COMMAND Valentin
LAURET Victor
MM-TDB
TP3 INFO 501

TP3 - Représentation visuelle d'objets

II - Préparation à faire avant le TP

Utilisation de Pygame

1) Les 4 lignes permettent de créer une fenêtre graphique de 300 par 200 qui s'affiche et disparaît instantanément.
2) Création d'une fenêtre graphique qui se ferme lorsque nous appuyons sur une touche quelconque du clavier.

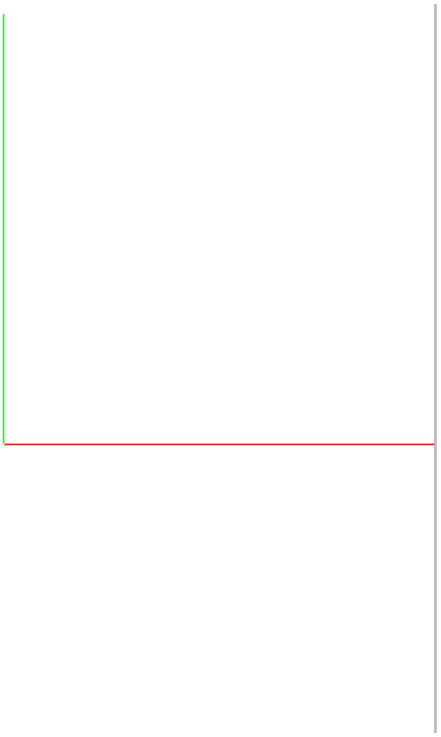
Utilisation de Pyopengl pour représenter des objets 3D

2) Nous nous sommes aidé des fonctions ainsi que de l'exemple dans le sujet de TP afin de tracer les axes suivant x, y et z avec une couleur rouge, vert et bleu. Ci-dessous le code qui nous avons exécuté :

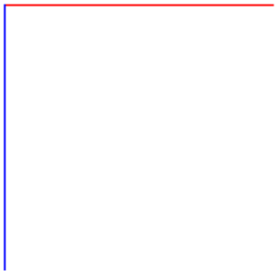
```python
import pygame
import OpenGL.GL as gl
import OpenGL.GLU as glu

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)
    glu.gluPerspective(45,(display[0] / display[1]),0.1,50)
    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([255, 0, 0]) # Indique la couleur du prochian 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, 255, 0]) # Indique la couleur du prochian 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.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((0, 0, -3)) # 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()
```

COMMAND Valentin
LAURET Victor
MM-TDB
TP3 INFO 501

Voici le résultat du code, nous avons obtenu un repère orthogonal que nous voyons dans le plan (x, y)



3) Dans cette question nous avons exécuté les lignes de codes correspondant à la question 3 ce qui qui a donc pour effet, la translation du repère de 2 suivant l'axe y et de -5 suivant l'axe z mais également la rotation du repère suivant l'axe x de 90°. Nous remarquons que l'axe y est maintenant l'axe sortant de l'écran .Voici ce que nous observons sur la fenêtre graphique:



COMMAND Valentin
LAURET Victor
MM-TDB
TP3 INFO 501

Découverte de l'environnement du travail du TP

1)

a) Dans la méthode processKeyDownEvent(self) nous voyons que lorsque nous tapons sur la touche "z" nous effectuons une rotation autour de l'axe z dans le sens anti-horaire grâce au signe "-" devant la valeur 2.5 , le "Z" nous permet de faire une rotation toujours autour de l'axe z dans le sens horaire. La touche "a" a pour but d'effacer l'affichage des axes.

b) Le chaînage de l'appel des méthodes setParameter() et display() sont possibles car ces méthodes retournent un objet de type configuration. Le traitement particulier sur screenPosition doit être effectué si l'on souhaite voir le résultat de notre fonction.

