
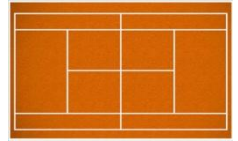


TP utilisant Pygame

Pygame est un module disponible pour Python qui permet de réaliser des programmes ayant une partie graphique plus évoluée et plus dynamique que la console habituelle.

L'objet de ce premier TP, est de réaliser un jeu (style pong) : l'écran est un terrain de tennis (ou autre), une balle  pourra rebondir sur les bords de la fenêtre. Une raquette est manipulable par le joueur, elle permet d'empêcher la balle de sortir du côté gauche de la fenêtre. Toutes les évolutions seront possibles ensuite en fonction du temps disponible. Il est évident que seule une toute petite partie des fonctionnalités de Pygame seront explorées, vous trouverez les ressources documentaires ici : <https://www.pygame.org/docs/>



Le programme réalisé ci-dessous utilisant Pygame peut servir de base pour développer d'autres productions, vous pouvez donc copier/coller cet exemple comme point de départ.

<pre>import pygame,sys,random from pygame.locals import * pygame.init() LargFen=717;HautFen=447 fen=pygame.display.set_mode((LargFen,HautFen)) pygame.display.set_caption("Tennis") fond=pygame.image.load("terraintennis.jpg").convert() balle=pygame.image.load("balle.png").convert() son=pygame.mixer.Sound("pong.wav") pgmFini=False while pgmFini==False: #insertion du fond dans la fenêtre fen.blit(fond,(0,0)) #gestion des événements clavier, souris ... for evenement in pygame.event.get(): if evenement.type==QUIT: pgmFini=True #mettre ici tout ce qui permet l'affichage de la fenêtre #mise à jour de la fenêtre pygame.display.update() pygame.quit() sys.exit()</pre>	<p>Import des modules Pygame, sys et de vos modules import des variables et constantes de Pygame</p> <p>Mise en route du moteur graphique de Pygame</p> <p>Taille de la fenêtre graphique en pixel</p> <p>Initialisation de la variable fen qui contient "un objet fenêtre". Vous verrez en terminale ce que cela signifie.</p> <p>On donne un titre à cette fenêtre</p> <p>Initialisation de 3 variables : une contenant l'image du terrain, une avec l'image de la balle et une avec un son qui sera émis quand la balle ricoche contre un mur.</p> <p>Début de la boucle principale qui s'exécute en permanence</p> <p>On insère le fond dans la fenêtre graphique au point haut gauche de la fenêtre(ce qui efface la fenêtre)</p> <p>On gère les événements (déplacement, clic de souris, pression d'une touche du clavier, ...)</p> <p>Plusieurs événements permettent de fermer la fenêtre, tous déclenchent l'événement de type QUIT</p> <p>Mettre ici tout ce qui permet le réaffichage de la fenêtre, couche après couche.</p> <p>Déclenchement du réaffichage de la fenêtre</p> <p>La fin de la boucle principale est terminée (car Fini=True), on arrête le moteur graphique de Pygame et on rend le contrôle au système d'exploitation</p>
---	---

Plusieurs remarques sont nécessaires dans l'ordre d'apparition :

- Quand vous exécuterez ce programme, la fenêtre graphique va s'afficher, la boucle principale se met en route et le réaffichage est permanent, suffisamment rapidement pour qu'on ne s'en rende pas compte. En fonction de l'occupation du processeur principal et du processeur graphique, le nombre d'images affichées chaque seconde varie, vous n'avez pas à vous en préoccuper, pour le moment au moins.
 - Pour arrêter le programme, commencez toujours par fermer la fenêtre graphique avant de revenir à eduPython sinon, risque de blocage.
 - Dans ce TP, il faut commencer par choisir l'image de votre terrain, puis vous adapterez la taille de la fenêtre en fonction de cette image. D'où le choix surprenant des dimensions 717 et 447 choisies ici. Les dimensions sont en pixels. Un conseil : Laisser les fichiers contenant les images, les sons dans le même dossier que le programme. Astuce : Il est préférable d'utiliser deux variables pour enregistrer les dimensions de la fenêtre et ensuite de toujours positionner les éléments en fonction de ces deux variables. Ainsi il suffit de modifier les contenus des deux variables si on désire changer le look du fond de la fenêtre.
- Choisir une image pour le terrain, une pour la balle et éventuellement un fichier wav pour le bruitage. Faire un copier-coller du programme ci-dessus et le modifier suivant les noms de fichier, taille du terrain. A ce stade, la balle est absente (normal, elle n'a pas encore été placée dans la fenêtre), rien ne se passe, sauf que votre programme actualise la fenêtre à raison d'un peu plus de 1730 images/sec sans rien changer !!, il ne fait rien d'autre que d'attendre un clic de souris, la pression d'une touche du clavier ...)

