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 :

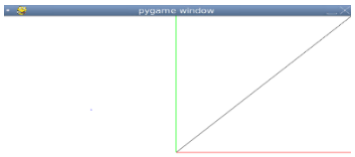
```
import pygame
import OpenGL.GL as gl
import OpenGL.GLU as glu

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 :



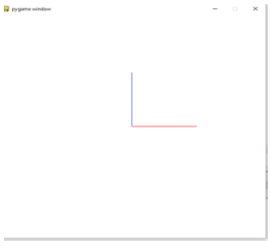
Nous avons en rouge l'axe x, en vert l'axe y, en bleu l'axe z normal au plan (Oxy) et en noir, l'axe défini dans l'énoncé

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 :

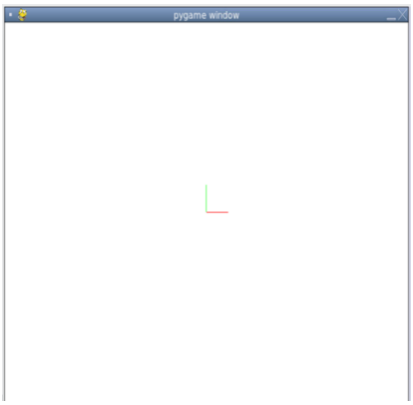


6. Question (1a). Découverte de l'environnement du travail du TP

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.

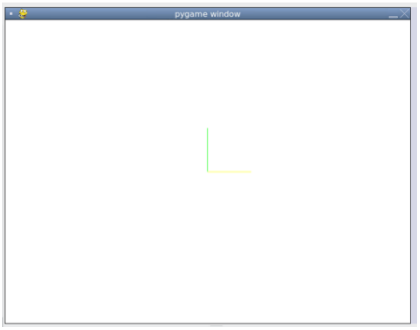
7. Question (1b).

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.

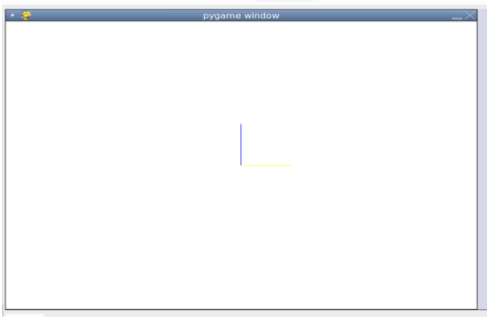


8. Question (1c).

Il nous est demandé d'ajouter une instruction dans la méthode initializeTransformationMatrix() de sorte que l'axe z soit vertical 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

9. Question (1d)

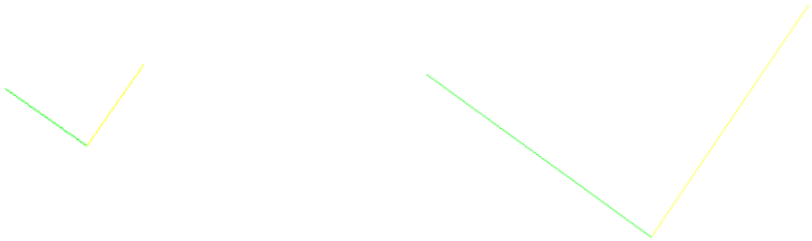
Il nous est demandé d'écrire une méthode processKeyDownEvent qui permet d'effectuer des rotations de +-2,5 degrés autour de l'axe z et de pouvoir grossir ou réduire l'affichage. Le facteur d'échelle demandé pour un zoom positif de 1.1 et un zoom négatif de 1/1.1.

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 :



10. Question (1e)

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.

11. Question (1f)

Dans cette question, il nous est demandé d'écrire la méthode processMouseMotionEvent() qui permet d'effectuer des rotations aux objets via le clic gauche de la souris ainsi que des translations via le clic droit de la souris tout en bougeant la souris.

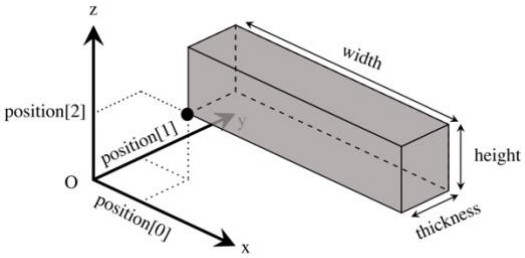
```
# Processes the MOUSEMOTION event
def processMouseMotionEvent(self):
    if pygame.mouse.get_pressed()[0] == 1:
        gl.glRotatef(self.event.rel[0], 0, 0, 1)
        gl.glRotatef(self.event.rel[1], 1, 0, 0)
    elif pygame.mouse.get_pressed()[2] == 1:
        gl.glTranslatef(self.event.rel[0]/10,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

