# PROJET : SPACE INVADERS

12.04.2021

Carl Gauss Rugero
L2 Informatique Internationale

Cours : PCO
Prof : Catherine Dezan

## Introduction

L'objectif du projet est de construire un jeu vidéo classique intitulé Space Invaders. Le principe du jeu est de détruire des aliens regroupés en Bunker par l'intermédiaire d'un tireur ou Defender qui lance des tirs, en se déplaçant horizontalement sur l'écran.

En usant du langage Python et de l'éditeur de texte de Microsoft  Visual Studio Code, on peut se demander comment le jeu est conçu dans les moindres détails.

Sous forme de documentation, on tentera tout d'abord d'expliquer le but et les fonctionnalités du jeu; ensuite, on expliquera dans des détails techniques, les différentes classes qui composent le programme avec leur structures et responsabilités fonctionnelles; après, on parlera des relations entre les différents objets du système et enfin, des particularités du jeu qui ne sont pas visibles à la simple lecture du code.

## But et Fonctionnalités du jeu : Space Invaders

Space Invaders est un jeu classique de 20ème siècle qui a pour but de détruire à l'aide d'un canon à laser. Il consiste à éliminer les aliens avant qu'il n'atteignent la terre càd le bas de l'écran ou avant qu'il ne touche le Defender 3 fois avec leurs tirs à laser (bullets); il est donc impératif que le Defender esquive les tirs des aliens et se protège en tirant ou en se cachant derrière des barrières; sinon c'est GAME OVER !

Le jeu commence de lui-même, le joueur se doit d'utiliser 3 touches du clavier en alternance. Les flèches DROITE et GAUCHE pour déplacer le Defender et la touche SPACE pour tirer sur les aliens. Au fur et à mesure que le Defender touche un alien, il gagne un point. De même, lorsque le Defender est touché par l'alien, il perd une vie.

Le jeu est gagné d'une seule manière : le Defender doit éliminer tous les aliens (Fleet) sans que sa vie ne soit nulle.

Le jeu est perdu de 2 manières : D'une part, si les aliens tuent toutes les vies du Defender, d'autre part, si les aliens parviennent à rejoindre le sol ou bas de l'écran.

## Détails techniques du jeu : Space Invaders

Grâce à l'éditeur de texte VsCode, on a pu importer la bibliothèque graphique pour le langage Python qui permet la création d'interfaces graphiques et la bibliothèque de génération de nombres aléatoires.

```
import tkinter as tk
```

```
from tkinter import

import random as rd

from winsound import *
```

L'écriture du code du programme est fait de 8 classes pythons à savoir :

- La classe **Defender()** qui permet de tirer sur les aliens.
- La classe **Fleet()** qui permet de regrouper un groupe d'aliens pour tirer sur le Defender.
- La classe **Alien()** qui représente un alien qui a pour but de tuer un Defender.
- La classe **Bullet()** qui représente les balles du Defender et les aliens qui s'entretuent.
- La classe **Shield()** qui représente une barrière de protection se trouvant devant l'objet Defender.
- La classe **multiple_shields()** qui représente la multiplication des shields en horizontal.
- La classe **Game()** qui permet de faire les différents jeux de données
- La classe **SpaceInvaders()** est la classe principale qui permet la création de la fenêtre principale et son exécution.

## La classe Defender()

Les accesseurs de cette classe permettent de mettre les coordonnées du Defender à jour. La méthode install_in() permet au Defender de se positionner grâce aux accesseurs et à la méthode prédéfinie canvas.create_image(...). La méthode keypress() va permettre de bouger le Defender en utilisant le clavier

- D'une part, avec event.keysym ==’Left’, on appuie sur la touche Flèche Gauche, qui permet le Defender d'aller à gauche;
- D'autre part, event.keysym == ‘Right’, qui permet au Defender de partir à droite.
- Et enfin, event.keysym == ‘space’, qui permet de lancer des tirs.

La méthode move_in() favorise le déplacement horizontal du Defender et Fire(), de vérifier les tirs du Defender via

```
self.bullet.id = self.bullet.install_in(self.canvas)

self.fired_bullets.append(self.bullet.id)
```

La méthode move_bullet() quant à elle, gère le déplacement, les tirs et la limite des bullets qui sortent du Defender. La méthode defender_touched_by() gère la collision du Bullet sur le Defender. Elle mémorise les coordonnées du Defender, met dans une liste

```
x,y,x1,y1= self.canvas.bbox(self.defender_object)
```

```
collapsed = canvas.find_overlapping(x,y,x1,y1)
```

Si le bullet touche le Defender, le Defender perd une vie et supprime le bullet du jeu.

## La classe Fleet()

Dans cette classe, on définit  deux groupements d'accesseurs qui définissent l'espacement entre les aliens

```
def get_dx(self):
        return self.alien_x_delta
    def get_dy(self):
        return self.alien_y_delta
    def set_dx(self,new_dx):
        self.alien_x_delta=new_dx
```

et les dimensions du Fleet

```
def get_width(self):
        return self.width
    def get_height(self):
        return self.height
```

La méthode install_in() permet connaître le positionnement du Fleet et créer une matrice qui définit la largeur et la longueur d'un alien et, les espacements entre aliens.

La méthode move_in() permet de faire bouger l'objet Fleet() et de contrôler son déplacement dans le canvas. Si dépassement au niveau des côtés, revenir au côté opposé,

et si dépassement au niveau du bas de l'écran, supprimer tout le contenu du canvas et renvoyer le message GAME OVER.