Nous avons exécuté le code fourni dans le sujet nous permettant de changer la couleur des axes.

c) Nous avons rajouté la ligne en surbrillance gl.glRotatef(-90,1,0,0) qui permet donc de faire une rotation qui nous permet d'afficher l'axe z verticalement et l'axe y en profondeur.

```
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)
```

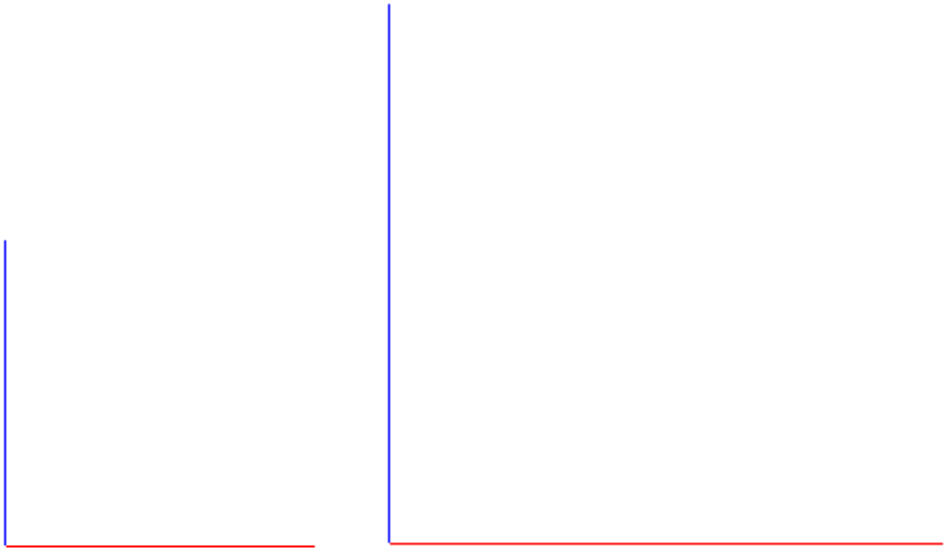
COMMAND Valentin
LAURET Victor
MM-TDB
TP3 INFO 501

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

d) Nous avons ajouté à la méthode processKeyDownEvent deux conditions afin de pouvoir effectuer un zoom positif et négatif à l'aide des touches pageup et pagedown du clavier. Pour cela nous avons utilisé la méthode glScalef qui était donnée dans le TP. Voici ci-dessous le code correspondant:

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

    # Draws or suppresses the reference frame
    elif self.event.dict['unicode'] == 'a' or self.event.key == pygame.K_a:
        self.parameters['axes'] = not self.parameters['axes']
        pygame.time.wait(300)
    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 ci-dessous deux exemples, la première image représente ce que nous voyons de base lors de l'exécution, la deuxième image est un zoom positif avec la touche pageup et la troisième est un zoom négatif avec la touche pagedown



Base

Agrandi

Réduit

COMMAND Valentin
LAURET Victor
MM-TDB
TP3 INFO 501

e) Dans cette question nous nous sommes inspirés de la question précédente pour faire un zoom positif et négatif à l'aide de la molette de la souris.

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

f) Nous voulons effectuer une rotation lorsque nous maintenons le clic gauche de la souris et une translation lorsque nous maintenons le clic droit. Pour ce faire nous avons testé lequel des boutons de la souris était enfoncé. Nous avons relevé le déplacement effectué par celle-ci pour effectuer une rotation ou une translation. Dans notre programme nous pouvons voir que nous avons calculé un angle lorsque le clic gauche est maintenu et ceci nous permet de faire une rotation qui a pour valeur cet angle. Nous avons mis un coefficient dans la translation car nous avons jugé que le déplacement était trop rapide sans ce coefficient.

```
# Processes the MOUSEMOTION event
def processMouseMotionEvent(self):
    if pygame.mouse.get_pressed()[0]==1:
        x=self.event.rel[0]
        y=self.event.rel[1]
        t=math.atan2(y,x)
        gl.glRotate(t, 1, 0, 1)
    elif pygame.mouse.get_pressed()[2]==1:
        x=self.event.rel[0]
        y=self.event.rel[1]
        gl.glTranslatef(0.01*x,0,0.01*y)
```

COMMAND Valentin
LAURET Victor
MM-TDB
TP3 INFO 501

IV - Création d'une section

2)

a)

Nous nous sommes inspirés du fichier Configuration.py pour écrire notre méthode generate(self) dans notre classe Section. Cette méthode est divisée en deux parties le self.vertices, permet de créer les 8 sommets d'une section qui est un parallélépipède. La seconde partie permet de créer les faces de ce parallélépipède avec les points que nous avons créés précédemment.

```
# 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'],0],
            [self.parameters['width'],self.parameters['thickness'],self.parameters['height']]
            ]
    self.faces = [
            [0, 3, 2, 1],
            [1,2,7,5],
            [4,6,7,5],
            [0,3,6,4],
            [0,4,5,1],
            [3,6,7,2]
            ]
```

b)

Les codes ci-dessous nous ont permis de créer une section ayant pour origine le sommet situé en bas à gauche de la section en (1,1,0) de longueur 7 et de hauteur 2.6.
glPushMatrix() et glPopMatrix() permettent de sauvegarder et de récupérer une matrice.
A l'aide de l'exemple du sujet nous avons pu déduire le reste du code pour créer une section.

