PIK Théo
CAMUS Loïc
22/11/2021 & 07/12/2021

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. On a choisi la représentation de maisons à partir d'objets simples que l'on va construire progressivement comme les murs, les portes, les fenêtres…

II. Préparation

1. Utilisation de Pygame

Question (1) :

Il s'agit ici de copier-coller les lignes indiquées dans le sujet de TP.

```
import pygame
pygame.init() # permet d'initialiser le module pygame.
ecran = pygame.display.set_mode((300, 200)) # permet de définir une fenêtre dans laquelle seront visualisés les résultats du programme
pygame.quit() #permet de quitter le module pygame.
```

Lorsque l'on lance le programme une fenêtre s'ouvre puis se referme directement.

Question (2) :

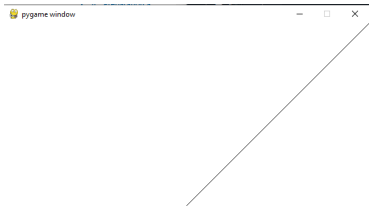
```
import pygame


pygame.init()
ecran = pygame.display.set_mode((300, 200)) # definition de la fenêtre.

continuer = True
while continuer:
    for event in pygame.event.get(): # boucle qui attend qu'on événement quelconque se produise
        if event.type == pygame.KEYDOWN:
            continuer = False  # si un événement se présente alors le programme se ferme.

pygame.quit()
```

Lorsqu'on lance le programme, une fenêtre pygame s'ouvre. Celle-ci se ferme lorsqu'on appuie sur n'importe quel bouton.

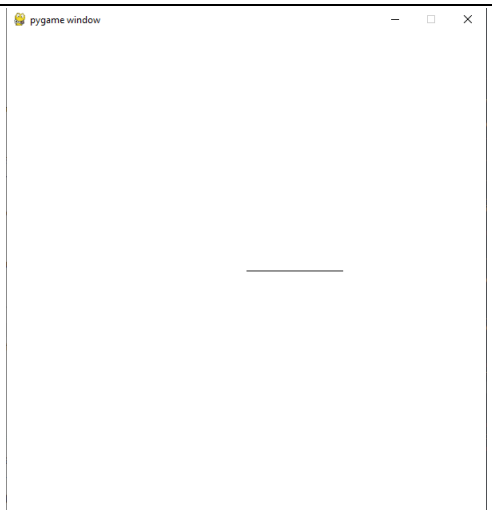
2. Utilisation de Pyopengl pour représenter des objets 3D

Question (1) et (2) :

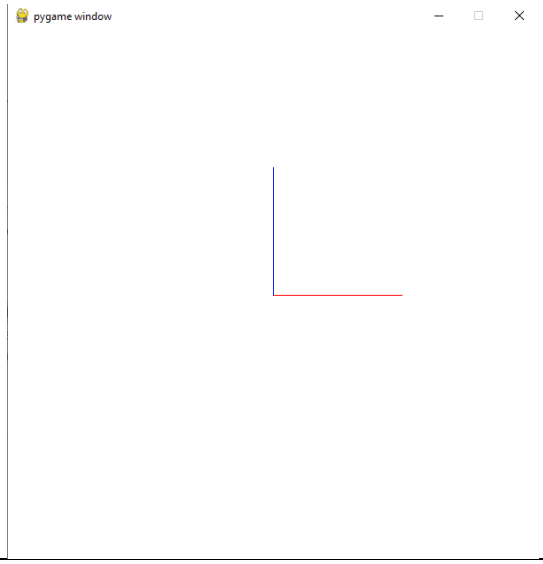
Il s'agit ici de copier-coller le programme dicté dans le sujet. En lançant le programme, le résultat suivant est obtenu : un trait :



Question (3)



Après avoir tracé les 3 axes x en rouge, y en vert et l'axe z en bleu, on obtient le résultat suivant :



Remarque : l'axe « y » n'apparaît pas, car nous avons fait une rotation de -90 °, de plus comme on est face à la figure, on ne voit pas le 3ème axe.

3. Découverte de l'environnement du travail du TP

Question (1).a) :

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

- Explication onglet « Configuration » :

Le fichier **Configuration.py** commence par dessiner une fenêtre de 800 par 800, ensuite, celui-ci permet d'afficher des axes de différentes couleur.

