Rapport de TP  
FI3 – INFO501 Année 2021-2022  
Polytech Annecy-Chambéry – Université Savoie Mont-Blanc

LAUBERT Théoxane TDB1 MM3  
09/12/2021

**Rapport de TP3 – Représentation visuelle d’objets**

1. **Préparation**

**Utilisation de Pygame**

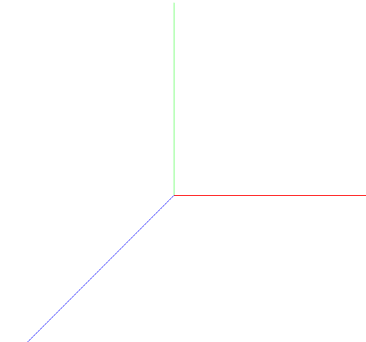
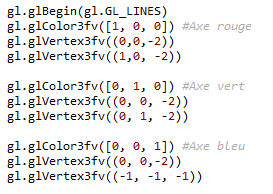
1. Afin de comprendre Pygame, on réalise un petit essai de quelques lignes données. La première ligne permet l’import du module pygame dans python, il n’est pas présent sinon et donc inutilisable. Ensuite on réalise une initialisation de pygame, sans cela on ne peut toujours pas utiliser le module.   
   La troisième ligne initialise une variable écran en lui donnant la valeur pygame.display.set\_mode((300,200)). Cette fonction crée une fenêtre pour affichage, ici il s’agit de notre écran de jeu. En spécifiant (300,200) on initialise le format de la fenêtre en même temps.Enfin la quatrième ligne quitte la fonction, en effet quand on initialise le module il faut aussi le quitter pour « nettoyer » ?

Lorsqu’on exécute le code, une petite fenêtre apparait, correspondant au format (300,200).

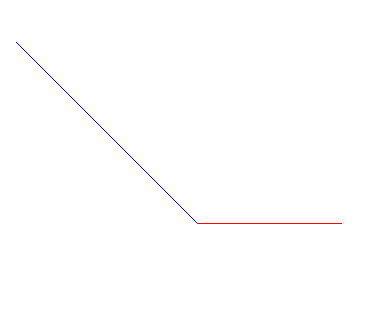
1. La fenêtre se ferme car nous ne lui donnons pas d’autres instructions à réaliser. Pour cela nous testons le deuxième code proposé dans l’énoncé. On a donc ajouté 5 lignes au code précédent. La première ligne crée une variable booléenne continuer initialisée à True. Ensuite une boucle while permet de laisser l’affichage tant qu’il n’y a aucun évènement. La 3ème ligne est une boucle for, c’est-à-dire que si il y a un évènement, on vérifie le type de l’évènement. Ici on vérifie que l’évènement corresponde à un appui sur une touche du clavier. Si on appuie sur une touche, la variable continuer prend la valeur False et la boucle while se termine. La fenêtre d’affichage se ferme alors.

**Utilisation de Pyopengl pour représenter des objets 3D**

1. En ajoutant la ligne glu.gluPerspective (45,display[0]/display[1],0.1,50), la fenêtre s’affiche et reste tant que je n’appuie pas sur la croix de la fenêtre.
2. En réalisant le code suivant et en exéutant, nous obtenons une fenêtre d’affiahcge avec troix axes, x, y et z en rouge, vert et bleu



1. En ajoutant les lignes pour la rotation et la translation, on obtient bien l’axe bleu et rouge grâce à une translation et une rotation de -90°.



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

Q1) a) Analyse des fichiers main.py et Configuratio.py :

**main.py** : Ce fichier permet l’import des modules nécessaires et contient les fonctions de réalisation du sujet, chacune portant le nom Q.. correspondant aux questions de ce TP.

**Configuration.py** : 1 : \_\_init\_\_(self, parameters = {}) : Création de la classe Configuration (14 à 50). Initialisation de la classe et paramétrisation des axes. Initialisation de la fenêtre d’affichage. Il y a également l’initialisation dans les modules Pygame et OpenGl puis on génère des coordonnées.

2 : initializePyGame(self): Initialisation de Pygame comme vu lors de la préparation du TP.

3 : initializeOpenGL(self): Initialisation de OpenGl comme vu lors de la préparation du TP.

4 : initializeTransformationMatrix(self): Initialisation des matrices transformation grâce à OpenGl.

5 : getParameter(self, parameterKey): Methode get des paramètres : permet d’afficher les paramètres sans faire appel au constructeur.

6 : setParameter(self, parameterKey, parameterValue): Fonction de paramétrage et qui permet d’effectuer une transformation suivant les coordonnées de l’écran.

7 : generateCoordinates(self): Fonction qui permet de générer les coordonnées des vertices et edges.

