CHAIX Adrien
TO Nicolas
Date du TP (07/12/2021)

Rapport de TP3 – Représentation visuelle d'objets

I. Introduction

L'objet de ce TP est de représenter des objets en 3D dans une fenêtre graphique dans laquelle, on peut effectuer des opérations de zoom, de translations et de rotations. On s'intéressera aux modules PyGame et PyOpenGL.

II. Préparation à faire avant le TP

1. Question (1).

Il nous est demandé d'écrire un code simple sur Spyder.
Voici le code :

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

En exécutant le code, une fenêtre Pygame de dimension 300 x 200 s'ouvre puis se coupe. Cela correspond bien à ce qui est écrit dans le programme : il affiche à l'écran une fenêtre pygame, cette fenêtre est de dimension 300 x 200 puis le programme fait quitter la fenêtre.

2. Question (2).

Dans cette question, il est demandé d'écrire un programme similaire à la 1re question cependant, il y a des conditions qui s'ajoutent.
Voici le code :

```
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()
```

Le programme fait ouvrir une fenêtre Pygame de dimension 300 x 200.
La fenêtre reste ouverte tant qu'il n'y a pas un événement de type événement pygame. S'il y a un événement du type : appuyer sur touche du clavier, la fenêtre se fermera.

3. Question (1). : Utilisation de Pyopengl pour représenter des objets 3D

Il nous est demandé de rentrer le code suivant :

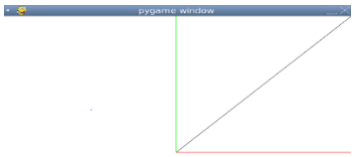
```
import pygame
import OpenGL.GL as gl
import OpenGL.GLU as glu

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. 18
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
            exit()
```

4. Question (2).

Il nous est demandé de tracer des axes en utilisant les fonctions glBegin, glColor3fv, glVertex3fv et glEnd.

```
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)

    glu.gluPerspective(45, (display[0] / display[1]), 0.1, 50.0)
    gl.glTranslatef(0.0, 2, -5)
    gl.glRotatef(-90, 1, 0, 0)

    gl.glBegin(gl.GL_LINES) # Indique que l'on va commencer un trace en mode lignes (segments)

    gl.glColor3fv([0, 0, 0]) # Indique la couleur du prochian segment en RGB
    gl.glVertex3fv((0,0, -2)) # Premier vertice : départ de la ligne
    gl.glVertex3fv((1, 1, -2)) # Deuxième vertice : fin de la ligne

    gl.glBegin(gl.GL_LINES) # Indique que l'on va commencer un trace en mode lignes (segments)
    gl.glColor3fv([0, 0, 0]) # Indique la couleur du prochian segment en RGB
    gl.glVertex3fv((0,0, -2)) # Premier vertice : départ de la ligne
    gl.glVertex3fv((1, 1, -2)) # Deuxième vertice : fin de la ligne

    gl.glEnd() # Find du tracé
    pygame.display.flip() # Met à jour l'affichage de la fenêtre graphique
```

Voici ce que nous renvoie le programme :



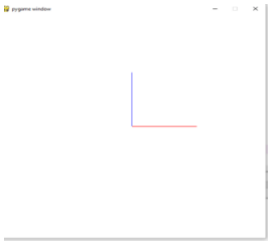
Nous avons en rouge l'axe x, en vert l'axe y, en bleu l'axe z normal au plan (Oxy) et en noir, l'axe défini dans l'énoncé

5. Question (3).

Il nous est demandé d'effectuer une translation de 2 en y et -5 en z puis une rotation de -90 degrés.

```
glu.gluPerspective(45, (display[0] / display[1]), 0.1, 50.0)
gl.glTranslatef(0.0, 2, -5)
gl.glRotatef(-90, 1, 0, 0)
```

Voici ce que renvoie le programme :

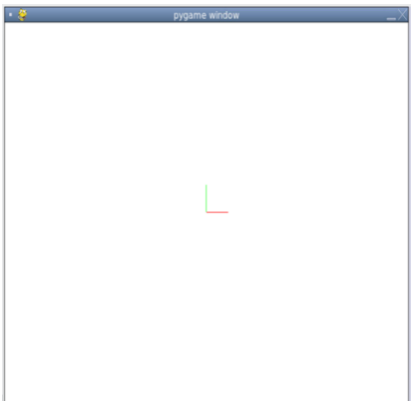


6. Question (1a). Découverte de l'environnement du travail du TP

Il nous est demandé de rentrer la commande return Configuration () :

```
def Q1a():
    return Configuration()
```

Voici ce que renvoie le programme :



Le programme nous renvoie les trois axes x, y, z dans leur couleur respective.

