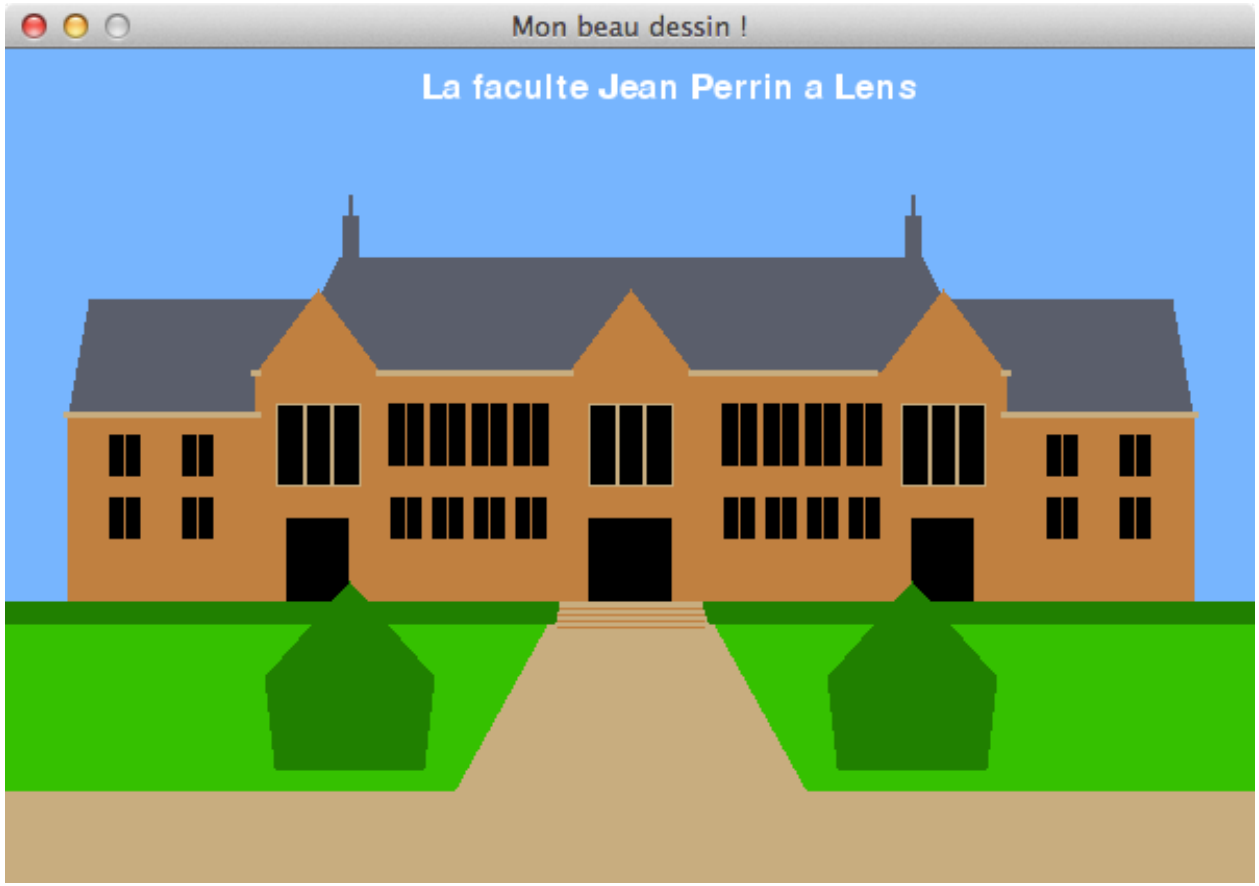
Cours 7

Graphismes

Aujourd'hui. . .

Nous allons faire un beau dessin !



La bibliothèque Pygame

Pour installer Pygame, visitez l'adresse : `http://pygame.org`

Pygame est une bibliothèque graphique parmi tant d'autres, comme tkinter, PySDL, PyQt, wxPython, etc.

Pygame est une bibliothèque créée, entre autres, pour faciliter :

- le dessin des formes graphiques basiques ;
- l'affichage des images type « bitmap » ;
- les animations ;
- le traitement du clavier, la souris, les manettes de jeu, etc. ;
- le traitement de son.

L'algorithme du programme

Un programme qui utilise Pygame pour afficher des objets graphiques doit suivre un schéma particulier pour pouvoir fonctionner correctement.

Que ce soit un jeu, une animation ou un simple dessin, le schéma est toujours le même.

