

Rapport de TP3 – Représentation visuelle d'objets

I/ Introduction

L'objectif de ce TP est de s'approprier la représentation d'objets 3D à l'écran dans une fenêtre graphique qui permet des opérations de zoom, rotation et translations.

II – Travail préparatoire

Utilisation de Pygame

1)

Explication du code :

La première ligne « import pygame » permet d'importer le package pygame. La deuxième ligne permet d'initialiser. A la troisième ligne « ecran = pygame.display.set_mode((300, 200)) » permet de créer une fenêtre en lui passant des paramètres de dimension, ces paramètres indiquent que la fenêtre fera 200 pixels de haut et 300 pixels de large. La quatrième ligne permet de nettoyer, c'est-à-dire libérer les ressources dont nous n'avons plus besoin.

Lorsque l'on exécute le code, une fenêtre noire apparaît qui se referme instantanément.

2)

Lorsque nous lançons le second programme, la fenêtre ne se referme plus immédiatement. Mais dès que l'on appuie sur une touche, la fenêtre se ferme immédiatement.

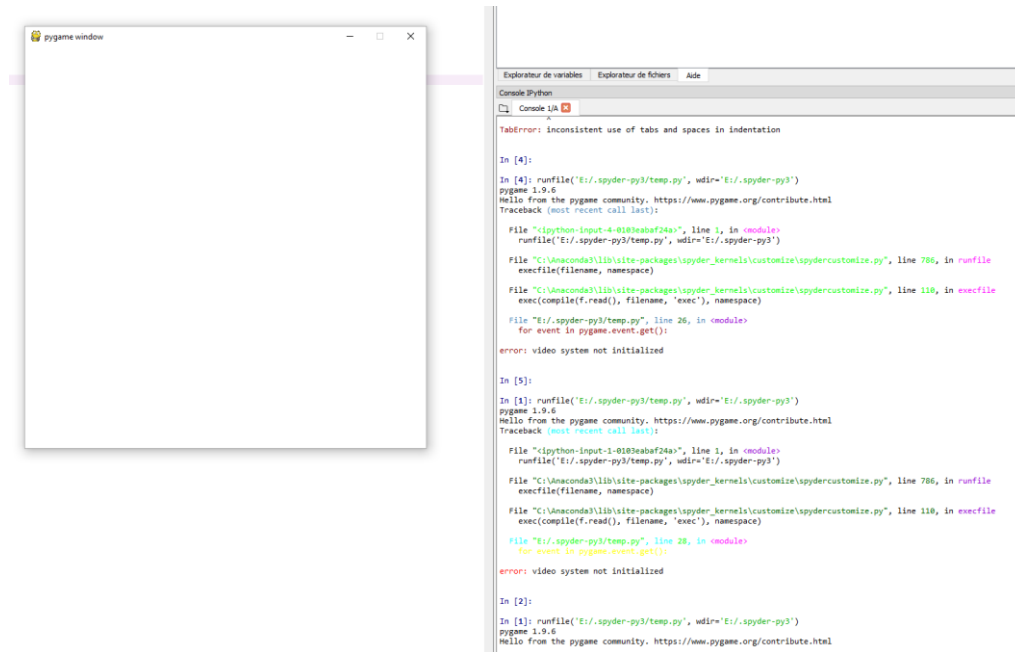
Explication du code :

La première ligne permet de garder la fenêtre ouverte. Les boucles while et for permettent de récupérer tous les fichiers créés de pygame. Les lignes 4 et 5 vérifient si l'on appuie sur une touche du clavier (KEYDOWN) et si c'est le cas, permettent de stopper la fonction continue. La boucle s'arrête et la dernière ligne permet de fermer la fenêtre.

Utilisation de Pyopengl pour représenter des objets 3D

1)

Nous avons effectué le code suivant afin de gérer la perspective :
perspective = glu.gluPerspective (45, 1, 0.1, 50)



Il n'y a pas d'erreurs.

