**PROJET TSI**

**OpenGL**



2021-2022

SOMMAIRE :

* I) Présentation du jeu
* II) Création des objets :

II.1) Personnage

II.2) Créations du plateau

II.3) Création des murs

II.4) Création des obstacles

II.5) Message de fin de partie

* III) Explication des méthodes :

III.1) Update\_key

III.2) Gestion du saut

III.3) Camera

III.4) Plateau

III.5) Mur

III.6) Obstacles

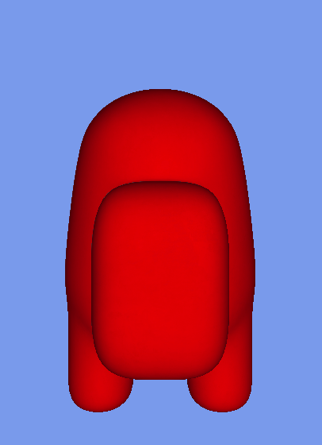
**I) PRESENTATION DU JEU**

Le but de ce projet est de crée un jeu à l’aide de OpenGL en python. Nous avons choisi de réaliser un jeu de parcours d’obstacle à la façon de « Subway surfer ». Le but est simple, le personnage se situe entre deux murs de part et d’autre de lui, qui délimite les bords de la carte, le personnage incarné par le joueur doit éviter les obstacles qui apparaisse devant lui. L’objectif est de parcourir la plus grande distance sans toucher un obstacle.

**II) CÉATIONS DES OBJETS**

**II.1) PERSONNAGE**

On a choisi pour notre personnage un avatar du jeu « Among us ». Le modèle à une taille de [1,1,1,1].

Ce modèle est ajouter à la liste des objets (objs) à la position d’indice 0 de la classe ViewerGL.

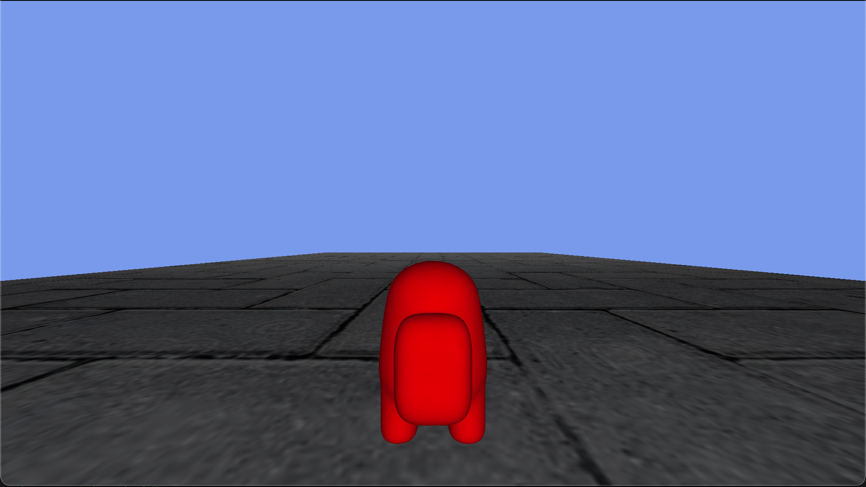
Une image contenant texte

Description générée automatiquement

**II.2) CRÉATION DU PLATEAU**

Pour la création du plateu on créé 3 plan carré de 50 bloc de côté, avec le premier plan centré en 0, Le duxième à +50 sur z et le troisième à +100 sur z. Ces trois plans sont stockés dans la liste des objets (objs) de la position d’indice -5 à -3 de la classe ViewerGL.

Nous verrons pourquoi nous avons créé 3 plateau dans la partie III.4)



Une image contenant texte

Description générée automatiquement

**II.3) CRÉATION DES MURS**

Pour la création des murs nous avons d’abord créé un cube de 1 de côté sur Blender.

Nous avons chargé une seul fois la texture sur le GPU, puis créé le mur de droite à la position +4 sur x tout le long du plateau sur 3 blocs de haut, puis créé le mur de gauche à la position -4 sur x tout le long du plateau sur 3 blocs de haut. Tous les blocs sont stockés dans la liste des objets (objs) de la position d’indice 1 à 241 de la classe ViewerGL.

Une image contenant texte

Description générée automatiquementUne image contenant orange, brique, bâtiment, matériau de construction

Description générée automatiquement

**II.4) CRÉATION DES OBSTACLES**

Les obstacles utilises les mêmes formes et même texture que les blocs qui composent les murs. Ces obstacles sont générer à des positions aléatoires comprises entre [-3, 3] sur x entre [0, 4] sur y et entre [0, 24] sur z. Tous les obstacles sont stockés dans la liste des objets (objs) de la position d’indice 242 à 246 de la classe ViewerGL.