IV. Création d'une section

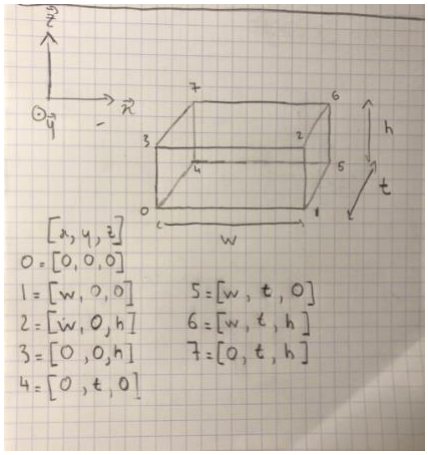
12. Question (2a)

On s'intéresse ici à la création d'une section.

On a un parallélépipède à 8 sommets et 6 faces.
On nous demande d'écrire la méthode generate(self) de la classe Section.



Afin de rentrer les coordonnées des points pour permettre au programme de reconnaître les sommets et faces, nous avons au préalable effectuer un schéma au brouillon.




```
# Defines the vertices and faces
def generate(self):
    self.vertices = [
            [0,0,0],
            [self.parameters['width'],0,0],
            [self.parameters['width'],0,self.parameters['height']],
            [0,0,self.parameters['height']],
            [0,self.parameters['thickness'],0],
            [self.parameters['width'],self.parameters['thickness'],0],
            [self.parameters['width'],self.parameters['thickness'],self.parameters['height']],
            [0,self.parameters['thickness'],self.parameters['height']],
            ]


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

13. Question (2b)

Dans cette question, on nous demande dans un premier temps d'analyser la question (2b) et de décrire ce que fait cette commande.

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

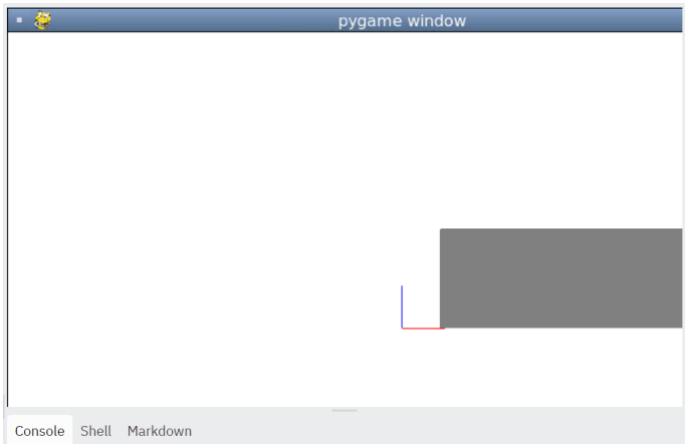
Cette commande permet d'ajouter une section de largeur (width) 7 et de hauteur (height) 2,6 en partant du point d'origine [1,1,0].

Ensuite, on nous demande d'écrire la méthode **draw()** pour la classe Section afin de tracer les faces de la section en gris.

La méthode **draw()** s'écrit :

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

    self.drawEdges()
    gl.glPushMatrix()
    gl.glTranslatef(self.parameters['position'][0],self.parameters
    ['position'][1],self.parameters['position'][2])
    gl.glRotatef(self.parameters['orientation'],0,0,1)

    gl.glPolygonMode(gl.GL_FRONT_AND_BACK,gl.GL_FILL)
    gl.glBegin(gl.GL_QUADS)
    gl.glColor3fv(self.parameters['color'])

    for k in self.faces:
      for j in k:
        gl.glVertex3fv(self.vertices[j])

    gl.glEnd()
    gl.glPopMatrix()
```

En exécutant la classe main en activant la question 2b, la fenêtre nous affiche ceci :



14. Question (2c)

Dans cette question, il nous est demandé dans un premier temps d'écrire la méthode drawEdges() dans la classe Section. Cette méthode sera aussi incluse dans la méthode draw()

```
# Draws the edges
def drawEdges(self):

    def draw(self):

      self.drawEdges()
    gl.glPushMatrix()
    gl.glTranslatef(self.parameters['position'][0],self.parameters
    ['position'][1],self.parameters['position'][2])
    gl.glRotatef(self.parameters['orientation'],0,0,1)

    gl.glPolygonMode(gl.GL_FRONT_AND_BACK,gl.GL_LINE)
    gl.glBegin(gl.GL_QUADS)
    gl.glColor3fv(self.parameters['edges'])

    for k in self.faces:
      for j in k:
        gl.glVertex3fv(self.vertices[j])

    gl.glEnd()
    gl.glPopMatrix()
```

