Chapitre 3

Concepts du jeu vidéo et premiers pas à propos de Pygame

1. Introduction

Le présent chapitre se propose à la fois de présenter différents concepts relatifs à Pygame d'une part, et d'autre part de commencer à expliquer comment Pygame apporte des solutions pratiques face à ces concepts. Il nous faut notamment expliquer ce qu'est une boucle de jeu, puis présenter la notion de gestion des collisions. Ce dernier aspect est particulièrement fastidieux à gérer sans Pygame. Nous prendrons un petit exemple d'un jeu dont la gestion des collisions est réalisée sans Pygame. Celui-ci nous permettra ainsi de démontrer à quel point Pygame est pratique pour la gestion de cet aspect si central en développement de jeux vidéo.

2. La boucle de jeu

Avant d'aller plus loin, il faut définir une notion fondamentale du développement de jeux vidéo : la boucle de jeu. Parfois appelée « boucle d'animation », elle se retrouve dans (presque) tous les jeux vidéo et constitue à ce titre une sorte de colonne vertébrale.

Elle correspond à une boucle infinie. Évidemment, il faut pouvoir l'interrompre. Pour cela, on code une action utilisateur (par exemple, appuyer sur la touche [Esc] du clavier). Un appui sur cette touche interrompt la boucle infinie et donc la partie en cours.

Que fait donc cette boucle à chaque itération ? Peu ou prou, les étapes suivantes :

1. Vérifier que les conditions d'arrêt ne sont pas atteintes et, si elles le sont, interrompre la boucle.

2. Mettre à jour les ressources nécessaires pour l'itération courante.

3. Obtenir les entrées soit issues du système, soit issues de l'interaction avec le joueur.

4. Mettre à jour l'ensemble des entités qui caractérisent le jeu.

5. Rafraîchir l'écran.

À la fin d'une itération, la boucle infinie se poursuit et on passe à l'itération suivante.

Idéalement, chaque itération doit avoir la même durée que toutes les autres pour permettre une certaine fluidité dans le déroulement du jeu. Comme nous le verrons par la suite, c'est aussi l'un des avantages de Pygame qui offre des outils de gestion du temps. Pour information, on travaille en général avec une modalité de 30 images par seconde avec Pygame, ce qui revient à voir la boucle infinie du jeu réaliser 30 itérations par seconde.

3. Présentation de Pygame

Cette section est avant tout là pour contextualiser Pygame et expliquer comment ce framework est conçu, pourquoi et dans quels contextes.

Pygame est une bibliothèque logicielle, c'est-à-dire un ensemble de ressources logicielles, dédiée au développement de jeux vidéo temps réel. Cette bibliothèque a également la spécificité et le grand intérêt d'être un logiciel libre, dans la mesure où elle est distribuée sous licence GNU LGPL.

Ce qu'il faut bien comprendre, c'est que pour faire un jeu, il faut nécessairement que des éléments de code interagissent avec des parties « bas niveau » de l'ordinateur, c'est-à-dire avec le matériel, que ce soit l'écran, la carte son, le clavier, la souris, voire d'autres périphériques dédiés au jeu. Ces interactions « bas niveau » sont complexes, en général développées avec des langages comme le langage C, et utilisent la bibliothèque SDL (*Simple DirectMedia Layer*) qui n'est pas toujours très simple à prendre en main.

La bibliothèque Pygame constitue une couche qui se situe au-dessus de SDL et qui simplifie l'accès et la manipulation de ces différents périphériques de manière simple et intuitive, qui plus est en développant avec le langage Python.

Pygame offre des modules simplifiant tous les aspects du jeu : entre autres le graphisme, le son, les interactions avec la souris ou le clavier, la gestion des événements.

4. Installation de Pygame

Le moyen le plus simple d'installer Pygame est d'utiliser le programme `pip`. Ainsi, la ligne de commande suivante permet d'ajouter la dernière version de Pygame.

```
--------------------------------------------------------------------------------

pip install pygame

--------------------------------------------------------------------------------
```

Une fois installé, vous pouvez immédiatement vérifier la version de Pygame, avec la commande suivante lancée dans un environnement Python :

```
import pygame

--------------------------------------------------------------------------------
```

On obtient :

```
> pygame 2.1.2 (SDL 2.0.18, Python 3.8.8)
```

Il est par ailleurs possible d'installer Pygame avec des programmes incluant de nombreux autres outils, comme Anaconda par exemple.