8 : add(self, x): Surcharge de l’addition

9 : draw(self): Affichage des axes x y et z

10 : processKeyDownEvent(self): Fonction permettant de reconnaître les évènements clavier. Un appui sur « z » permet la rotation de la figure selon y (sens trigonométrique). Un appui sur « Z » permet cette même rotation en sens anti-trigonométrique. L’appui sur « a » fait disparaître ou apparaître la figure.

11 : processMouseButtonDownEvent(self): Fonction permettant de reconnaître les évènement de la souris.

12 : processMouseMotionEvent(self): Fonction gérant les évènements liés à la souris.

13 : display(self): Fonction permettant l’affichage et la coordination de différents évènement et fonctions.

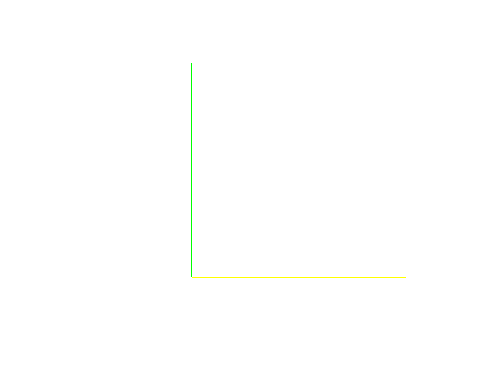
Q1)b) Lorsqu’on utilise la ligne de commande :

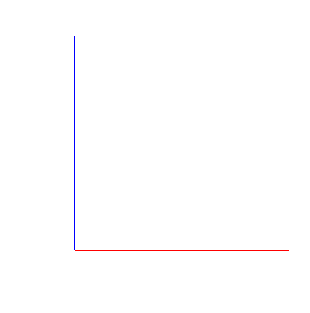
Configuration**({**'screenPosition'**:** **-5,** 'xAxisColor'**:** **[1,** **1,** **0]}).**display**()**

On saute les setParameter et l’affichage se fait directement.

On peut chaîner l’appel des méthodes setParameter() et display() car ils appartiennent à la classe configuration, on peut les mettre à la suite pour rendre l’écriture plus légère.

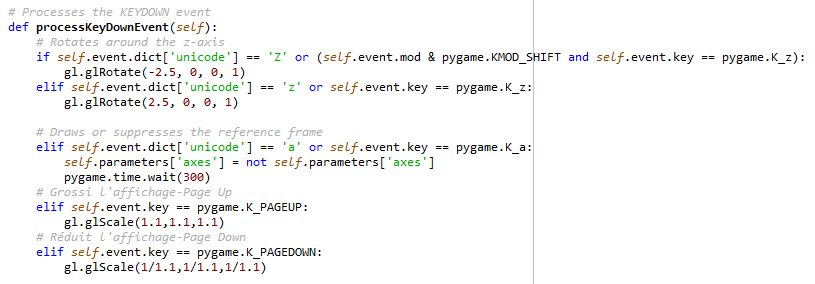
Le traitement particulier pour « screenPosition » est nécessaire pour initialiser la position de l’objet. Pour cela on appelle initializeTransformationMatrix() pour une mise à jour.

Voici ce que l’on obtient lorsqu’on modifie les paramètres des axes.

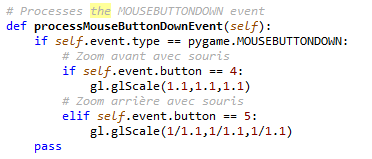
Q1)c) On ajoute gl.glRotate(-90,1,0,0)dans initializTransformationMatrix() pour que l’axe z soit vertical et x horizontal : (x rouge, y vert, z bleu). On obtient bien ceci lors d l’affichage :

1. **Mise en place des interactions avec l’utilisateur avec Pygame**

1)d) Voici le code permettant de gérer l’appui sur les touches page up et page down :

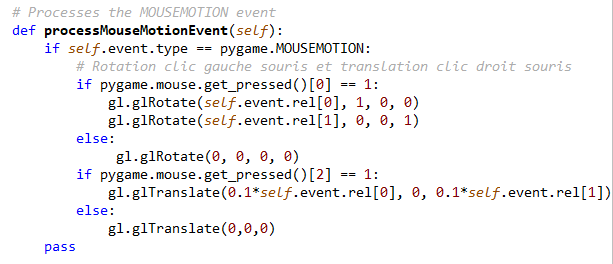


1)e) Afin de gérer l’évènement de la molette qui descend, voici ce que l’on écrit dans processMouseButtonDownEvent :



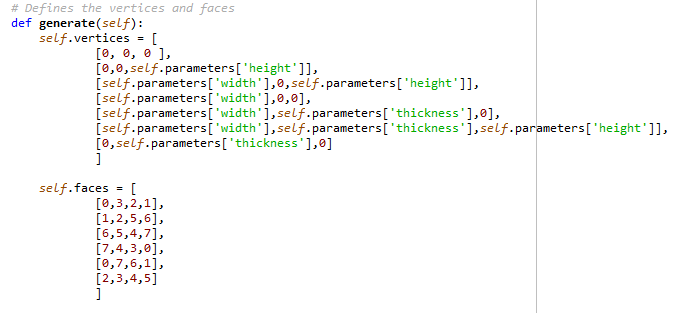
En réalisant un test on peut vérifier que cela fonctionne correctement.

