TP : Programmer un jeu

Le but de ce TP est d'utiliser une bibliothèque de programmation graphique pour développer un jeu.

1 La bibliothèque pygame

On utilisera la bibliothèque pygame (`https ://www.pygame.org/docs/`) qui permet à un programme python de dessiner à l'écran, de gérer le clavier, la souris, ...

1.1 Importer la bibliothèque

```
import pygame
from pygame.constants import *
```

1.2 Initialiser la bibliothèque

Et créer une fenêtre de 640 pixels de large et 480 pixels de hauteur

```
pygame.init()
fenetre = pygame.display.set_mode((640, 480))
```

1.3 Dessiner dans la fenêtre

Un rectangle :

```
pygame.draw.rect(fenetre,          # fenêtre sur laquelle on dessine
                 (255,0,0),        # couleur (rouge)
                 (10, 15, 4, 5),   # rectangle à dessiner (x,y,l,h)
                 3                 # épaisseur du trait en pixels
                )
```

Si l'épaisseur du trait est égale à 0, le rectangle est rempli avec la couleur

Attention ! le système de coordonnées utilisé par pygame a son axe y orienté vers le bas. Le point de coordonnées $(0\,,0)$ se trouve en haut à gauche de l'écran.

Une ellipse :

```
pygame.draw.ellipse(fenetre,          # fenêtre sur laquelle on dessine
                    (0,255,0),        # couleur (vert)
                    (10, 15, 4, 5),   # rectangle contenant l'ellipse
                    3                 # épaisseur du trait en pixels
                   )
```

Si l'épaisseur du trait est égale à 0, l'ellipse est rempli avec la couleur

Une ligne :

```
pygame.draw.line(fenetre,           # fenêtre sur laquelle on dessine
                 (0,0,255),         # couleur (bleu)
                 [10, 15], [4, 5],  # (x1,y1) début et (x2, y2) fin
                 3                  # épaisseur du trait en pixels
                )
```

Une image :

```
image = pygame.image.load("nom_de_fichier.jpg")  # Charge l'image
image.convert()                                  # Accélère l'affichage
fenetre.blit(image,(x,y))                        # Affiche l'image
```

1.4 Gérer des événements

```
continuer = True
while(continuer):                    # Boucle principale du jeu
  for event in pygame.event.get():   # Boucle sur les événements en attente
    if event.type == QUIT:           # L'utilisateur demande à quitter l'application
      continuer = False
      pygame.quit()
    if event.type == KEYDOWN:        # L'utilisateur appuie sur une touche
      if event.key == K_DOWN:
        # Faire quelque-chose si l'utilisateur appuie sur
        # la flèche du bas.
      elif event.key == K_UP:
        # Faire autre chose si l'utilisateur appuie sur
        # la flèche du haut
```

La bibliothèque pygame gère également les événements `KEYUP`, `MOUSEMOTION`, `MOUSEBUTTONDOWN`, `MOUSEBUTTONUP`, ... Voir la documentation

2 Snake

Le but de ce TP est de programmer le jeu "Snake" où le joueur dirige un serpent qui doit manger des pommes. À chaque pomme ingurgitée, le serpent grandit un peu.

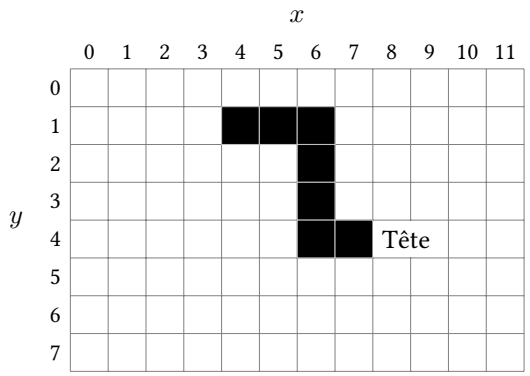
La difficulté du jeu vient du fait que le serpent ne doit pas rencontrer d'obstacle, son corps constituant un obstacle ainsi que les bords de l'écran. Plus le serpent mange de pommes, plus il gagne de points. Mais il grandit également et il lui devient de plus en plus difficile d'éviter les collisions avec son corps.