Suite des remarques :

- Les variables fen, fond, balle, son contiennent des objets. Il s'agit d'une structure de données que nous étudierons plus tard, il s'agit de données comme les variables classiques mais elles disposent de fonctions qui leur sont propres (Cf poly sur les chaînes de caractères qui sont un bon exemple).
- La balle est une image qui peut avoir un fond transparent ou non, la modification sera expliquée plus loin.
- La boucle principale a une fonction principale : Le réaffichage permanent de la fenêtre. Plus la charge de travail du processeur est importante, plus les réaffichages seront espacés dans le temps. Le projet actuel est suffisamment basique pour que les rafraîchissements soient très rapides. Dans cette boucle, on doit donc concevoir le réaffichage de toute la fenêtre en commençant par les images de fond puis incruster toutes les autres images "par dessus".
- Concernant la gestion des événements : la fonction "pygame.event.get" renvoie une liste d'attente de tous les événements non encore traités (pression ou relâchement d'une touche du clavier, déplacement de la souris, mouvement de la molette ...). La boucle "for" permet de les prendre en compte, les uns après les autres. Le tableau de droite donne quelques exemples de type d'événements (variable event.type). Le premier type d'événement "QUIT" est appelé lorsque l'on ferme la fenêtre, par un clic sur la croix rouge de fermeture de la fenêtre ou par un raccourci clavier (Alt+F4). Suivant le type d'événement, on a un complément d'informations, par exemple quelle est la touche pressée, quel bouton de souris est pressé, quel est l'emplacement de la souris, etc.

QUIT	none
ACTIVEEVENT	gain, state
KEYDOWN	key, mod, unicode, scancode
KEYUP	key, mod
MOUSEMOTION	pos, rel, buttons
MOUSEBUTTONUP	pos, button
MOUSEBUTTONDOWN	pos, button
JOYAXISMOTION	joy, axis, value
JOYBALLMOTION	joy, ball, rel

```

1 if event.type == KEYDOWN:
2     if event.key == K_SPACE:
3         print("Espace")
4     if event.key == K_RETURN:
5         print("Entrée")

```

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

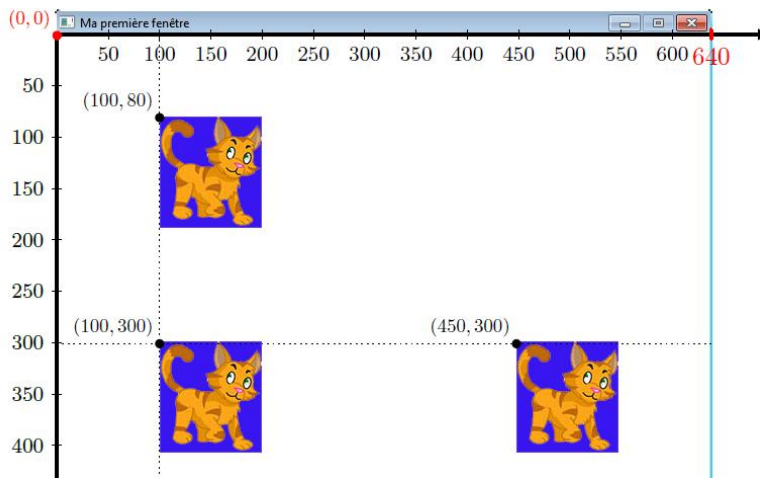
```

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

```

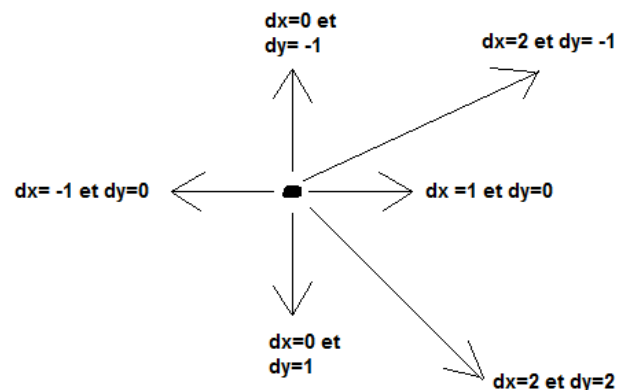
Vous trouverez les codes de chaque touche du clavier sur internet (mots clé : pygame touches clavier)