Grâce à la méthode install_alien_bullet(), les bullets des aliens ont un positionnement initial; mais dans ce cas qui varie avec la définition de la bibliothèque aléatoire réservée aux bullets des aliens

```
rand=rd.randint(0,len(self.aliens_fleet)-1)
alienx1,alieny1,alienx2,alieny2=self.canvas.bbox(self.aliens_fleet[rand])
```

Après on appelle la classe Bullet pour donner ces coordonnées aléatoires dans ses setteurs

La méthode fire() permet la configuration des bullets de telle manière que lorsque le bullet est tiré, recharger un nouveau bullet au niveau des aliens en redonnant de nouvelles coordonnées aléatoires et en ajoutant le bullet tiré dans la liste fired_bullet

```
self.bullet.id = self.bullet.install_in(self.canvas)

self.fired_bullet.append(self.bullet.id)
```

La méthode move_bullet() sert à gérer les déplacements du bullet de manière à ce que lorsque le bullet dépasse le cadre du jeu, on le supprime et un autre est généré.

La méthode manage_touched_aliens_by() gère la collision avec les bullets du Defender. S'il existe encore des aliens dans le Fleet, on définit de nouvelles coordonnées des bullets des aliens et si le bullet touche l'alien, on fait appel à la méthode de alien_touched() pour gérer cette collision, on supprime le bullet et l'alien du canvas.

```
self.alien.alien_touched_by(self.canvas,self.defender.fired_bullets[i])

canvas.delete(self.defender.fired_bullets[i])

canvas.delete(self.aliens_fleet[j])
```

## La classe Alien()

Classique, la méthode install_in() permet le positionnement de l'objet alien via la création d'une image maintenant

```
self.id = canvas.create_image(x, y, image=self.alien, tags="alien")
```

La méthode move_in() permet de faire bouger l'objet alien et la méthode alien_touched_by() permet de créer une animation lorsque le bullet touche l'alien via ce code

```
explosion = canvas.create_image(x+(x1-x)/2, y1+(y1-y)/2,
image=self.explosion, tags="explosion")

        canvas.after(100,canvas.delete,explosion)
```

## La classe Bullet()

Les accesseurs permettent de mettre à jour les coordonnées du bullet. Les méthodes install_in(), move_in() et move_in_aliens() permettent la création d'un objet rouge qui sera un bullet

```
self.id = canvas.create_oval(x,y,x+r,y+r,fill="red")
```

et de faire bouger ce bullet du Defender d'une part

```
canvas.move(self.id, 0, -self.speed)
```

et d'autre part, des aliens

```
canvas.move(self.id, 0, self.speed)
```

## Les classes Shield() et multiple_shields()

De part les méthodes classiques de positionnement install in(), on a une méthode shield_touched() permet de contrôler la collision d'un bullet avec un shield avec le même processus que sur le Fleet : Mémorisation et suppression des bullets tirés d'une part, et mémorisation et suppression des shields d'autre part.

```
canvas.delete(fleet.fired_bullet[j])

del fleet.fired_bullet[j]

canvas.delete(self.list_shields[k])

del self.list_shields[k]

res=True

res1=True

break
```

## La classe Game()

Cette classe appelle la majorité des classes dans la méthode __init__() car elle permet la manipulation des différentes classes pour faire des jeux de données ou des suppositions.

```
self.fleet = Fleet()

self.multiple_shields=multiple_shields()

self.defender = Defender()

self.defender.install_in(self.canvas)

self.fleet.install_in(self.canvas)

self.multiple_shields.install_in(self.canvas,50,600)
```

Cette classe comporte plusieurs méthodes d'appel importantes. La méthode start_animation() permet l'affichage de la méthode animation() qui permet la configuration des points, le score et la vie du Defender, l'affichage de la vie du Defender sous forme de Label, l'affichage du Titre principal.

L'accesseur set_score() permet de définir la valeur du score qui configurer dans la méthode animation(). Enfin, les méthodes move_bullets() et move_aliens_fleet() permettent de faire d'autres appels de classes sous-méthodes notamment Defender et Fleet().

### La classe Space Invaders()

La méthode __init__() permet de définir les dimensions de la fenêtre graphique et d'appeler la méthode game(). La méthode play() quant à elle permet l'activation du clavier et l'affichage de la fenêtre du jeu et le tour est joué. Il y a également un appel de la bibliothèque Audio qui va permettre de  jouer une musique en arrière-plan histoire d'animer le jeu.

```
self.musique_menu = PlaySound("audio1.wav", SND_FILENAME | SND_ASYNC) #
```

Le jeu commence grâce à ce code

```
jeu=SpaceInvaders()

jeu.play()
```

## Idées pour l'amélioration du projet

Pour la partie de IA (Intelligence Artificielle) envisagée dans le projet Space Invaders, on peut imaginer que les Aliens vont pouvoir analyser les mouvements du Defender; de ce fait, pour rendre la tâche plus difficile au joueur, il change de tactiques de jeu car les Aliens après qu'ils aient dépassé la moitié du trajet, ils vont mémoriser et comprendre la tactique du joueur et même sa manière de bouger, et donc leurs tirs vont être mieux ciblées dans le but de toucher le Defender.

Cela va pousser le joueur à être plus prudent à son jeu et ses mouvements, et surtout quand les Aliens ont déjà traversé la moitié du trajet.

## Conclusion

En somme, le jeu Space Invaders conçu avec le langage Python a été fait bout à bout avec des différentes classes et méthodes interconnectées. Des propositions supplémentaires ont été ajoutées comme l'indicateur de vie, le Fleet quand il touche le bas de l'écran, le jeu se termine.