Brice Le Bouille
Paul Pasquier

TP 3 : Représentation visuelle d’objet

Introduction

1. Préparation avant le Tp

A. Utilisation de Pygame

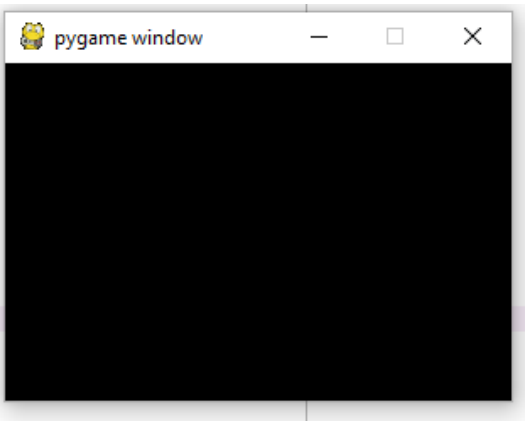
Question 1 :

```
 8 import pygame
 9 pygame.init()
10 ecran = pygame.display.set_mode((300, 200)
11 pygame.quit()
```

Ces quelques lignes permettent d'importer le module pygame. Celui-ci est directement initialisé afin d'utiliser la méthode pygame.display.set_mode(()). Cette méthode permet de définir l'écran utilisé. L'origine de l'affichage est en haut à gauche et augmente vers le bas et la droite avec les valeurs saisies. Cette méthode supprime tous types d'écran déjà présents. De plus, cette méthode ne nous restreint pas sur la modification des pixels plus tard.

Question 2 :

```python
import pygame


pygame.init()
ecran = pygame.display.set_mode((300, 200))

continuer = True
while continuer:
    for event in pygame.event.get():
        if event.type == pygame.KEYDOWN:
            continuer = False

pygame.quit()
```

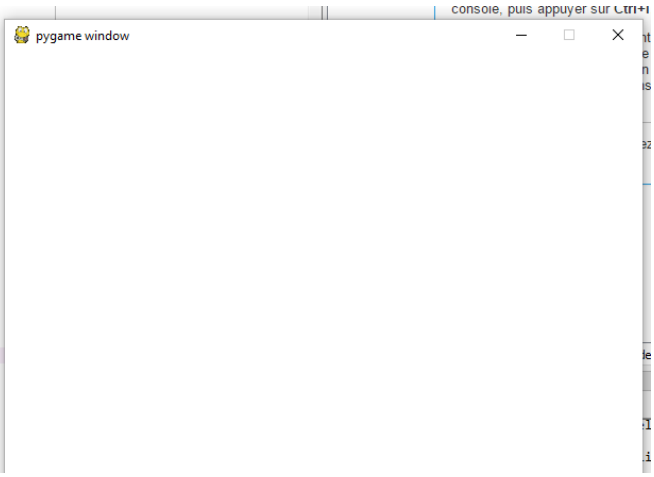


On constate effectivement l'ouverture d'une fenêtre. La méthode pygame.event.get crée un évènement. Celui est complété avec un booléen qui est tout le temps sur vrai. La fonction pygame.KEYDOWN permet de définir une action. En effet, en appuyant sur n'importe quelle touche du clavier la fenêtre contenant notre écran de ferme.

B. Utilisation de Pyopengl pour représenter des objets 3D

Question 1 :

