clic de souris. La méthode `get_pressed` renvoie un tuple constitué de 3 éléments. En l'absence de clic de souris, ce tuple est (0,0,0). En cas de clic sur le bouton gauche de la souris le tuple est (1,0,0). En cas de clic sur le bouton central le tuple est (0,1,0). En cas de clic sur le bouton droit le tuple est (0,0,1)

A faire vous même 10.

Saisissez, analysez et testez ce code. Cliquez avec votre souris (clic gauche et clic droit) dans la fenêtre `pygame`, observez attentivement la console

```
import pygame
surf = pygame.display.set_mode((800,600))
run = True
while run :
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
```

```
        run = False
      if event.type == pygame.MOUSEBUTTONDOWN :
        if pygame.mouse.get_pressed() == (1,0,0) :
          print ("clic bouton gauche")
        if pygame.mouse.get_pressed() == (0,0,1) :
          print ("clic bouton droit")
    surf.fill((0,0,0))
    pygame.display.flip()
  pygame.quit()
```

Il est aussi possible de récupérer les coordonnées du pointeur de la souris au moment du clic à l'aide de la méthode `mouse.get_pos`. Cette méthode renvoie un tuple (x,y) => coordonnées du pointeur de la souris au moment du clic

A faire vous même 11.

Saisissez, analysez et testez ce code. Cliquez avec votre souris (clic gauche) dans la fenêtre pygame, observez attentivement la console

```
import pygame

surf = pygame.display.set_mode((800,600))
run = True
while run :
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      run = False
    if event.type == pygame.MOUSEBUTTONDOWN :
      if pygame.mouse.get_pressed() == (1,0,0) :
        pos = pygame.mouse.get_pos()
        print(pos)
  surf.fill((0,0,0))
  pygame.display.flip()
pygame.quit()
```

Il est aussi possible de gérer des événements de type "clavier" :

A faire vous même 12.

Saisissez, analysez et testez ce code, observez la console lorsque vous appuyez sur les touches "Entrée", "Espace" et "A" (vous pouvez aussi constater qu'il ne se passe rien quand)

```
import pygame

surf = pygame.display.set_mode((800,600))
run = True
while run :
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      run = False
    if event.type == pygame.KEYDOWN :
      if event.key == pygame.K_SPACE :
        print ("vous avez appuyé sur la touche espace")
      elif event.key == pygame.K_a :
        print ("vous avez appuyé sur la touche A")
      elif event.key == pygame.K_RETURN :
        print ("vous avez appuyé sur la touche Entrée")
      else :
        print ("vous avez appuyé sur une touche")
  surf.fill((0,0,0))
  pygame.display.flip()
pygame.quit()
```

Vous trouverez la liste complète des touches dans la documentation officielle de pygame.

2 Le projet

Voici quelques exemples d' animation quand on attend qu' un programme se lance ou qu' un fichier se télécharge :

- https://www.graphisme-echirolles.com/wp-content/themes/cool-blog/assets/loader/style-5.gif
- http://incognitoheureux.i.n.pic.centerblog.net/dd449d89.gif
- https://www.tribalfrance.com/assets/img/gif-attente-10.gif
- https://www.emjysoft.com/images/site/attente.gif*
- https://www.mostiglass.fr/int/i_attente.gif
- https://www.maisonsablayrolles.fr/wp-content/themes/monamourparis/src/assets/loading4.gif
- https://www.pcindustriel.com/images/attente.gif
- https://portail.csb.qc.ca/pluriportail/images/Attente.gif
- https://cdn.dribbble.com/users/347696/screenshots/2665029/dribbble-progress-bar.gif
- https://d34u8crftukxnk.cloudfront.net/slackpress/prod/sites/6/2019-01_BrandRefresh_Old-to-New-Final.gif
- https://vectorified.com/images/waiting-icon-gif-20.png
- https://usagif.com/wp-content/uploads/loading-12.gif

Il y en a plein d' autres : Chargement données Pronote, Chargement vidéo Youtube, Chargement PIX, Attente Windows, Attente Ubuntu, …

Objectif :

1. Avec pygame, programmez une animation style barre de téléchargement
2. Avec pygame, programmez une animation circulaire
3. Inventez votre propre animation d' attente