COMMAND Valentin
LAURET Victor
MM-TDB
TP3 INFO 501

```python
# Draws the faces
def draw(self):
    # A compléter en remplaçant pass par votre code


    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)
    for i in self.faces :
        gl.glBegin(gl.GL_QUADS)
        gl.glColor3fv([0.5,0.5,0.5])
        for j in i:
            gl.glVertex3fv(self.vertices[j])
        gl.glEnd()
    gl.glPopMatrix()
```

```python
def Q2b():
    # Ecriture en utilisant le chaînage
    return Configuration().add(
            Section({'position': [1, 1, 0], 'width':7, 'height':2.6})
            )
```

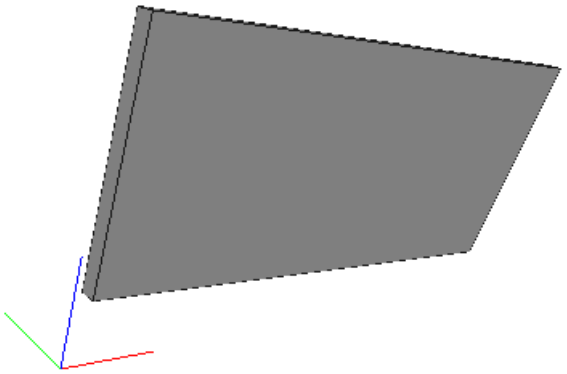
c) Dans cette question nous avons réutilisé le même raisonnement que précédemment pour dessiner les arêtes de nos sections. Nous avons choisi une couleur plus foncée pour les arêtes pour mieux les distinguer. Le code est pratiquement le même, nous avons juste changé un argument de PolygonMode pour créer une ligne à la place d'une face. Nous avons ajouté self.drawEdges() dans la méthode draw(self) pour que drawEdges(self) s'effectue avant draw(self).

COMMAND Valentin
LAURET Victor
MM-TDB
TP3 INFO 501

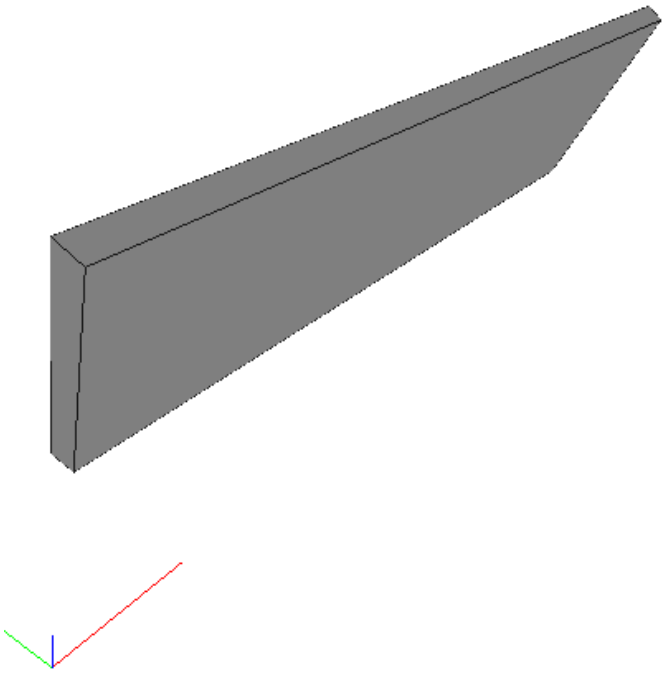
POLYTECH ANNECY-CHAMBÉRY

UNIVERSITÉ SAVOIE MONT BLANC

```python
# Draws the edges
def drawEdges(self):
    # A compléter en remplaçant pass par votre code
    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_LINE)
    for i in self.faces :
        gl.glBegin(gl.GL_QUADS)
        gl.glColor3fv([0.1,0.1,0.1])
        for j in i:
            gl.glVertex3fv(self.vertices[j])
        gl.glEnd()
    gl.glPopMatrix()

# Draws the faces
def draw(self):
    # A compléter en remplaçant pass par votre code
    self.drawEdges()
    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)
    for i in self.faces :
        gl.glBegin(gl.GL_QUADS)
        gl.glColor3fv([0.5,0.5,0.5])
        for j in i:
            gl.glVertex3fv(self.vertices[j])
        gl.glEnd()
    gl.glPopMatrix()
```