1)f) Afin de pouvoir déplacer les objets affichés en bougeant la souris, voici le code réalisé :

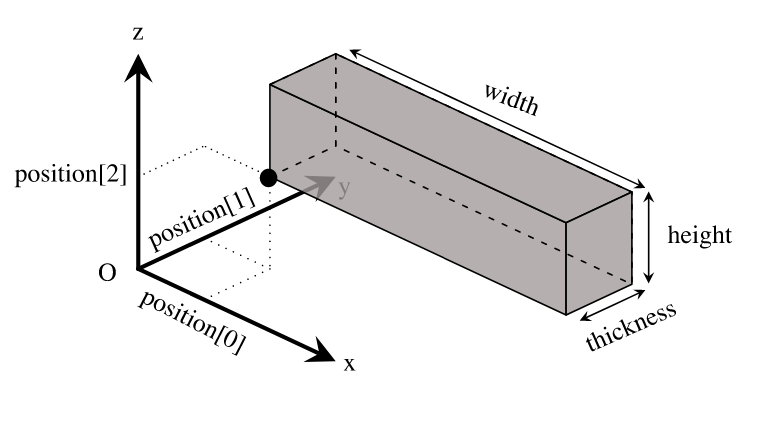
On crée la rotation autour de x avec une rotation autour de z dans le premier if.

Et une translation selon x t z lorsque le bouton droit est enfoncé.

1. **Création d’une section**

2)a) Pour créer les sommets et les faces d’une section orientée selon l’axe x et dont le coin bas gauche de la face externe est en (0,0,0) : voici le code écrit dans la méthode generate() : 

6



7

5

4

3

2

1

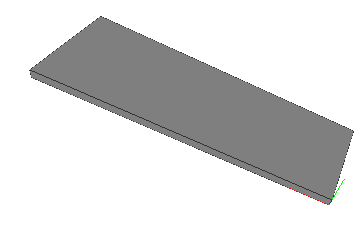
0

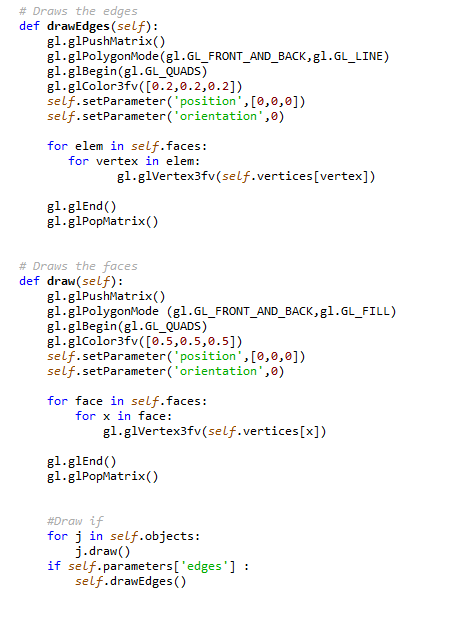
FIN Séance 1 TP3

2)b) Dans le fichier main, fonction Q2b(), l’instruction Configuration().add(section).display() signifie que l’on appelle la méthode add de configuration pour lui donner une section en argument. La fonction display() ajoutée à la fin de l’instruction permet l’affichage de cette section



Voici l’affichage obtenu en exécutant le main : il est possible de le déplacer à l’aide de la souris comme précédemment.

2)c) Voici la fonction réalisée drawEdges qui permet d’avoir les arrêtes noires et donc de mieux les visualiser, ainsi que draw() modifier comme demander ainsi que l’affichage obtenu :



1. **Création des murs**

3)a) Analyse de Wall.py :

Wall est une classe :   
**def \_\_init\_\_(self, parameters = {}) :**

- initialisation de la classe. On initialise d’abord la position, la hauteur, l’orientation, l’épaisseur et la couleur, comme cela a déjà été fait dans les précédentes classes.  
- self.objects = [] crée une liste d’objet vide.  
- self.parentSection crée la section principale   
- self.objects.append(self.parentSection) ajoute la section principale à la liste d’objet créée juste avant

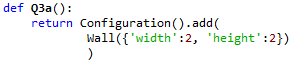
**def getParameter(self, parameterKey):** surcharge des paramètres du mur

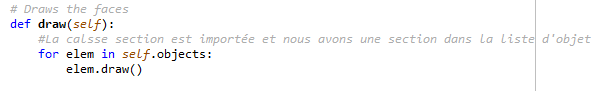
**def setParameter(self, parameterKey, parameterValue):** surcharge pour modifier valeur d’un parameter.

