BOUILLARD Justine
CHAVRIER Justine
Date du TP (06/12/2021)

Rapport de TP3 – Représentation visuelle d'objets

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 est translations.

II. Préparation à faire avant le TP

Utilisation de Pygame

1) 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, ces paramètres indiquent que le 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 pendant 2 secondes.

2) Nous avons essayé le nouveau code. La fenêtre reste ouverte. En revanche, lorsque l'on appuie sur une touche quelconque du clavier, la fenêtre se ferme :

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

pygame.quit()
```

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.

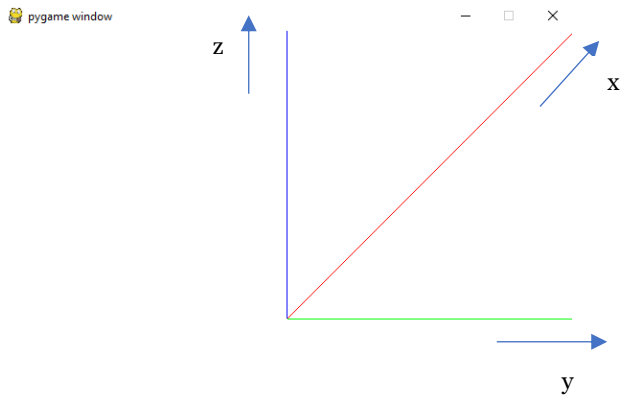
Utilisation de Pyopengl pour représenter des objets 3D

1) Pour gérer la perspective, en initialisant la matrice de perspective, nous avons écrit le code suivant : « perspective = glu.gluPerspective (45, 1, 0.1, 50) ».
Nous avons exécuté le fichier, il n'y a pas d'erreur.

2)Nous avons tracé les axes. Dans la ligne « gl.glColor3fv([0, 0, 0]) », le 1er chiffre correspond au rouge, le 2e correspond au vert et le 3e chiffre correspond au bleu. Voici notre code :

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

    gl.glColor3fv([255, 0, 0])
    gl.glVertex3fv((0,0, -2))
    gl.glVertex3fv((1, 1, -2))

    gl.glColor3fv([0, 255, 0])
    gl.glVertex3fv((0,0, -2))
    gl.glVertex3fv((1, 0, -2))


    gl.glColor3fv([0, 0, 255])
    gl.glVertex3fv((0,0, -2))
    gl.glVertex3fv((0, 1, -2))

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

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

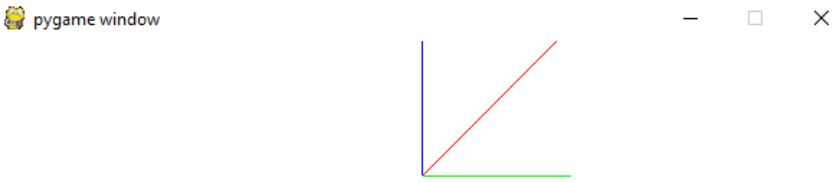
Voici ce que nous obtenons :



3) Nous avons fait la translation. Nous avons décalé y de 2 et z de -5. Voici notre code :

```
# Placer ici l'utilisation de gluPerspective.
glu.gluPerspective(45, (display[0] / display[1]), 0.1, 50.0)
gl.glTranslatef(0.0, 2, -5)
```

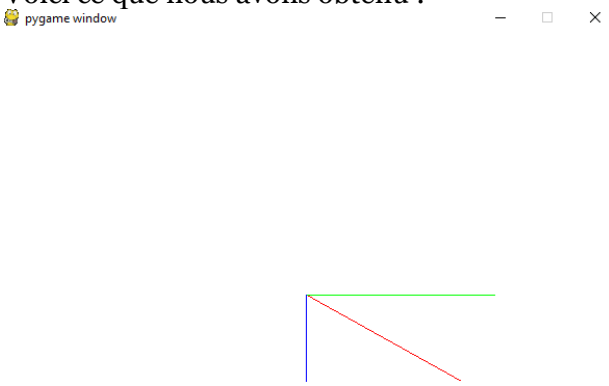
Voici ce que nous obtenons :



Nous avons fait la rotation. Nous avons fait une rotation de 120° autour de l'axe x. Voici notre code :

