PACCOUD William

Riss Ryan

04/01/2022

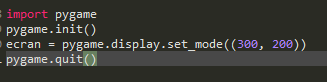
# Rapport de TP3 Final – Représentation visuelle d’objets

## Introduction

Ce TP porte sur la représentation visuelle d’objets. Pour cela, on va utiliser le moteur de jeu pygame pour obtenir une fenêtre dans laquelle nous allons pouvoir afficher des éléments 3D à l’aide du module OpenGL. Pour cela, il va falloir se familiariser avec l’espace 3D d’openGL et les différentes opérations possibles dans cet espace.

## Utilisation de Pygame

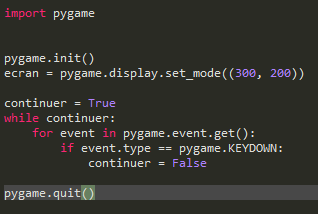
### Initialisation



Dans ce bout de code, d’abord on importe pygame. D’après la documentation, la méthode init() initialise tous les modules de pygame. La ligne pygame.display.set\_mode initialise une nouvelle fenêtre avec la résolution passée en argument qu’on va pouvoir manipuler en la récupérant dans une variable. Finalement, pygame.quit() désinitialise tous les modules pygames. D’après la documentation, cette ligne n’est pas nécessaire sauf si le programme doit continuer après car cette méthode est appelée quand l’interpreteur python se ferme.

Dans la pratique, on observe une fênetre qui s’ouvre et se ferme instantanément.

### Évènement

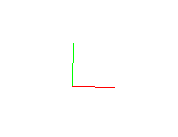


Cette fois-ci, on utilise la variable continuer pour créer une boucle infinie. A l’intérieur, on vérifie à l’aide la la méthode pygame.event.get qui retourne la liste de tous les évènements en attente si le type d’un des évènements correspond à la constante pygame.KEYDOWN qui correspond au type d’un évènement d’une touche pressé. Ainsi quand on presse n’importe quelle touche, la boucle s’arrête et la fenêtre se ferme.

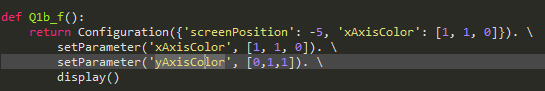
1. **Découverte de l’environnement du travail de TP**

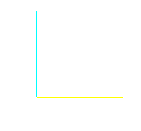
### Configuration





Dans le fichier main, on a plusieurs fonctions qui définissent plusieurs configurations. Pour cette question, on utilise la configuration par défaut qui initialise une fenêtre et dessine deux axes qui peuvent être tournés avec les touches du clavier. La configuration est une classe qui se charge d’initialiser les modules pygame et openGL et la position initiale de la caméra. Elle dispose d’une méthode add() pour ajouter des objets à dessinner et une méthode draw pour dessinner des axes et tous les objets ajoutés. Enfin, elle dispose d’une fonction display qui se charge d’afficher une fenêtre et gérer les évènements.

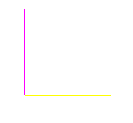




On observe que l’axe des X et des Y ont changé de couleur et que la caméra a avancé. En effet, on a modifié les paramètres screenPoistion, xAxisColor et yAxisColor.

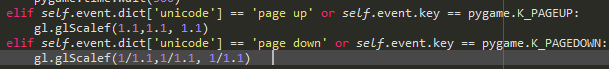
Il est possible de chaîner les méthodes car setParameter() retourne la nouvelle configuration. Modifier le paramètre screenPosition nécessite de recalculer la matrice de perspective par rapport à notre nouvelle position.



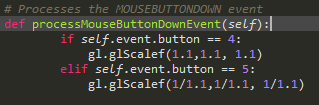


Pour que l’axe des z (dont on a changé la couleur pour mieux le distingué) soit représenté verticalement, il faut effectué une rotation de -90° sur l’axe des x.

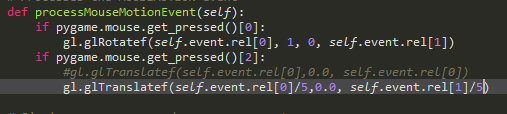
## Mise en place des interactions avec l’utilisateur avec Pygame



On souhaite ajouter la gestion du zoom. Pour cela on rajoute dans la méthode processKeyDownEvent les lignes ci-dessus pour détecter les touches page up et page down et effectuer un zoom avec la fonction gl.glScalef.



Pour que le zoom soit géré par la souris, on vérifie dans la fonction processMouseButtonDown et on vérifie la valeur du bouton.



Pour le déplacement des objets, on modifie la méthode processMouseMotionEvent pour effectuer une rotation ou une translation en fonction du déplacement de la souris celon le clic pressé. Pour la translation, on divise le déplacement de la souris par 5 pour qu’elle soit plus fluide.

