HUMBERT Ophélie
GERNIGANT Elsa
Date du TP (17/01/2022)

Rapport de TP3 (séance2)– Représentation visuelle d'objets

Introduction

Pour débuter cette seconde partie de TP, nous devions commencer à réellement définir des objets dans la maison. Pour mieux appréhender des termes nouveaux à manipuler, la création des différents objets est différenciée en plusieurs partie, de cette façon on commence par créer une simple section puis des murs puis les ouvertures (porte et fenêtre) pour finir par obtenir une maison. A la fin, il est également possible de dupliquer cette maison.
Le but de ce TP est de construire une maison en 3D et de pouvoir la visualiser dans l'espace.

V. Section 1 du TP

1. Question (3a).

```
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]

        # Objects list
        self.objects = []

        # Adds a Section for this object
        self.parentSection = Section({'width': self.parameters['width'], \
                                      'height': self.parameters['height'], \
                                      'thickness': self.parameters['thickness'], \
                                      'color': self.parameters['color'],
                                      'position': self.parameters['position']})
        self.objects.append(self.parentSection)
```

- On construit dans cet onglet, la classe Wall.

Tout d'abord on initialise les paramètres d'entrée qui sont « self » et « parameters = {} qui est un dictionnaire.

Pour commencer le code, s'il n'y a pas de position d'entrée, on lui donne par défaut la valeur [0,0,0].

Si « width » n'est pas déjà dans les paramètres alors on nous demande de rentrer le paramètre « width ».
La même commande est écrite pour le paramètre « height ».
Le paramètre « orientation » prend par défaut la valeur 0 également si la valeur n'est pas donnée précédemment en paramètre.
Le paramètre « thickness » a une valeur attribuée de 0.2 si la valeur n'est pas donnée précédemment en paramètre.
Le paramètre « color» a une valeur attribuée de [0.5,0.5,0.5] si la valeur n'est pas donnée précédemment en paramètre.

On initialise Self.objects comme une liste vide.

Self.parentSection prend en valeur les valeurs données dans la classe Section.

Pour finir self.objects prend les valeurs de self.parentSection

```
def draw(self):

    gl.glPushMatrix()
    gl.glRotate(self.parameters["orientation"],0,0,1)
    for x in self.objects:
        x.drawEdges()
        x.draw()

    gl.glPopMatrix()
```

- On crée une autre matrice grâce à « glPushMatrix puis on effectue une rotation sur la section et pour finir on appelle l'objet de draw().
  Puis on renvoie la matrice.

- A l'instar de l'instruction de la question Q2c dans le main, nous avons créé dans la fonction Q3a du fichier Main.py l'instruction pour créer un mur.

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

VI. Création d'une maison

Nous avons ajouté une fonction « glPushMatrix » pour copier une nouvelle matrice puis « glPopMatrix » pour la retourner.
Nous avons fait tourner la section en fonction du paramètre « orientation » grâce à Rotate.

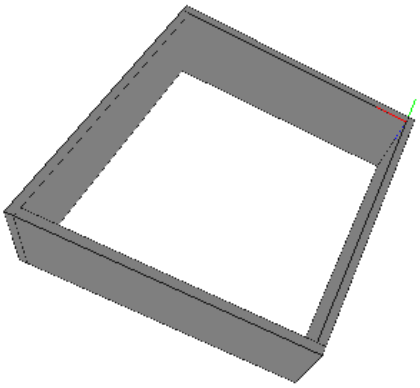
```
def draw(self):

    gl.glPushMatrix()
    gl.glRotate(self.parameters["orientation"],0,0,1)
    for x in self.objects:
        x.draw()

    gl.glPopMatrix()
```

Nous avons ajouté une fonction translate dans la Section.

```
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)
```