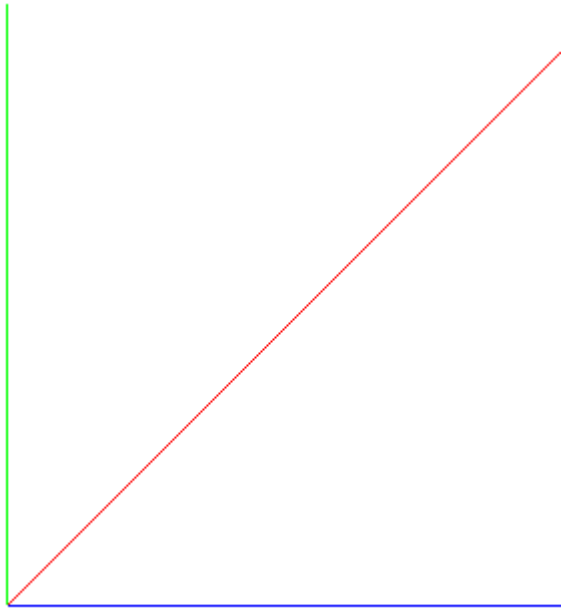
```python
import pygame
import OpenGL.GL as gl
import OpenGL.GLU as glu

if __name__ == '__main__':
    pygame.init()
    display=(600,600)
    pygame.display.set_mode(display, pygame.DOUBLEBUF | pygame.OPENGL)

    # Sets the screen color (white)
    gl.glClearColor(1, 1, 1, 1)
    # Clears the buffers and sets DEPTH_TEST to remove hidden surfaces
    gl.glClear(gl.GL_COLOR_BUFFER_BIT | gl.GL_DEPTH_BUFFER_BIT)
    gl.glEnable(gl.GL_DEPTH_TEST)

    # Placer ici l'utilisation de gluPerspective.

    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
        exit()
```



Premièrement les modules OpenGL.GL et OpenGL.GLU sont importés en plus de pygame. Ensuite tout se déroule main dans le main. La fenêtre est agrandi et transformée en blanc. Les méthodes du module gl. Permettent de manipuler directement l'écran comme par exemple sa couleur et les surfaces.

Question 2 :

Ces lignes de codes permettent d'afficher les lignes de graphiques, on répète 3 fois l'opération afin d'afficher les 3 axes.



On trace chaque axe avec une couleur différente.

Question 3 :

C. Découverte de l'environnement du travail du TP

Question 1.a :

Le code dans Q1 Configuration.py crée des évènements lorsqu'on appuie sur des touches est appuyée. Quand on appuie sur la touche "z" l'axe fait une rotation autour de l'axe z dans le sens positif. Alors que "Z" l'axe effectue une rotation dans le sens négatif. La touche "a" permet de faire disparaître et apparaître les éléments.
Le code permet d'afficher 2 axes : l'axe en rouge et l'axe y en vert.

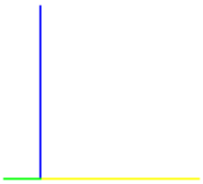
Question 1.b :

La modification du code rend l'axe x qui était rouge en vert. Screen position : -5, décale le graphique de -5.

Les setParameter() initialise le graphique et définit les axes, on finit avec un display pour les intégrer dans la fenêtre.

Question 1.c :

Pour que l'axe z apparaît verticalement, on ajoute le code qui suit à la méthode *initializeTransformationMatrix()* dans le programme Configuration.py. A l'aide de la méthode rotate appartenant au module gl nous réalisons un rotation de 90° dans le sens antihoraire sur x :



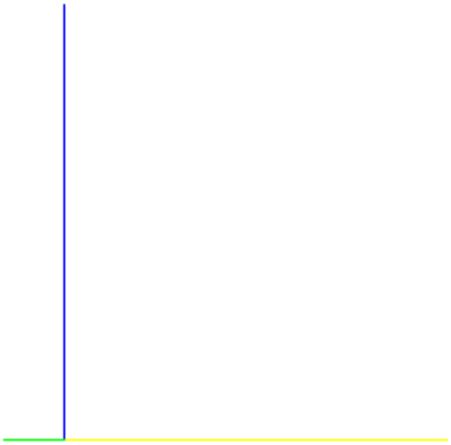
gl.glRotate(-90,1,0,0)

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

Question 1.d :

On ajoute à la gestion des touches via la méthode processKeyDownEvent(self), la gestion des touches permettant de zoomer et dézoomer. La touche "U" (Up) correspond à un zoom de *1.5 et "D" (down) à un dézoom de *0.5. Cela est géré grâce à une boucle elif permettant de proposer un cas par touche. Ensuite on associe un évent à chaque touche. Enfin le zoom se gère via la méthode glScale(..,..,...,).

```
# Grossi l'affichage-Page Up
elif self.event.dict['unicode'] == 'U':
    gl.glScale(1.5,1.5,1.5)
# Réduit l'affichage-Page Down
elif self.event.dict['unicode'] == 'D':
    gl.glScale(0.5,0.5,0.5)
```