```
    # Placer ici l'utilisation de gluPerspective.
    glu.gluPerspective(45, (display[0] / display[1]), 0.1, 50.0)
    gl.glTranslatef(0.0, 2, -5)
    gl.glRotatef(-120, 1, 0, 0)
```

Voici ce que nous avons obtenu :



Découverte de l'environnement de travail

1a) Nous avons écrit le code suivant :

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

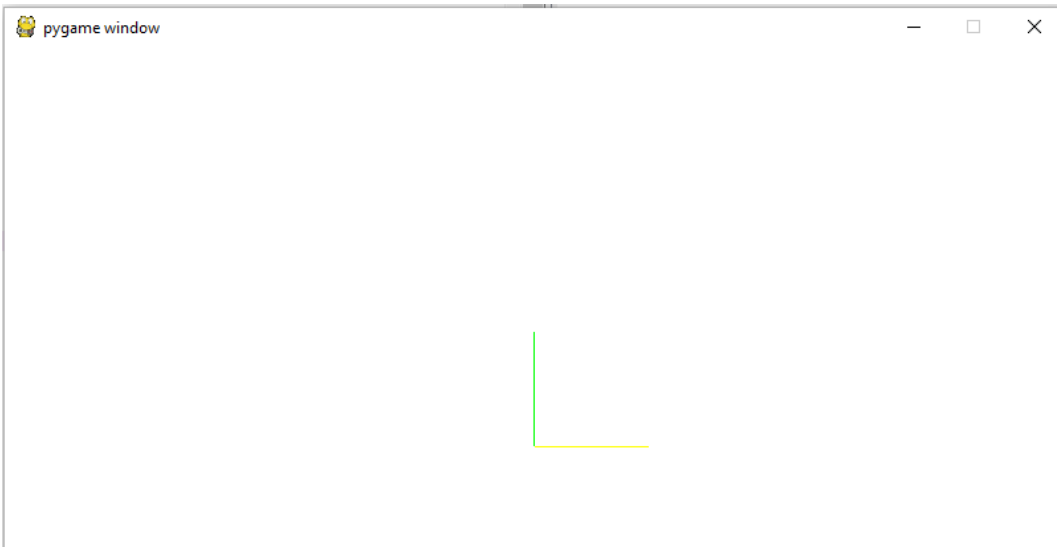
Nous remarquons que lorsque nous appuyons sur la touche a, le graphique disparaît, lorsque l'on réappuie sur la touche, le graphique réapparait. Lorsque l'on appuie sur la touche z, le graphique tourne dans le sens trigonométrique et sur la touche Z, le graphique tourne dans le sens horaire.

Le fichier main définit et utilise des configurations. Le fichier configuration crée l'environnement de l'affichage : crée les paramètres, peut ajouter des objets…
Les deux fichiers sont composés de fonctions.

1b)

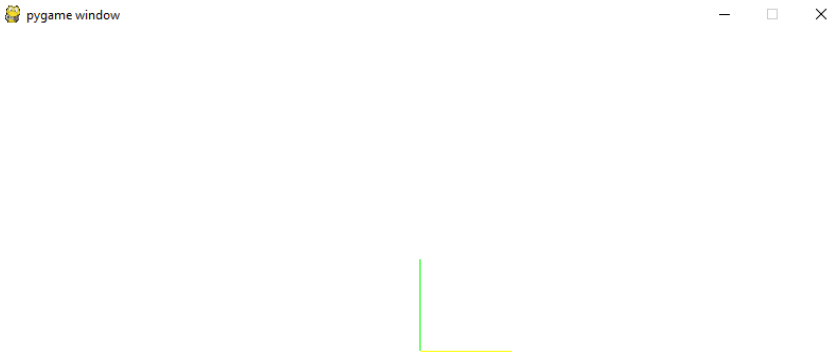
```
21 def Q1b_f():
22     return Configuration({'screenPosition': -5, 'xAxisColor': [1, 1, 0]}).display()
23
```

Ce code permet de modifier la couleur d'un axe :



