BECHARD Thomas

VEDIE Max

Date du TP (09/12/2021)

# Rapport de TP3 – Représentation visuelle d’objets

## Introduction

L’objectif de ce TP est de créer un programme nous permettant d’afficher un objet en 3D dans une fenêtre et de pouvoir le manipuler afin de changer de point de vue avec des zooms, des rotations ou des translations. Ici, nous essaierons de représenter une maison. Pour se faire, nous aurons besoin de manipuler deux modules différents : Pygame pour les interactions avec la fenêtre graphique et PyOpenGL pour l’affichage des objets graphiques à l’aide d’une bibliothèque particulière.

## Préparation à faire avant le TP

**Utilisation de Pygame**

### Quand nous utilisons le programme donné nous obtenons cette phrase :

### “Hello from the Pygame community. https://www.pygame.org/contribute.html” ²

La ligne 1 « import pygame » permet d’importer le module pygame dans notre code.

La ligne 2 « pygame.init() » est une simple initialisation du programme.

La ligne 3 « ecran = pygame.display.set\_mode((300, 200)) » nous ouvre un écran de 300 x 200 pixels.

La ligne 4 « pygame.quit() » est une fermeture du programme après utilisation.

### Lorsque nous appuyons sur n’importe quelle touche du clavier, cela ferme la fenêtre. Si nous

### regardons le code, la ligne « if event.type ==pygame.KEYDOWN : » est celle qui est à l’origine de cette fermeture de fenêtre.

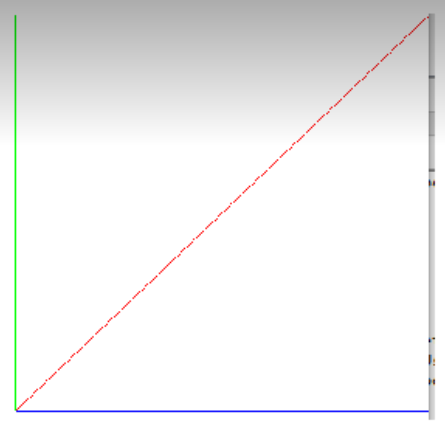
Une image contenant texte

Description générée automatiquement**Utilisation de PyOpenGL pour représenter des objets 3D**

1. On utilise le gluperspective :

« glu.gluperspective (45, 1, 0.1, 50) » avec les valeurs données dans l’énoncé, mais on a mis 1 à l’attribut « aspect » car ce n’est pas important pour l’instant. Quand on lance le programme, il n’y a pas d’erreur car nous avons l’affichage d’une fenêtre blanche.

1. Dans cette partie, le code qui nous ait donné permet de tracer 3 axes de différentes couleurs. Chaque couleur demandée est une couleur primaire c’est pour cela que les valeurs dans les parenthèses de la fonction gl.glColor() sont toutes à 255 ou 0. De plus, on a répété l’opération trois fois afin de faire les trois axes.

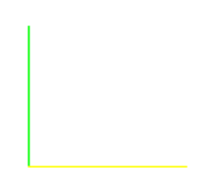


1. Dans la fonction gl.glRotatef(), nous avons mis la valeur -120 au lieu de -90 afin de tourner les axes. De fait, la figure ci-dessus, se retrouve à l’envers et a tourné autour de l’axe bleu.

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



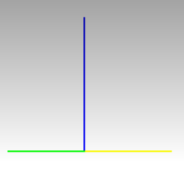
1. Dans notre code, la fonction processKeyDownEvent permet de créer une action avec l’appuie sur unetouche du clavier. Ici, le code affiche un repère avec l’axe x en rouge et l’axe y en vert. Ensuite, plusieurs touches sont utilisées. En effet, le « z » nous permet de faire tourner le repère dans le sens anti-horaire alors que le « Z » fait tourner dans le sens horaire, les rotations sont autour de l’axe z. Enfin, la touche « a » permet de masquer ou afficher le repère.



1. Le code suivant : Configuration({'screenPosition': -5, 'xAxisColor': [1, 1, 0]}).display() nous permet de changer la couleur de l’axe x rouge en jaune et de changer la taille des axes qui a augmenté. Nous sommes passés de -10 à -5 ce qui nous permet de doubler la taille.

