SALESSE Maxime
LE CLOËREC Maëlis
15/12/21

Rapport de TP3 – Représentation visuelle d'objet

I. Introduction

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

II. Section 1 du TP

1. Question (1).

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

Pygame.display. : Configure the display surface.
Pygame.display.set mode : Initialize a window or screen for display
Ici la taille de la fenetre est de 300*200.

Lorsqu'on lance le programme, un écran d'affiche de taille 300*200, qui disparait tres vite.
En effet le programme quitte directement la fenetre. Il n'y a pas de boucle

2. Question (2).

```
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()
```

Dans ce cas, grâce à la boucle while, l'écran s'affiche indéfiniment. Pour quitter le programme, il doit détecter que la touche « KEYDOWN » du clavier soit actionnée. Lorsqu'une touche est actionnée par l'utilisateur, la fenêtre se ferme.

III. Utilisation de Pyopengl pour représenter des objets 3D

1. Question (1).

On utilise la fonction **gluPerspective (fovy, aspect, zNear, zFar)**. On utilisera un Field of View de 45° Et comme plan de clipping near la valeur 0.1 Et le plan de clipping far, la valeur 50.

2. Question (2).

Tracer des axes en utilisant les fonctions **glBegin**, **glColor3fv** et **glVertex3fv** et enfin de la fonction **glEnd**

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

        #set screen to white
        gl.glClearColor(1,1,1,1)

        gl.glClear(gl.GL_COLOR_BUFFER_BIT | gl.GL_DEPTH_BUFFER_BIT)
        gl.glEnable(gl.GL_DEPTH_TEST)
        glu.gluPerspective(45, (1, 0.1, 50.0)


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

        gl.glColor3fv([0, 255, 0]) # Indique la couleur du prochian segment en RGB
        gl.glVertex3fv((0,0, -2)) # Premier vertice : départ de la ligne
        gl.glVertex3fv((10, 5, -2)) # Deuxième vertice : fin de la ligne

        gl.glColor3fv([255, 0, 0]) # Indique la couleur du prochian segment en RGB
        gl.glVertex3fv((0,0, -2)) # Premier vertice : départ de la ligne
        gl.glVertex3fv((10, 1, -2)) # Deuxième vertice : fin de la ligne

        gl.glColor3fv([0, 0, 255]) # Indique la couleur du prochian segment en RGB
        gl.glVertex3fv((0,0, -2)) # Premier vertice : départ de la ligne
        gl.glVertex3fv((1, 10, -2)) # Deuxième vertice : fin de la ligne



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


        gl.glTranslatef(0.0, 2, -5)
        gl.glRotatef(-90, 1, 0, 0)


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

IV. Découverte de l'environnement du travail du TP

1. Question (1.a).

a : permet de faire disparaitre la figure
Z : permet de faire tourner la figure dans le sens horaire, en appui long
z : permet de faire tourner la figure dans le sens trigonométrique (antihoraire), en appui-long.

Configaration.py donne les paramètres, accumule les objets que l'on veut afficher. Il genere la base.
Main.py : il exécute configuration.py

2. Question (1.b).

- La modification permet d'afficher l'écran et de changer la couleur de l'axe.

- Chaque méthode renvoie un objet, permettant aux appels d'être enchaînés dans une seule instruction sans nécessiter de variables pour stocker les résultats intermédiaires. Ici la fonction d'un objet qui renvoie l'objet, on peut ainsi mettre en cascade les différentes fonctions car on utilise directement le résultat.

- Le screenPosition dans le setter est nécessaire, car il sert a initialiser la nouvelle valeur de la position de l'écran, afin que ça ne reste pas indéfiniment sur la position du début.

3. Question (1.c).

```
def initializeTransformationMatrix(self):
    gl.glMatrixMode(gl.GL_PROJECTION)
    gl.glLoadIdentity()
    glu.gluPerspective(70, (self.screen.get_width()/self.screen.get_height()), 0.1

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

Pour avoir l'axe z en verticale, y en profondeur et x en horizontal, il faut donc faire une rotation de -90°C sur l'ancienne base.

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

1. Question (1.d).

Fonction zoom avec le clavier :

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

Ici le Zoom est de *1.1 à chaque fois qu'on appui sur la touche du haut du clavier.

2. Question (1.e).

Fonction zoom avec la souris :

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

3. Question (1.f).

Une rotation autour de x composée avec une rotation autour de z lorsque le bouton gauche est enfoncé et que la souris est déplacée.

Une translation selon l'axe x composée à une translation selon l'axe z lorsque le bouton droit est enfoncé et que la souris est déplacée.