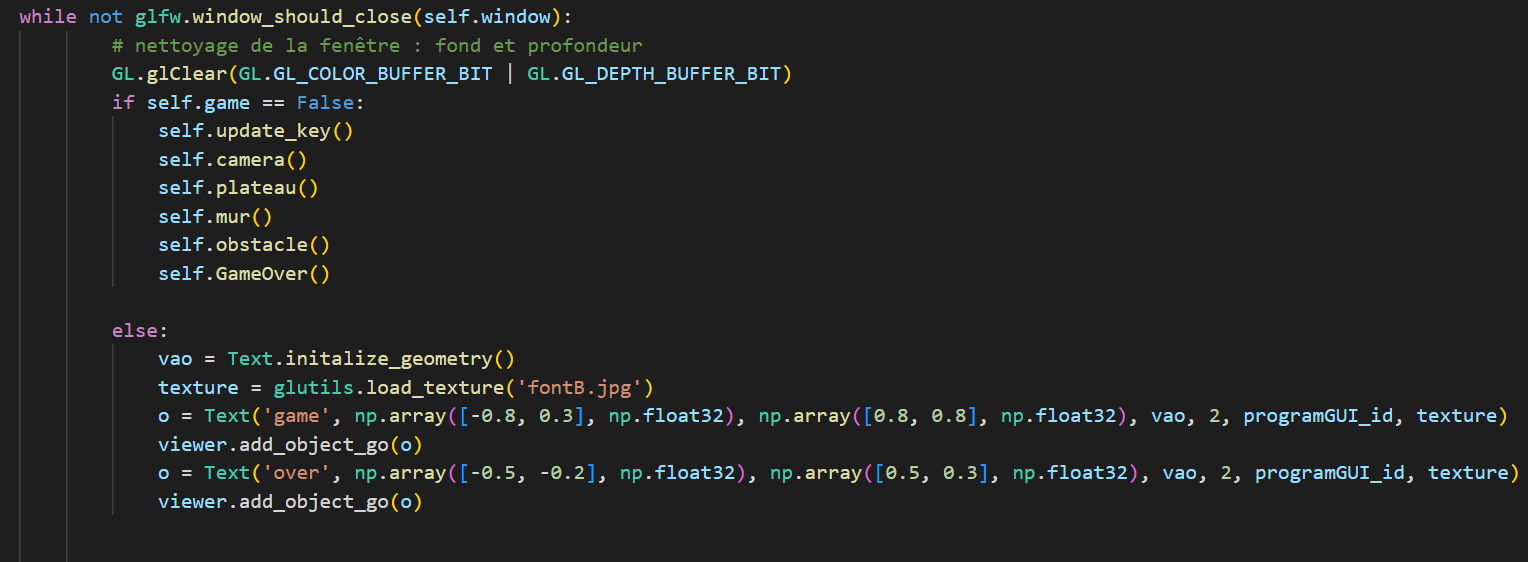
Une image contenant brique, orange, matériau de construction, pierre

Description générée automatiquement

**II.5) MESSAGE DE FIN DE PARTIE**

Une image contenant graphiques vectoriels

Description générée automatiquement Ce message est simplement un texte qui s’affiche sur l’écran lorsque le joueur à perdue. Une variable self.game passe à False et le message s’affiche et le jeu s’arrete en meme temps.



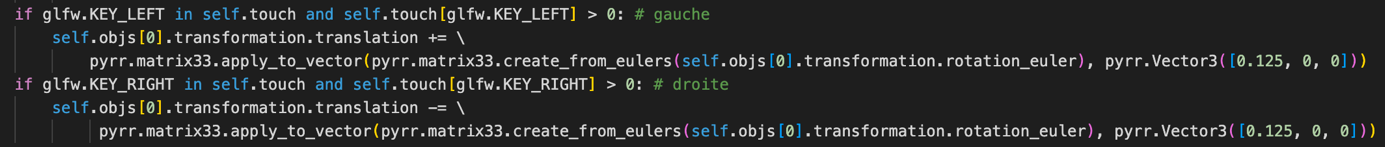
**III) Explications des méthodes**

**III.1) Update\_key**

La méthode Update\_key est méthodes qui gère tous les déplacements du personnage. Les touches utilisées pour se déplacer sont KEY\_LEFT pour se déplacer à gauche KEY\_RIGHT pour se déplacer à droite sur l’axe des x. Le personnage se déplace tout seul vers l’avant et sa vitesse augmente progressivement au cours du temps pour complexifier un peu le jeu.

