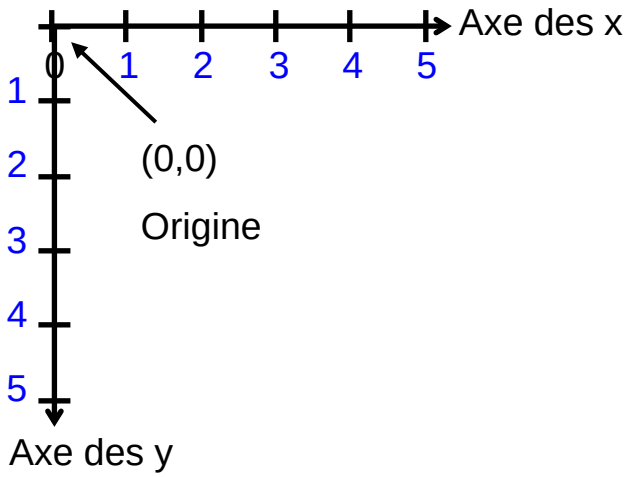
Introduction à Pygame

Les coordonnées écran de Pygame

L'origine des axes est située dans le coin en haut à gauche de l'écran.



Par rapport au système de coordonnées cartésien, l'axe des y est inversé.

D'autre part l'écran couvre uniquement le quart inférieur droit. Si on dessine un objet en utilisant des coordonnées négatives, le dessin sera réalisé hors écran.

La librairie Pygame

Cet ensemble de fonctions, permet de dessiner des formes, d'afficher des images, de les animer, d'interagir avec le clavier, la souris, de jouer des sons etc.

Un programme Pygame commence toujour par :

```
import pygame
pygame.init()
```

et se termine par

```
pygame.quit()
```

Surtout n'appelez pas votre programme pygame.py, car alors ce n'est plus la librairie pygame qui sera importée, mais votre programme.

La définition des constantes utilisées dans le programme

On peut définir, en début de programme les variables que l'on va utiliser.

Par exemple, on peut définir les couleurs utilisées, la valeur de pi, si l'on trace des cercles etc. Par convention, le nom des constantes est en majuscules.

Les couleurs peuvent être définies en donnant les valeurs de chaque composante : Rouge, Vert, Bleue.

```
NOIR = (0, 0, 0)
BLANC = (255, 255, 255)
VERT = (0, 255, 0)
BLEU = (0, 0, 255)
ROUGE = (255, 0, 0)
```

Cette application en ligne fournit la valeur de chacune des composantes de n'importe quelle couleur. https://www.webfx.com/web-design/color-picker/

Vous pouvez également utiliser les outils permettant de définir une couleur personnalisée dans paint ou dans Word.

Les valeurs peuvent être aussi définie de la façon suivante, 0x introduisant une valeur héxadécimale :

BLEU = (0x00, 0x00, 0xFF) #0 = zéro

Ouvrir une fenêtre

```
taille = (640, 480) #fenêtre de 640 pixels de large et 480 pixels de haut
ecran = pygame.display.set_mode(taille)
```

set_mode permet d'ouvrir une fenêtre, avec une zone de titre, des boutons pour minimizer ou fermer la fenêtre etc.

Pour mettre un titre dans la fenêtre :

```
pygame.display.set_caption(" mon premier jeu pygame ")
```

Interagir avec le joueur

Il s'agit maintenant de mettre en place le cœur du programme.

Celui-ci est composé d'une boucle qui ne prend fin que lorsque le joueur ferme la fenêtre.

```
arret = False
horloge = pygame.time.Clock()
```

Le jeu s'arrêtera lorsque arret prendra la valeur True.

horloge va permettre d'indiquer au programme quand il devra rafraichir l'écran.

La boucle principale :

```
#-------------------------------------------------------------
while not arret :
    #boucle des évènements
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            arret = True
        …
    #logique du jeu
    …
    #Code traçant les objets du jeu
    #commencer par effacer l'écran
    ecran.fill(BLANC)
    … Code traçant les objets du jeu
    #Mettre l'écran à jour
    pygame.display.flip()
    #Mise à jour de l'écran à raison de 60 images/seconde
    horloge.tick(60)
```

La boucle des évènements

Tous les évènements doivent être traités **dans une seule boucle**.