```
# Processes the MOUSEMOTION event
def processMouseMotionEvent(self):
    if pygame.mouse.get_pressed()[0] == 1 :
        gl.glRotatef(self.event.rel[0]/10,1,0,1)
    elif pygame.mouse.get_pressed()[2] == 1 :
        gl.glTranslatef(self.event.rel[0]/10,0,self.event.rel[0]/10)
```

VI. Création d'une section

1. Question (2.a).

Création des sommets et des faces :

```
# Defines the vertices and faces
def generate(self):

    #definir les 6 sommets
    self.vertices = [
            [0, 0, 0 ],
            [0, 0, self.parameters['height']],
            [self.parameters['width'], 0, self.parameters['height']],
            [self.parameters['width'], 0, 0],
            [0,self.parameters['thickness'],0],
            [0,self.parameters['thickness'],self.parameters['height']]
            [self.parameters['width'],self.parameters['thickness'],self.parameters['height']]
           [self.parameters['width'],self.parameters['thickness'],0

            ]
          #definir les 6 faces
    self.faces = [
           [0,1,2,3],
           [0,4,5,1],
           [4,5,6,7],
           [6,2,3,7],
           [1,5,6,2],
           [0,4,7,3],

            ]
```

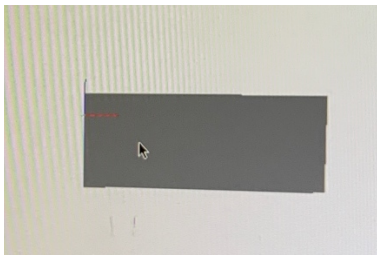
2. Question (2.b).

Remplir les 6 faces dans une couleur, modifiable :

```
# Draws the faces
def draw(self):

    for i in self.faces :
        gl.glPushMatrix()
        gl.glTranslatef(0.0,0.0,-2.0)
        gl.glPolygonMode(gl.GL_FRONT_AND_BACK,gl.GL_FILL)
        gl.glBegin(gl.GL_QUADS)
        gl.glColor3fv([0.5,0.5,0.5])
        gl.glVertex3fv(self.vertices[i[0]])
        gl.glVertex3fv(self.vertices[i[1]])
        gl.glVertex3fv(self.vertices[i[2]])
        gl.glVertex3fv(self.vertices[i[3]])
        gl.glEnd()
        gl.glPopMatrix()
```

Résultat :



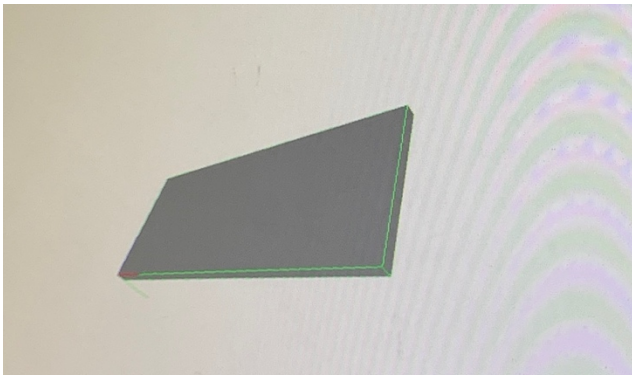
3. Question (2.c).

Dessiner les arêtes de la section :

```
# Draws the edges
def drawEdges(self):
    for i in self.faces :
        gl.glPushMatrix()
        gl.glTranslatef(0.0,0.0,0.0)
        gl.glPolygonMode(gl.GL_FRONT_AND_BACK,gl.GL_LINE)
        gl.glBegin(gl.GL_QUADS)
        gl.glColor3fv([0.5,250,0.5])
        gl.glVertex3fv(self.vertices[i[0]])
        gl.glVertex3fv(self.vertices[i[1]])
        gl.glVertex3fv(self.vertices[i[2]])
        gl.glVertex3fv(self.vertices[i[3]])
        gl.glEnd()
        gl.glPopMatrix()

