ORSET Emma
ERUAM Hervé
14/12/2021

Rapport de TP3 – Représentation visuelle d'objets

I. Introduction

On s'intéresse dans ce TP à la représentation d'objets 3D à l'écran dans une fenêtre graphique qui permet des opérations de zoom, rotation est translations. On a choisi la représentation de maisons à partir d'objets simples que l'on va construire progressivement comme les murs, les portes, les fenêtres… Pour ce faire, nous utiliserons les modules Pygame et PyOpenGL.

II. Travail préparatoire

1. Utilisation de Pygame

1. Ce code permet d'afficher une fenêtre, qui se ferme instantanément.

2. La fenêtre reste ouverte jusqu'à ce qu'on appuie sur une touche quelconque. Cela est possible grâce à « event » qui contient tous les événements en attente. Dans le code, on lui demande de se fermer lorsqu'on appuie sur une touche.

2. Utilisation de Pyopengl pour représenter des objets 3D

Dans cette partie, nous avons suivit les instructions du TP. Nous n'avons pas eu de problème.

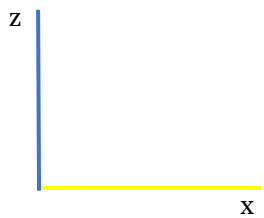
3. Découverte de l'environnement du travail du TP

a) Dans le fichier main, il y a plusieurs fonctions qui définissent les configurations associées aux différents exercices.
Dans le fichier configuration, nous avons une classe Configuration avec un constructeur qui définit les paramètres passés en paramètre de la fonction. Ceux qui ne sont pas passé ont des valeurs par défaut. Il y a aussi quelques méthodes qui initialisent les différentes librairies, et qui initialisent certains paramètres. Il y a aussi des méthodes pour dessiner/afficher les formes et gérer les événements clavier.
A l'écran, on obtient une fenêtre avec les axes x et y affichés en rouge et vert.

b)

- Cela a changé la couleur de l'axe x qui est maintenant jaune, et cela a également fait un zoom.
- Le chainage est possible car a chaque appel de méthode, ces dernières retourne l'objet instancié.
- Toutes les transformations dépendent du paramètre « screenPosition »

c) Nous avons tourné l'axe de 90° afin d'avoir l'axe z devant et l'axe y en profondeur (donc invisible pour le moment). Nous obtenons la figure :



III. Mise en place des interactions avec l'utilisateur avec Pygame

On s'intéresse maintenant à gérer les interactions possibles de notre application avec l'utilisateur à l'aide de PyGame. Tous ces éléments sont gérés dans la méthode display de la classe Configuration.

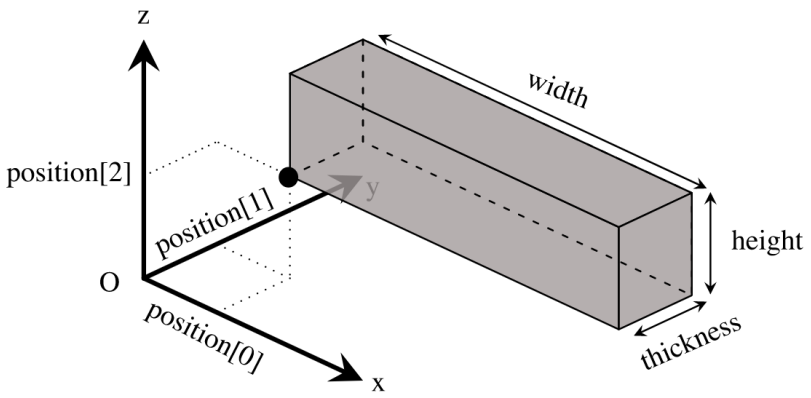
d) Nous avons trouvé dans la documentation les touches K_PAGEUP et K_PAGEDOWN. Ensuite, nous avons utilisé la méthode glScalef pour donner un zoom identique aux trois axes.

e) Nous avons voulu suivre les instructions mais le bouton de la molette semble ne pas fonctionner correctement.

f) Nous n'avons pas eu de mal pour la rotation mais la translation nous a demandé plus de réflexion car il fallait régler la sensibilité (nous l'avons divisé par 10), ce qui n'était pas demandé.