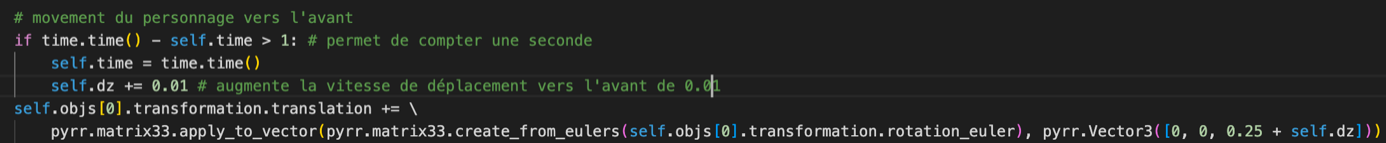
KEY\_LEFT et KEY\_RIGHT:

Lorsque le joueur maintient la touche KEY\_LEFT ou la touche KEY\_RIGHT le personnage se déplace à gauche ou à droite de façon constante (± 0.125 sur x).



Déplacement vers l’avant :

Le personnage se déplace tout seul vers l’avant c’est-à-dire sur l’axe z, de plus la vitesse à laquelle il se déplace augment progressivement au cours du temps. L’augmentation est de + 0.01 par seconde.



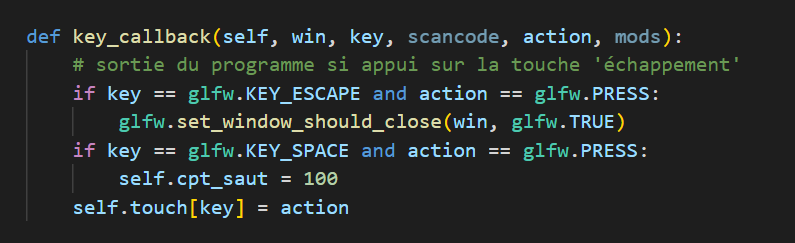
**III.2) Gestion du saut**

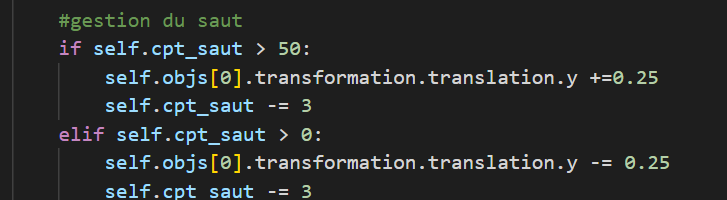
Nous avons réalisé un saut en cloche. Pour cela nous avons commencé par initialiser un conteur self.cpt\_saut = 0

Ensuite au moment de l’appuie sur espace, ce conteur passe a 100, cela va avoir pour conséquence de rentrer dans le if défini dans la méthode update\_key.

Celui-ci va faire monter notre personnage sur l’axe des y et décrémenter le conteur quand celui-ci arrive a 50 (la moitié), notre personnage redescend.

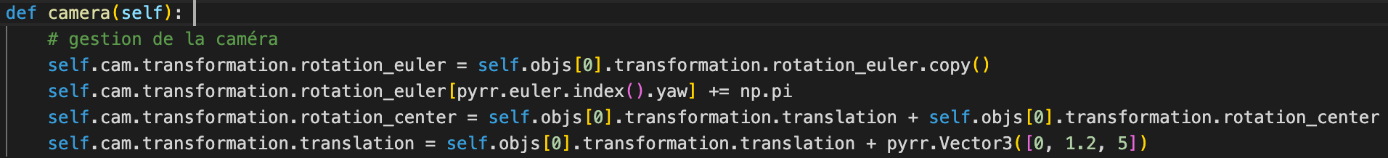
Nous avons détecté un bug que nous avons pas eu le temps de résoudre. En effet si on appuie plusieurs fois sur espace, notre personnage s’envole.





**III.2) Camera**

La méthode caméra permet de réactualiser la position de la caméra de façon à toujours voir le personnage en vue à la troisième personne, quelle que soit les déplacements du personnage.



**III.3) Plateau**

La méthode plateau permet de créer un plateau de taille infini avec seulement trois objet. Comme vue précédemment nous avons créé trois plateaux carrés de 50 blocs, lorsque l’un de ces trois plateaux passe derrière la position du personnage selon z, c’est-à-dire au de vue du joueur, celui-ci se téléporte à +150 blocs devant le joueur. Se procéder permet d’avoir l’impression que le plateau est infini.

Une image contenant texte

Description générée automatiquement

**III.4) Mur**

La méthode mur permet de créer un mur de taille infinie de part et d’autre du joueur avec les objets murs créés précédemment. À chaque fois que l’un des murs passe derrière la position du personnage selon z, celui-ci est téléporter à +80 blocs devant le joueur. Se procéder permet d’avoir l’impression que les murs sont de taille infinie.