**def findSection(self, x):** trouve un emplacement pour ajouter un objet.

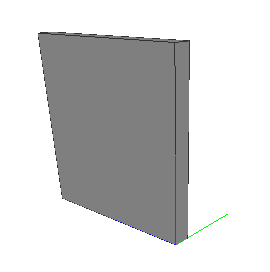
**def add(self, x):**  fonction d’addition des objets

**def draw(self):** function de dessin des objets.

Instruction dans main.py :

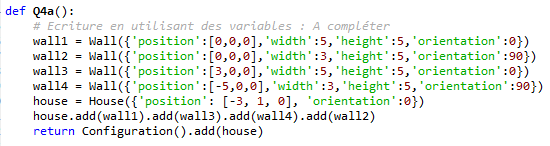
Instruction dans Wall.py :   
J

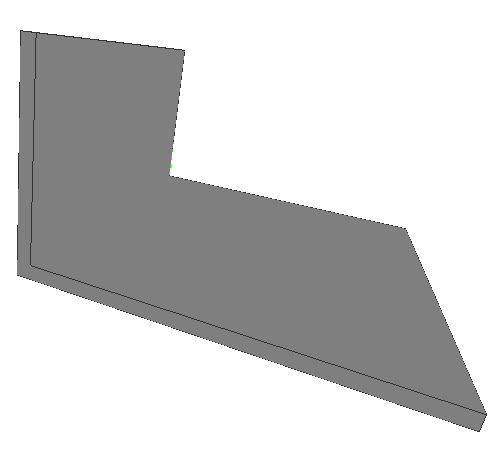
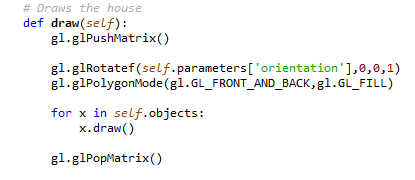
Afin de vérifier que la taille de mon mur soit conforma à ce que je demande, j’ai réalisé un carré 2\*2 et j’ai obtenu le résultat suivant :



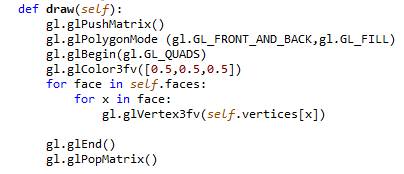
1. **Création d’une maison**

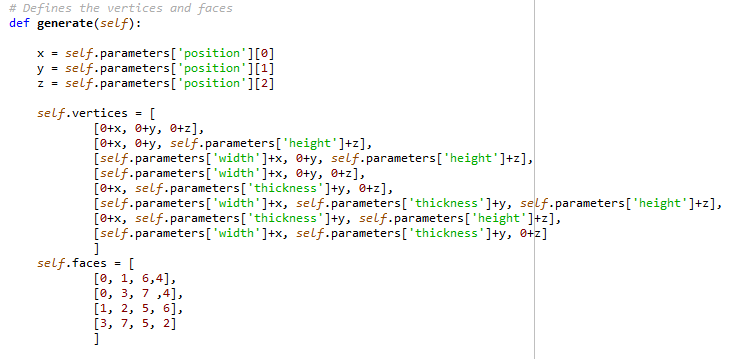
En suivant les étapes et en écrivant le code suivant, j’obtiens 2 murs perpendiculaires. En réalités les 4 sont créés mais se superposent. Je n’arrive pas à créer le changement d’origine pour régler le problème. Pour tenter d’avancer je vais donc rester dans cette situation.



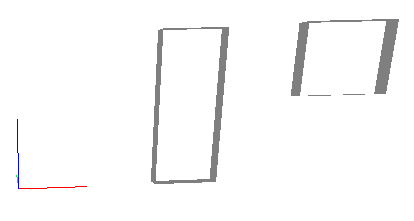


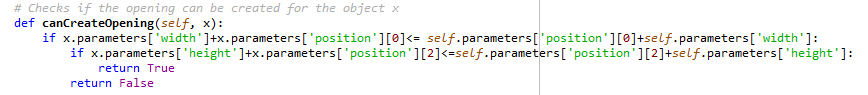
**VII Création d’une ouverture**

5)a)

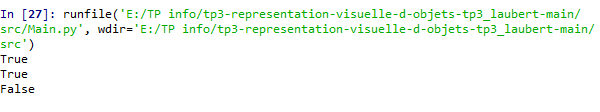


On obtient donc les ouvertures suivantes :



5)b) Il faut vérifier que les ouvertures ne dépassent pas le mur :

Voici le résultat obtenu : la troisième ouverture n’est pas possible car elle dépasse du mur



FIN séance 2 TP3