## Création d’une section

Une image contenant texte

Description générée automatiquement

Pour créer une surface, on crée d’abord les différents points de la section à partir de la position de la section et des ses différents paramètres, puis on définit les six faces à partir de ces points.

Une image contenant texte

Description générée automatiquement

Dans la fonction draw, on trace les différentes points de la section qui seront reliés en quadrilatères grâce au mode approprié tout en redéfinissant une matrice de projection propre à la section tout en conservant la matrice initiale à l’aide des fonctions glPushMatrice et glPopMatrice.

Une image contenant flèche

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Pour afficher les arêtes de la section, on suit le même principe que pour les faces, cette fois en utilisant le mode correspondant pour dessiner des lignes.

Une image contenant texte

Description générée automatiquement

Dans la fonction draw, on rajoute une condition pour que les arêtes soit dessinées avant les faces si le paramètre edges est vrai.

Une image contenant texte, équipement électronique

Description générée automatiquement

## Création des murs

Dans le constructeur de la classe Wall, on observe qu’un mur est composé de plusieurs objets qui le compose, ici dans l’exemple seulement une section qu’on ajoute à la liste de ses objets. Ainsi, pour dessiner un mur il faudra dessiner chacun de ses objets.

Une image contenant texte

Description générée automatiquement

Pour la fonction draw, il suffit donc pour chaque objet du mur créer une matrice de projection temporaire, effectuer la rotation et appeler la fonction draw de l’objet.

Une image contenant texte

Description générée automatiquement

## Création d’une maison

La méthode draw de la classe House est identique à celle d’un mur :

Une image contenant texte

Description générée automatiquement

Pour dessiner une maison, on crée deux murs sans orientations et deux murs à 90° et on place correctement les différents murs en fonction de la position du premier et de leur taille :

Une image contenant texte

Description générée automatiquement

Une image contenant texte, carte de visite

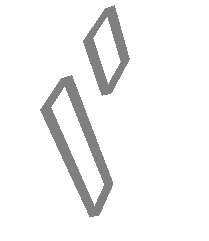
Description générée automatiquement

## Création d’ouverture

Pour compléter la classe Opening, on reprend le code de la classe Section car une ouverture n’est enfait qu’une section à laquelle on retire les faces avant et arrière, ce qui donne le code suivant :

Une image contenant texte

Description générée automatiquement



Dans la fonction canCreateOpening, on vérfie que l’ouverture a bien la même thickness que la section et que l’ouverture ne dépasse pas de cette dernière.

Une image contenant texte

Description générée automatiquement

On vérifie, les tests sont bien cohérents :

Une image contenant texte

Description générée automatiquement

La dernière ouverture ne peut pas être créée car elle dépasserait la section en hauteur.

Pour la fonction createNewSections, on vérifie que chaque surface a lieu d’être créée avant de l’ajouter à la liste des nouvelles sections.

Une image contenant texte

Description générée automatiquement

En testant, on obtient bien les résultats désirés :

Une image contenant texte

Description générée automatiquement

Ici, la fonction enumerate sert à obtenir le couple (index de l’item, item) dans la boucle for.

Avec le code suivant :

wall = Wall({'width':7, 'height':2.6,})

opening1 = Opening({'position': [2, 0, 0], 'width':0.9, 'height':2.15, 'thickness':0.2, 'color': [0.7, 0.7, 0.7]})

section = wall.findSection(opening1)

section[0] vaut 0 et section[1] est la section parente du mur car l’ouverture peut être insérer sur celle-ci.

Pour la fonction add de la classe Wall, on commence par trouver la section sur laquelle insérer l’ouverture, on créer ensuite les nouvelles sections qu’on insère dans les objets du mur après avoir retirer l’ancienne section.

Une image contenant texte

Description générée automatiquement



## Conclusion

Dans cette première partie de TP, nous avons pu nous familiariser avec le fonctionnement interne d’un moteur 3D, sans surprise une bonne compréhension des matrices est nécessaires pour comprendre ce que fait le moteur lors des différentes opérations de translations, rotations, etc… . Il est intéressant mais néanmoins un peu fastidieux de créer des représentations 3D avec OpenGL car l’outil reste relativement bas niveau et plusieurs étapes sont nécessaires pour créer une géométrie dans l’espace. Enfin, nous avons pu découvrir comment deux outils comme pygame et OpenGL peuvent interagir pour créer différentes interactions comme zoomer, se déplacer dans l’espace à l’aide des évènements pygame.

La deuxième partie concernant la création de maison nous a permis d’expérimenter avec des objets composées de plusieurs sous objets et de comprendre un peu mieux les dessous des logiciels de modélisation, en effet insérer une ouverture dans une géométrie n’est pas si simple qu’il n’y parait, c’est toute la géométrie qu’il faut redéfinir pour créer cette ouverture.