On obtient la fenêtre suivante :

Question (1).b) :

On obtient le résultat suivant :

Ensuite, après avoir modifié les paramètres à l'aide de la commande setParameter() on obtient :

On peut voir que 2 axes ont été tracés, un axe vert et un axe jaune, ces deux axes ont pour origine le centre de la fenêtre. Il est ensuite est possible à l'aide des boutons « z » et « Z » de procéder à la rotation des axes, et enfin le bouton « a », lui, permet de cacher ou de montrer ces derniers.

- Le chaînage de l'appel des méthodes **setParameter()** et **display()** est possible car la commande « setParameter » renvoie un objet de type « Configuration », c'est pourquoi nous pouvons aligner plusieurs « setParameter ».

- Ce traitement particulier doit être effectué car

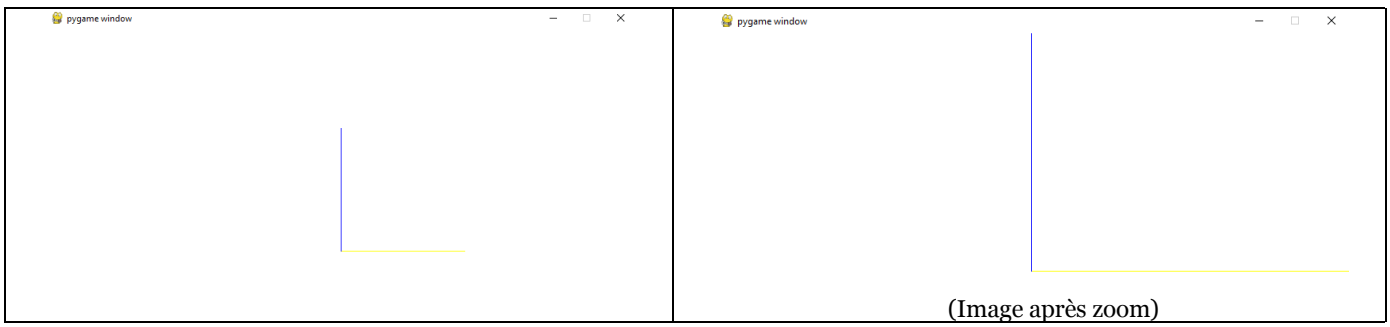
Question (1).c) :

```
69      # Initializes the tranformation matrix
70      def initializeTransformationMatrix(self):
71          gl.glMatrixMode(gl.GL_PROJECTION)
72          gl.glLoadIdentity()
73          glu.gluPerspective(70, (self.screen.get_width()/self.screen.get_height()), 0.1, 100.0)
74
75          gl.glMatrixMode(gl.GL_MODELVIEW)
76          gl.glLoadIdentity()
77          gl.glTranslatef(0.0,0.0, self.parameters['screenPosition'])
78          gl.glRotatef(-90, 1, 0, 0)
```

Pour que l'axe *x* soit représenté horizontalement et que l'axe *y* devienne alors la profondeur il faut rajouter la commande « gl.glRotatef(-90, 1, 0, 0) » (située à la ligne 78).

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

1. Question (1). d) :



(Image après zoom)

```
150         # Zoom:
151         if self.event.key == pygame.K_PAGEUP:
152             gl.glScalef(1.1,1.1,1.1)
153 
154         if self.event.key == pygame.K_PAGEDOWN:
155             gl.glScalef(1/1.1,1/1.1,1/1.1)
```

Cette fonction permet de zoomer et dézoomer notre figure selon les échelles 1.1 ou 1/1.1 (à l'aide du clavier).

2. Question (1). e) :

Le code est le suivant :

```
157     # Processes the MOUSEBUTTONDOWN event
158     def processMouseButtonDownEvent(self):
159        if self.event.button == 4:
160             gl.glScalef(1.1,1.1,1.1)
161 
162        if self.event.button == 5:
163             gl.glScalef(1/1.1,1/1.1,1/1.1)
164 
```