```
def Q1b_f():
    return Configuration({'screenPosition': -5, 'xAxisColor': [1, 1, 0]}). \
        setParameter('xAxisColor', [1, 1, 0]). \
        setParameter('yAxisCo lor', [0,1,1]). \
        display()

```

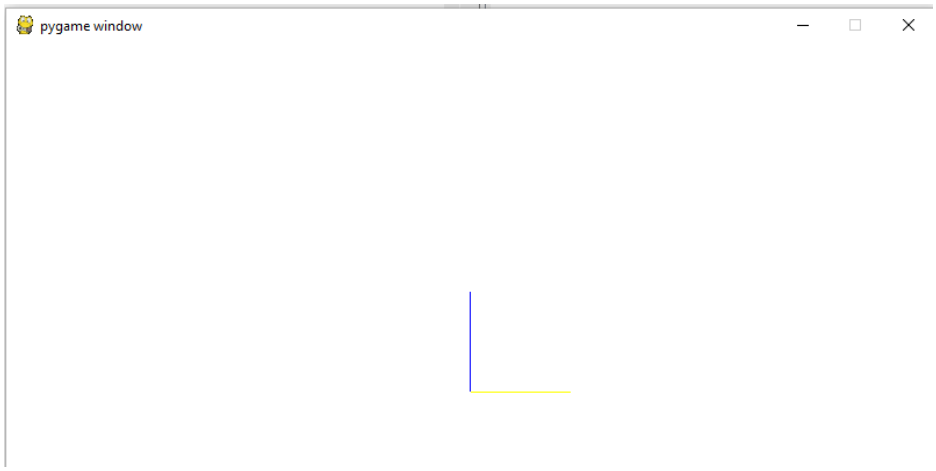
Ce code permet également de modifier la couleur d'un axe :



1c) 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 :

```
# Initializes the tranformation matrix
def initializeTransformationMatrix(self):
    gl.glMatrixMode(gl.GL_PROJECTION)
    gl.glLoadIdentity()
    glu.gluPerspective(70, (self.screen.get_width()/self.screen.get_height()), 0.1, 100.0)

    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 :

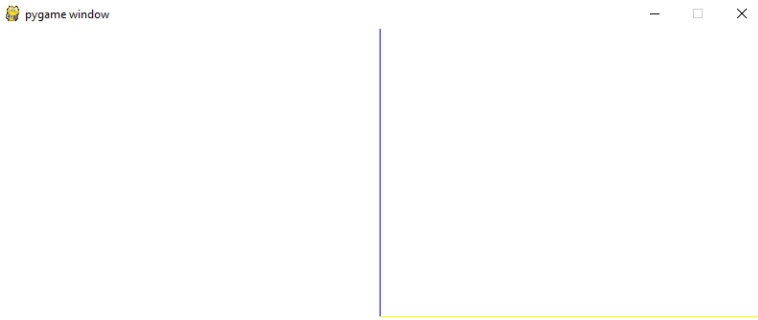


III. Mise en place des interactions avec l'utilisateur avec Pygame

1d) Nous avons ajouté la gestion des touches « Page Up », « Page Down » pour, respectivement, grossir ou réduire l'affichage. Pour cela, 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 notre code :

```
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)
```

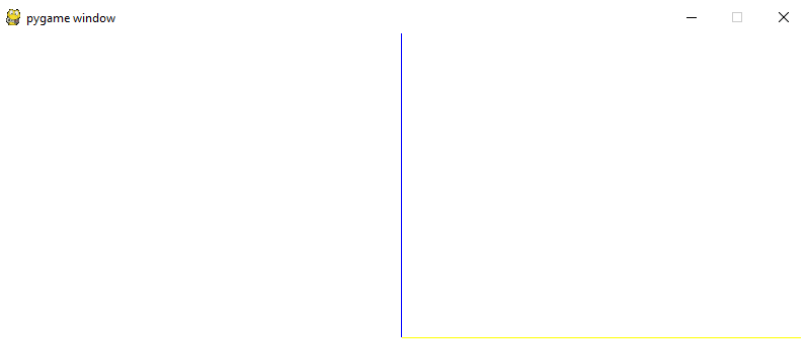
Voici ce que nous obtenons :



1e) Nous avons écrit le code permettant de zoomer avec la roulette de la souris. 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 notre code :

```
# 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)
```

Notre fonction fonctionne correctement, voici un exemple d'axe que nous pouvons obtenir :



1f) Nous avons écrit la fonction qui permet de gérer le déplacement d'objets. Voici notre code :

```
# Processes the MOUSEMOTION event
def processMouseMotionEvent(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)
```

Nous pouvons déplacer les axes, notre code fonctionne correctement :

IV. Création d'une section

2a) Nous avons écrit la fonction generate qui permet de créer les sommets et les faces d'une section orientée selon l'axe x et dont le coin bas gauche de la face externe est en (0, 0, 0). Voici notre code :