2)

Voici le code que nous avons effectué ainsi que notre résultat obtenu :

```
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.

    glu.gluPerspective(45, 1, 0.1, 50)

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

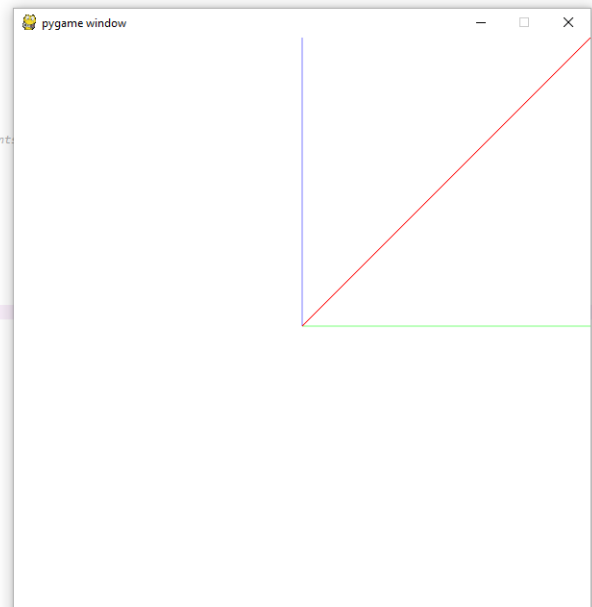
    gl.glColor3fv([255, 0, 0]) # Indique la couleur du prochain segment en RGB
    gl.glVertex3fv((0, 0, -2)) # Premier vertice : départ de la ligne
    gl.glVertex3fv((1, 1, -2)) # Deuxième vertice : fin de la ligne

    gl.glColor3fv([0, 255, 0]) # Indique la couleur du prochain segment en RGB
    gl.glVertex3fv((0, 0, -2)) # Premier vertice : départ de la ligne
    gl.glVertex3fv((1, 0, -2)) # Deuxième vertice : fin de la ligne

    gl.glColor3fv([0, 0, 255]) # Indique la couleur du prochain segment en RGB
    gl.glVertex3fv((0, 0, -2)) # Premier vertice : départ de la ligne
    gl.glVertex3fv((0, 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

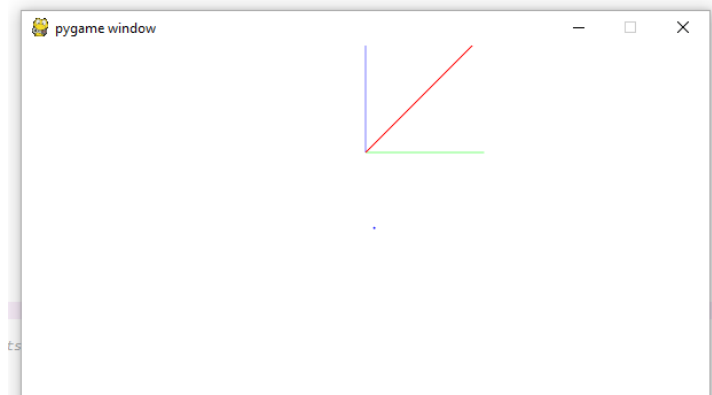
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                exit()
```



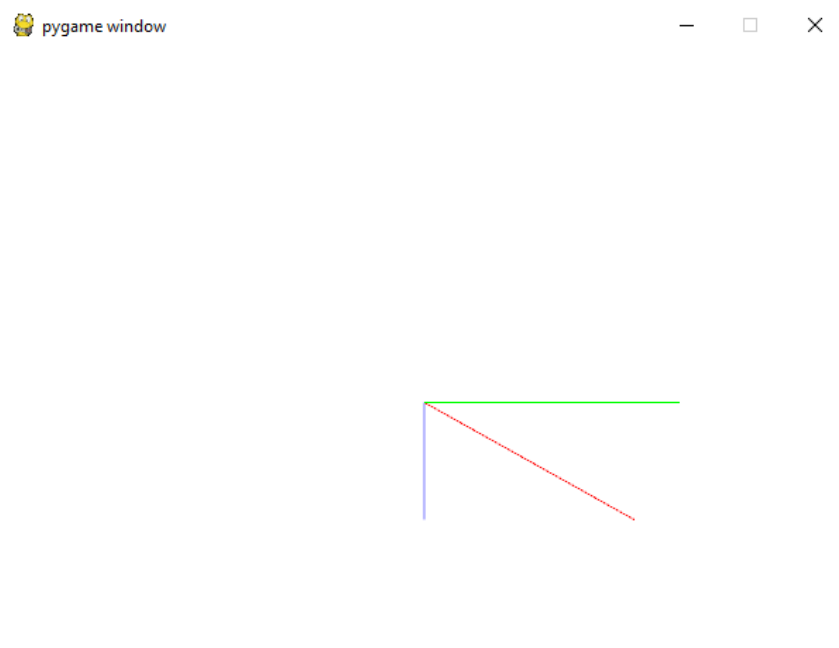
Dans la ligne « `gl.glColor3fv([0, 0, 0])` », le 1^{er} chiffre correspond au rouge, le 2^e correspond au vert et le 3^e chiffre correspond au bleu.