Au moment de la rédaction de ce livre, la version de Pygame (comme indiqué ci-dessus) est la version 2.1.2, incluant la version de SDL 2.0.18.

Il peut être intéressant de commencer par visiter le site officiel du projet Pygame (https://www.pygame.org), d'une part pour découvrir les différentes ressources qui pourront vous être utiles par la suite, mais d'ores et déjà pour vérifier quel est le numéro de la version la plus récente. Par ailleurs, le site officiel propose plusieurs alternatives d'installation selon votre système d'exploitation.

Que vous soyez sous Windows, Linux, ou macOS, l'installation de Pygame est extrêmement simple. Rendez-vous pour commencer à l'adresse web suivante :
https://www.pygame.org/download.shtml

La page de téléchargement est organisée selon le système d'exploitation utilisé :

- Sur Windows, téléchargez le programme d'installation (.msi) adéquat, puis exécutez-le.
- Sur Linux, choisissez la ressource correspondant à votre distribution (Ubuntu, Debian, Fedora, etc.), téléchargez-la puis exécutez-la avec la commande d'installation indiquée.
- Sur macos, téléchargez la ressource adéquate (.dmg), puis exécutez le programme d'installation inclus (.mpkg).

L'installation de Pygame à laquelle vous avez procédé a également et de façon transparente installé SDL.

Même si l'installation de Pygame semble s'être bien déroulée (pas de message d'erreur à l'horizon), il s'agit tout de même de vérifier que tout est correctement en place. Faites l'import de Pygame dans une ligne de commande Python pour vérifier qu'une version de Pygame est bien proposée et qu'elle est cohérente avec ce que vous avez pu lire sur le site web.

Si problème, il se peut qu'il soit dû soit à un problème de cible 32 ou 64 bits, soit à une version de Pygame ne correspondant pas à celle de Python sur votre machine.

5. Les modules composant Pygame

Cette section recense les différents modules qui composent Pygame, ce qui donne un premier aperçu des possibilités. Avec seulement quatre ou cinq de ces modules (ou classes) qu'on utilise le plus souvent en général (*event*, *image*, *draw*, *surface*, *time*, par la suite *sprite*), il est possible de réaliser des jeux vidéo avancés.

- `cdrom` : gestion des lecteurs CD/DVD.
- `cursors` : gestion du curseur de la souris.
- `display` : configuration de la surface d'affichage.
- `draw` : dessin de formes surfaciques.
- `event` : gestion des événements graphiques ou relatifs aux périphériques.
- `font` : gestion des polices de caractères.
- `image` : gestion des images.
- `joystick` : gestion des périphériques de jeu.
- `key` : gestion du clavier.
- `mixer` : gestion générale du son.
- `mouse` : gestion de la souris.
- `movie` : gestion des aspects vidéo.
- `music` : gestion de la musique.
- `overlay` : gestion avancée de la vidéo.
- `pygame` : gestion de la bibliothèque Pygame elle-même.
- `rect` : gestion spécifique de la forme rectangulaire.
- `sndarray` : gestion de données sonores.
- `sprite` : gestion des sprites, objets de haut niveau utilisés dans Pygame.
- `surface` : gestion avancée d'images.
- `surfarray` : gestion et manipulation d'images.

- `time` : gestion du temps et du rafraîchissement de l'écran.
- `transform` : transformation et déplacement des images.

6. Réalisation d'un premier jeu graphique : fusée et planètes

L'idée ici est de réaliser un petit jeu dont le but est de piloter un engin volant et d'éviter certains obstacles : des planètes, deux en l'occurrence, tombant du ciel. La fusée doit les éviter en allant à gauche ou à droite. Ce sera là l'action du joueur : déplacer à gauche ou à droite (latéralement uniquement) la fusée pour éviter les planètes. Si une planète touche la fusée, c'est évidemment perdu.