En effet la fonction **pygame.event.get()** récupère tous les évènement et les place dans une liste.

Le programme utilise une boucle for, qui va chercher un à un les évènements inscrits dans la liste et teste ensuite leur nature.

On parcourt ainsi la liste de tous les évènements reçus.

Clavier

Les évènements claviers sont : KEYDOWN, KEYUP valables pour toutes les touches.

Pour une touche en particulier il faut rajouter un test qui détermine la touche pressée :

```
if event.type == KEYDOWN and event.key == K_SPACE :
    … #la touche espace a été enfoncée.
```

Ou si l'on veut tester plusieurs touches :

```
if event.type == KEYDOWN :
    if event.key == K_SPACE :
    … #la touche espace a été enfoncée.
    if event.key == K_RETURN :
    … #la touche entrée a été enfoncée.
```

event.key peut prendre les valeurs suivantes :

Lettres : K_a ... K_z

Nombres : K_0 ... K_9

Controles:
K_TAB, K_RETURN, K_ESCAPE, K_SCROLLOCK, K_SYSREQ, K_BREAK, K_DELETE, K_BACKSPACE, K_CAPSLOCK, K_CLEAR, K_NUMLOCK

Ponctuation: K_SPACE, K_PERIOD, K_COMMA, K_QUESTION, K_AMPERSAND, K_ASTERISK, K_AT, K_CARET, K_BACKQUOTE, K_DOLLAR, K_EQUALS, K_EURO, K_EXCLAIM, K_SLASH, K_BACKSLASH, K_COLON, K_SEMICOLON K_QUOTE, K_QUOTEDBL, K_MINUS, K_PLUS, K_GREATER, K_LESS

Parenthèses:
K_RIGHTBRACKET, K_LEFTBRACKET, K_RIGHTPAREN, K_LEFTPAREN

Touches F:
K_F1 ... K_F15

Touches d'édition:
K_HELP, K_HOME, K_END, K_INSERT, K_PRINT, K_PAGEUP, K_PAGEDOWN
K_FIRST, K_LAST

Clavier numérique:
K_KP0 ... K_KP9, K_KP_DIVIDE, K_KP_ENTER, K_KP_EQUALS, K_KP_MINUS
K_KP_MULTIPLY, K_KP_PERIOD, K_KP_PLUS

SHIFT ,CTL,ALT etc:
K_LALT, K_RALT, K_LCTRL, K_RCTRL, K_LSUPER, K_RSUPER, K_LSHIFT,
K_RSHIFT, K_RMETA, K_LMETA

Flèches:
K_LEFT, K_UP, K_RIGHT, K_DOWN

Autres:
K_MENU, K_MODE, K_PAUSE, K_POWER, K_UNDERSCORE, K_HASH

Souris:

Les évènements souris sont : MOUSEBUTTONDOWN et MOUSEBUTTONUP pour tous les boutons de la souris.

Pour savoir quel bouton a été enfoncé ou relaché, il faut ajouter un test sur **event.button** qui peut prendre les valeurs :

1 bouton gauche

2 bouton milieu

3 bouton droite

4 molette haut

5 molette bas.

event.pos renvoie dans un tuple les coordonnées de la souris au moment du clic.

event.pos[0] = abscisse_clic et **event.pos[1] = ordonnee_clic**

Tracer un rectangle qui rebondit

Nous allons mettre en œuvre Pygame, dans un programme qui trace un rectangle qui se déplace dans la fenêtre et rebondit sur les bords.

Tracer un rectangle

La définition de la fonction dessinant un rectangle est :

pygame.draw.rect(Surface, color, Rect, width=0): return Rect

Surface = variable représentant l'écran

Color = couleur des bords du rectangle ou dremplissage

Rect = rectangle définit par coordonnées de son coin en haut à gauche, largeur, hauteur

Width = épaisseur du trait. Si 0 ou si cette valeur est omise, le rectangle sera plein.