En appuyant sur “U”

En appuyant sur “D”

Question 1.e :

Il faut maintenant réaliser la même opération mais en utilisant la molette de la souris. Il faut maintenant créer une autre méthode processMouseButtonDownEvent(self). Il faut ensuite créer un évent pour le zoom et un pour le dézoom. La molette avant correspond à la touche 4 de la souris et la molette arrière à la touche 5 de la souris.

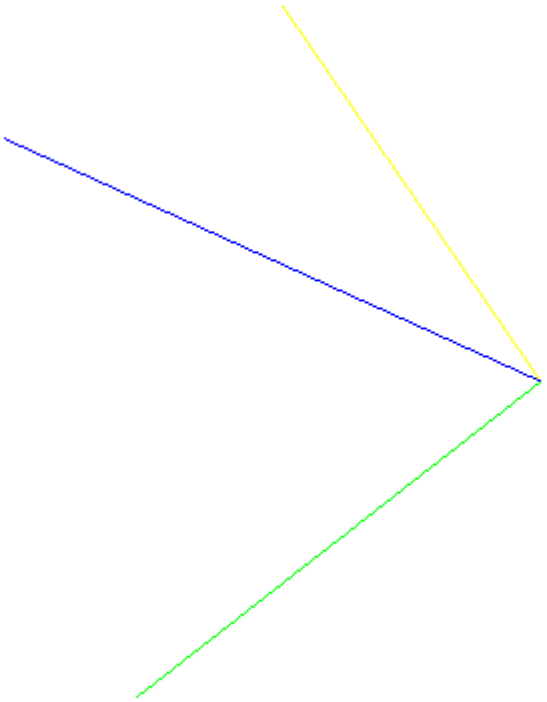
```
# Processes the MOUSEBUTTONDOWN event
def processMouseButtonDownEvent(self):
    if self.event.type == pygame.MOUSEBUTTONDOWN:
        # zoom avant avec la molette de la souris
        if self.event.button ==4:
            gl.glScale(1.5,1.5,1.5)
        # zoom avant avec la molette de la souris
        elif self.event.button == 5:
            gl.glScale(0.5,0.5,0.5)
```

Question 1.f :

Cette question va nous permettre de pouvoir observer nos axe dans tous les angles possible en déplaçant celui grâce à notre curseur de souris.
Ici le clic gauche correspond à la rotation, on a donc get_pressed()[0] qui correspond au clic gauche de la souris, avec le if on autorise les rotation faites par le clic de la souris, le sinon correspond au non clic et donc à la rotation = 0.

Alors que le clic droit correspond à la translation get_pressed()[2] qui correspond au clic droit de la souris, si les conditions du if sont respectées alors on peut translater ici un clic droit sinon pas de translation quand il n’y a pas de clic.

```python
    def processMouseMotionEvent(self):
        if self.event.type == pygame.MOUSEMOTION:
        # rotation clic gauche souris et translation clic droit souris
            if pygame.mouse.get_pressed()[0]==1:
                gl.glRotate(self.event.rel[0],1,0,0)
                gl.glRotate(self.event.rel[1],0,0,1)
            else:
                gl.glRotate(0,0,0,0)
            if pygame.mouse.get_pressed()[2]==1:
                gl.glTranslate(0.1*self.event.rel[0], 0,0.1*self.event.rel[1])
            else:
                gl.glTranslate(0,0,0)
```



--> Rotation

3. Création d'une section

Question 2.a :

Question 2.b :

Question 2.c :

Question 2.d :

4. Création des murs

Question 3.a :

5. Création d’une maison

Question 4.a :

6. Création d’ouvertures

Question 5.a :

7. Pour finir…

Conclusion