Jourdan Samuel
De Carné Aymeric
Date du TP (09/12/2021)

Rapport de TP3 – Représentation visuelle d'objects

I. Introduction

L'objectif de ce TP est de représenter des objects visuelles 3D. Ici on réalisera la représentation d'une maison à partir d'objet simple.

II. Préparation à faire avant le TP

A) Utilisation de Pygame

1. Question (1)
Lorsque l'on exécute l'action pygame se ferme directement.

2. Question (2)
Par la suite on indique que la fenêtre reste ouverte tant qu'on ne l'a pas fermée en créant une variable continuer égale à True. Ainsi avec une boucle while, tant qu'on n'a pas changé la variable en False la fenêtre reste ouverte.

B) Utilisation de Pyopengl

3. Question (1)

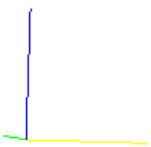
4. Question (2)
Le but de cette question est de tracer des axes, avec x,y et z respectivement en rouge, vert et bleu.
Le résultat est le suivant :

5. Question (3)

C) Découverte de l'environnement de travail

6. Question (1a)

7. Question (1b).
Cette modification va changer la couleur des axes affichés et va en même temps réaliser un zoom sur ces mêmes axes.



Axe en couleur

Le Chainage est possible car la Class le permet

8. Question (1c).
Pour que z soit représenté verticalement et que y devient la profondeur on doit faire une rotation selon l'axe x, dans la méthode initializeTransformationMatrix().

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

1. Question (1d)

Pour réaliser le zoom nous avons utilisé l fonction Scalef trouvé dans la documentation. Ensuite nous avons simplement écrit un programme où chaque axe est multiplié par 1.1 pour zommer (grâce à la touche Page Up) et multiplié par 1/1.1 pour dézoomer (grâce à la touche Page Down).

2. Question (1e).

Pour réaliser le zoom avec la molette, nous utilisons comme précédemment la fonction Scalef. Dans la fonction processMouseButtonDownEvent(), grâce à la fonction if/elif nous avons demandé que lorsque l'on tourne la molette vers l'avant (symbolisé par la valeur 4) nous zoomons et quand nous tournons dans le sens opposé (symbolisé par la valeur 5) nous dézoomons.

3. Question (1f).

Le but de cette question est de déplacer les objets par rotation en appuyant sur la touche gauche de la souris.

Si pygame.mouse.get_pressed()[0] est égale à 1 le bouton est enfoncé. Donc nous demandé grâce à la fonction if et glRoatate que l'objet se déplace par rotation si pygame.mouse.get_pressed()[0] == 1

On doit ensuite déplacer les objets par translation en appuyant sur la touche droite de la souris.

De la même manière que pour la rotation, sachant que pygame.mouse.get_pressed()[2] égale à 1 indique le bouton droit enfoncé, grâce à elif et glTranslate, l'objet est translaté si pygame.mouse.get_pressed()[2] == 1

Pour la translation on a divisé par 100 pour diminuer la sensibilité.

IV. Création d'une section

1. Question (2a)

Pour cette question il faut créer des sommets et les faces.

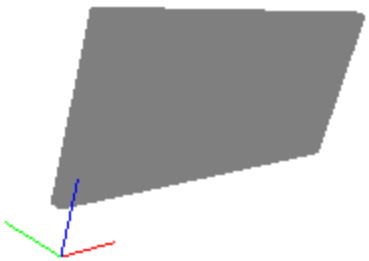
Pour les sommets il suffit de suivre le modèle de construction des sommets. Le seul point à faire attention et de mettre les sommets dans un ordre logique (même s'il il n'y a pas un seul ordre possible) pour faciliter la création des faces. En effet pour construire les faces on s'appuie sur les 4 sommets qui la constitue. On doit alors simplement appliqué la méthode de construction de face

2. Question (2b)

La fonction Q2b() dans le fichier main.py, Configuration().add(section).display() permet de renvoyer le repère ainsi que la section ajouté (la fonction add permettant l'addition donc ici l'ajout d'une deuxième fonction). , A noté que dans le cas où le code de la section ne marche pas ou code n'est pas encore crée, le repère sera quand même affiché.