- Important, toutes les incrustations d'images dans la fenêtre ne seront visibles qu'au prochain rafraîchissement de la fenêtre, c'est à dire lors de l'exécution de : "pygame.display.update()"
- Toutes les coordonnées sont données dans le repère ci-contre, l'axe des abscisses est classique (gauche vers droite), l'axe des ordonnées pointe vers le bas de la fenêtre.



Dans la mesure où la balle va se déplacer on va noter x_b, y_b ses coordonnées à tout instant.

- Initialiser x_b et y_b à l'emplacement d'où partira la balle au démarrage du programme, par exemple : $x_b = 3 * \text{LargFen} // 4$ et $y_b = \text{HautFen} // 2$
- Initialiser largbal et hautbal avec la largeur et la hauteur de la balle.
A noter on peut utiliser la fonction `get_rect()` qui renvoie un objet de type rectangle. A ce titre `R=raquet.get_rect()` permet ensuite de connaître sa hauteur à tout moment grâce à `R.height`, sa largeur est `R.width` (d'autres informations sont disponibles)
- Dans la boucle principale, insérer l'image de la balle dans la fenêtre (avec "fen.blit ...")
- Initialiser deux variables dx et dy (les coordonnées du vecteur vitesse) qui vont contenir le pas de déplacement de la balle à chaque rafraîchissement de la fenêtre. Vous pourriez commencer par 1 et 1, ainsi au démarrage du programme, la balle partira vers la droite et en bas. En augmentant les valeurs de dx et de dy on accélère le mouvement, mais si les valeurs sont trop grandes, le déplacement de la balle deviendra saccadé.
- Il ne reste plus qu'à provoquer dans la boucle principale le déplacement de la balle avec : $x_b = x_b + dx$ et $y_b = y_b + dy$



Si on considère que les impacts sur les bords de la fenêtre s'effectuent sans effet, le changement de trajectoire de la balle est simple : Si la balle touche l'un des bords droit ou gauche, il suffit de changer le signe de dx ($dx = -dx$) et si elle touche le bord haut ou bas de la fenêtre, on change le signe de dy.

- Ajouter à votre programme le test de collision de la balle avec les bords de la fenêtre. Attention de bien tenir compte de la largeur et de la hauteur de l'image de la balle, par exemple : si la future abscisse de la balle (son point haut-gauche) est à gauche de l'écran ou si le point situé en haut à droite de l'image est à droite de la fenêtre, on change la direction de la balle et on fait un bruit (ou plus exactement : on demande au son qui est contenu dans la variable Son de s'exécuter une fois).

```
if xb+dx<0 or xb+largbal+dx>largfen:  
    dx=-dx  
    Son.play()
```

Il est probable qu'à ce stade, votre jeu fonctionne mais qu'il soit injouable tellement la balle se déplace vite. Nous allons donc ralentir la boucle principale.

Dans la boucle principale, par exemple après la mise à jour de la fenêtre (c'est-à-dire après la ligne `pygame.display.update()`), insérer la commande : `pygame.time.delay(5)` ce qui fait patienter l'exécution du programme pendant 0.005 seconde. Ce paramètre sera à adapter en fonction de la vitesse de l'ordinateur et des tâches à accomplir.

On va maintenant ajouter sur le bord gauche de la fenêtre, une raquette manipulable par le joueur. Celui-ci pourra la faire monter ou descendre à l'aide des flèches haut et bas. Si la balle percute la raquette, elle rebondit, si la balle sort de la fenêtre par la gauche, le programme s'arrête.

- Choisir une image de raquette (un simple rectangle conviendra) de dimensions en rapport avec votre fenêtre
- Charger l'image dans une variable, choisir deux variables pour désigner les coordonnées du point haut gauche de la raquette : `xraq` et `yraq` ainsi que deux variables : `largraq` et `hautraq` pour sa largeur et sa hauteur.

