Snake Python
Documentation

EPSI B3 - Classe 1 2020/2021

Membres projet :

- DRAPEAU Fabien
- PLATON Samuel
- LE MAUFF Loan
- MERCERON Simon

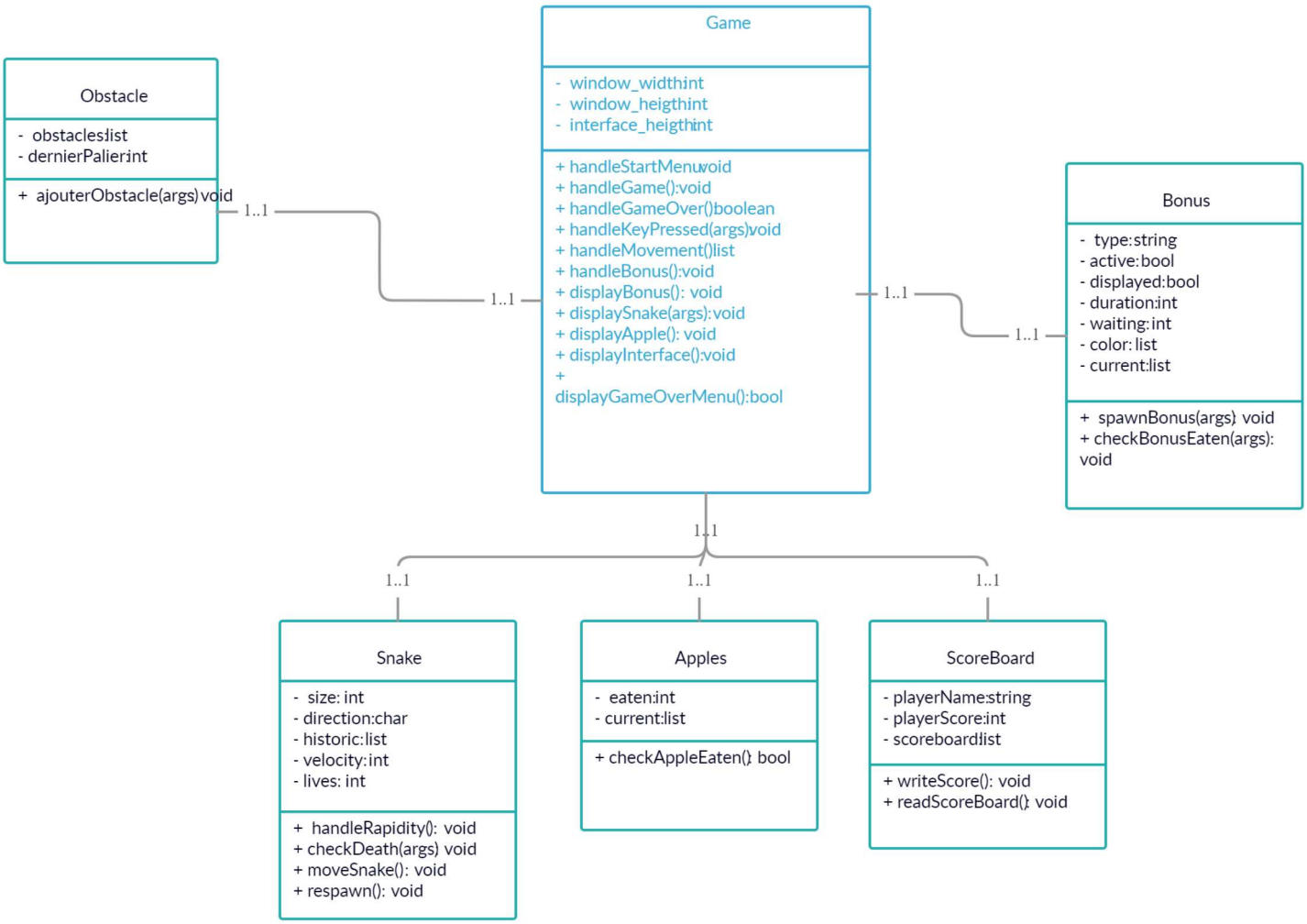
epsi

Lien du projet : https://github.com/SamuelPlaton/epsi-python-snake

Dépendances :

Au niveau des dépendances de librairies, seul *pygame* est nécessaire pour la version actuelle du projet, il est alors possible de l'installer en effectuant la commande :
pip install pygame **OU** *pip install -r requirements.txt*

Diagramme UML :



Classes :

Game :

Attributs :

apples Apples() : Object Apples()
snake Snake() : Object Snake()
scoreboard Scoreboard() : Object ScoreBoard()
window_width: int largeur du tableau
window_heigth: int hauteur du tableau
interface_heigth: int hauteur de l'interface (Qui affiche score et vies)

Méthodes:

__init__ : Initialise et lance la partie
handleStartMenu() : Affiche le menu
handleGame() : Affiche la partie de snake
handleGameOver() : Gère le game over du joueur
handleKeyPressed() : Gère lorsque le joueur appuie sur un bouton
handleMovement() : Affiche le prochain mouvement du snake sur l'interface
handleBonus(): Gère les effets des bonus
displaySnake() : Affiche graphiquement le snake
displayApple() : Affiche graphiquement la pomme
displayInterface() : Affiche graphiquement l'interface, comprenant score et vies
displayObstacle(): Affiche graphiquement les obstacles
displayBonus(): Affiche graphiquement les bonus
handleGameOverMenu() : Affiche le GameOver
handleScoreBoard() : Affiche le scoreBoard au GameOver

Snake :

Attributs :

size int: Taille du snake
direction char: Direction du snake
historic list: liste des positions du snake
velocity int: rapidité du snake
lives int: nombre de vies restantes

Méthodes:

__init__ : Initialise le snake
handleRapidity() : Gère la rapidité du snake selon la taille

checkDeath(): Check si le serpent perd une vie ou non, s'il se prend un mur, un obstacle ou lui-même

moveSnake() : Décide des prochaines coordonnées du snake selon sa direction

respawn() : Fais respawn le snake dans un ordre bien précis selon sa taille

Apples :

Attributs :

eaten int: nombre de pommes mangées

current list: tableau x y comprenant la position de la pomme actuelle

Méthodes:

__init__ : Initialise les pommes

checkAppleEaten() : Vérifie si la pomme a été mangée, si oui change les coordonnées de la prochaine

ScoreBoard :

Attributs :

playerName string: Nom du joueur en cours

playerScore int: Score du joueur en cours

scoreBoard list: ScoreBoard des 10 meilleurs joueurs

Méthodes:

__init__ : Initialise le scoreboard

writeScore() : A la fin de la partie, écrit le score dans les nouvelles données

readScoreBoard(): Lit le score et nom des 10 meilleurs joueurs

Bonus:

Attributs:

type: Bonus ou malus

active: savoir si le bonus/malus est actif sur le joueur

displayed: le bonus/malus est afficher

duration: durée du bonus/malus lorsque actif

waiting: temps d'affichage si non ramassé

color: couleur du bonus

current : position du bonus/malus

Méthodes

__init__ : initialise les bonus/malus

spawnBonus: fais spawn le bonus en déterminant son type (bonus/malus), son effet, et ses coordonnées

checkBonusEaten: vérifier si le bonus a été mangé, si oui enclenche l'effet sur le joueur pour une durée déterminée

obstacles:

Attributs:

obstacles[]: position de l'obstacle

dernierPalier: nombre de pomme mangé avant apparition de l'obstacle

Méthodes:

__init__:initialise obstacles

ajouterObstacle: place l'obstacle sur le jeu en définissant des coordonnées aléatoires