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

    self.faces = [
            [0,3,2,1],
            [4,7,6,5],
            [0,4,5,1],
            [3,7,6,2],
            [0,4,7,3],
            [1,5,6,2]
            ]
```

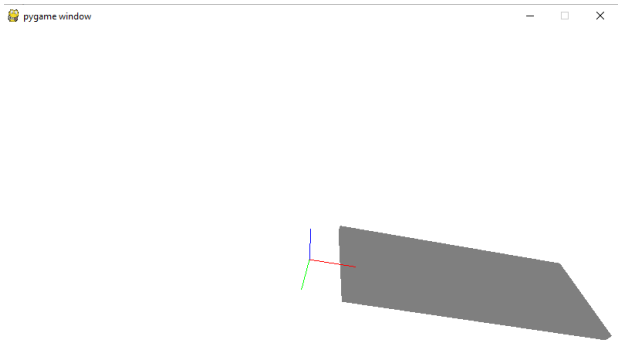
Nous avons défini les coordonnées des sommets en utilisant les paramètres width, height et thickness.

2b)La méthode Q2b du main permet d'ajouter dans Configuration une section qui se trouve à la position (1,1,0), de largeur 7 et de hauteur 2,6.
L'instruction « Configuration().add(section).display() » permet d'afficher la section.

Nous avons écrit la méthode draw qui permet de dessiner une section. Voici notre code :

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

    for face in self.faces:
        gl.glPushMatrix()
        gl.glTranslate(self.parameters['position'][0],self.parameters['position'][1],self.parameters['position'][2])
        gl.glRotate(self.parameters['orientation'],0,0,1)

        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[face[0]])
        gl.glVertex3fv(self.vertices[face[1]])
        gl.glVertex3fv(self.vertices[face[2]])
        gl.glVertex3fv(self.vertices[face[3]])
        gl.glEnd()
        gl.glPopMatrix()
```

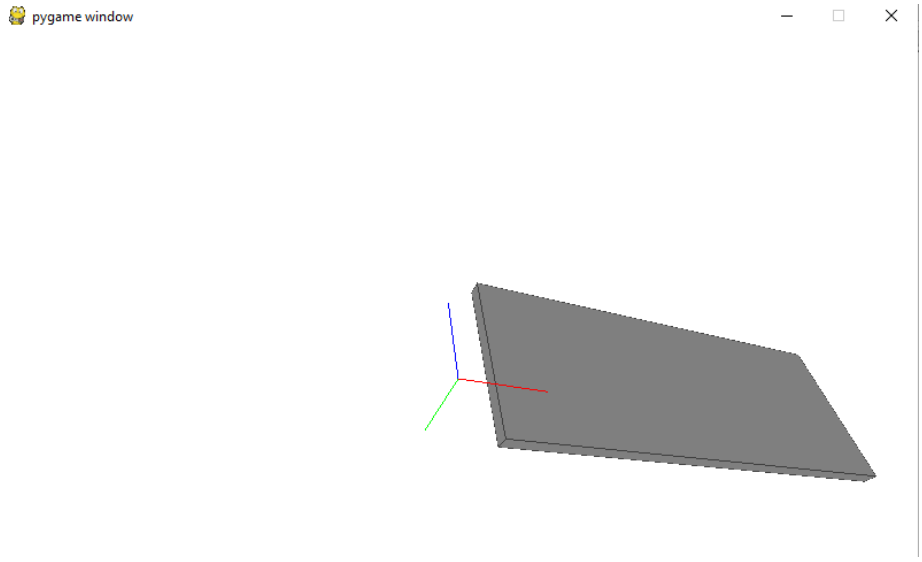
Voici ce que nous obtenons lorsque nous exécutons la fonction Q2b :



2c) Nous avons écrit la méthode drawEdge qui permet de dessiner les arêtes de la section. Pour cela, nous avons également modifié la méthode draw afin que la fonction drawEdge soit exécutée en premier. Voici notre code :

```python
# Draws the edges
def drawEdges(self):
    gl.glPushMatrix()
    gl.glTranslate(self.parameters['position'][0],self.parameters['position'][1],self.parameters['position'][2])
    gl.glRotate(self.parameters['orientation'],0,0,1)
    gl.glPolygonMode(gl.GL_FRONT_AND_BACK, gl.GL_LINE)
    for i in self.faces:
        gl.glBegin(gl.GL_QUADS)
        gl.glColor3fv([self.parameters['color'][0]*0.5,self.parameters['color'][1]*0.5,self.parameters['color'][2]*0.5])
        for j in i:
            gl.glVertex3fv(self.vertices[j])
        gl.glEnd()
    gl.glPopMatrix()



# Draws the faces
def draw(self):
    if self.parameters['edges']:
        self.drawEdges()


    for face in self.faces:
        gl.glPushMatrix()
        gl.glTranslate(self.parameters['position'][0],self.parameters['position'][1],self.parameters['position'][2])
        gl.glRotate(self.parameters['orientation'],0,0,1)

        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[face[0]])
        gl.glVertex3fv(self.vertices[face[1]])
        gl.glVertex3fv(self.vertices[face[2]])
        gl.glVertex3fv(self.vertices[face[3]])
        gl.glEnd()
        gl.glPopMatrix()
```

