La bibliothèque `python` : `pygame` [http://www.pygame.org/] permet de gérer graphisme et événements.

Nous avions vu au 1er trimestre que l'on pouvait appeler une bibliothèque à l'aide de `import` [*e.g.* `from math import sqrt`].

Pour charger la bibliothèque `pygame` et toutes les fonctions possibles, commencer par écrire :

```
import os, pygame as pg
from pygame.locals import *
pg.init()                       # initialise pygame sur notre systeme
PATH="C:/monRepertoire/ISN"     # pyzo travaille avec os.getcwd()
```

rq : ⚠ Pour éviter tout problème de chemin, on peut définir la variable `PATH` à l'aide de la fonction `os.path.join()`. Laquelle s'occupera d'ajouter les "/" ou "\".

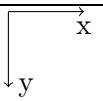
Ainsi `C:/monRepertoire/ISN` pourra s'écrire `os.path.join("C:","monRepertoire","ISN")`.

I Découverte des surfaces

On pourra se référer à la page [http://www.pygame.org/docs/ref/display.html] (et les autres /draw.html, /image.html, /surface.html).

On définit une fenêtre-écran à l'aide des lignes suivantes :

```
rwindows = Rect(0,0,800,600)    # definit un rectangle 800*600
screen = pg.display.set_mode(rwindows.size, 0)    # screen est la surface originale
```

⚠ l'origine (0,0) est en haut à gauche et (800,600) est en bas à droite. 

1 Dessiner avec pygame

On affiche un écran de couleur (RGB - rappel [http://rapidtables.com/web/color/RGB_Color.htm]) :

```
WHITE, RED, GREEN, BLUE, BLACK = (255,255,255), (255,0,0), (0,255,0), (0,0,255), (0,0,0)
fond = pg.Surface(rwindows.size)      # fond est une surface additionnelle
fond.fill(WHITE)
fond.convert()              # conversion au format de couleurs de screen
screen.blit(fond, (0,0))    # fond est ajoute a screen
pg.display.flip()           # tous les elements de screen sont affiches
```

Exo 1 : Proposer un programme qui affiche un fond d'écran rouge puis jaune.

On peut dessiner des polygones, cercles, lignes, ellipses, etc. en spécifiant la surface dans laquelle on les dessine (ici `fond`), leur couleur RGB, leurs caractéristiques géométriques puis L précise la largeur du train (si L n'est pas fourni l'objet sera rempli de la couleur voulue).

```
pg.draw.polygon(fond, WHITE, [(xA,yA),(xB,yB),(xC,yC)], L)    # polygone
pg.draw.circle(fond, RED, (x0,y0), r, L)                      # cercle (O,r)
pg.draw.line(fond, RED, (xA,yA), (xB,yB), L)                  # segment [AB]
pg.draw.ellipse(fond, WHITE, (x0,y0,xH,yH))                   # ellipse
screen.blit(fond, (0,0))    # NE PAS OUBLIER de re-associer fond a screen
pg.display.flip()           # puis de re-afficher screen
```

Exo 2 : Proposer un programme qui affiche sur un écran de couleur bleu un soleil et deux nuages.

2 Afficher une image

On peut souhaiter afficher une image ou une photo (supposé déposées dans le répertoire `PATH/images/`).

```
img = pg.image.load(os.path.join(PATH,'images','MonImage.png')) # creer une surface img
img.get_width(), img.get_height()                               # largeur/hauteur de l'image
screen.blit(img,(100,50)) # place l'image a 100 pixels du bord gauche & 50 du bord haut
pg.display.flip()         # affiche le nouvel ecran
```

Exo 3 : Afficher une image (*e.g.* `ball.gif`) en haut à droite et en bas à gauche de la fenêtre. Préciser les dimensions horizontales ainsi que verticales de l'image.

3 Mouvement des dessins/images

Afin de mieux voir les modifications, on dessinera une échelle définie par la portion de code suivante :

```
fond.fill(BLUE)                    # fond bleu uniforme
for i in range(0,800,100):
    pg.draw.line(fond,WHITE, (i,10),(i,100),5)
```

⚠ en fin de boucle on pourra ajouter `pg.time.delay(10)` pour ralentir chaque tour de 10 ms.

On souhaite afficher l'image précédente et la déplacer de gauche à droite

```
for i in range(50,200):
     fond.blit(img, (i,50))        # place l'image dans le fond au point (i,50)
     screen.blit(fond,(0,0))       # tout le fond est re-associe a l'ecran
     pg.display.flip()             # reaffiche l'ecran
```

? Que se passe-t-il ? Quel est le problème ?

Afin de palier ce problème, on peut utiliser la fonction `scroll()`. À la place du code précédent, essayer le code suivant (avec ou sans **rectangle**) :

```
rimg=img.get_rect()                # definit un rectangle des dimensions de l'image
fond.blit(img,(100,50))            # associe l'image au fond
fond.convert()
for i in range(50,200):
     fond.scroll(1,0)              # decale l'ecran de dx=1 et dy=0
     screen.blit(fond,(0,0))       # tout le fond est re-associe
     pg.display.update()          ## essayer aussi avec : .update(rimg)
```

? Est-ce que cela fait exactement ce que l'on veut ?