3)

Voici la translation que nous avons effectuée, Nous avons décalé y de 2 et z de -5. Voici notre résultat :



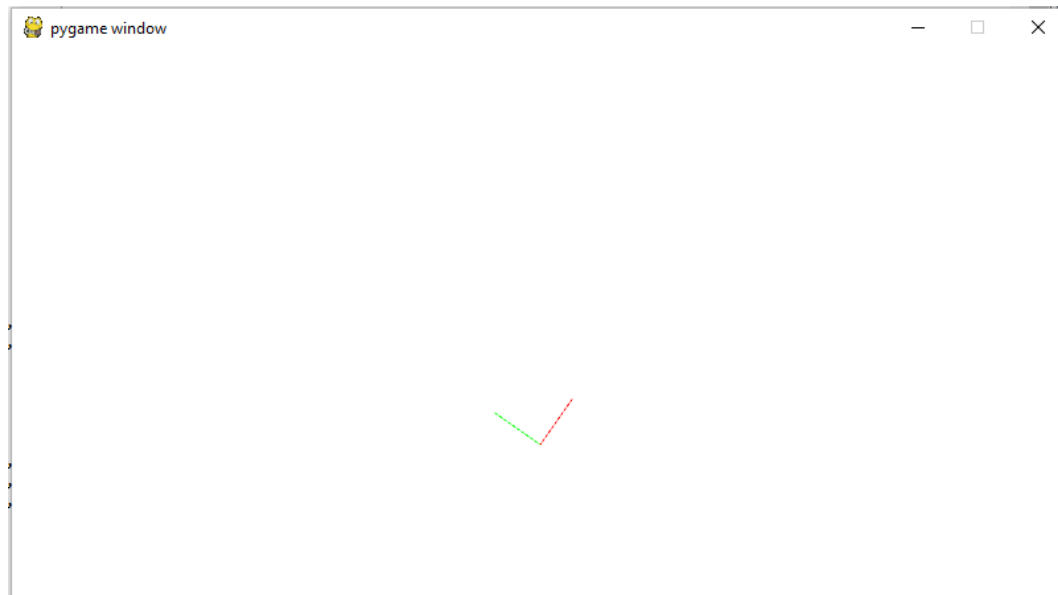
Voici la rotation que nous avons effectuée. Nous avons effectué une rotation de 120 degrés autour de l'axe x. Voici notre résultat :



Découverte de l'environnement de travail

1) a)

Nous remarquons que lorsque nous appuyons sur la touche a, le graphique disparaît, lorsque l'on réappuie sur la touche, le graphique réapparaît. Lorsque l'on appuie sur la touche z, le graphique tourne dans le sens trigonométrique et sur la touche Z (maj+z), le graphique tourne dans le sens horaire.

