Méthodologie de la programmation

Chapitre 4

Pygame

Pablo Rauzy <pr@up8.edu>
pablo.rauzy.name/teaching/mp

Pygame

- On va étudier une grosse bibliothèque en particulier : Pygame.
- Pygame sert principalement à faire des jeux vidéos.
- Site web : `https://pygame.org/`.
- Documentation : `https://pygame.org/docs/`.

Pourquoi Pygame

- *Logiciel libre*.
- Multi-plateformes : GNU/Linux, Windows, Mac OS X.
- Respecte bien les conventions Python.

Programmer un jeu

- Lorsqu'on programme un jeu il faut gérer :
  - l'affichage,
  - l'interaction avec l'utilisateur,
  - le temps,
  - (parfois) le réseau.

Affichage

- Dans un jeu, l'affichage est critique.
- Beaucoup de composants à gérer tout en restant fluide.
- Utilisation de la technique du *double buffer*.
- Automatique dans les bibliothèques haut niveau comme Pygame.

Le double buffer

- Si on essaye d'afficher plein d'éléments à l'écran, un scintillement peut se produire et casser la fluidité de l'affichage.
- La technique du double buffer consiste à avoir deux zones d'affichage, et d'alterner entre les deux.
- De cette manière, on dessine toujours sur la zone "cachée", et aucun artefact gênant ne peut arriver à l'écran.
- Chaque fois qu'on a fini de dessiner, on inverse les deux zones et celle qu'on vient de dessiner est affichée d'un seul coup.

Le double buffer avec Pygame

- La bibliothèque Pygame fait tout ça pour vous si vous le lui demandez :

```
1  import sys
2  import pygame
3  from math import cos
4
5  pygame.init()
6
7  size = width, height = 800, 100
8  screen = pygame.display.set_mode(size, pygame.DOUBLEBUF)
9  black, white = pygame.Color(0,0,0), pygame.Color(255,255,255)
10 screen.fill(white)
11
12 i = 0
13 while True:
14     for event in pygame.event.get():
15         if event.type == pygame.QUIT:
16             sys.exit()
17
18     y = int((height / 2) * cos(i / 50) + (height / 2))
19     x = i % width
20     i += 1
21     screen.set_at((x, y), black)
22     pygame.display.flip()
```

Composants

- Il est évidemment mieux de faire le moins d'opérations d'affichage possible.
- Par exemple dans un jeu de plateau, il est possible de construire le plateau au début du niveau puis de le garder en mémoire déjà construit pour l'afficher d'un seul coup.
- Pour les autres composants du jeu, par exemple une balle en mouvement qui va systématiquement bouger, on peut les mettre dans un conteneur qui va permettre d'itérer dessus pour les afficher.

```
1 bg = pygame.Surface(size, pygame.SRCALPHA, 32)
2 pygame.draw.line(bg, black, (30,20), (30,100), 10)
3 pygame.draw.line(bg, black, (30,100), (70,100), 10)
4 pygame.draw.line(bg, black, (80,50), (120,20), 10)
5 pygame.draw.line(bg, black, (120,20), (120,100), 10)
6     # update display
7     screen.blit(bg, (0, 0))
```

Sprites

- De la même manière, il faut mieux éviter de charger plein de petites images.
- On préfère donc charger une seule grosse image qui sert pour plusieurs *sprites*.

```
player = pygame.image.load('player.png').convert()
player.set_colorkey(white)
up = pygame.Rect(25, 0, 25, 31)
down = pygame.Rect(0, 0, 25, 31)
left = pygame.Rect(25, 31, 25, 31)
right = pygame.Rect(0, 31, 25, 31)
    # update display
    screen.blit(player, position, direction)
```

Interagir avec l'utilisateur

- Les entrées utilisateurs arrivent par la souris, le clavier, et/ou un joystick, dans le cas des jeux.
- La gestion de ces périphériques repose sur le principe d'évènements.
- Chaque fois que l'utilisateur est actif, un évènement est généré :
  - mouvement de la souris,
  - clic du bouton gauche / droit / milieu,
  - appui sur une touche,
  - ...
- Quand c'est applicable, l'évènement contient des informations utiles :
  - position du curseur de la souris,
  - quelle touche du clavier a été enfoncée,
  - ...

Boucle de gestion des évènements

- La gestion des évènements doit se faire en continue.
- Elle est donc placée au début de la boucle principale du programme.
- La structure de la boucle principale d'un jeu va être :

```
done = False
while not done:
    for event in pygame.event.get():
        # manage events
    # clear screen
    screen.fill(white)
    # update display
    screen.blit(bg, (0, 0))
    screen.blit(player, position, direction)
    # flip double buffer
    pygame.display.flip()
```

Gestion des évènements

- Pygame fourni une gestion complète et simple des évènements.
- La méthode `pygame.event.get()` renvoie la liste des évènements en attente.
- Chaque évènement a un `type` et contient des informations additionnelles.
  - Un évènement souris contiendra aussi les boutons cliqués et la position du curseurs.
  - Un évènement clavier contiendra aussi la touche appuyée et d'éventuels modificateurs.

```
1   for event in pygame.event.get():
2       if event.type == pygame.QUIT:
3           done = True
4       if event.type == pygame.KEYDOWN:
5           if event.key == pygame.K_UP:
6               direction = up
7               position[1] -= 2
8           elif event.key == pygame.K_DOWN:
9               direction = down
10              position[1] += 2
11          elif event.key == pygame.K_LEFT:
12              direction = left
13              position[0] -= 2
14          elif event.key == pygame.K_RIGHT:
15              direction = right
16              position[0] += 2
```

On essaye ?

- Regardons ensemble le code que l'on vient d'écrire et ce qu'il produit...