On va donc associer l'image à l'écran (ou une surface autre que le fond). À chaque étape, on réaffiche le morceau de fond qui a été écrasé par l'image et on réaffiche l'image.

```
screen.blit(fond,(0,0))                    # tout le fond est re-associe a l'ecran
for i in range(50,200):
     rimg = Rect( (i,50),img.get_size() )# rimg : rectangle de dimensions de img
     screen.blit(fond,rimg,rimg)           # seul le fond contenu dans rimg est re-associe
     screen.blit(img,rimg)                 # img est associe a l'ecran
     pg.display.update()
```

Exo 4 : a) Expliquer les 3 phénomènes que vous avez visualisés.

b) Proposer un code qui fasse traverser la balle sur la largeur de la fenêtre sans faire bouger l'écran.

4 Afficher un texte

En plus d'une image, on peut afficher un message dans le shell ou dans la fenêtre graphique.

```
police = pg.font.SysFont("monospace",15)   # on definit la police et la taille
imageTxt =police.render("Hello World!",True,RED)   # imageTxt contient le msg
screen.blit(imageTxt,(200,100))     # on positionne l'image
pg.display.flip()                   # on affiche le nouvel ecran
```

Exo 5 : Afficher un texte de couleur blanche en bas de la fenêtre graphique.

II Gestion d'événements

On pourra se référer à la page [http://www.pygame.org/docs/ref/event.html] (et les autres /mouse.html, /key.html).

`pygame` peut détecter des actions clavier ou souris, ses actions s'appelent des `events`. Un événement est guetté par la fonction `pygame.event.wait()`, `pygame.event.poll()` ou bien `pygame.event.get()`.

⚠ Il est conseillé avant chaque essai de faire un `restart` du `shell`.

1 Actions souris

Ici on détecte la fermeture de la fenêtre à l'aide de la souris, on peut écrire :

```
bool = True
while bool :
    event = pg.event.poll()    # on recupere le premier element de la liste
    if event.type == QUIT :    # un clic a ete fait sur la croix "fermer la fenetre"
        bool = False
pg.quit()                      # on quitte pygame
```

Un événement de souris peut être de trois types :

a) `MOUSEMOTION` : la souris se déplace
b) `MOUSEBUTTONDOWN` : un bouton est appuyé
c) `MOUSEBUTTONUP` : un bouton est relâché

Par exemple, ici on détecte les coordonnées du pointeur de la souris au moment d'appuyer sur le bouton.

```
if event.type == MOUSEBUTTONDOWN :
    x,y = pg.mouse.get_pos()       # on recupere les coordonnees du clic
    print("clic aux coordonnees ({0},{1})".format(x,y))
```

Exo 6 : a) Ajouter ce code dans la boucle `while bool:` précédente et essayer.
b) Écrire un code qui affiche les coordonnées du point où l'utilisateur à cliquer avec la souris.

2 Actions clavier

On peut connaître un événement clavier grâce à la fonction `key.get_pressed()` qui renvoie l'état courant du clavier (*cf.* corrige Exo9) ou bien en récupérant l'`event` associé qui ne possède alors que deux types :

a) `KEYDOWN` : la touche est enfoncée
b) `KEYUP` : la touche est relâchée

Ici selon la touche enfoncée (`KEYDOWN`), un message est écrit dans le shell :

```
if event.type == KEYDOWN:
    if event.key == pg.K_a :
        print("vous avez appuye sur la touche a")
    if event.key in {pg.K_UP, pg.K_DOWN, pg.K_LEFT, pg.K_RIGHT} :
        print("vous avez appuye sur une fleche")
    if event.key == pg.K_ESCAPE :
        bool = False
```

Exo 7 : Ajouter ce code dans la boucle `while bool:` précédente et essayer.

Voici quelques noms de touche (on considère le clavier américain) :

- pg.K_DOWN
- pg.K_UP
- pg.K_LEFT
- pg.K_RIGHT
- pg.K_PAGEDOWN
- pg.K_PAGEUP
- pg.KP_0 ; ...
- pg.KP_9
- pg.K_DELETE
- pg.K_RETURN
- pg.K_F1 ; ...
- pg.K_F15
- pg.K_SPACE
- pg.K_ESCAPE
- pg.K_a ; ...
- pg.K_z

Travail à faire pour la prochaine séance

Exo 8 : a) Écrire un code qui affiche un écran noir. Cet écran devient bleu si la touche 'b' est enfoncée, devient vert si la touche 'v' est enfoncée et enfin rouge si la touche 'r' est enfoncée.
b) Modifier le code afin que la couleur s'affiche 2 secondes puis le fond noir revienne.
c) Modifier le programme afin que la couleur redevienne noire au moment où l'utilisateur relâche la touche.
Exo 9 : Permettre à l'utilisateur de déplacer une image horizontalement avec les touches directionnelles.