La valeur retournée par la fonction est le rectangle passé en paramètre. On peut ignorer cette valeur de retour.

```
pygame.draw.rect(ecran,NOIR,[20,20,50,50])
```

dessine un rectangle noir de 50 pixels de large et 50 pixels de haut, dont le coin en haut à gauche est situé à 20 pixels en x et 20 pixels en y, par rapport au coin en haut à gauche de la fenêtre.

Pour déplacer le rectangle, il suffit dans la boucle après avoir effacé l'écran, de redessiner le rectangle **à de nouvelles coordonnées** :

```
#Code traçant les objets du jeu
    #commencer par effacer l'écran
    ecran.fill(BLANC)
    #Code traçant les objets du jeu
    Tracer le rectangle aux coordonnées x, y
    Calculer les nouvelles coordonnées du rectangle
```

Nous allons rajouter **avant** la boucle principale 4 variables :

```
#coordonnées de départ du rectangle
rect_x = 20
rect_y = 20
```

```
#déplacement du rectangle en x et y
rect_change_x = 5
rect_change_y = 5
```

et dans la boucle principale :

```
#Code traçant les objets du jeu
#commencer par effacer l'écran
    ecran.fill(BLANC)
    pygame.draw.rect(ecran, NOIR, [rect_x, rect_y, 50, 50])
    #changer la position du rectangle
    rect_x+=rect_change_x
    rect_y+=rect_change_y
```

Lorsque le rectangle dépasse les limites de la fenêtre, il disparait.

Il faut donc rajouter quelques lignes de code, pour faire rebondir le rectangle sur les bords de la fenêtre. Pour cela, il suffit d'inverser le signe du déplacement en x ou en y.

```
if rect_y > 430 or rect_y < 0:
        rect_change_y = rect_change_y * -1
    if rect_x> 590 or rect_x < 0:
        rect_change_x = rect_change_x *-1
```

Le code complet de ce premier programme Pygame

```
import pygame
pygame.init()
#----------Nos constantes----------
NOIR = (0, 0, 0)
BLANC = (255, 255, 255)
VERT = (0, 255, 0)
BLEU = (0, 0, 255)
ROUGE = (255, 0, 0)
```

```
#-------Créer une fenêtre ---------
taille = (640, 480) #fenêtre de 640 pixels de large et 480 pixels de haut
ecran = pygame.display.set_mode(taille)
pygame.display.set_caption(" mon premier jeu pygame ")
#-------Variables du jeu -----------
arret = False
horloge = pygame.time.Clock()
#coordonnées de départ du rectangle
rect_x = 20
rect_y = 20
#déplacement du rectangle en x et en y
rect_change_x = 5
rect_change_y = 5
#----Boucle principale ---------
while not arret:
    #boucle des évènements
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            arret = True

     #logique du jeu
     #pour l'instant rien

    #Code traçant les objets du jeu
    #commencer par effacer l'écran
    ecran.fill(BLANC)
    pygame.draw.rect(ecran, NOIR, [rect_x, rect_y, 50, 50])
    #changer la position du rectangle
    rect_x+=rect_change_x
    rect_y+=rect_change_y
```

```
    #faire rebondir le rectangle sur les bords
    if rect_y > 430 or rect_y < 0:
        rect_change_y = rect_change_y * -1
    if rect_x> 590 or rect_x < 0:
        rect_change_x = rect_change_x *-1
    #Mettre l'écran à jour
    pygame.display.flip()
    #Mise à jour de l'écran à raison de 60 images/seconde
    horloge.tick(60)

pygame.quit()
```

Modifier le déplacement du rectangle avec les flèches

Les flèches « à droite », « à gauche », « vers le haut », « vers le bas » doivent nous permettre de modifier la trajectoire du rectangle. Il suffit de changer le sens du déplacement en multipliant le déplacement par -1, si le sens du déplacement ne correspond pas à celui de la flèche.

Nous modifions la boucle for qui extrait les évènements de la liste d'évènements :

```
    #boucle des évènements
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            arret = True
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT and rect_change_x > 0:
                rect_change_x *= -1
            if event.key == pygame.K_RIGHT and rect_change_x < 0:
                rect_change_x *= -1
            if event.key == pygame.K_UP and rect_change_y > 0 :
                rect_change_y *= -1
            if event.key == pygame.K_DOWN and rect_change_y < 0 :
                rect_change_y *= -1
```