1. Importe la bibliothèque Pygame.
2. Initialise Pygame.
3. Initialise l'horloge.
4. Ouvre la fenêtre de l'application.
5. Tant que le programme n'est pas terminé :
   - 5.1 Traite les événements.
   - 5.2 Ajuste la vitesse de la boucle.
   - 5.3 Dessine.
6. Termine le programme.

Nous allons illustrer l'utilisation de Pygame en utilisant le code source du dessin d'un simple rectangle comme exemple.

Initialisation du programme

Comme toute autre bibliothèque en Python, Pygame doit être importée avant son utilisation :

```
import pygame
```

Note : en effet, nous pouvons aussi utiliser « `from pygame import *` », mais ceci est déconseillé pour éviter que des fonctions soient redéfinies.

Ensuite, pour pouvoir utiliser les fonctionnalités de Pygame dans notre programme, nous devons l'initialiser.

```
pygame.init()
```

Initialisation de l'horloge

Dans l'étape suivante, nous devons initialiser l'horloge :

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

La variable `horloge` contient maintenant une référence à l'horloge.

Ceci permettra l'exécution de l'étape « Ajuster la vitesse de la boucle », qui sera expliquée plus tard.

Ouverture de la fenêtre de l'application

La prochaine étape est l'ouverture de la fenêtre de notre application.

```
surface = pygame.display.set_mode((600, 400))
pygame.display.set_caption('Mon beau dessin !')
```

La première instruction ci dessus ouvre une fenêtre de taille 600×400 pixels pour l'application. **Les deux parenthèses sont nécessaires !!**

La variable `surface` contient une référence à la surface de la fenêtre de l'application.

C'est sur cette `surface` que nous allons dessiner les objets.

La deuxième instruction ci dessus assigne un titre à la fenêtre de l'application.

La boucle principale du programme

La boucle principale du programme correspond à l'instruction :

« Tant que le programme n'est pas terminé : »

Cette boucle est nécessaire. Elle maintient ouverte la fenêtre de l'application jusqu'à ce que l'utilisateur demande sa fermeture.

Voici le code schématique de la boucle principale :

```
continuer = True
while continuer :

    # Traite les événements.

    # Ajuste la vitesse de la boucle.

    # Efface l'écran.

    # Dessine les objets.

    # Met à jours l'écran.
```

Terminer Pygame

Avant de terminer le programme, il faut terminer Pygame :

```
pygame.quit()
```

Cela peut éviter le blocage de certaines interfaces de Python, comme Thonny, par exemple.

Traitement des événements

C'est à l'intérieur de la boucle principale du programme que nous devons traiter les événements causés par l'utilisateur (clavier, souris, etc.).

Sans ce traitement, l'utilisateur ne peut pas interagir avec le programme. Il ne pourra même pas fermer la fenêtre du programme !

Le traitement des événements doit lui aussi être programmé comme une boucle :

1. Pour chaque événement `event` détecté :
   - 1.1 Si `event` est égal à « quitter » :
     - 1.1.1 Quitter.
   - 1.2 Si `event` est égal à « touche du clavier » :
     - 1.2.1 Faire quelque chose.
   - 1.3 Si `event` est égal à « button de la souris » :
     - 1.3.1 Faire quelque chose.
   - 1.4 ...

Traitement des événements

Code schématique du traitement des événements :

```
# Pour chaque événement detecté :
for event in pygame.event.get():

    # Si l'utilisateur a cliqué sur fermer
       fenêtre :
    if event.type == pygame.QUIT:
        terminer = True

    # Si une touche du clavier a ete appuyee :
    if event.type == pygame.KEYDOWN :
        # Remplir avec code qui traite cet é
           vénement.

    # Si le button de la souris a ete appuye :
    if event.type == pygame.MOUSEBUTTONDOWN :
        # Remplir avec code qui traite cet é
           vénement.
```

Ajuster la vitesse de la boucle

Avant de dessiner, nous devons demander au programme qu'il attende un peu.

C'est ici que nous utilisons la variable `horloge` qui à été initialisée dans l'étape « Initialisation de l'horloge ».

Voici la commande pour ajuster la vitesse de la boucle :

```
horloge.tick(40)
```

La valeur 40 passée en argument signifie que la boucle sera exécuté 40 fois par seconde (40 FPS).

Effacer l'écran

Avant de dessiner les objets, il est parfois utile d'effacer la surface de la fenêtre de l'application avec l'instruction `fill` :

```
surface.fill((0, 0, 0))
```

Attention : dans cet exemple, `surface` correspond à la variable définie auparavant avec l'instruction `pygame.display.set_mode`.

L'argument `(0, 0, 0)` correspond au code de la couleur du fond de l'écran. Dans ce cas, la couleur noir.