La fonction setparameter() permet, elle aussi, de changer la couleur des axes.

L’appel des méthode setparameter() et display() est possible grâce à la création de la fonction display() contenu dans la classe configuration.



1. Chaque axe est mis à -90, pour que les axes changent de position. Le fait de tous les faire bouger de -90 permet de rendre l’axe z à la place de l’axe y, l’axe y à la place de l’axe x et l’axe x à la place de z.

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

1. Nous souhaitons grossir ou rétrécir l’affichage des axes. Nous avons donc mis les lignes de code suivantes dans la classe configuration et dans la méthode « processKeyDownEvent() »

* Grossissement de l'affichage-Page Up

elif self.event.dict['unicode'] == 'U':

gl.glScale(1.1,1.1,1.1)

Ici nous avons mis les valeurs 1.1 pour grossir chaque axe de 0.1 quand nous appuyons sur la touche « U »

* Réduction de l'affichage-Page Down

elif self.event.dict['unicode'] == 'D':

gl.glScale(1/1.1,1/1.1,1/1.1)

Ici nous avons diminué la taille des axes d’environ 0.1 quand nous appuyons sur la touche « D »

1. Le code suivant permet de faire en sorte que la molette de la souris d’ordinateur contrôle le zoom de la fenêtre. C’est la méthode processMouseButtonDownEvent() qui permet cette opération.

Une image contenant texte

Description générée automatiquement

1. Le code suivant utilise la même méthode que la question précédente et permet cette fois à la souris de contrôler le déplacement grâce au clic droit et la rotation du repère grâce au clic gauche.

Une image contenant texte

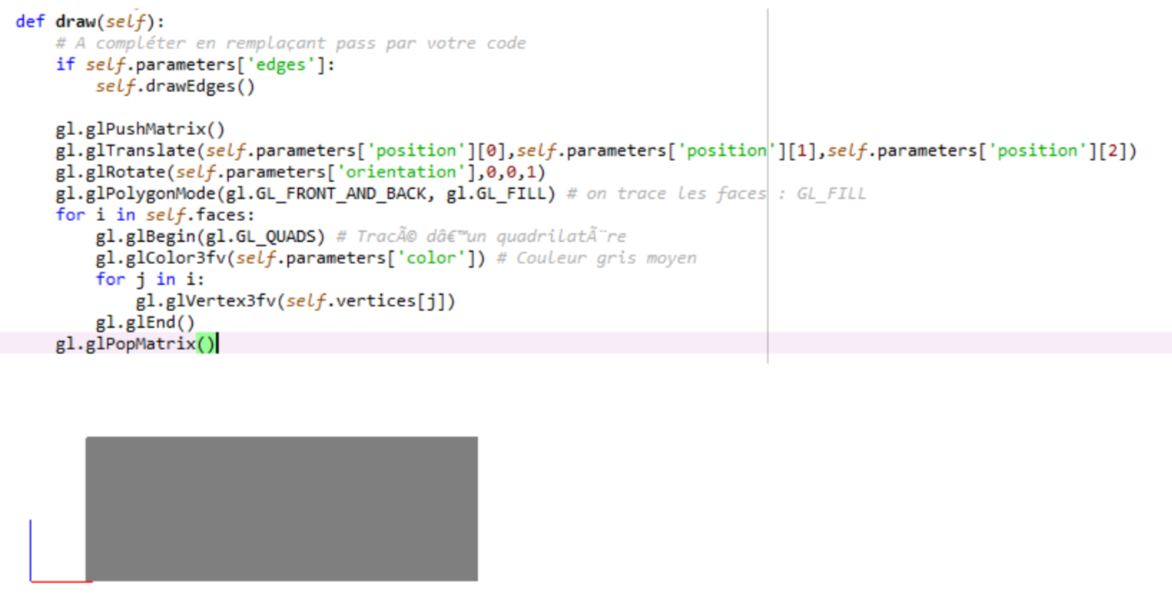
Description générée automatiquement

## Création d’une section

1. Grâce aux lignes suivantes, nous avons la possibilité de créer des points qui deviendront des sommets. Nous prenons toutes les possibilités différentes pour créer les points avec la fonction « self.paramaters[‘…’] ». Les « … » étants remplacés par width, thickness et height, respectivement suivant les axes x,y ou z qui correspondent à la largeur, l’épaisseur et la hauteur. Ces sommets sont au nombre de huit et sont nommés de 0 à 7. Il est alors possible de relier 4 de ces sommets entre eux afin d’obtenir une face. C’est la deuxième partie du code.



