Projet - jeu Snake

Tom Avenel

2022 / 2023

Résumé

L'objectif de ce projet est de coder un jeu du serpent en pygame.

Table des matières

Programmation orientée objet

Classe Case

Créer une classe `Case` servant à représenter une case du plateau.

Cette classe a 2 attributs : `x` et `y` (coordonnées de la case).

Implémenter les méthodes spéciales :

- `__eq__(self, other) -> bool` retournant `True` si et seulement si `self` et `other` représentent la même case (mêmes coordonnées).
- `__repr__(self) -> str` et `__str__(self) -> str` permettant de transformer une instance de `Case` en une chaîne de caractères représentant l'instance (on pourra utiliser la même implémentation dans les 2 méthodes).

Classe Snake

Créer une classe `Snake` permettant de représenter le serpent. Un serpent est composé d'une liste de `Case`.

Ajouter des méthodes :

- `head(self) -> Case` : retourne la `Case` de la tête du serpent.
- `get_all_cases(self) -> list[Case]` : retourne l'ensemble des `Case` représentant le serpent
- `move(self, new_case: Case, remove_tail: bool)` : permet de déplacer le serpent vers une nouvelle case.
  - **Attention : `new_case` peut valoir `None` - dans ce cas, le serpent continue à se déplacer dans la même direction que précédemmment !**
- `is_out_of_frame(self) -> bool` : retourne `True` si le serpent sort du cadre du jeu (on utilisera `WINDOW_X-10` et `WINDOW_Y-10` comme limites du cadre).
- `is_head_touching_body(self) -> bool` : retourne `True` si la tête du serpent touche son corps.

Programme pygame

Boucle principale

En s'inspirant du canevas de code ci-dessous, coder la boucle principale permettant de déplacer le serpent lorsque le joueur appuie sur les touches du clavier.

On déplacera le serpent de 10 **unités dans une direction à chaque itération.**

```
# importing libraries
import pygame
import time
import random

SNAKE_SPEED = 10

# Window size
WINDOW_X = 400
WINDOW_Y = 300

# defining colors
BLACK = pygame.Color(0, 0, 0)
WHITE = pygame.Color(255, 255, 255)
GREEN = pygame.Color(0, 255, 0)

# Initialising pygame
pygame.init()

# Initialise game window
pygame.display.set_caption('Jeu du serpent')
game_window = pygame.display.set_mode((WINDOW_X, WINDOW_Y))

# FPS (frames per second) controller
fps = pygame.time.Clock()

# Main Function
while True:

    # handling key events
    for event in pygame.event.get():
```

```
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP:
                # Déplacer le serpent de 10 unités vers le haut, ...
                pass
            if event.key == pygame.K_DOWN:
                pass
            if event.key == pygame.K_LEFT:
                pass
            if event.key == pygame.K_RIGHT:
                pass

    # Déplacer le serpent

    # ------------------------ #
    # Représentation graphique
    game_window.fill(BLACK)

    # Dessin du fruit (case `fruit`)
    #pygame.draw.rect(game_window, WHITE, pygame.Rect(fruit.x, fruit.y, 10, 10))

    # Dessin du serpent
    for case in snake.get_all_cases():
        pygame.draw.rect(game_window, GREEN, pygame.Rect(case.x, case.y, 10, 10))

    # Affichage du score
    # surface = pygame.font.SysFont('times new roman', 20).render('Score : ' + str(score), Tr
    # rect = surface.get_rect()
    # game_window.blit(surface, rect)

    # Mise à jour de l'écrant
    pygame.display.update()

    # Frame Per Second /Refresh Rate
    fps.tick(SNAKE_SPEED)
```

Gestions des fruits

Génération des fruits

Ajouter une fonction dans le programme principal : `new_fruit()`. Cette fonction crée une nouvelle case aléatoire représentant le fruit à faire manger par le serpent. Appeler cette fonction pour générer un nouveau fruit en début de partie et lorsque le serpent mange un fruit (il y a donc toujours un et un seul fruit sur le plateau).

Ajout du score

Ajouter une gestion du score : chaque fruit mangé par le serpent rapporte **10 points**.

Manger un fruit fait grossir le serpent

Si le serpent mange un fruit (si la tête du serpent traverse un fruit), le corps du serpent grossit.

Game Over

Ajouter les tests permettant de gérer la fin du jeu.

Une fois le jeu terminé, faire exécuter le code suivant :

```
# after 2 seconds we will quit the program
time.sleep(2)
pygame.quit()
quit()
```

Ce document a été généré grâce à l'outil pandoc.

Les sources au format Markdown de ce document sont disponibles sur le site web.