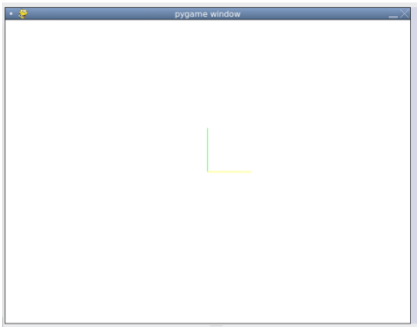
7. Question (1b).

Il nous est demandé d'expliquer ce que faisait la commande donnée dans l'énoncé.

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

```
Configuration({'screenPosition': -5, 'xAxisColor': [1, 1, 0]}). \
    setParameter('xAxisColor', [1, 1, 0]). \
    setParameter('yAxisCo lor', [0,1,1]). \
    display()
```

Ces commandes permettent de changer la couleur des axes mais aussi la profondeur.

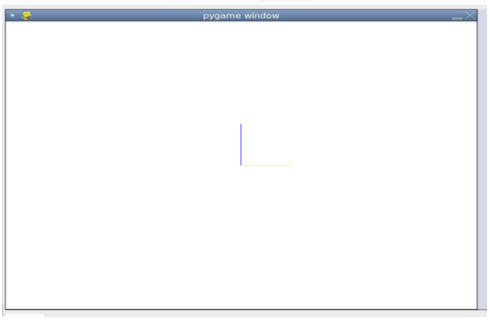


8. Question (1c).

Il nous est demandé d'ajouter une instruction dans la méthode initializeTransformationMatrix() de sorte que l'axe z soit vetical et l'axe x soit horizontal tandis que l'axe y devient la profondeur.

```
gl.glRotatef(-90, 1, 0, 0)
```

Cette instruction permet de faire une rotation de 90 degrés.



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

1. Question 1.D

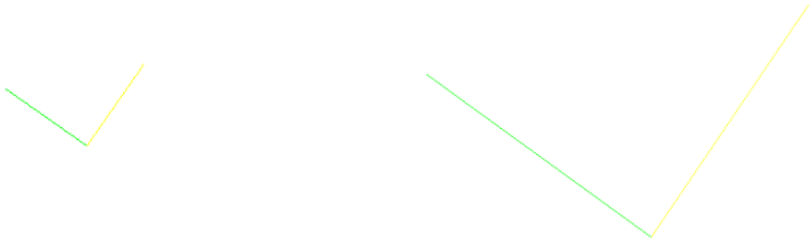
Il nous est demandé d'écrire une méthode processKeyDownEvent qui permet d'effectuer des rotations de +-2,5 degrés autour de l'axe z et de pouvoir grossir ou réduire l'affichage. Le facteur d'échelle demandé pour un zoom positif de 1.1 et un zoom négatif de 1/1.1.

Voici le programme entré :

```
# Processes the KEYDOWN event
def processKeyDownEvent(self):
    # Rotates around the z-axis
    if self.event.dict['unicode'] == 'Z' or (self.event.mod & pygame
        gl.glRotate(-2.5, 0, 0, 1)
    elif self.event.dict['unicode'] == 'z' or self.event.key == pyga
        gl.glRotate(2.5, 0, 0, 1)

    # Draws or suppresses the reference frame
    elif self.event.dict['unicode'] == 'a' or self.event.key == pyga
        self.parameters['axes'] = not self.parameters['axes']
        pygame.time.wait(300)

    if self.event.key == pygame.K_PAGEUP:
        gl.glScale(1.1,1.1,1.1)

    elif self.event.key == pygame.K_PAGEDOWN:
        gl.glScale(1/1.1,1/1.1,1/1.1)
```

La touche « Z » permet une rotation de -2,5 degrés et la touche « z » une rotation de +2,5 degrés tandis que les touches « PageUp » et « PageDown » permettent d'effectuer un zoom grâce à la fonction gl.glScale

On a testé le programme et il y a aucune erreur :



2. Question 1.E

Il nous est demandé d'écrire la méthode processMouseButtonDownEvent qui permet d'effectuer un zoom via la molette de la souris. On associe en plus à l'événement qui prend les valeurs 4 ou 5.

```
# Processes the MOUSEBUTTONDOWN event
def processMouseButtonDownEvent(self):
    if self.event.button == 5: #la molette descend
        gl.glScalef(1/1.1,1/1.1,1/1.1)

    elif self.event.button == 4: #la molette descend
        gl.glScalef(1.1,1.1,1.1)
```

On a testé le programme et il y a eu aucune erreur, le zoom s'effectue correctement avec la molette.

IV. Conclusion

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