Il est possible de tinter l'écran avec n'importe quelle couleur. (Nous verons les codes de couleurs par la suite.)

Dessin des objets

Dans notre exemple, nous allons simplement dessiner un rectangle blanc. Voici donc l'instruction pour le faire :

```
pygame.draw.rect(surface, (255, 255, 255), (50,
    50, 200, 200))
```

Encore une fois, la variable `surface` correspond à la variable définie auparavant avec l'instruction `pygame.display.set_mode`.

Le tuple `(255, 255, 255)` correspond à la couleur du rectangle : blanc.

Le tuple `(50, 50, 200 200)` correspond, dans l'ordre :

- La coordonée `x` du rectangle `50`.
- La coordonée `y` du rectangle `50`.
- La largeur du rectangle `200`.
- La hauteur du rectangle `200`.

Mise à jours de l'écran

Après avoir dessiné sur la surface de l'écran, nous devons mettre à jours l'écran pour que les objets soient affichés :

```
pygame.display.update()
```

Attention : **il est nécessaire d'utiliser cette instruction pour que les objets soient affichés à l'écran.**

Cette instruction doit être exécutée une seule fois, à la fin de la boucle.

Le programme complet

Vérifiez le programme complet dans le fichier `affiche_rectangle.py`.

Couleurs

Avant de commencer à dessiner les objets, il est parfois utile de définir les couleurs qui seront utilisées.

Pygame utilise le système de couleurs RVB (rouge, vert, bleu), parfois aussi appelé RGB (de l'anglais : red, green, blue).

Chaque couleur est un mélange de ces trois couleurs représenté par un triplet d'entiers entre 0 et 255.

Par exemple :

- la couleur blanche est représentée par le code (255, 255, 255) ;
- la couleur noir est représentée par le code (0, 0, 0).

La valeur 0 signifie qu'il n'y a aucun pigment de la couleur correspondante dans le mélange alors que 255 signifie qu'il y a autant que possible du pigment correspondant dans le mélange.

Couleurs

Définitions de quelques couleurs au format RVB :

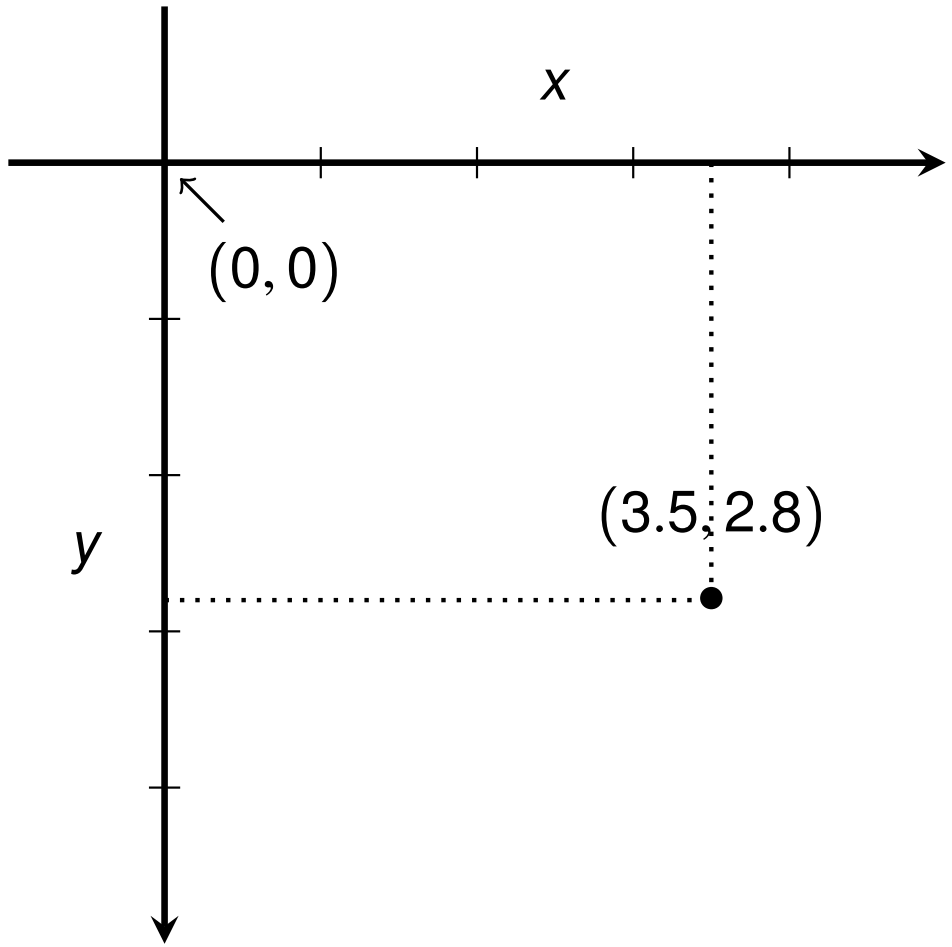
```
NOIR        = (  0,   0,   0)
BLANC       = (255, 255, 255)
BLEU_CIEL   = (119, 181, 254)
VERT        = (  0, 192,   0)
VERT_FONCE  = (  0, 128,   0)
BEIGE       = (200, 173, 127)
MARRON      = (192, 128,  64)
ARDOISE     = ( 90,  94, 107)
```

