Bases de l'utilisation de Pygame

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

Remarques préliminaires :

a) Pour interagir avec pygame on utilise des évènements qui peuvent être un clic de la souris, un appui sur une touche etc….Ces évènements sont stockées dans une file qui s'appelle pygame.event. Les événements sont traités dans leur ordre chronologique. On peut avoir le type de l'évènement grâce à la commande event.type. Avec des instructions conditionnelles, on peut gérer les actions correspondant aux différents évènements.

b)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..).

1) **Création d'une fenêtre** :

Voici le schéma d'un programme créant une fenêtre Pygame :

```
import pygame  #importation de pygame
import os
import traceback #module pour récupérer des infos sur les erreurs
from pygame.locals import * #on importe les constantes de pygame
pygame.init()  #on lance pygame
#toujours encadrer vos programmes pygame par try et finally ce qui permet de fermer correctement la fenêtre pygame en cas d'erreur
try:
    #pour positionner la fenêtre sur l'écran à la position (400,600).
    os.environ['SDL_VIDEO_WINDOW_POS']="400,600"
    #création d'une fenêtre
    fenetre=pygame.display.set_mode((640,480))#fenêtre de taille 640*480
    continuer=1
    #boucle perpétuelle qui permet de garder la fenêtre ouverte
     while continuer:
        for event in pygame.event.get():
            #pygame prend le premier évènement de la file
            if event.type==QUIT:
                #l'évènement QUIT correspond au clic sur la croix
                continuer=0  #permet de quitter la boucle
```

```
except :
    traceback.print_exc()
finally:
    pygame.quit()
    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()

Mise en œuvre : Créer une fenêtre de taille 300*200 coloriée en bleu

2) **Image**.

Pour afficher une image :

- Il faut charger l'image (après avoir lancé pygame) :
  image1=pygame.image.load("mon_image.jpg").convert()
  Entre les guillemets on met l'adresse de l'image par rapport au dossier où se trouve le programme (voir les adresses dans la page sur le HTML sur le site).
  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))
  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))
  Remarque : dans la boîte à couleur on peut "récupérer" les valeurs RGB d'une couleur d'une image en utilisant la petite "pipette"
- redimensionner une image :
  image1=pygame.transform.scale(image1,(30,30)) pour avoir une image 30*30

Mise en œuvre : Afficher l'image « perso1.png » 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.

3) **Evènements** .

Exemple :

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