Ayant pour but de tracer les faces de la section en gris, nous avons écrit une la méthode draw, en utilisant les fonctions glPushMatrix() et glPopMatrix() respectivement au début et à la fin afin de créer la matrice. A l'intérieur, en s'inspirant du code donné dans l'énoncé, nous avons à l'aide de deux boucles for … in range, appelé tous les sommets de la section en faisant face par face.

Nous savons également que la couleur grise est codée par (0,5 ; 0,5 ; 0,5). Ce qui nous a permis à l'aide de glColor3fv d'afficher la section en gris.



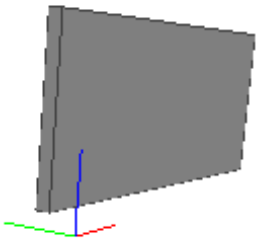
Création de la section

3. Question (2c)

Le but est maintenant de dessiner les arrêtes de la section.

Pour écrire la méthode draw.Edges, nous avons construit notre programme de la même manière que pour draw. Ensuite pour que la méthode draw.Edges() soit exécuté en premier lorsque le paramètre Edge est True nous avons mis dans la méthode draw une fonction if qui demande simplement que ce paramètre soit respecté.

Nous avons mis les arêtes d'une couleur plus foncé pour qu'elles ressortent mieux (avec les valeurs suivantes (0,25 ; 0,25 ; 0,25).



Ajout des arêtes sur la section

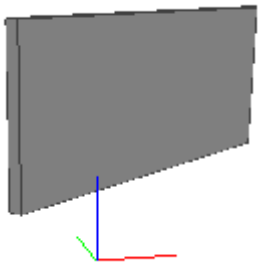
V. Création des murs

1. Question (3a)

En analysant le fichier Wall.py, on remarque que crée une section automatiquement, la section parente.

Nous avons ensuite, comme tous à l'heure crée une matrice avec glPushMatrix() et glPopMatrix(). Nous avons réalisé la rotation avec glRotate. Pour simplifier la programmation de la méthode nous avons rappelé Section avec from Section import Section. Il a ensuite suffi d'afficher les faces et les arêtes de la section grâce à une boucle for i in range.

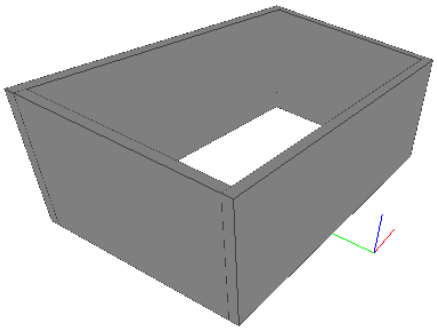
Section avec une rotation



VI. Création d'une maison

On veut maintenant construire une maison constituée de 4 murs.
Pour cela on crée une méthode draw qui affiche les murs avec for i in range qui ajoute loqiquement le nombre de mur de la maison.
Dans Main.py on indique les données de nos 4 murs.

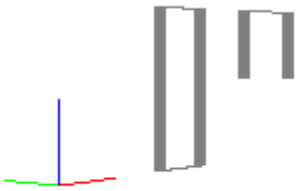
Maison réalisé grâce au 4 sections de mur



VII. Création d'ouverture

1. Question (5a)

On s'intéresse maintenant aux ouvertures (porte et fenêtres) dans les murs.
Dans la Class Opening, on constitue notre méthode draw.
De la même manière que précédemment on crée une matrice avec glPushMatrix() et glPopMatrix() en s'aidant ensuite de for … in range et glTranslate



Ouvertures (portes et fenêtre)

2. Question (5b)

L'objectif est maintenant de modifier la Class Section en ajoutant la méthode canCreateOpening(self, x), qui permet de déterminer si la taille de l'ouverture est compatible avec la taille de la section.