# Draws the faces
def draw(self):
    if self.parameters['edges'] == TRUE :
        self.drawEdges()
    for i in self.faces :
        gl.glPushMatrix()
        gl.glTranslatef(0.0,0.0,-2.0)
        gl.glPolygonMode(gl.GL_FRONT_AND_BACK,gl.GL_FILL)
        gl.glBegin(gl.GL_QUADS)
        gl.glColor3fv([0.5,0.5,0.5])
        gl.glVertex3fv(self.vertices[i[0]])
        gl.glVertex3fv(self.vertices[i[1]])
        gl.glVertex3fv(self.vertices[i[2]])
        gl.glVertex3fv(self.vertices[i[3]])
        gl.glEnd()
        gl.glPopMatrix()
```

Résultat :



Ici on aperçoit les arrêtes qui sont symbolisés par les traits verts

VII. Création des murs

1. Question (3.a).

Dessiner le mur avec ses arêtes :

```
def draw(self):
    for i in range (len(self.objects)):
        self.objects[i].draw()
```

```
def Q3a():
    mur = Wall({'position': [1, 1, 0], 'width':5, 'height':2.6, 'orientation' : 0,'thickness' : 0.2,'color' : [10, 10, 10] })
    return Configuration().add(mur)
```

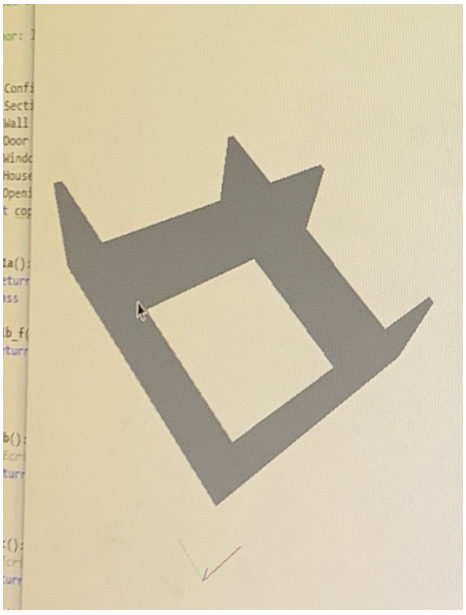
VIII. Création d'une maison

```
# Draws the house
def draw(self):
    for i in range (len(self.objects)):
        self.objects[i].draw()
```

```
def Q4a():

    wall1 = Wall({'position': [1, 1, 0], 'width':5, 'height':2, 'orientation' : 0,'thickness' : 0.2,'color' : [10, 10, 10] })
    wall2 = Wall({'position': [1.2, 1, 0], 'width':5, 'height':2, 'orientation' : 90,'thickness' : 0.2,'color' : [10, 10, 10] }
    wall3 = Wall({'position': [1, 6, 0], 'width':5, 'height':2, 'orientation' : 0,'thickness' : 0.2,'color' : [10, 10, 10] })
    wall4 = Wall({'position': [6, 1, 0], 'width':5, 'height':2, 'orientation' : 90,'thickness' : 0.2,'color' : [10, 10, 10] })
    house = House({'position': [-3, 1, 0], 'orientation':0})
    house.add(wall1).add(wall3).add(wall4).add(wall2)
    return Configuration().add(house)
```

Resultat :



IX. Conclusion

Ce TP à été réalisé lors de 2 séances. Lors de la première séance nous nous sommes arrêtés avant la chapitre IV, création de section. Lors de cette séance nous n'avons pas rencontré de difficultés majeurs, mise à part le dézoom avec le clavier sur lequel nous avons passés plus de temps. Lors de la deuxième séance nous avons dû se réapproprier le sujet. Puis nous avons passé du temps sur la création d'une section, qui nécessitait une compréhension complète des méthodes à adopter. Nous n'avons cependant pas réussi à atteindre la partie « ouverture » dans le temps qui nous étaient impartis.
Lors de ces séances nous avons pu apprendre de nombreuses choses, et conforter d'autre. Ce TP est très concret, et les évolutions de notre programme était visible avec l'affichage sur l'écran de notre figure.