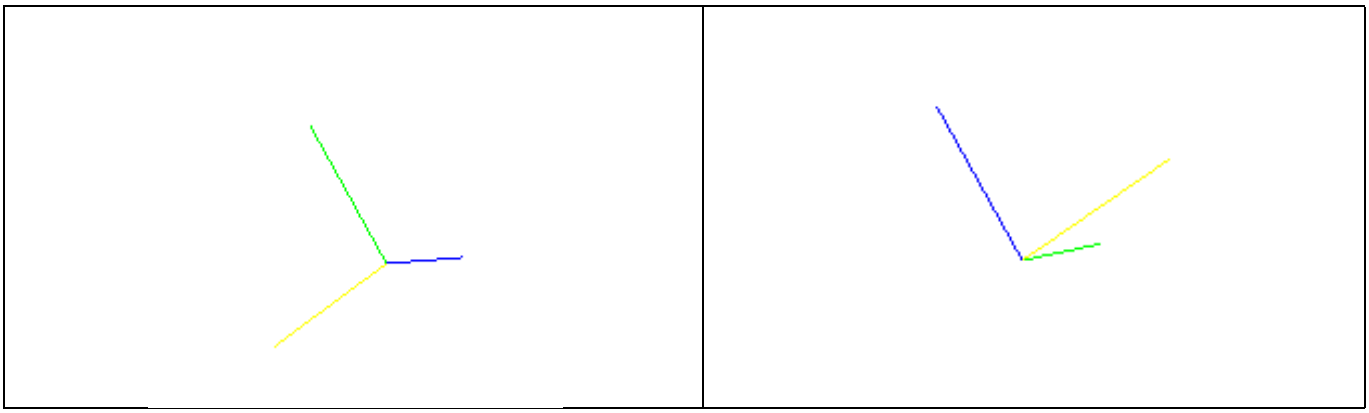
Le résultat est le même que le précédent mais cette fois-ci la molette de la souris est utilisée pour réaliser l'action.

3. Question (1). f) :

Il est ici question d'effectuer une rotation de l'objet à l'aide du bouton gauche de la souris et une translation avec le bouton droit de cette dernière.
Le code est le suivant :

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

    if pygame.mouse.get_pressed()[0]:
        gl.glRotate(self.event.rel[0], 0,0,1)
        gl.glRotate(self.event.rel[0],0,1,0)
```

Et on obtient le résultat suivant :



IV. Création d'une section

1. Question (2).a) :

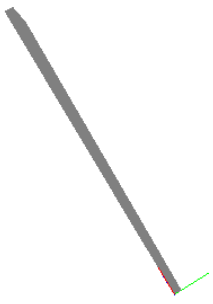
Ici, il est question de définir les paramètres de notre section, en l'occurrence ici, les sommets et les faces :

```
# Defines the vertices and faces
def generate(self):
    self.vertices = [
            # Définir ici les sommets

            [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 = [
            # définir ici les faces
             [0, 3, 2, 1],
             [2, 3, 6, 7],
             [6, 7, 5, 4],
             [0, 4, 5, 1],
             [1, 2, 7, 5],
             [0, 3, 6, 4]