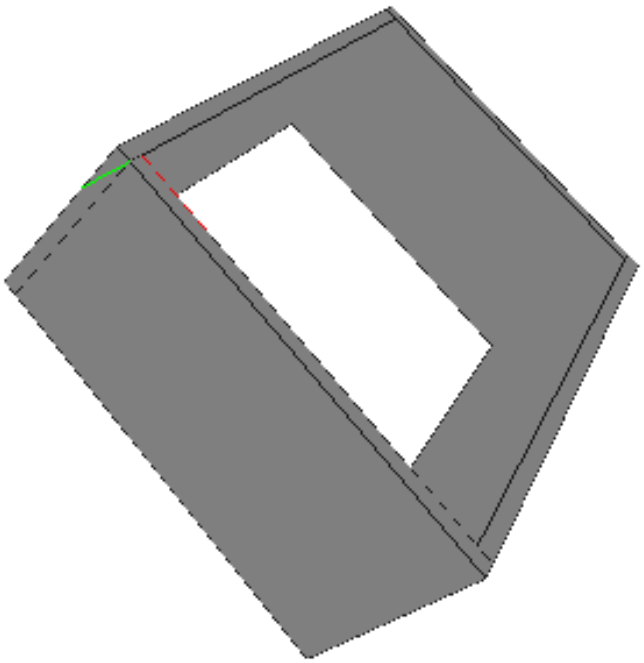
Puis nous avons donné des coordonnées pour placer les différents murs dans l'onglet « Main ».

```
def Q4a():
    # Ecriture en utilisant des variables : A compléter
    wall1 = Wall({'position': [0, 0, 0], 'width':7, 'height':2.6, 'thickness': 0.2})
    wall2 = Wall({'position': [-6.8, 6.8, 0], 'width':7, 'height':2.6, 'thickness': 0.2, 'orientation': 180})
    wall3 = Wall({'position': [-6.8, 0, 0], 'width':7, 'height':2.6, 'thickness': 0.2, 'orientation': 90})
    wall4 = Wall({'position': [0, 6.8, 0], 'width':7, 'height':2.6, 'thickness': 0.2, 'orientation': -90})
    house = House({'position': [-3, 1, 0], 'orientation':0})
    house.add(wall1).add(wall3).add(wall4).add(wall2)
    return Configuration().add(house)
```



Après nous avons transformé le carré obtenu en rectangle. Premièrement en changeant la valeur de « width » de deux cotés opposés puis en modifiant les positions afin de coller les rectangles.

```
def Q4a():
    # Ecriture en utilisant des variables : A compléter
    wall1 = Wall({'position': [0, 0, 0], 'width':5, 'height':2.6, 'thickness': 0.2})
    wall2 = Wall({'position': [-4.8, 6.8, 0], 'width':5, 'height':2.6, 'thickness': 0.2, 'orientation': 180})
    wall3 = Wall({'position': [-6.8, 0, 0], 'width':7, 'height':2.6, 'thickness': 0.2, 'orientation': 90})
    wall4 = Wall({'position': [0, 4.8, 0], 'width':7, 'height':2.6, 'thickness': 0.2, 'orientation': -90})
    house = House({'position': [-3, 1, 0], 'orientation':0})
    house.add(wall1).add(wall3).add(wall4).add(wall2)
    return Configuration().add(house)
```



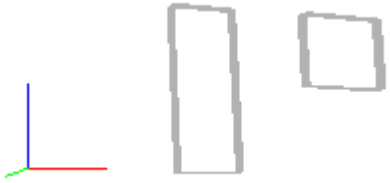
VII. Création d'ouvertures

1. Question(5a)

Nous nous sommes inspirées de ce que nous avions fait dans la Class Section pour réaliser cette partie. Cependant on a retiré les deux faces qu'il est a en moins ici.

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

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)
    gl.glBegin(gl.GL_QUADS) # Tracé d'un quadrilatère
    gl.glColor3fv(self.parameters['color'])
    gl.glVertex3fv([0, 0, 0])
    gl.glVertex3fv([0, 0, self.parameters['height']])
    gl.glVertex3fv([0, self.parameters['thickness'], self.parameters['height']])
    gl.glVertex3fv([0, self.parameters['thickness'],0])
    gl.glEnd()

    gl.glPolygonMode(gl.GL_FRONT_AND_BACK, gl.GL_FILL)
    gl.glBegin(gl.GL_QUADS) # Tracé d'un quadrilatère
    gl.glColor3fv(self.parameters['color'])
    gl.glVertex3fv([0, 0, self.parameters['height']])
    gl.glVertex3fv([0, self.parameters['thickness'], self.parameters['height']])
    gl.glVertex3fv([self.parameters['width'], self.parameters['thickness'], self.parameters['height']])
    gl.glVertex3fv([self.parameters['width'], 0, self.parameters['height']])
    gl.glEnd()

    gl.glPolygonMode(gl.GL_FRONT_AND_BACK, gl.GL_FILL)
    gl.glBegin(gl.GL_QUADS) # Tracé d'un quadrilatère
    gl.glColor3fv(self.parameters['color'])
    gl.glVertex3fv([self.parameters['width'], self.parameters['thickness'], self.parameters['height']])
    gl.glVertex3fv( [self.parameters['width'], 0, self.parameters['height']])
    gl.glVertex3fv([self.parameters['width'], 0, 0])
    gl.glVertex3fv( [self.parameters['width'], self.parameters['thickness'],0])
    gl.glEnd()

