SANGOUARD Marine

MONNIER Marine

14/12/2021

# Rapport de TP3 – Représentation visuelle d’objets

## Introduction

Lors de ce TP, nous allons réaliser des représentations d’objets 3D à l’écran dans une fenêtre graphique qui permet des opérations de zoom, de rotations et de translations. Par exemple, nous allons représenter des maisons à partir d’objets simples que l’on va construire progressivement comme les murs, les portes et les fenêtres. Pour cela, nous allons nous utiliser 2 modules : Pygame et PyOpenGL

## Préparation à faire avant le TP

### Utilisation de Pygame

### À l’aide de la documentation de Pygame, expliquer ces 4 lignes.

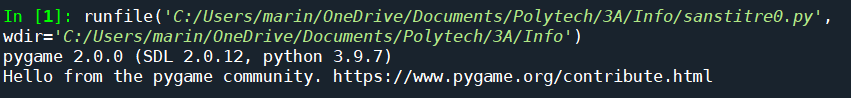
import pygame => importe le package pygame avec tous ses modules disponibles

pygame.init() => permet d’initialiser chacun de ces modules

pygame.display.set\_mode() => créé une fenêtre graphique, ici “ecran” est en fait un objet “Surface” qui est visible par l’utilisateur, avec comme propriétés : largeur 300 et hauteur 200.

pygame.quit() => quitte/supprime/ferme la fenêtre graphique

Rien ne se passe, il y a seulement le message ci-dessous qui apparaît :



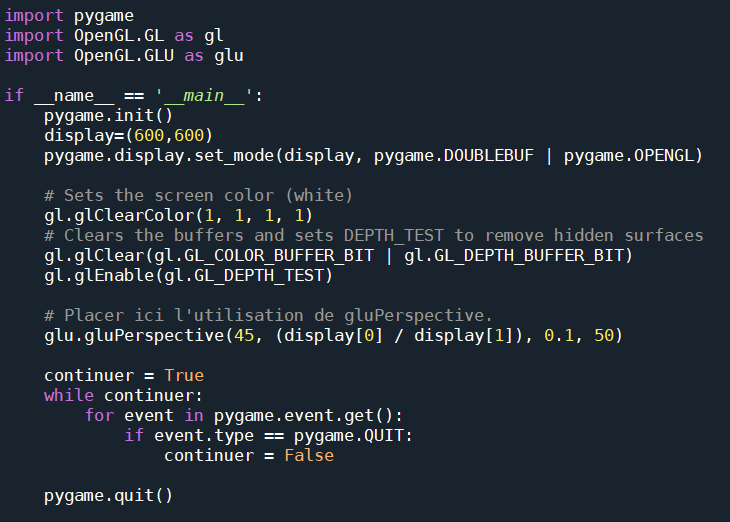
### Appuyer sur un bouton quelconque du clavier et observez ce qu’il se passe. Expliquer à l’aide du code et de la documentation de Pygame, comment on est parvenu à ce résultat.

Une fenêtre s’est ouverte. Si l’on appuie sur une touche du clavier, cette dernière se referme. En effet, la boucle “while” avec la variable “continuer = True” permet d’avoir la fenêtre qui reste ouverte. Elle se fermera que lorsque l’on ressortira de cette boucle, c’est à dire si le type d’un événement “pygame.KEYDOWN” a lieu (if event.type (in pygame.event.get()) == pygame.KEYDOWN). D’après la documentation, les événements pygame.KEYDOWN interviennent lorsque les boutons du clavier sont enfoncés et relâchés.

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

### Initialisez la matrice de perspective puis exécuter le fichier et voir s’il n’y a pas d’erreurs.

Le programme ne marchait pas, il y avait un problème avec le “exit()” donc nous avons repris la boucle “while” du programme précédent pour fermer la fenêtre mais cette fois-ci en utilisant “pygame.QUIT” au lieu de “pygame.KEYDOWN”. Nous avons ainsi le programme suivant :



### Utiliser cet exemple pour tracer un segment pour chaque axe x, y et z en couleurs rouge, vert et bleu respectivement.

### Découverte de l’environnement

### Analyser le fichiers main.py et Configuration.py de manière précise et détaillée, les points principaux du code et du résultat obtenu à l’écran lorsque l’on appuie sur les touches « a », « z » et « Z ».

Après avoir ajouté return Configuration(), on peut utiliser certaines touches.

La touche ‘a’ permet d’afficher ou de ne plus afficher le repère.

La touche ‘z’ permet de faire tourner le repère autour de l’axe z dans le sens antihoraire, ‘Z’ dans le sens horaire.

En effet, la def « Q1a » utilise la classe Configuration programmée dans le fichier configuration et la méthode utilisée lorsque l’on appuie sur ces touches est celle appelée “processKeyDownEvent”

Cette méthode appelle la fonction « rotate » avec un déplacement vers les x positifs (donc sens antihoraire) lorsqu’elle détecte l’appui de la touche ‘z’, ou avec un déplacement vers les x négatifs (donc sens horaire) lorsqu’elle détecte l’appui des touches Shift et ‘z’. On utilise l’attribut dict pour gérer indifféremment l'utilisation d’un clavier Qwerty ou Azerty.

L’appuie sur la touche ‘a’ désactive les paramètres des axes (pour désactiver l’affichage) après un léger temps de latence (utilisation de wait(300))

### Analyser la modification des paramètres par défaut, puis Expliquer pourquoi le chaînage de l’appel des méthodes setParameter() et display() est possible. Expliquer pourquoi un traitement particulier doit être effectué dans le « setter » pour le paramètre screenPosition.

La ligne « Configuration({'screenPosition': -5, 'xAxisColor': [1, 1, 0]}).display() » permet de configurer les paramètres par défauts des axes. Ici on attribue la couleur jaune à l’axe x; y et z gardent leur valeur RVB initiales.

L’ajout de « setParameter('xAxisColor', [1, 1, 0]).setParameter('yAxisColor', [0,1,1]) » permet de modifier ces paramètres. Ici on met x en jaune (ce qui ne change rien) et y en cyan.

On peut chaîner la méthode display après la méthode setParameter car la méthode setParameter renvoie son paramètre self, et c’est ce qui est nécessaire d’avoir en entrée pour utiliser la méthode display.

Si le paramètre à modifier est un code couleur d’axe, il suffit dans le setter de mettre à jour ce paramètre en changeant sa valeur dans son dictionnaire.

Or si le paramètre à modifier est ScreenPosition, il est nécessaire d’appeler la fonction initializeTransformationMatrix(). Cette méthode permet de placer l’écran à la position z demandée (donc le placer plus ou moins proche).

### Ajouter une seule instruction à la méthode initializeTransformationMatrix() pour que l’axe z soit représenté verticalement sur l’écran et que l’axe x soit représenté horizontalement.

Nous avons rajouter « gl.glRotatef(-90, 1, 0, 0) » afin de faire une rotation de 90° selon l’axe x (basculement vers l’extérieur).

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

Test question 1a



etc

## Conclusion

Expliquer ici l’état d’avancement du TP actuel, les difficultés principales que vous avez rencontrées ainsi que ce que vous avez appris.