2.1 Programme de base

On utilisera le programme de base disponible à l'adresse suivante : `http ://tsi-troyes-physique.bitballoon.com/fichiers/info/snake_base.py`

Ce programme permet de gérer le déplacement du serpent à l'écran.

Le serpent est représenté par une liste de points `[[x1,y1], [x2, 2], ...]`. Le premier point de la liste est la tête et le dernier point est le bout de la queue du serpent.



`S = [[7,4],[6,4],[6,3],[6,2],[6,1],[5,1],[4,1]]`

Le jeu se déroule dans une fenêtre de 640 pixels de large et 480 pixels de haut séparée en 64×48 petits carrés de 10 pixels de côté. Le corps du serpent est aussi constitué de 10 pixels de côté.

La variable `dir` est une liste `[dx,dy]` indique le déplacement suivant x et y de la tête du serpent à chaque itération. par exemple `dir=[1,0]` correspond à un serpent se dirigeant vers la droite et `dir=[0,1]` correspond à un serpent se dirigeant vers le bas.(l'axe y est orienté vers le bas)

2.2 Développement

Il faut maintenant compléter le programme de base afin de le transformer en un jeu un peu plus intéressant.

2.2.1 Collisions

Le serpent ne doit pas pouvoir sortir de l'écran. Il faut donc ajouter des bords à la zone de jeu (choisir une couleur) et gérer les collisions du serpent avec le bord. Si le serpent entre en collision avec le bord de l'écran, le joueur a perdu, le jeu doit recommencer. On peut également coder une variante où lorsque le serprent sort de l'écran, il rentre par le bord opposé.

Le serpent ne doit pas pouvoir traverser son corps. S'il rentre en collision avec son corps, le joueur a perdu.

Le serpent ne peut pas faire demi-tour. Il faudra donc interdire ce déplacement car il provoque une collision du serpent sur lui-même.

2.2.2 Manger des pommes

L'objectif du serpent est d'ingurgiter le maximum de pommes. À chaque pomme gobée, le serpent grandit un peu. Il faut donc faire apparaître des pommes à l'écran. Lorsque la tête du serpent passe sur la pomme il se passe plusieurs choses :

- La pomme disparait
- Une nouvelle pomme apparait ailleurs (au hasard)
- Le serpent grandit d'une unité
- Le joueur gagne des points

2.2.3 Gérer le score

Lorsque le serpent mange une pomme, le joueur gagne des points. Il faut afficher les points quelque part à l'écran et les rappeler au moment où le joueur perd la partie.

On pourra gérer une liste de *high scores* avec les noms des joueurs correspondants. Cette liste devra être stockée dans un fichier pour être conservée lorsque le programme est arrêté.

2.2.4 Aller plus loin

À ce stade nous avons un jeu qui commence à être jouable. Pour en faire un vrai bon jeu, il y a encore un peu de travail. C'est là que la créativité du développeur intervient. On pourra par exemple :

- Travailler les graphismes, des petits carrés de couleur unie ne sont pas forcément très attirants. On pourra afficher des images, pour avoir une pomme qui ressemble à une pomme, et un serpent qui ressemble à un serpent. On pourra aussi améliorer le graphisme des murs et de l'espace où se déplace le serpent.
- Un jeu qui ne fait pas de bruit ça n'est pas vraiment un jeu. On pourra utiliser la bibliothèque `pygame` pour jouer une petite musique entrainante pendant la partie, et des effets sonores quand le serpent mange une pomme où lorsqu'il percute un mur. (documentation : `https ://www.pygame.org/docs/ref/mixer.html` et `https ://www.pygame.org/docs/ref/music.html`). Vous pourrez trouver des effets sonores à l'adresse suivante : `https ://freesound.org/`
- Multijoueurs ! Jouer à plusieurs c'est plus amusant, vous pouvez ajouter un second serpent (ou plus !) et transformer le jeu en compétition. Le but devient de manger autant de pommes que possible pour grandir et bloquer le joueur adverse. Le dernier serpent en vie gagne.