1. Nous utilisons le chainage pour écrire la ligne « Configuration().add(section).display() » c’est-à-dire que nous lions avec des « . » la classe configuration avec les deux fonctions add() et display(). Grâce à ça, nous pouvons ajouter une section en tant qu’objet afin de pouvoir ensuite la tracer et la visualiser.



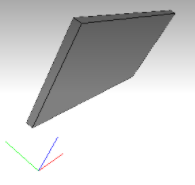
1. Pour tracer les arrêtes, il faut reprendre à peu près le code pour afficher les sections. Le seul truc qui change est que nous remplaçons le GL\_FILL qui trace une face par GL\_LINE qui trace des lignes. Nous avons également changé les paramètres de la couleur pour obtenir du noir qui se démarquera de la couleur grise des faces.

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquementCe petit bout de code permet de faire passer les arrêtes en premier afin de les rendre voyantes. La couleur noire était la plus adaptée pour voir au mieux les arrêtes. Vous pouvez visualiser le rendu avec les arrêtes noires juste ci-dessous :



## Création des murs

Dans la classe wall.py, nous pouvons voir l’initialisation des paramètres par défaut qui seront pris par le programme quand l’utilisateur a oublié de les annoncer. Nous nous sommes inspirés de la méthode draw() de la classe configuration pour cela, et dans la classe main, on a utilisé un code similaire à celui de l’affichage de la section dans la partie précédente. Nous avons changé l’objet section par l’objet wall et nous avons ajouté le paramètre orientation afin de pivoter la section de 90°.

Une image contenant texte

Description générée automatiquement

* Classe Wall.py

Une image contenant texte

Description générée automatiquement

* Une image contenant texte, carte de visite

  Description générée automatiquementClasse main

Nous obtenons bien une section correspondant à un mur orienté de manière verticale.

## Création d’une maison

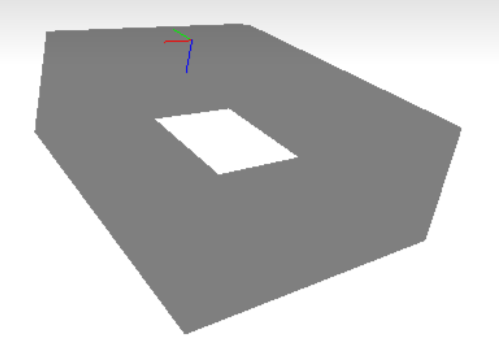
Dans la classe house, nous n’avons fait que reprendre la fonction draw() de la classe wall vue précédemment. Puis dans la classe main, en revanche, nous avons mis quatre murs que nous avons déplacé afin de former les quatre murs collés d’une maison. Il suffit de modifier la position et l’orientation pour chaque mur.

Une image contenant texte

Description générée automatiquement

* Classe main
* Classe house

Voilà le résultat final :



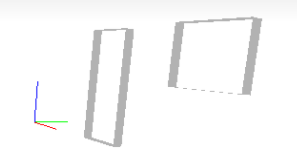
## Création d’ouverture

1. Nous voulons, ici, créer les contours des ouvertures pour les fenêtres et la porte. Pour se faire, il faut reprendre les fonctions draw() et generate() utilisées dans la classe section afin de créer des sections de différentes tailles pour faire les rebords d’une porte ou d’une fenêtre. Les seules choses modifiées sont donc la longueur et la largeur des sections ainsi que l’épaisseur.