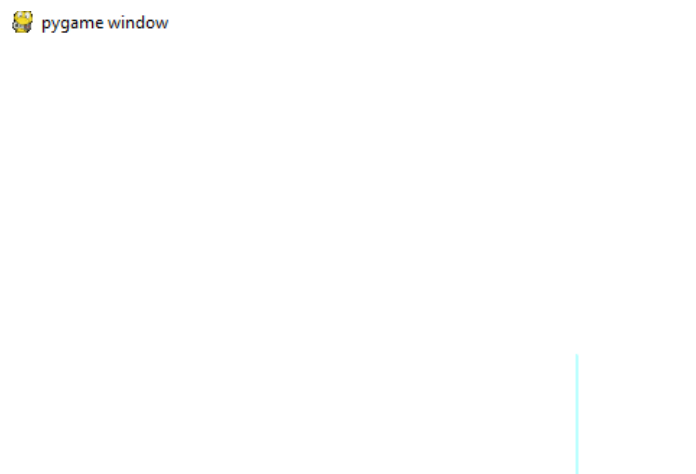


Dans le Main, on importe toutes les classes de chaque module. On peut exécuter l'ensemble des modules avec tous les paramètres inscrit dans le Main.

1) b)

Le code en question permet de modifier la couleur des axes

Voici ce que nous obtenons :



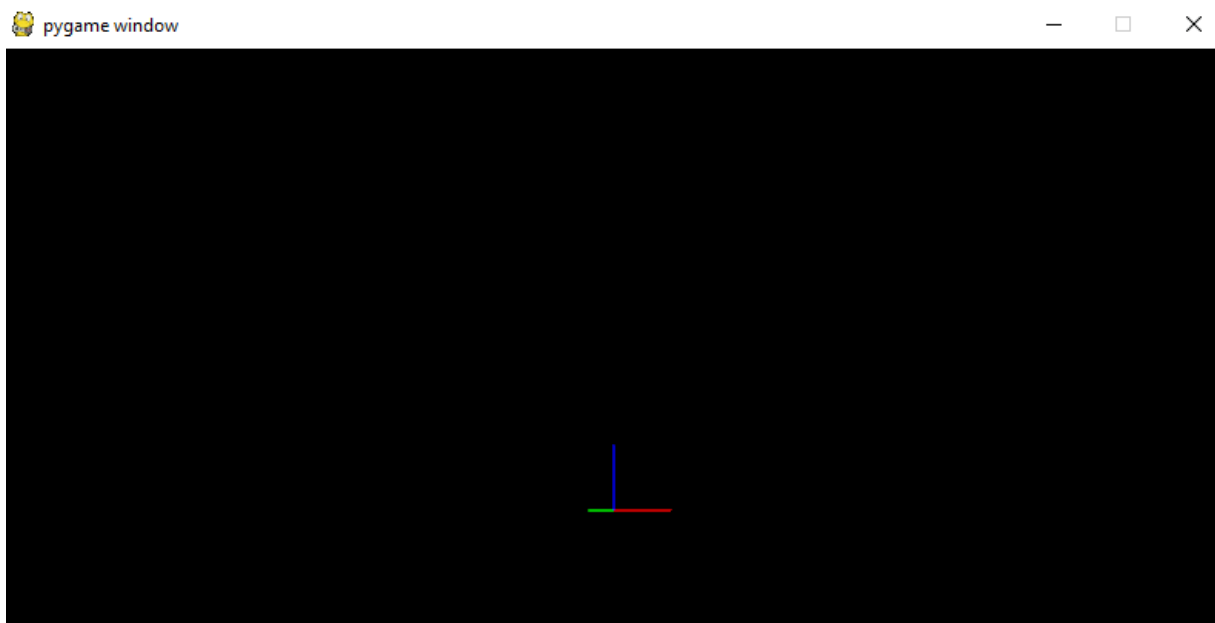
Le chainage de l'appel des méthodes SetProperty() et display est disponible car pour utiliser display et donc afficher les axes, il faut les paramétrer à l'aide de SetProperty.

1) c)

Pour que l'axe z soit représenté verticalement sur l'écran et que l'axe x soit représenté horizontalement, nous avons fait une rotation de 90° autour de l'axe x. Voici notre code que nous avons modifié :

```
gl.glMatrixMode(gl.GL_MODELVIEW)
gl.glLoadIdentity()
gl.glTranslatef(0.0,0.0, self.parameters['screenPosition'])
gl.glRotatef(-90,1,0,0)
```

Voici ce que nous obtenons :



On a bien l'axe z qui est représenté verticalement sur l'écran et l'axe qui est représenté horizontalement.

