Chapitre 5
L'ajout de sons dans un jeu Pygame

1. Introduction

Animer un jeu vidéo, c'est d'abord mettre en place le visuel et donner le mouvement à certains objets graphiques. Mais cela consiste également à ajouter du son au jeu, par exemple un fond sonore, une musique, pour ainsi améliorer l'expérience utilisateur. Il y a dans Pygame un module pour cela. Son étude et sa mise en œuvre constituent l'essentiel du présent chapitre.

Le module qui permet la gestion des sons en Pygame s'appelle *pygame.mixer.* Il contient deux grandes notions :

- Le sous-module *music* qui gère la musique de fond. Il n'y en a qu'une à la fois.
- L'objet *Sound* de *mixer* que l'on peut instancier plusieurs fois pour s'en servir par exemple pour les effets sonores du jeu.

2. La gestion du son avec Pygame

En premier lieu, il est nécessaire d'initialiser ce module grâce à la fonction *pygame.mixer.init.* Cet appel doit être fait explicitement. Le module *mixer* inclut un certain nombre d'outils relatifs aux effets sonores grâce à la classe *Sound.* Ce module inclut également une sorte de sous-module, *music*, dévolu à la gestion du fond sonore.

2.1 Les modules pygame.mixer et pygame.mixer.music

Remarque

Une documentation des modules *mixer* et *music* de Pygame est disponible dans le chapitre Les principaux modules Pygame.

2.1.1 Le module pygame.mixer.music (fond sonore)

Ce module Pygame permet de gérer le fond sonore (unique). Ci-après les principales fonctions disponibles dans *music*.

- *pygame.mixer.music.load* permet de charger un fichier de fond sonore.
- *pygame.mixer.music.play* permet de jouer/lire le fond sonore.
- *pygame.mixer.music.rewind* permet de reprendre au début le fond sonore.
- *pygame.mixer.music.stop* permet d'arrêter la lecture du fond sonore.
- *pygame.mixer.music.pause* permet de faire une pause dans la lecture.
- *pygame.mixer.music.set_volume* permet de régler le volume du fond sonore.
- *pygame.mixer.music.get_volume* permet de connaître le volume courant du fond sonore.

2.1.2 Le module pygame.mixer (effets sonores)

Pour les effets sonores, il faut préalablement créer (instancier) un objet de type *Sound* avant d'accéder à un ensemble de fonctions comparables à celles du sous-module *music*.

pygame.mixer.Sound permet de créer un nouvel objet de type *Sound*.

Une fois créé l'objet de type *Sound*, on peut utiliser une de ses fonctions. Entre autres :

- *pygame.mixer.Sound.play* permet de jouer le son.
- *pygame.mixer.Sound.stop* permet de stopper la diffusion du son.

2.2 Les fichiers son

Voici un court rappel sur les différents formats et extensions de fichiers sonores qui sont utilisables avec Pygame.

Deux formats sont particulièrement recommandés :

- le format WAV (*Waveform Audio File Format*)
- le format ouvert et libre OGG

Ils sont recommandés, car leur utilisation ne pose aucun problème, quelle que soit la plateforme. Il n'en va pas de même du format de compression audio MP3 par exemple, qui selon la documentation officielle peut induire des soucis d'utilisation sous certaines plateformes, en particulier avec la distribution Linux Debian (*« On some systems an unsupported format can crash the program, e.g. Debian Linux. Consider using OGG instead. »*).

Comment obtenir ou afficher des fichiers son pour un fond sonore ou pour des effets sonores ? On peut par exemple obtenir de la musique publiée sous licence libre sur Internet. On peut également créer ses propres fichiers son. Par exemple en les enregistrant grâce à l'enregistreur d'un smartphone ou en utilisant le logiciel libre d'enregistrement Audacity, qui permet aussi de faire du montage audio et d'exporter des sons dans le format de son choix, le format OGG en particulier.

2.3 La notion de channel (canal) dans Pygame

Pygame offre une notion supplémentaire dans la gestion du son. Il s'agit de la notion de channel (canal, en français). Un jeu possède plusieurs canaux son. Ainsi, on peut affecter tel son au canal numéro 1 et tel autre son au canal numéro 2. Il est ainsi possible de jouer simultanément des sons en activant leurs lectures sur des canaux différents comme le suggère l'exemple théorique suivant.

```
pygame.mixer.set_num_channels(2)

sound_1 = pygame.mixer.Sound("son1.ogg")
sound_2 = pygame.mixer.Sound("son2.ogg")

canal_1 = pygame.mixer.Channel(0)
canal_2 = pygame.mixer.Channel(1)

canal_1.play(sound_1)
canal_2.play(sound_2)
```

Remarque

Cette notion est largement abordée dans les sections Le module mixer et music au chapitre Les principaux modules Pygame.