COMMAND Valentin
LAURET Victor
MM-TDB
TP3 INFO 501



V - Création des murs

3)

a) Dans cette question nous nous sommes inspirés de la méthode que nous avons réalisée à la question précédente. Nous pouvons voir quelques changements au niveau de la boucle for, ici nous n'avons pas directement accès aux faces comme à la question précédente, nous sommes donc passé par parentSection qui contient ainsi la face. Mise à part ce changement nous sommes sur le même code.

```
# Draws the faces
def draw(self):
    # A compléter en remplaçant pass par votre code
    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.parentSection.faces:
        gl.glBegin(gl.GL_QUADS)
        gl.glColor3fv([0.1,0.1,0.1])
        for j in i:
            gl.glVertex3fv(self.parentSection.vertices[j])
        gl.glEnd()
    gl.glPolygonMode(gl.GL_FRONT_AND_BACK, gl.GL_FILL)
    for k in self.parentSection.faces:
        gl.glBegin(gl.GL_QUADS)
        gl.glColor3fv([0.5,0.5,0.5])
        for l in k:
            gl.glVertex3fv(self.parentSection.vertices[l])
        gl.glEnd()
    gl.glPopMatrix()
```

COMMAND Valentin
LAURET Victor
MM-TDB
TP3 INFO 501

```
def Q3a():
    return Configuration().add(
            Wall({'position': [1, 0, 0], 'width':5, 'height':5})
            )
```

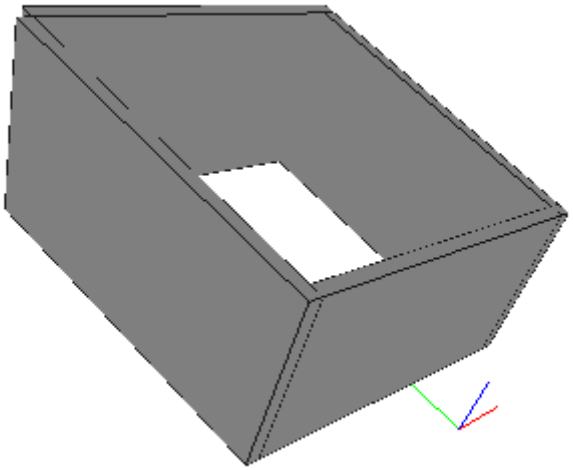
VI - Création d'une maison

Nous nous sommes inspirés de la méthode draw() plus particulièrement de la fonction for x in self.objects: x.draw() de la classe Configuration() pour afficher plusieurs objets en même temps. Ici nous avons pu afficher les 4 murs de la maison.

```
# Draws the house
def draw(self):
    # A compléter en remplaçant pass par votre code
    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()
```

COMMAND Valentin
LAURET Victor
MM-TDB
TP3 INFO 501

```python
def Q4a():
    # Ecriture en utilisant des variables : A compléter
    wall1 = Wall({'position': [0, 0, 0], 'width':5, 'height':2.6, 'edges': True, 'orientation':0})
    wall2 = Wall({'position': [5, 0, 0], 'width':5, 'height':2.6, 'edges': True,'orientation':90})
    wall3 = Wall({'position': [0, 0, 0], 'width':5, 'height':2.6, 'edges': True,'orientation':90})
    wall4 = Wall({'position': [0, 5, 0], 'width':5, 'height':2.6, 'edges': True,'orientation':0})
    house = House({'position': [-3, 1, 0], 'orientation':0})
    house.add(wall1).add(wall3).add(wall4).add(wall2)
    return Configuration().add(house)
```



Pour conclure ce TP nous a appris à faire une représentation 3D d'un objet dans une fenêtre graphique. Il nous a également appris à se déplacer dans l'espace à l'aide de notre clavier et de notre souris. Malheureusement, nous n'avons pas pu finir la dernière partie de ce TP qui consistait à faire des ouvertures afin de pouvoir mettre des portes et des fenêtres.