De plus, nous avons changé le fond d'écran pour plus de visibilité.

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

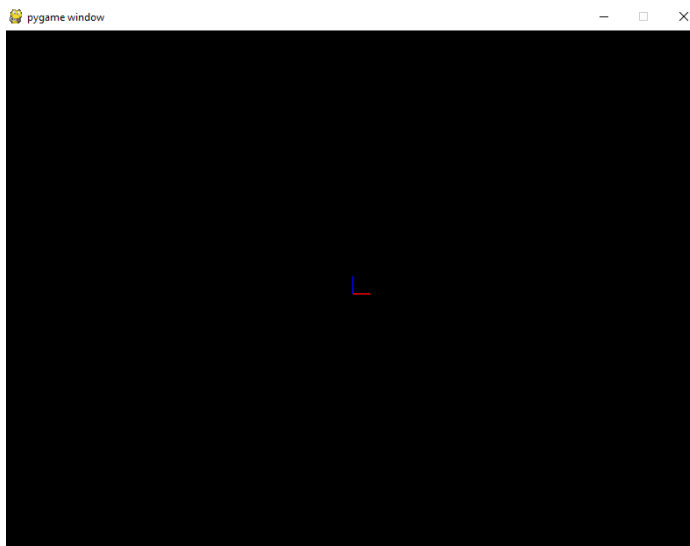
1) d)

Voici le code que nous avons effectué :

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

Nous avons ajouté pour les trois axes un facteur d'échelle pour un zoom positif de 1.1 et pour un zoom négatif de 1/1.1.

Voici les résultats que nous avons obtenu :



1) e)

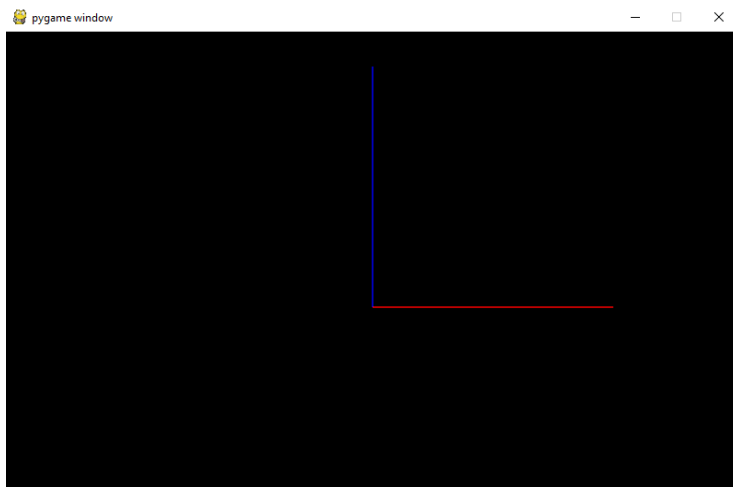
L'objectif est maintenant de pouvoir zoomer à l'aide de la roulette de la souris. Voici le code que nous avons réalisé :

```
# Processes the MOUSEBUTTONDOWN event
def processMouseButtonDownEvent(self):
    if self.event.button == 4:
        gl.glScalef(1.1,1.1,1.1)
    elif self.event.button == 5:
        gl.glScalef(1/1.1,1/1.1,1/1.1)
```

Comme pour la question précédente, nous avons ajouté pour les trois axes un facteur d'échelle pour un zoom positif de 1.1 et pour un zoom négatif de 1/1.1.



Voici un exemple que nous avons obtenue :



1) f)

Voici le code que nous avons effectué :

```
# Processes the MOUSEMOTION event
def processMouseEvent(self):
    if self.event.type == pygame.MOUSEMOTION:
        if pygame.mouse.get_pressed()[0]==1:
            gl.glRotate(self.event.rel[1], -1, 0, 0)
            gl.glRotate(self.event.rel[0], 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)
```

Quand on clique et que l'on bouge avec la souris, tous les axes bougent. Le code de fonctionne donc bien correctement.