Pour que l'ouverture puisse appartenir à la section, l'épaisseur de la section et de l'ouverture doivent être égale alors que la hauteur et la largeur de l'ouverture doivent être inférieur à celle de la section.

Nous avons plus qu'à écrire un programme avec if et elif.

Dans Main.py le dernier affichage vaut False car pour opening3 la position en hauteur plus la hauteur de la position dépasse les 2,6 de hauteur de la section (1,7+1=2,7>2,6)

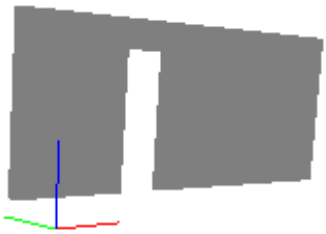
```
True
True
False
```

Résultat Vérifier le bon fonctionnement de votre méthode (respectivement pour opening1, opening2 et opening3)

3. Question (5c)

On décompose maintenant la section en plusieurs nouvelles sections avec la création de la méthode CreateNewSection(self, x).

Une ouverture créera 4 nouvelles sections. Nous avons donc fait une boucle for i in range(4) pour paramétrer ces 4 sections. Et à l'aide de if et elif pour chaque section nous avons déterminé la hauteur, la largeur et la position initiale de la nouvelle section. Nous avons également noté l'épaisseur des sections qui sont équivalentes à l'épaisseur de la section initiale.



Création de mur ou de fenêtre dans la section

4. Question (5d)

La fonction enumerate permet de déterminer la position et le caractère d'un object.

Si l'on exécute on remarque que Section [0] est égal à 0 car c'est le premier élément et Section [1] renvoi l'adresse de la section.

Pour la suite de cette question nous avons eu un problème pendant longtemps.

Premièrement la fenêtre ne s'affichait pas quand on créait l'ouverture de la porte avant la fenêtre alors que dans l'autre sens cela fonctionnait. Nous avons compris que cela venait du fait de la fonction canCreateOpening() vérifie la possibilité de créer une ouverture dans une section en prenant comme origine ne point en bas à gauche de cette section. Or lorsque la porte est créé, si l'on prend le point en bas à gauche de la section à droite de la porte comme référence, la fenêtre ne rentre pas dans cette section alors que si on prend le point en bas à gauche du mur en référence la fenêtre rentre.

Afin de résoudre ce problème, nous avons soustrait les coordonnées du point de référence en x et z de la section afin de simuler un départ du point de référence du mur.

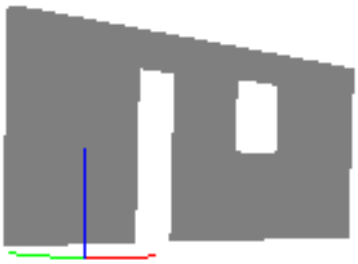
Après cela la fenêtre s'affichait mais pas correctement car le point de référence de la section et du mur était différent ainsi le décalage de les coordonnées du point de référence de la fenêtre n'était pas dans le bon repère.

Etant donné que le point de référence du mur n'était pas en (0,0,0) par rapport au repère de base, nous ne pouvions pas enlever les valeurs des points coordonnées au décalage de la fenêtre.

Après avoir tester plusieurs possibilités infructueuses, nous avons modifié le prototype de la méthode createNewSections afin d'y intégrer les cordonnées de référence du mur.

Grâce à cela nous avons réussi à calculer le bon décalage et donc positionner la fenêtre correctement.

La méthode createNewSections étant appélé dans d'autre méthode, l'utilisateur n'a pas besoin d'entrer le nouveau paramètre dans le Main. Cependant la question 5c1 et 5c2 appelle cette fonction dans le Main, nous avons donc dû rajouter manuellement ce paramètre dans le Main pour ces deux questions.



Insertion de la fenêtre et de la porte sur la section du mur

VIII. Pour finir

1. **Question (6)**
   Nous n'avons pas eu le temps de faire cette dernière question.

2. **Question (7)**