            ]
```

2. Question(2).b) :

Par la suite, il est nécessaire de modéliser la section que nous avons définie précédemment à l'aide de la fonction suivante :

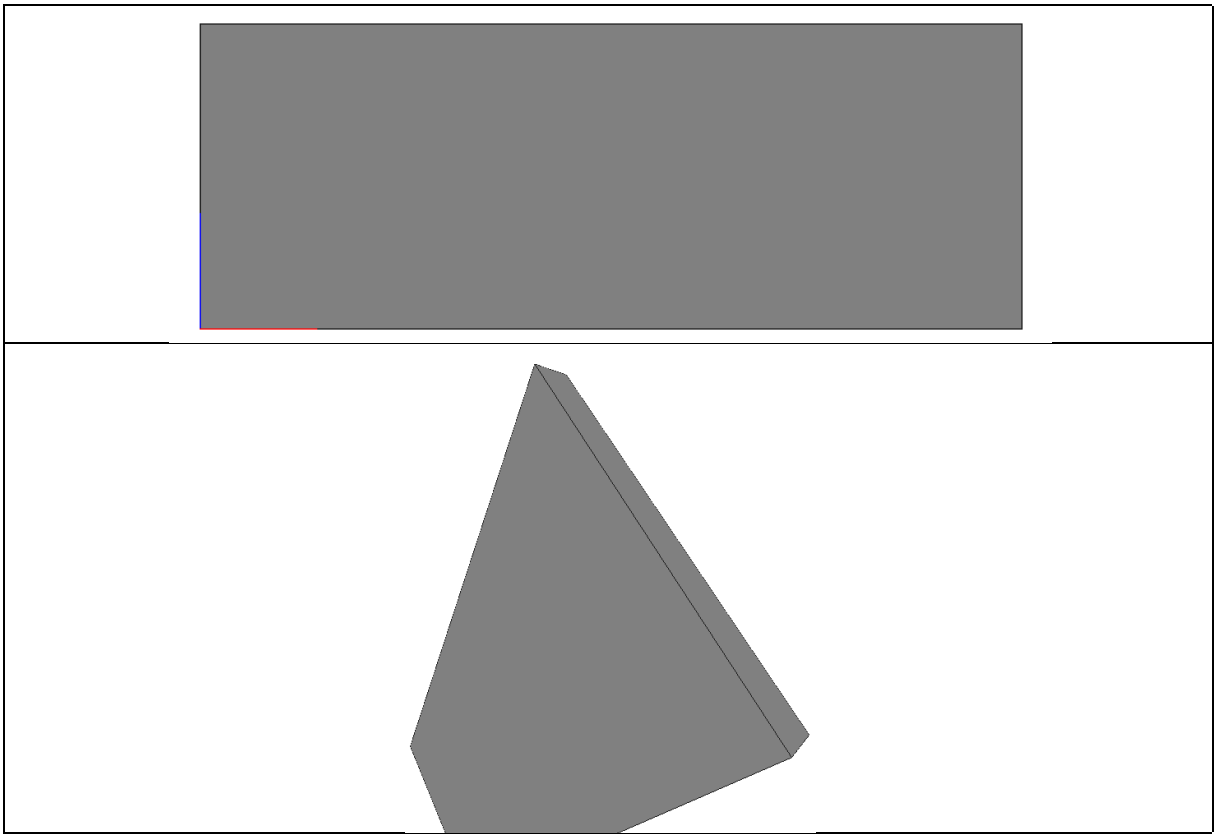
```
# Draws the faces
def draw(self):
    gl.glPushMatrix()
    # A compléter en remplaçant pass par votre code
    gl.glPolygonMode(gl.GL_FRONT_AND_BACK, gl.GL_FILL) # on trace les faces : GL_FILL
    gl.glBegin(gl.GL_QUADS) # Tracé d'un quadrilatère
    gl.glColor3fv([0.5, 0.5, 0.5]) # Couleur gris moyen
    for i in self.faces:
      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()
```



3. Question (2).c) :

Il s'agit ici de tracer les arrêtes de la section, elles seront ici affichées en noir comme l'indique le code suivant et on obtient les images suivantes :

```
# Draws the edges
def drawEdges(self):
    # A compléter en remplaçant pass par votre code
    gl.glPolygonMode(gl.GL_FRONT_AND_BACK, gl.GL_LINE) # on trace les arrêtes : GL_LINE
    # A compléter en remplaçant pass par votre code
    gl.glBegin(gl.GL_QUADS) # Tracé d'un quadrilatère
    gl.glColor3fv([0.1, 0.1, 0.1]) # Couleur gris moyen
    for i in self.faces:
      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()
```



V. Création des murs :

1. Question (3).a) :

Le code du fichier « Wall.py » reprend les différentes fonctions qui ont été utilisé dans la partie précédente. En effet le programme génère une section définie, avec :

```
# 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]
```

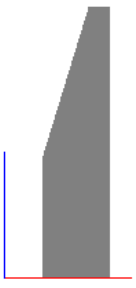
Pour commencer on a défini les fonctions qui permettent de gérer la translation et l'orientation autour de l'axe « z » du mur :

```
# Draws the faces
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['position'][2])
    gl.glRotatef(self.parameters['orientation'],0,0,1)
    for x in self.objects:
        x.draw()
    gl.glPopMatrix()
```

- Orientation de 90° :

Pour orienter le mur, on ajoute « orientation » avec une certaine valeur qui va permettre de définir l'orientation du mur autour de l'axe « z ».