```
if event.type==KEYDOWN:   #appui sur une touche
        if event.key==K_UP:   #la touché est la flèche vers le haut
                déplacer le personnage  d'une case vers le haut…(voir plus loin)
        if event.key==K_DOWN:
                déplacer le personage d'une case vers le bas
```

Les différents types d'évènements :

- 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)
  On peut alors récupérer la touche appuyée avec la commande :
  event.key qui peut prendre différentes valeurs.
  Pour avoir une liste complète:
  http://thepythongamebook.com/fr:glossaire:p:pygame:keycodes
  - K_a pour le a (et pareil pour le reste de l'alphabet)
  - K_0 pour les chiffres en haut du clavier
  - K_F1 pour la touche F1
  - K_SPACE pour la touche espace
  - K_RETURN pour la touche ENTER
  - K_ESCAPE pour la touche Echap

- K_UP pour la flèche vers le haut (DOWN, LEFT, RIGHT)
- K_KP0 pour le 0 du pavé numérique

Si on maintient la touche enfoncée on peut répéter l'évènement avec la commande :
pygame.key.set_repeat(400,30) 400 est le délai (en millisecondes) avant que la répétition ne commence et à partir de là, un nouvel évènement est généré toutes les 30 ms. Si votre touche provoque un déplacement et que vous appuyez 550ms sur la touche, le déplacement va se faire une première fois au début, il y aura 400ms d'attente , puis il y aura un déplacement à 400ms, à 430 ms, à 460 ms….Le délai est là pour qu'il n'y ait pas de répétitions indésirables.
On peut récupérer les caractères tapés (par exemple si le joueur tape son nom et qu'on veut le récupérer) , après un if event.type==KEYDOWN:
carac=event.dict['unicode'] si on a tapé la touche K_a alors carac='a'

- 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'évènements peut rapidement remplir la file des évènements….

Pour attendre les évènements :

- pygame.event.wait() :
  attend le prochain évènement et le programme se met en pause en attendant
- pygame.event.poll():
  fait la même chose, mais le programme continue à tourner.

Mise en œuvre :Crée une fenêtre de taille 600*400 et écrire un programme qui affiche « perso1.png » à l'endroit où on a cliqué avec la souris. Comment faire pour qu'un seul personnage soit affiché à chaque fois ?

4) **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 :
  objet_rect=objet.get_rect()
  #objet_rect vaut alors (x,y,largeur,hauteur) où (x,y) est la position de l'objet
  #et largeur et hauteur sont les dimensions du rectangle qui entoure l'image.

- pour créer un rectangle :
  mon_rect=pygame.Rect(x,y,largeur, hauteur) où x,y correspond à la position du rectangle
- pour déplacer un rect (et donc l'objet qui est dedans) :
  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 attributs d'un rect appelé r :

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

Mise en œuvre : Afficher « perso1.png » , récupérer son « rect » et l'afficher dans la console . Recommencer, mais choisir vous-même le « rect » pour que le personnage soit au centre de l'écran.

Exemple de déplacement d'un personnage :

```
import pygame
from pygame.locals import *
import traceback
pygame.init()
try:
    fenetre=pygame.display.set_mode((640,480))
    fenetre.fill((255,255,255)) #on remplit la fenêtre de blanc
    image1=pygame.image.load("perso1.png").convert_alpha()
    image1_rect=image1.get_rect() #on crée un rectangle entourant l'image
    fenetre.blit(image1,image1_rect) #on blitte l'image dans ce rectangle
    pygame.key.set_repeat(400,30)  #on active la répétition des touches
    pygame.display.flip()
    continuer=1
    while continuer:
        for event in pygame.event.get():
            #attention, toujours prévoir un moyen de sortir de la boucle
            if event.type==QUIT:
                continuer=0
```

```
        elif event.type==KEYDOWN:
            if event.key==K_LEFT:
                image1_rect=image1_rect.move(-5,0)
                 #on déplace le rectangle de l'image de 5 pixel vers la gauche
                if image1_rect.left<0:
                 #si le bord gauche de l'image sort du cadre, on remet l'image à droite
                    image1_rect.left=610
            if event.key==K_RIGHT:
                image1_rect=image1_rect.move(5,0)
                if image1_rect.right>640:
                    image1_rect.right=30
            #si on ne re-remplit pas le fond, on verra l'objet aux deux positions
            fenetre.fill((255,255,255))
            fenetre.blit(image1,image1_rect)
            pygame.display.flip()
except :
    traceback.print_exc()
finally:
    pygame.quit()
    exit()
```

Mise en oeuvre : 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 ».

5) **Les textes**

Voici comment afficher une chaîne.

- chaine="ma chaine sur une seule ligne" (penser à la méthode format pour afficher une chaîne contenant des données variables, par exemple le nom du joueur ou le score)
- font=pygame.font.SysFont("broadway",24,bold=False,italic=False) pour choisir la police broadway , en taille 24, pas de gras, pas d'italique.
- text=font.render(chaine,1,(R,G,B)) pour créer l'objet texte
- fenetre.blit(text,(30,30)) où (30,30) correspond à la position du texte
- fenetre.display.flip() pour rafraîchir l'affichage

Mise en oeuvre : 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 .

6) **Les dessins**

On peut dessiner des formes dans pygame :

- un rectangle : mon_rectangle=pygame.draw.rect(surface,color,rect,épaisseur) 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) ,épaisseur est l'épaisseur du trait (0 donne un rectangle plein)
- un cercle : mon_cercle=pygame.draw.circle(surface,color,pos_centre,rayon,épaisseur) pos_centre est du type (x,y)
- une ligne : ma_ligne=pygame.draw(surface,color,position_départ,position_arrivée,épaisseur)

Inutile de blitter, mais il faut rafraîchir 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
surf.fill((R,G,B), rect) remplira le rect seulement où rect est un pygame.Rect ou (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.

Mise en œuvre : écrire un programme qui trace les segments entre les points où on aura cliqué successivement avec la souris.

7) **Le temps**

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

import pygame