    gl.glPolygonMode(gl.GL_FRONT_AND_BACK, gl.GL_FILL)
    gl.glBegin(gl.GL_QUADS) # Tracé d'un quadrilatère
    gl.glColor3fv(self.parameters['color'])
    gl.glVertex3fv([self.parameters['width'], self.parameters['thickness'],0])
    gl.glVertex3fv( [0, self.parameters['thickness'],0])
    gl.glVertex3fv([0, 0, 0])
    gl.glVertex3fv([self.parameters['width'], 0, 0])
    gl.glEnd()
    gl.glPopMatrix()
```

Voici ce qu'on a obtenue :



2. Question (5b)

Une ouverture peut être possible si et seulement si tous ces points appartiennent à la section. C'est pourquoi dans le code ci-dessus on contrôle premièrement que les coordonnés du point donné dans les paramètres soient dans l'intervalles des coordonnées du mur. C'est-à-dire entre 0 et 7 sur l'axe des x, entre 0 et 2.15 sur l'axe des z et soit égale à 0 sur l'axe des y.
Et enfin, on vérifie que le reste des valeurs est dans le bon intervalle (voir le code ci-dessous).
Le code renverra True si l'ouverture répond à toutes les contraintes.

```
def canCreateOpening(self, x):
    return x.parameters['position'][0] >= 0 and x.parameters['position'][0]< 7\
    and x.parameters['position'][0] + x.parameters['width'] >= 0 and x.parameters['position'][0] + x.parameters['width'] <self.parameters['width'] \
    and x.parameters['position'][1] == 0\
    and x.parameters['thickness']== self.parameters['thickness']\
    and x.parameters['position'][2] >= 0 and x.parameters['position'][2]<self.parameters['height']\
    and x.parameters['position'][2] + x.parameters['height'] >= 0 and x.parameters['position'][2] + x.parameters['height'] < self.parameters['height']
```

En exécutant le main on obtient :

```
In [127]: runfile('C:/Users/gernigae/Documents/tp3-representation-visuelle-d-objets-
eit_tp3_humbert_gernigant-main/src/Main.py', wdir='C:/Users/gernigae/Documents/tp3-
representation-visuelle-d-objets-eit_tp3_humbert_gernigant-main/src')
True
True
False
```

Le dernier affichage est False car pour l'opening3 1.7+1=2.7>2.6 (2.6 étant la hauteur de la section), donc l'ouverture n'est pas possible.

3. Question (5c)

```
def createNewSections(self, x):
    liste=[]
    s=0
    if x.parameters['position'][0]-self.parameters['position'][0]!=0:
        s=Section({'width':x.parameters['position'][0]-self.parameters['position'][0], 'height':2.6, 'thickness':0.2, 'position': self.parameters['position']})
        liste.append(s)
    if self.parameters['position'][1]-x.parameters['position'][1]!=0:
        s=Section({'width':x.parameters['width'], 'height':self.parameters['position'][1]-x.parameters['position'][1], 'thickness':0.2,\
                   'position':[x.parameters['position'][0]-self.parameters['position'][0],0,self.parameters['position'][1]-x.parameters['position'][1]] })
        liste.append(s)
    if x.parameters['position'][1]-self.parameters['position'][1]!=0:
        s=Section({'width':x.parameters['width'], 'height':x.parameters['position'][1]-self.parameters['position'][1], 'thickness':0.2,\
                   'position':[x.parameters['position'][0]-self.parameters['position'][0],0,x.parameters['position'][1]-self.parameters['position'][1]] })
        liste.append(s)
    if self.parameters['position'][0]-x.parameters['position'][0]!=0:
        s=Section({'width':self.parameters['position'][0]-x.parameters['position'][0], 'height':self.parameters['position'][2], 'thickness':0.2,\
                   'position':[self.parameters['position'][0]-x.parameters['position'][0],0,self.parameters['position'][2]] })
        liste.append(s)

    return liste
```

Pour cette question nous avons essayé de crée la fonction qui retourne la liste des sections engendrée par la création d'une ouverture. Pour cela nous avons ajouté des conditions pour que les sections existent c'est-à-dire il faut que leur largeur ou leur hauteur soit différent de 0. Ensuite nous avons crée la section grâce à la classe section en complétant les valeurs des paramètres en fonction des sections. Puis nous ajoutons cette section dans la liste « liste ». Enfin on retourne la liste « liste ».

VIII. Conclusion

Nous avons appris à modéliser différents objets en 3D avec python grâce à ce TP. Même si parfois les premiers essais n'étaient pas les bons, nous avons au fil du TP réussi à corriger nos erreurs.
Malheureusement le TP s'arrête à la question 5c, car nous n'avons pas eu le temps de trouver comment corriger notre erreur à cette question. Néanmoins nous avons quand déposé notre code pour montrer nos recherches.
Pour finir, ce TP nous a appris de nombreuses choses sur le codage avec les Class mais également à utiliser le module pygame. Nous savons maintenant comment créer une maison jusqu'au mur et au fenêtre grâce à Python.