```
def Q3a():
    return Configuration().add(
        Wall({'position': [1, 1, 0], 'width':7, 'height':2.6, 'orientation':90})
        )
```



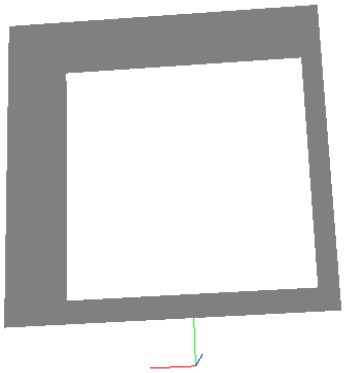
VI. Création d'une maison :

Pour définir les 4 murs, on fait comme à la question précédente, dans la fonction draw, on 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['position'][2])
     gl.glRotatef(self.parameters['orientation'],0,0,1)
     for x in self.objects:
         x.draw()
     gl.glPopMatrix()
```

Puis dans le **main**, dans la fonction « Q4a », on complète le code par :

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



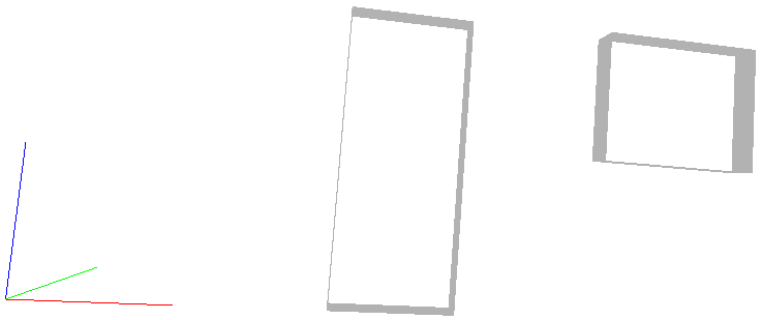
(Résultat obtenue)

Remarque : les murs font 7 de longueur mais il faut prendre en compte l'épaisseur du mur pour le dernier afin que ce dernier soit bien aligné.

VII. Création d'ouverture :

1. Question (5). a)

Il est ici question de visualiser deux ouvertures s'apparentant à une porte et une fenêtre comme illustré ci-dessous (voir programme à la suite) :



```python
    # Defines the vertices and faces
    def generate(self):
        self.vertices = [
                # Définir ici les sommets
                [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 = [
                # définir ici les faces
                 #[0, 3, 2, 1],
                 [2, 3, 6, 7],
                 #[6, 7, 5, 4],
                 [0, 4, 5, 1],
                 [1, 2, 7, 5],
                 [0, 3, 6, 4]
                ]

    # Draws the faces
    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['position'][2])
        # A compléter en remplaçant pass par votre code
        gl.glPolygonMode(gl.GL_FRONT_AND_BACK, gl.GL_FILL) # on trace les faces : GL_FILL
        gl.glBegin(gl.GL_QUADS) # Tracé d'un quadrilatère
        gl.glColor3fv(self.parameters['color']) # Couleur gris moyen


        for i in self.faces:
          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()
```

Remarque : cette fois-ci il a été seulement nécessaire de modéliser 4 faces de la section afin que celles-ci soient creusées.

2. Question (5). b)

```python
# Checks if the opening can be created for the object x
def canCreateOpening(self, x):
    # A compléter en remplaçant pass par votre code

    ouverture=True
    #sert à savoir si on peut placer l'ouverture dans la section
    if (self.parameters['height']>=x.getParameter('height')+x.getParameter('position')[2]-self.parameters['position'][2])==False:
        ouverture = False

    #sert à savoir si la position de l'ouverture ne dépasse pas à droite et à gauche
    if (x.getParameter('position')[2]>=self.parameters['position'][2])==False and (x.getParameter('position')[0]>=self.parameters['position'][0])==False:
        ouverture = False

    #sert à savoir si l'ouverture ne dépasse pas en hauteur de la section
    if (self.parameters['width']>=x.getParameter('width')+x.getParameter('position')[0]-self.parameters['position'][0])==False:
        ouverture=False

    return ouverture
```

```
IPdb [4]: runfile('D:/Théo/POLYTECH/INFO501/tp3-representation-visuelle-d-objets-tp3_pik_camus-main/src/Main.py', wdir='D:/Théo/POLYTECH/INFO501/tp3-representation-visuelle-d-objets-tp3_pik_camus-main/src')
True
True
False
```

Il y a donc ici la première ouverture possible, la deuxième ouverture possible et cependant pour la troisième ouverture elle est impossible car ne rentre pas dans la section.

VIII. Conclusion

Durant ce TP, il a été possible de découvrir la modélisation d'objets en 3D à l'aide du code Python. La présence de nombreuses listes complexes a augmenté notre vigilance car il a été nécessaire de bien organiser les données afin d'obtenir le résultat demandé. Nous n'avons malheureusement pas eu le temps de finaliser la réalisation de la maison en 3D cependant nous avons pu comprendre la méthodologie à appliquer pour la modélisation de cette dernière.