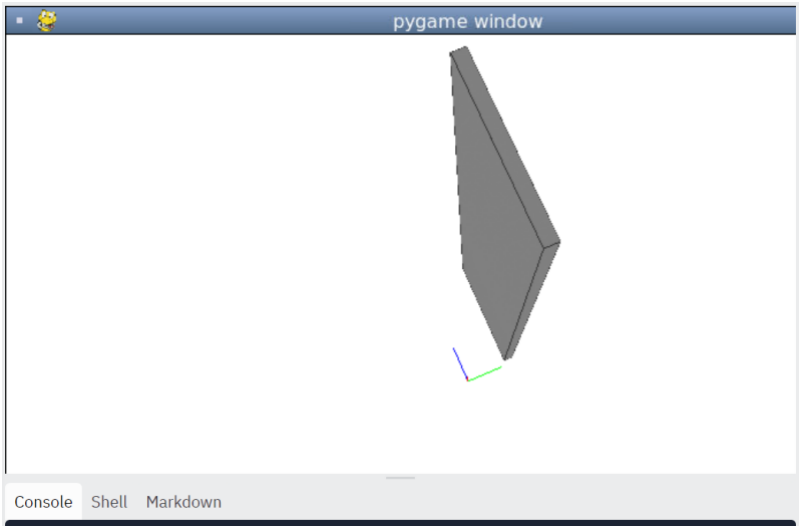
Afin d'inclure cette méthode dans la méthode draw(), on a modifié la méthode :

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

    gl.glPushMatrix()
    gl.glTranslatef(self.parameters['position'][0],
    self.parameters['position'][1],self.parameters
    ['position'][2])
    gl.glRotatef(self.parameters['orientation'],0,0,
    1)
    self.drawEdges()
    for i in self.faces:
      gl.glPolygonMode(gl.GL_FRONT_AND_BACK,
      gl.GL_FILL)
      gl.glBegin(gl.GL_QUADS)
      gl.glColor3fv(self.parameters['color'])
      for k in self.faces:
        for j in k:
          gl.glVertex3fv(self.vertices[j])

      gl.glEnd()
    gl.glPopMatrix()
```

Pour la couleur des arrêtes, nous avons choisi de les assombrir par un gris plus foncé de code couleur :

```
if 'color' not in self.parameters:
    self.parameters['color'] = [0.5, 0.5, 0.5]
if 'edges' not in self.parameters:
    self.parameters['edges'] = [0.2, 0.2, 0.2]
```

En exécutant la question 2c dans la classe **main()**, et en faisant une certaine rotation pour bien faire apparaitre notre parallélépipède, la fenêtre affiche :



V. Création des murs

15. Question (3a)

Dans cette question, il nous est demandé d'analyser le fichier wall.py

```
class Wall:
    # Constructor
    def __init__(self, parameters = {}) :
        # Parameters
        # position: position of the wall
        # width: width of the wall - mandatory
        # height: height of the wall - mandatory
        # thickness: thickness of the wall
        # color: color of the wall

        # Sets the parameters
        self.parameters = parameters

        # Sets the default parameters
        if 'position' not in self.parameters:
            self.parameters['position'] = [0, 0, 0]
        if 'width' not in self.parameters:
            raise Exception('Parameter "width" required.')
        if 'height' not in self.parameters:
            raise Exception('Parameter "height" required.')
        if 'orientation' not in self.parameters:
            self.parameters['orientation'] = 0
        if 'thickness' not in self.parameters:
            self.parameters['thickness'] = 0.2
        if 'color' not in self.parameters:
            self.parameters['color'] = [0.5, 0.5, 0.5]
```

Cette commande permet de créer un mur. Elle va créer ce mur en rentrant des valeurs dans la classe. Par ailleurs, ces valeurs doivent vérifier des conditions programmées dans la commande.

Dans un second temps, on nous demande d'écrire une suite d'instructions dans la méthode draw de la classe Wall, elle permettra de dessiner le mur avec ses arêtes.

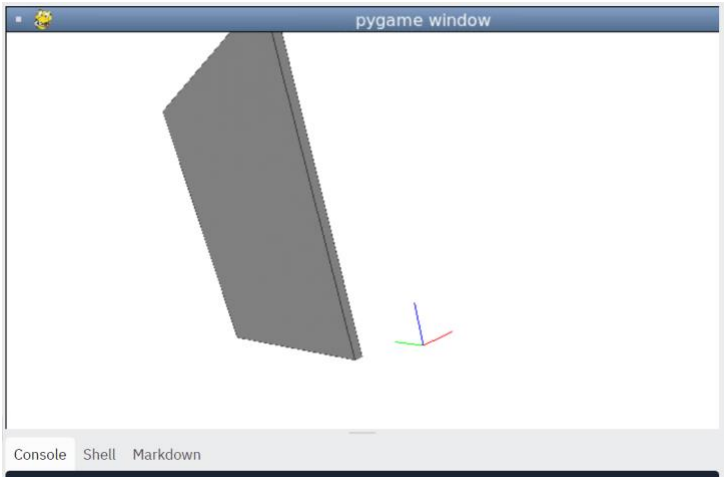
La méthode **draw()** de la classe **Wall** s'écrit :

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

    gl.glPushMatrix()
    gl.glRotatef(self.parameters['orientation'],0,0,1)
    for objet in self.objects:
     objet.draw()
    gl.glPopMatrix()
```