Voici ce que nous obtenons lorsque nous exécutons la fonction Q2c() du fichier main.py :



Nous avons effectué le test section, nous n'avons pas d'erreur :

```
----------------------------------------------------------------------
Ran 9 tests in 0.004s

OK
An exception has occurred, use %tb to see the full traceback.

SystemExit: False

C:\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3304: UserWarning: To exit: use 'exit',
'quit', or Ctrl-D.
  warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)

In [5]:

In [5]:
```

V. Création des murs

3a) Dans le constructeur du fichier Wall, il y a une série de tests qui vérifient si les paramètres de parameters sont bien présents. S'il en manque, des valeurs seront fixées.

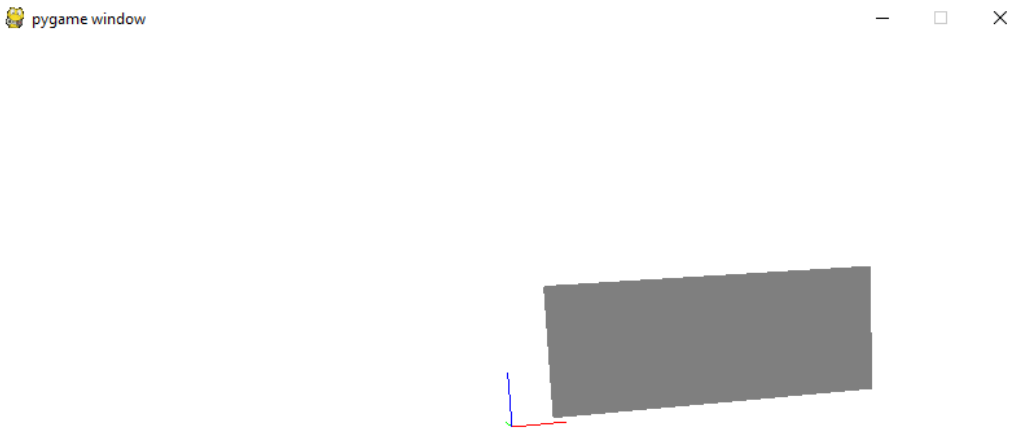
Nous avons la méthode draw dans le fichier wall. Elle permet de dessiner le mur avec ses arêtes. Voici notre code :

```
# Draws the faces
def draw(self):
    gl.glPushMatrix()
    gl.glRotate(self.parameters['orientation'],0,0,1)
    for i in self.objects:
        i.draw()
    gl.glPopMatrix()
```

Nous avons enfin écrit la fonction Q3a dans le main :

```
def Q3a():
    return Configuration().add(Wall({'position': [1, 1, 0], 'width':7, 'height':2.6, 'edges': True}))
```

Voici ce que nous obtenons :