Dans la boucle principale vous détectez déjà des événements comme la pression d'une touche. Cependant on ne peut pas utiliser cette détection d'événements, en effet on peut penser que l'utilisateur pourra presser la flèche haut pour faire monter la raquette et qu'il laissera le doigt appuyé sur la touche. Or au moment précis où il appuie, un événement est bien détecté mais ensuite plus aucun événement n'apparaît jusqu'à ce qu'il la relâche. Il faut donc connaître à tout instant quelles sont les touches pressées. Pygame en gère la liste et peut donc vous renseigner à tout instant. Dans l'exemple ci-contre, on teste si la flèche haut est actuellement pressée, et si oui, on remonte la raquette de 2 pixels.

```
ListTouche=pygame.key.get_pressed()  
if ListTouche[pygame.K_UP]==True:  
    yraq=yraq-2
```

- Créer de même la gestion de la flèche bas. Pour les deux mouvements possibles, assurez-vous que la raquette ne puisse sortir de l'écran.
- Le point le plus délicat est désormais de gérer l'impact éventuel de la balle contre la raquette ou contre le bord gauche de la fenêtre.
- Votre boucle principale doit avoir pris un embonpoint qui vous empêche de la voir sur un écran : Découper votre programme en plusieurs fonctions.

Evolutions possibles :

- Faire varier la vitesse de la balle, ajouter une seconde raquette, modifier les angles de trajectoires de la balle...
- Compléter les sons suivants les obstacles rencontrés par la balle
- Afficher des scores
- Déplacer la raquette avec la souris

Afficher une chaîne de caractères

Supposons que l'on veuille afficher la chaîne `Chn="truc à afficher"`.

Il faut commencer par déclarer la police utilisée, on peut le faire au début du programme ou à tout moment :
`font=pygame.font.SysFont("broadway",24,bold=False,italic=False)` pour choisir la police `broadway`, en taille 24, pas de gras, pas d'italique.

`texte=font.render(Chn,1,(R,G,B))` pour créer l'objet texte dans la couleur (R,G,B)

Ensuite le texte s'insère dans une fenêtre comme on insère une image :

`fen.blit(texte,(30,30))` : ce qui place le texte dans la fenêtre `fen` au point de coordonnées (30,30)

Mise en place de la transparence d'une image

Si la transparence existe déjà dans l'image, il suffit en Python de charger l'image puis prendre en compte la transparence grâce à la méthode `convert_alpha()` en suivant cet exemple :

```
MonImage = pygame.image.load("image.png").convert_alpha()
```

Si la transparence n'existe pas dans l'image, on charge l'image et on déclare la couleur à transformer en "couleur transparente"

```
MonImage = pygame.image.load("image.png").convert()
```

```
MonImage.set_colorkey(Coul) où Coul est la couleur à transformer au format (R,G,B)
```

Mesurer le temps, une durée

On peut utiliser `pygame.time.get_ticks()` : cette fonction renvoie le temps écoulé depuis l'exécution de `pygame.init()` en millisecondes. Si on veut mesurer la durée d'un événement, il suffit donc de faire une différence.

Un fond uni sans image

Au lieu d'écrire : `fen.blit(fond,(0,0))` où `fond` contient l'image de fond, on saisit : `fen.fill((r,v,b))` où `r,v,b` sont trois octets définissant la couleur du fond

Pour activer une fonction à intervalle régulier

Avant la boucle principale, on définit une variable `noEvenementPerso = USEREVENT+1`. (`noEvenementPerso` doit avoir une valeur comprise entre `USEREVENT`, une variable gérée par `pyGame` et `NUMEVENTS` (numéro maximal d'un événement, c'est une constante de `Pygame`))

```
pygame.time.set_timer(depla,150) #pour mettre l'évènement dans la file d'attente des événements toutes les 150 ms.
```

Ensuite dans la boucle principale, il ne reste plus qu'à tester l'arrivée de l'événement :

```
for event in pygame.event.get():
```

```
    if event.type==noEvenementPerso:
```

```
        #on appelle la fonction à exécuter
```

Faire apparaître un événement de manière aléatoire

Il suffit de mettre dans la boucle principale, un test du style :

```
if randint(1,100)<=60:  
    action
```

pour que action s'exécute dans 60% des cas.

Mettre une musique de fond

Dans les initialisations :

```
fondsonore =pygame.mixer.Sound("BFLATBLS.wav") #Chargement d'un son
```

Puis on lance la musique avec fondsonore.play()