Puis enfin, on nous demande d'écrire des instructions pour créer un mur en lien avec notre section.

```
def Q3a():
    return Configuration().add(Wall({'position': [1, 1, 0],'width':7,
    'height':7,'orientation':90}))  #créé un mur de taille 7x7x0,2
    orienté de 90° d'angle
```

Dans la classe main, en activant la question 3a, la fenêtre affiche :



VI. Création d'une maison

16. Question (4a)

Nous allons maintenant utiliser la classe House.
Dans cette question, il nous est demandé d'écrire la méthode draw(), similaire à celle de la classe Configuration.

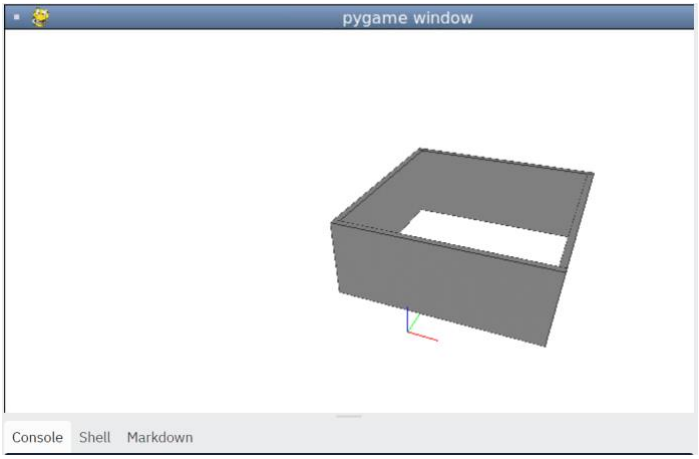
La méthode **draw()** s'écrit :

```
# Draws the house
def draw(self):
    # A compléter en remplaçant pass par votre code
    gl.glPushMatrix()
    gl.glTranslatef(self.parameters['position'][0],
    self.parameters['position'][1], self.parameters
    ['orientation'],0,0,1)

    for objet in self.objects:
      objet.draw()
    gl.glPopMatrix()
```

Dans un second temps, on nous demande de modifier la question 4a du fichier Main.py afin de créer une maison.

```
def Q4a():
    # Ecriture en utilisant des variables : A compléter
    wall1 = Wall({'position':[0,0,0], 'width':7, 'height':2.6})
    wall2 = Wall({'position':[0,7,0], 'width':7, 'height':2.6,
    'orientation':0})
    wall3 = Wall({'position':[0.2,-7,0], 'width':7, 'height':2.6,
    'orientation':90})
    wall4 = Wall({'position':[0.2,-0.2,0], 'width':7, 'height':2.6,
    'orientation':90})
    house = House({'position': [-3, 1, 0], 'orientation':0})
    house.add(wall1).add(wall3).add(wall4).add(wall2)
    return Configuration().add(house)
```

En exécutant la classe main, la fenêtre nous affiche :



Nos 4 murs sont bien positionnés sans débordements ou intersections.

VII. Création d'ouvertures

17. Question (5a)

Dans cette question, on nous demande de compléter la classe Opening.

```
# Draws the faces
def draw(self):
    # A compléter en remplaçant pass par votre code
    gl.glPushMatrix()
    gl.glRotatef(self.parameters['orientation'],0,0,1)

    gl.glPopMatrix()
```

Dans un second temps, on nous demande d'exécuter le fichier Main.py avec la fonction de la question 5a afin d'afficher une image représentant la figure de l'énoncé.

Malheureusement, je n'ai pas réussi à afficher les figures avec ma machine personnelle que ce soit sur Spyder ou Replit.

19. Question (5b)

Dans cette question on nous demande d'écrire une méthode qui retourne True lorsqu'une ouverture représentée par x peut être incluse dans la section et retourne False si cela n'est pas possible.

...

VIII. Conclusion

Afin de conclure notre étude, nous pouvons dire que ce TP a été très instructif, regroupant de nombreuses fonctions permettant de renforcer nos connaissances. Nous avons pu apprendre à créer de nombreux objets ou encore à les pivoter en assignant des commandes.