Pour cela, nous allons écrire un petit code en Pygame en détaillant le plus possible chaque étape.

Le code est muni d'un fichier requirements.txt qui vous autorise à créer et activer un environnement virtuel. Ici, il n'y a qu'un seul module, en l'occurrence Pygame.

L'un des intérêts de cette démarche est d'expliquer et d'illustrer simplement deux grands aspects du développement Pygame, à savoir :

- le système de coordonnées dans une fenêtre Pygame,
- la gestion des collisions, c'est-à-dire le code qui permet de gérer la rencontre de deux objets graphiques (si l'on veut par exemple simuler graphiquement le rebond d'une balle sur le sol).

La suite du livre détaille avec précision chaque aspect abordé. Mais ce premier développement offre une vue globale de ce qu'est Pygame et de ce qu'il permet de faire.

6.1 Les images utilisées

L'engin volant est défini par une image au format JPEG ou au format PNG. Vous pouvez utiliser une de vos photographies ou une de vos créations graphiques réalisées avec un logiciel de dessin ou encore une image trouvée sur le Web, sous réserve qu'elle soit sous licence permettant une telle réutilisation gratuite et éventuellement sans nécessité d'attribution. C'est le cas de l'image de fusée utilisée ici. Elle est stockée dans le répertoire du projet sous le nom FUSEE.png. Le véhicule volant a une orientation verticale, en vue d'un défilement automatique du jeu orienté vers le haut. Idéalement enfin, l'image choisie doit avoir un fond transparent.

L'image de planète que la fusée doit éviter est stockée sous le nom PLANETE.png.

6.2 La fenêtre du jeu

Créer la fenêtre du jeu est la première étape. Il s'agit de la fenêtre qui accueille les éléments graphiques qui composent le jeu.

Sans surprise, on commence par importer Pygame :

```
import pygame
```

Puis, on initialise Pygame :

```
pygame.init()
```

Les deux lignes précédentes sont systématiquement présentes dès lors que l'on utilise Pygame.

Définissons maintenant les dimensions de la fenêtre :

```
HAUTEUR_FENETRE = 600
LARGEUR_FENETRE = 600
```

Puis, nous définissons la couleur de fond de la fenêtre :

```
COULEUR_FOND = (255, 255, 255)
```

Et enfin, nous utilisons la commande d'affichage de la fenêtre :

```
ECRAN = pygame.display.set_mode((LARGEUR_FENETRE, HAUTEUR_FENETRE))
```

Si vous lancez le programme, la fenêtre du jeu s'affiche avec un fond blanc. Aucune indication sur la durée de l'affichage n'est apportée, on la voit donc s'ouvrir et se fermer le temps d'une fraction de seconde.

6.3 La boucle du jeu

Pour définir la boucle du jeu, nous commençons par définir un booléen : si sa valeur est `True`, la boucle se poursuit, elle s'interrompt sinon.

```
ARRET_JEU = False
```

La fonction `pygame.event.get()` permet d'intercepter tous les événements entrants notamment depuis le clavier, la souris, etc. Si on appuie sur la touche [Esc], alors le jeu est interrompu. Ceci est permis en utilisant les événements du clavier suivants : touche appuyée (`pygame.KEYDOWN`) et touche [Esc] ou [Echap] (`pygame.K_ESCAPE`).

On définit donc ainsi la boucle de jeu :

```
while not ARRET_JEU:
    for event in pygame.event.get():
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                ARRET_JEU = True
```

À ce stade, le code est le suivant :

```
import pygame
pygame.init()
HAUTEUR_FENETRE = 600
LARGEUR_FENETRE = 600
COULEUR_FOND = (255, 255, 250)
ECRAN = pygame.display.set_mode((LARGEUR_FENETRE, HAUTEUR_FENETRE))
# booléen de gestion de la boucle
ARRET_JEU = False
while not ARRET_JEU:
    for event in pygame.event.get():
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                ARRET_JEU = True
```

La fenêtre s'affiche et l'affichage s'interrompt dès qu'on appuie sur [Esc]. Nous sommes pour le moins encore assez loin de quelque chose d'amusant. Nous allons donc entrer dans le vif du sujet et commencer à parler de fusée et de planètes.