Pour trouver plus de couleurs au format RVB, regardez l'adresse :
`http://fr.wikipedia.org/wiki/Liste_de_noms_de_couleur`

Dessiner les objets

Pour dessiner des objets sur la surface de l'écran nous devons passer les coordonnées des objets que nous voulons dessiner.

Le système des coordonnées de l'ordinateur est semblable au système Cartésien, mais avec l'axe *y* inversé.



Dessiner un rectangle

Syntaxe :

```
pygame.draw.rect(Surface, couleur, coords, largeur=0)
```

`Surface` : surface où l'objet sera dessinée.
(Définie auparavant avec `pygame.display.set_mode`.)

`couleur` : couleur de l'objet.

`coords` : une séquence de quatre entiers. Les deux premiers correspondent aux coordonnées (x, y) de l'objet. Les deux autres à la largeur et à la hauteur de l'objet.

`largeur:` (paramètre optionnel, valeur par défaut = 0) un entier qui correspond à la largeur de la bordure de l'objet.

Exemple :

```
pygame.draw.rect(surface, BLANC, (50, 50, 200,
    200))
```

Dessiner une ligne

Syntaxe :

```
pygame.draw.line(Surface, couleur, debut, fin, largeur=1)
```

`Surface:` surface où l'objet sera dessiné.

`couleur:` couleur de l'objet.

`debut:` coordonnées du début de la ligne.

`fin:` coordonnées de la fin de la ligne.

`largeur:` (optionnel, défaut = 1) largeur de la ligne.

Exemple :

```
pygame.draw.line(surface, BLANC, (265, 268),
    (335, 268))
```

Dessiner d'autres objets

Il y a plusieurs instructions pour dessiner. Leur syntaxe est semblable.

Syntaxes :

```
pygame.draw.ellipse(Surf, couleur, coords, largeur=0)
```

```
pygame.draw.arc(Surf, couleur, debut, fin, largeur=0)
```

```
pygame.draw.polygon(Surf, couleur, points, largeur=0)
```

```
pygame.draw.circle(Surf, couleur, pos, rayon, largeur=0)
```

Pour plus d'objets, regarder :
`http://pygame.org/docs/ref/draw.html`

Dessiner du texte

L'écriture d'un texte en mode graphique consiste en trois étapes :

1. Sélection de la police.
2. Création de l'image du texte.
3. Affichage du texte.

Exemple :

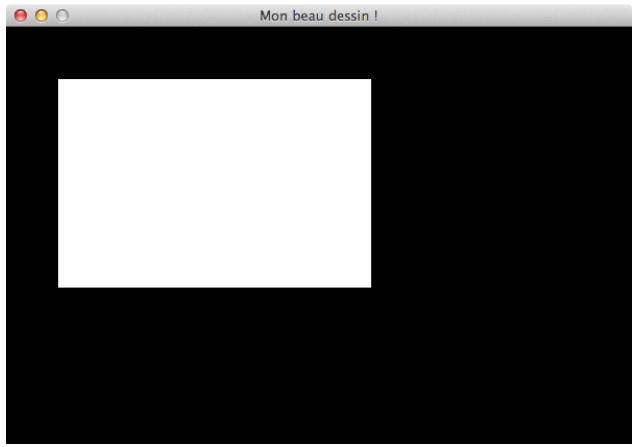
```
# Seléction de la police default et taille 25.
police = pygame.font.Font(None, 25)

# Création de l'image du texte.
texte = police.render('Mon texte', True, NOIR)

# Affichage du texte aux coordonnées (250, 250).
surface.blit(texte, (250, 250))
```

Fichiers source complets

- `affiche_rectangle.py`



- `affiche_jean_perrin/`
  - `__init__.py`
  - `__main__.py`
  - `jean_perrin.py`
  - `couleur.py`