VI. Création d'une maison

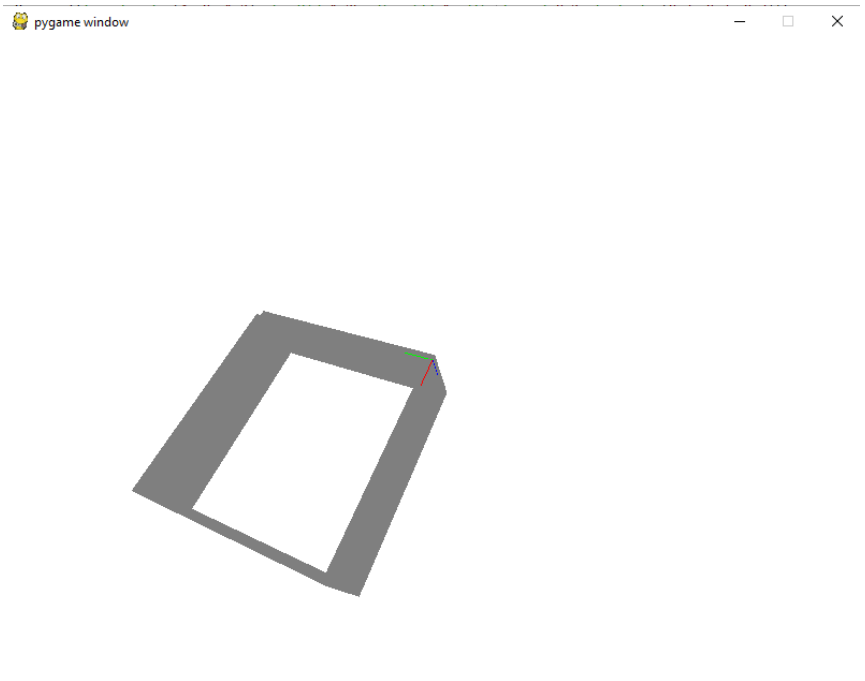
Nous avons écrit la méthode draw dans la classe House. Voici notre code :

```
# Draws the house
def draw(self):
    gl.glPushMatrix()
    gl.glTranslate(self.parameters['position'][0], self.parameters['position'][1], self.parameters['position'][2])
    gl.glRotate(self.parameters['orientation'],0,0,1)
    for i in self.objects:
        i.draw()
    gl.glPopMatrix()
```

Nous avons également écrit la méthode Q4a :

```
def Q4a():
    # Ecriture en utilisant des variables : A compléter
    wall1 = Wall({'position': [0, 0, 0], 'width':7, 'height':2.6, 'edges': True})
    wall2 = Wall({'position': [0, 0, 0], 'width':7, 'height':2.6, 'edges': True, 'orientation':90})
    wall3 = Wall({'position': [0, 7, 0], 'width':7, 'height':2.6, 'edges': True})
    wall4 = Wall({'position': [0, -7, 0], 'width':7, 'height':2.6, 'edges': True, 'orientation':90})
    house = House()
    house.add(wall1).add(wall3).add(wall4).add(wall2)
    return Configuration().add(house)
```

Voici ce que nous obtenons :

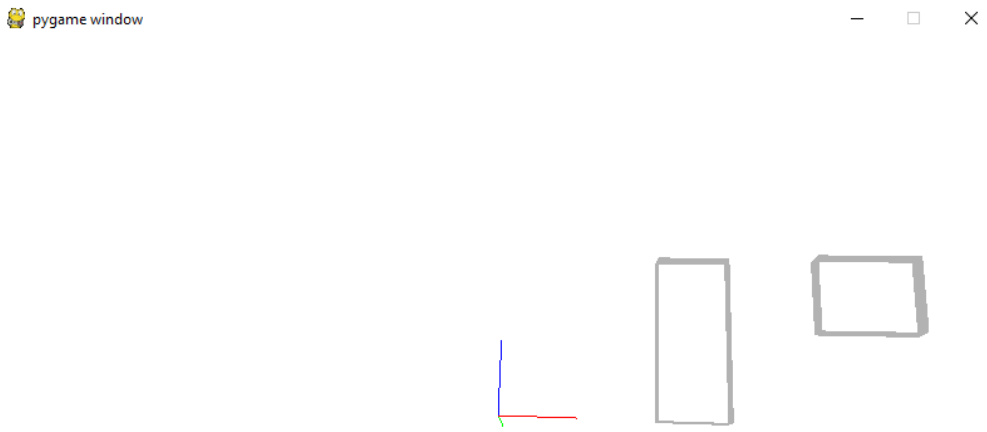


VII. Création d'ouvertures

Nous avons complété la classe opening :

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

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

# Draws the faces
def draw(self):
    gl.glPushMatrix()
    gl.glTranslate(self.parameters['position'][0],self.parameters['position'][1],self.parameters['position'][2])
    gl.glPolygonMode(gl.GL_FRONT_AND_BACK, gl.GL_FILL) # on trace les faces : GL_FILL
    for i in self.faces:
        gl.glBegin(gl.GL_QUADS) # TracÃ© dâ€™un quadrilatÃ¨re
        gl.glColor3fv(self.parameters['color']) # Couleur gris moyen
        for j in i:
            gl.glVertex3fv(self.vertices[j])
        gl.glEnd()
    gl.glPopMatrix()
```

Voici ce que nous obtenons :



VIII. Conclusion

Ce tp nous a permis de découvrir une nouvelle utilisation de python. C'était intéressant de voir que l'on peut créer des objets grâce à du code. Malheureusement, nous n'avons pas eu le temps de finir le tp et donc de construire une maison complète.