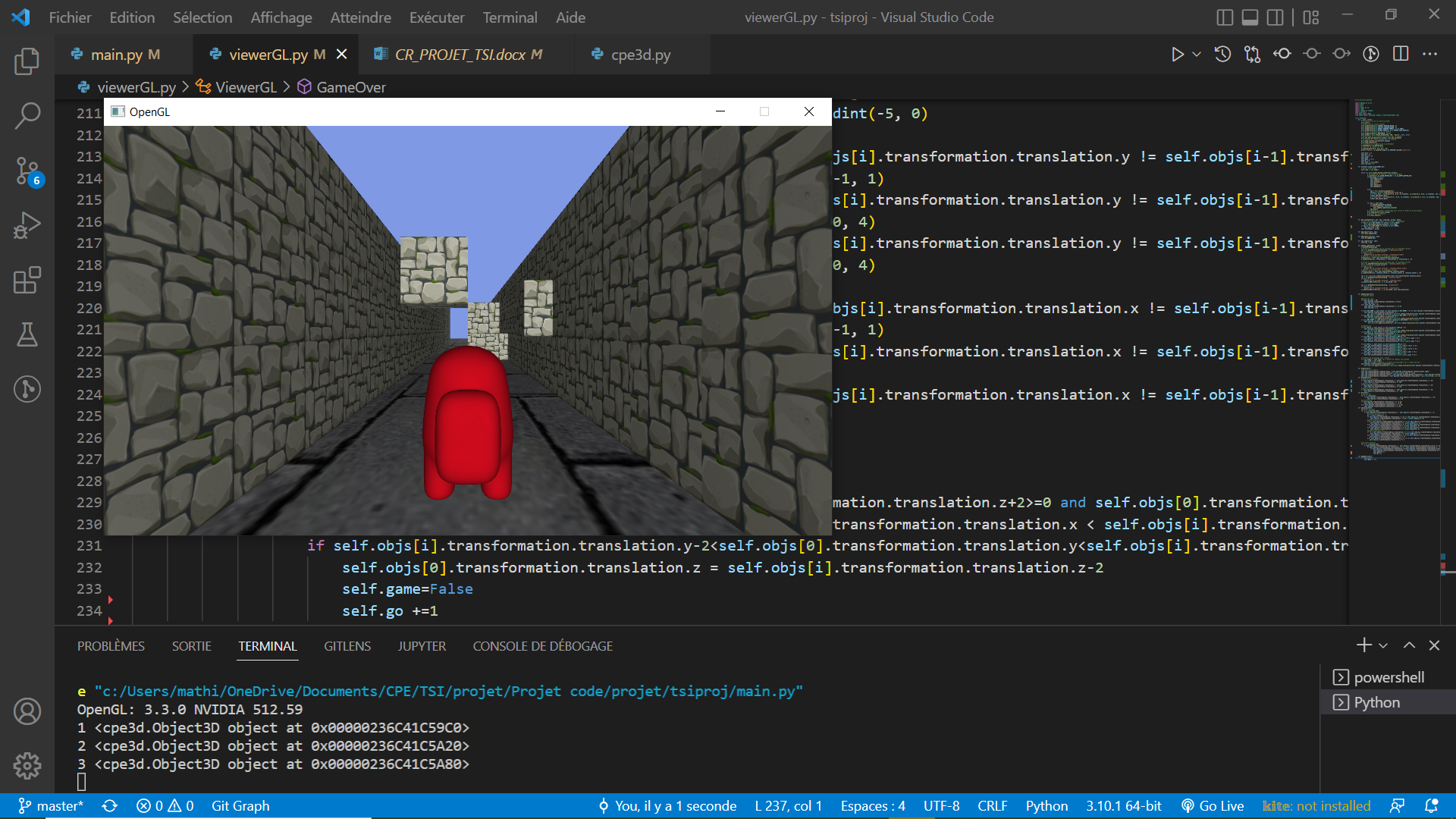
Cette méthode gère aussi les collisions du personnage avec les murs. Lorsque le personnage touche un mur sa position sur x reste inchangé jusqu’à ce que le joueur change de direction.

Une image contenant texte

Description générée automatiquement

**III.5) Obstacles**

La méthode obstacle permet de faire réapparaitre les obstacles créés précédemment, à des positions aléatoires, devant le personnage. À chaque fois que l’un des obstacles passe derrière la position du personnage selon z, celui-ci est téléporter à +50 + randint(-5,0) blocs devant le joueur selon z, avec une position sur x comprise entre [-2,2] et une position sur y comprise entre [0, 3]. Cette méthode gère aussi les collisions entre le personnage et les différents obstacles. Detecter une position exacte n’est pas fonctionel car il y a peu de chance que le personnage atteigne un valeur précise. Donc la detection des collisions se fait entre la position de notre personnage et l’obstacle avec une marge de 0.3 selon z

Une image contenant texte

Description générée automatiquement