IV. Création d'une section

On s'intéresse maintenant à la création d'une section. Une section correspond à un parallélépipède à 8 sommets et 6 faces :



a) Nous avons défini chaque point selon width, height et thickness. L'ordre dans lequel nous les avons définies nous donne les numéros de nos point. On peut alors définir les faces. A l'affichage, nous obtenons bien un « mur » qui débute à la position 0,0,0.

b) Nous avons passé plus de temps sur cette question car nous avions un problème avec la couleur, mais finalement nous avons trouvé.

c) Nous avons suivi le TP et nous n'avons pas eu de problème.

V. Création des murs

On s'intéresse maintenant à la création d'un mur. Celui-ci correspond à un objet contenant une liste d'éléments à tracer (fenêtres, portes et sections) dans notre application. Dans un premier temps, nous allons considérer qu'un mur n'est constitué que d'une seule section.

a) On remarque que Wall prend plusieurs paramètres avec des valeurs par défaut, sous la forme d'un dictionnaire. On créé une section qui est associée au mur, Wall hérite de la classe Section. Section est une classe parente.

VI. Création d'une maison

Pour afficher une maison (draw()) nous nous plaçons dans une sous matrices ou nous effectuons la translation puis la rotation nécessaire avant d'afficher chaque objet contenu dans la maison a l'ecran.

Après avoir géré la création de murs (pour l'instant uniquement constitués d'une section parente), on va construire une maison en créant 4 instances de murs. Celle-ci sera gérée par la classe House du fichier House.py

VII. Création d'ouvertures

a) Pour créer les ouvertures, nous nous sommes basés sur les fonctions generate et draw de la class Section en retirant simplement deux des 6 faces.

b) Dans cette question, le plus compliqué a été de trouver les bonnes conditions à appliquer. Le dernier affichage de la question Q5b vaut false car on essaye de créer une ouverture de 1m de haut a une hauteur de 1.7m (soit le point le plus haut à 2.7m de haut) dans une section de 2.6 m de haut. Celle-ci dépasserait dont de 10cm ce qui n'est pas possible.

c) Dans la même idée que pour la question précédente, le plus dur était ici de choisir les bons paramètres à donner aux nouvelles sections.

d) La fonction enumerate permet d'associer un entier a chaque élément de la liste (sous la forme d'un tuple) qui correspond à la position de celui dans la liste. Dans le cas de la fonction findSection, avoir la position dans la liste d'objet de la section retournée permettras de la supprimer le moment venu dans la fonction add().

A l'exécution du 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] contient la section qui constitue le mur (la section parente) puisqu'aucun n'élément n'a été ajouté au mur, il n'est toujours composé que d'une section.

VIII. Conclusion

Ce TP a été plus difficile que les précédents, notamment pour comprendre la logique derrière le programme avec les draw de chaque objet « en cascade » de même que pour les rotations/translations de chaque objet. Nous n'avons pas pu terminer la partie « Pour finir … » à cause d'un problème avec la fenêtre qui ne s'affiche pas complétement et n'avons pas eu le temps de chercher une solution. Le TP était tout de même intéressant et nous avons pu appréhender les représentations 3D pour la première fois.

PS : Nous nous demandons s'il n'y a pas une erreur à la ligne 87 du test_opening.py. Cette partie est censé tester le cas où il n'y a pas de section à droite mais il place l'ouverture de test en x=0 et y=7-0.9. Est-ce que ce ne se serait pas l'inverse ?

opening1 = Opening({'position': [0, 7-0.9, 0.1], …
deviendrait
opening1 = Opening({'position': [7-0.9, 0, 0.1], …