3. Exemple d'utilisation du son avec Pygame

Le but de ce petit exemple est d'utiliser les deux aspects présentés précédemment. Ainsi nous allons créer une fenêtre de jeu à laquelle on associe un fond sonore (*music*) ainsi que quelques effets sonores déclenchés par l'appui de boutons du clavier.

- Le fond sonore est un sifflement diffusé en boucle.
- L'appui sur la touche o diffuse un cocorico.
- L'appui sur la touche c diffuse un bruit de corneille.
- L'appui sur la touche v diffuse une sonnette de vélo.

En appuyant sur la touche [Flèche en haut], on augmente le volume de chacun des quatre sons. En appuyant sur la touche [Flèche en bas], on diminue ce même volume.

On commence nécessairement par initialiser le module *mixer*.

```
pygame.mixer.init()
```

Puis on crée le fond sonore d'après un fichier son OGG qui inclut un sifflement mélodique d'une durée de quelques dizaines de secondes.

```
SIFFLEMENT = pygame.mixer.music.load("sifflement.ogg")
```

Ensuite, on peut jouer ce fond sonore avec la fonction *play* qui prend deux paramètres :

- Le premier permet d'indiquer combien de fois on désire boucler sur le son (ici 10 fois).
- Le second, une valeur décimale, indique en secondes le moment du morceau qui constitue le début de la diffusion.

```
pygame.mixer.music.play(10, 0.0)
```

Nous définissons maintenant les trois effets sonores.

```
COQ = pygame.mixer.Sound("coq.ogg")
CORNEILLE = pygame.mixer.Sound("corneille.ogg")
VELO = pygame.mixer.Sound("vélo.ogg")
```

Ils sont déclenchés par l'appui d'une touche du clavier dédiée.

```
elif event.key == pygame.K_o:
 COQ.play()
elif event.key == pygame.K_c:
 CORNEILLE.play()
elif event.key == pygame.K_v:
 VELO.play()
```

Enfin, on gère le volume en l'augmentant ou en le diminuant, en commençant par obtenir le volume courant auquel on ajoute ou on retranche 0.1. En effet, le volume s'exprime entre 0.0 et 1.0 ; par défaut, le volume est à 1.0.

```
elif event.key == pygame.K_DOWN:
 VOLUME = pygame.mixer.music.get_volume() - 0.1

 pygame.mixer.music.set_volume(VOLUME)
 COQ.set_volume(VOLUME)
 CORNEILLE.set_volume(VOLUME)
 VELO.set_volume(VOLUME)

elif event.key == pygame.K_UP:
 VOLUME = pygame.mixer.music.get_volume() + 0.1

 pygame.mixer.music.set_volume(VOLUME)
 COQ.set_volume(VOLUME)
 CORNEILLE.set_volume(VOLUME)
 VELO.set_volume(VOLUME)
```

Le code complet du programme est le suivant :

```
import pygame, sys

pygame.init()
pygame.mixer.init()

# COULEURS
COULEUR_BLANCHE = pygame.Color(255, 255, 255)

# FENETRE DE 400 SUR 400
ECRAN = pygame.display.set_mode((400,400))
ECRAN.fill(COULEUR_BLANCHE)
pygame.display.set_caption("Chapitre 5, du son")

SUITE = True

# FOND SONORE
SIFFLEMENT = pygame.mixer.music.load("sifflement.ogg")
pygame.mixer.music.play(1, 0.0)

# EFFETS SONORES
COQ = pygame.mixer.Sound("coq.ogg")
CORNEILLE = pygame.mixer.Sound("corneille.ogg")
VELO = pygame.mixer.Sound("vélo.ogg")

# BOUCLE DE JEU
while SUITE:

 for event in pygame.event.get():
   if event.type == pygame.QUIT:
     SUITE = False
   elif event.type == pygame.KEYDOWN:
     if event.key == pygame.K_ESCAPE:
       SUITE = False
     elif event.key == pygame.K_o:
       COQ.play()
     elif event.key == pygame.K_c:
       CORNEILLE.play()
```

```
        elif event.key == pygame.K_v:
            VELO.play()
        elif event.key == pygame.K_DOWN:
            VOLUME = pygame.mixer.music.get_volume() - 0.1
            pygame.mixer.music.set_volume(VOLUME)
            COQ.set_volume(VOLUME)
            CORNEILLE.set_volume(VOLUME)
            VELO.set_volume(VOLUME)
        elif event.key == pygame.K_UP:
            VOLUME = pygame.mixer.music.get_volume() + 0.1
            pygame.mixer.music.set_volume(VOLUME)
            COQ.set_volume(VOLUME)
            CORNEILLE.set_volume(VOLUME)
            VELO.set_volume(VOLUME)

    pygame.display.flip()
```