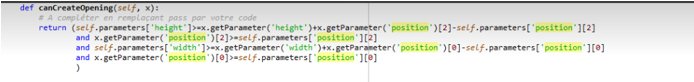
Une image contenant texte

Description générée automatiquement

Ce qui nous donne le résultat suivant :

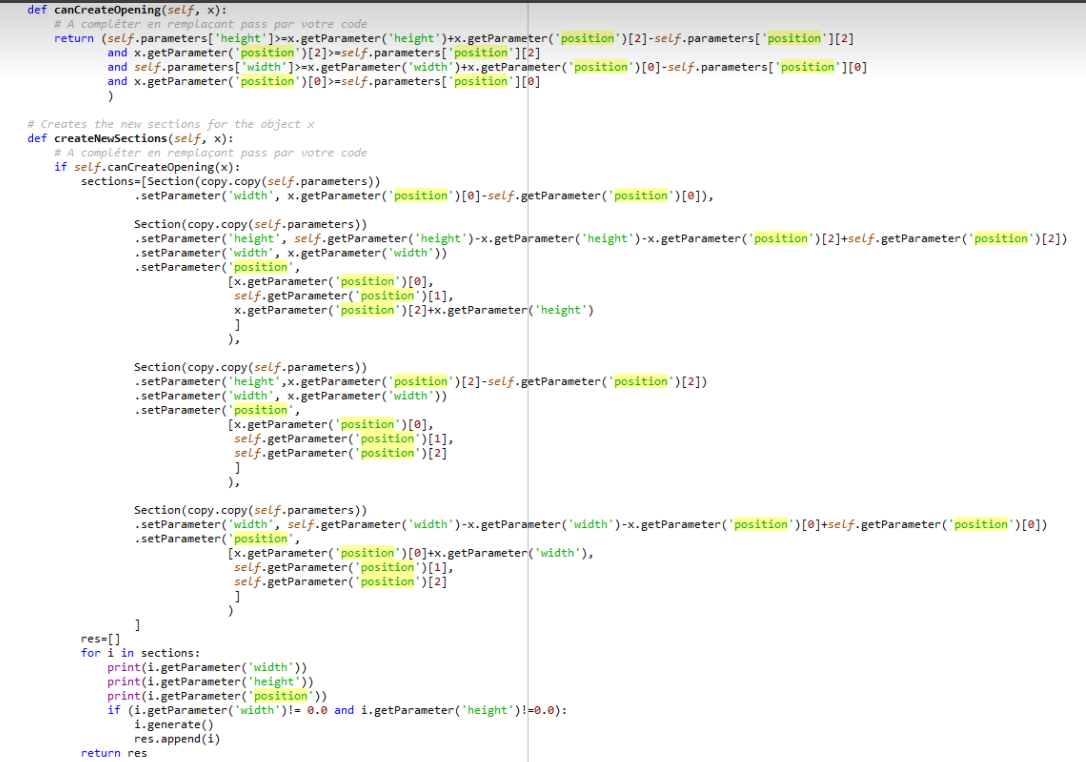


1. Dans cette partie, c’est la classe Section que nous avons agrémenter d’une fonction canCreateOpening() que nous avons établi comme cela :



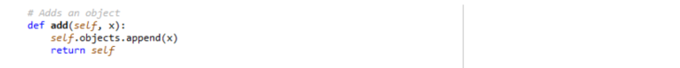
Cette fonction va pouvoir retourner *True* si on peut rajouter une ouverture et *False* sinon en comparant les informations données par les paramètres de position, de largeur, de hauteur de l’ouverture avec celles fournies par les paramètres de la section.

1. Nous avons terminé ce TP par l’écriture de la fonction createNewSections() de la classe Section qui nous permet d’avoir la liste des sections engendrées par la création de l’ouverture. Nous n’avons cependant pas pu visualiser les représentations des figures par faute de temps. Le code ainsi obtenu est donné la page suivante :



1. Dans la méthode findSection() le rôle de enumerate() de**parcourir le self.objects tout en gardant une trace de l’index de l’élément actuel dans une variable de compteur.**

La méthode add() a été rédigé simplement comme cela :



Cette fonction ajoute l’insertion que l’on veut faire grâce au append(x). Nous pas pu approcher les derniers points du TP de la partie « Pour finir… » pareillement par faute de temps.

## Conclusion

Au travers de cette étude nous avons pu parfaire notre apprentissage du langage Python en utilisant la modélisation 3D du logiciel afin de réaliser des murs d’une maison mais également des ouvertures pour placer des portes ou des fenêtres. Nous pourrions assimiler notre travail à des logiciels de modélisation 3D connus comme SOLIDWORKS ou Sweet Home 3D où il est question de modélisation et de manipulation dans l’espace en 3 dimensions.