```
import os
import traceback
from pygame.locals import *
pygame.init()
try:
    fenetre=pygame.display.set_mode((640,480))
    fenetre.fill((255,255,255))
    image1=pygame.image.load("perso1.png").convert_alpha()
    image1_rect=image1.get_rect()
    fenetre.blit(image1,image1_rect)
    pygame.display.flip()
    #on crée un évènement qui n'est pas un évènement prédéfini par le système : il
        faut lui donner un numéro pour que pygame le gère
    depla=USEREVENT+1 #on donne un numéro à l'évènement entre USEREVENT
        #et NUMEVENTS (qui sont des constantes de Pygame :sur mon ordinateur
        # cela se situe entre 25 et 31. On peut les faire afficher dans la console)
        #on peut aussi utiliser directement un numéro à la place de "depla", ici 25)
    pygame.time.set_timer(depla,150)  #l'évènement va se mettre dans la file des
        #évènements toutes les 150 ms
    continuer=1
    while continuer:
        for event in pygame.event.get():
            if event.type==QUIT:
                continuer=0
            elif event.type==depla:  #gestion de l'évènement répétitif
            #on fait ce que l'on veut, ici on déplace le personnage en diagonale
                image1_rect=image1_rect.move(3,3)
                fenetre.fill((255,255,255))
                fenetre.blit(image1,image1_rect)
                pygame.display.flip()
except :
    traceback.print_exc()
finally:
    pygame.quit()
    exit()
```

8) **Animation**

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.

Voici un exemple :

```
import sys,pygame
import traceback
from pygame.locals import *
blue=145,197,235
clock=pygame.time.Clock()
 #voir à la fin, on veut choisir le nombre d'images par secondes
def image(chaine):
    """fonctions qui charge les sprites et rend le fond transparent etc.."""
    im=pygame.image.load(chaine)
    im=im.convert_alpha(im)
    return im
try:
    screen=pygame.display.set_mode((400,151))
    b1=image("course4.png")
    b2=image("course2.png")
    b3=image("course3.png")
    b=[b1,b2,b3]
    #on fait une liste des images du personnage dans différentes positions
    continuer=1
    i=0   #index de l'image qu'on va afficher
    while continuer:
        for event in pygame.event.get():
            if event.type in (QUIT, KEYDOWN):#pour quitter
                continuer=0
        screen.fill(blue)
        #on va afficher successivement les images de la liste en revenant au début
        #quand on est à la fin (avec i%3)
        screen.blit(b[i%3],(180,54))   #(180,54) est la position où on blitte
        i=i+1   #pour afficher ensuite l'image suivante
        clock.tick(10) #on limite le nombre d'images à  10 images par seconde.
                        #sinon on ne voit rien, c'est trop rapide
        pygame.display.flip()
except :
    traceback.print_exc()
finally:
    pygame.quit()
```

Beaucoup d'autres instructions existent et sont décrites dans la documentation de pygame (en anglais) : **http://www.pygame.org/docs/ref/**

Transparence dans GIMP :

avec les sprites :
sélectionner l'un des sprite
sélectionner rect
édition copier
édition coller comme……image

Dans la nouvelle image :
sélection par couleur
cliquer sur le fond
puis
sélection inverser
puis
copier
coller comme…..image

exporter……..choisir png