SCHERRER Mathieu
BARROUX Hugo
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. 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 à faire avant le TP

Utilisation de Pygame

1. Expliquer ces 4 lignes :

```
1  import pygame
2  pygame.init()
3  ecran = pygame.display.set_mode((300, 200))
4  pygame.quit()
```

Tout d'abord, on importe la librairie pygame. Ensuite, on initialise pygame et on configure l'affichage avec les dimensions 300 de large et 200 de haut et enfin on quitte l'affichage. Ainsi, on obtient une fenêtre qui s'ouvre et se referme instantanément.

2. Que ce passe-t-il avec les lignes suivantes ?

```
1  import pygame
2
3
4  pygame.init()
5  ecran = pygame.display.set_mode((300, 200))
6
7  continuer = True
8  while continuer:
9      for event in pygame.event.get():
10         if event.type == pygame.KEYDOWN:
11             continuer = False
12
13 pygame.quit()
```

Tout d'abord, on importe la librairie pygame. Ensuite, on initialise pygame et on configure l'affichage avec les dimensions 300 de large et 200 de haut. Puis, on crée une condition qui décrit que dès que l'on touche une touche du clavier, la fenêtre se ferme instantanément. Pygame.event.get gère les évènements entrants des différents appareils (clavier, souris). On défini avec Pygame.keydown qu'un clique sur le clavier déclenchera l'évènement : ici continuer passe à FALSE et l'intreface quitte pygame.

Utilisation de Pyopengl pour représenter des objets 3D

3. Utiliser la fonction gluPerspective (fovy, aspect, zNear, zFar).

```
# Placer ici l'utilisation de gluPerspective.
glu.gluPerspective (45, 1, 0.1, 50)
```

4. **Utiliser cet exemple pour tracer un segment pour chaque axe x, y et z en couleurs rouge, vert et bleu respectivement.**

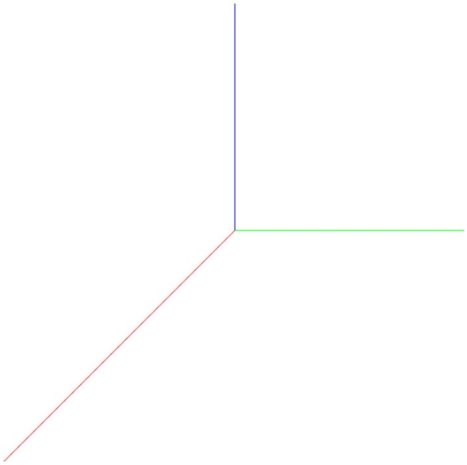
```
gl.glBegin(gl.GL_LINES) # Indique que l'on va commencer un trace en mode lignes (segments)

gl.glColor3fv([255, 0, 0]) # Indique la couleur du prochian segment en RGB
gl.glVertex3fv((0,0, -1)) # Premier vertice : départ de la ligne
gl.glVertex3fv((-1, -1, 0)) # Deuxième vertice : fin de la ligne

gl.glColor3fv([0, 255, 0]) # Indique la couleur du prochian segment en RGB
gl.glVertex3fv((0, 0, -1)) # Premier vertice : départ de la ligne
gl.glVertex3fv((1, 0, 0)) # Deuxième vertice : fin de la ligne

gl.glColor3fv([0, 0, 255]) # Indique la couleur du prochian segment en RGB
gl.glVertex3fv((0, 0, -1)) # Premier vertice : départ de la ligne
gl.glVertex3fv((0, 1, 0)) # Deuxième vertice : fin de la ligne
```

Voici l'affichage :

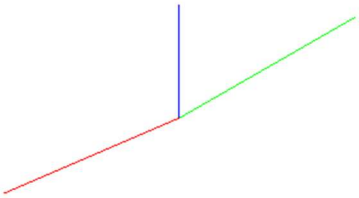


5. **Utiliser la fonction glTranslatef après la fonction gluPerspective pour faire une translation de l'affichage de z=−5 Et de y=2.**

On entre le code suivant :

```
glu.gluPerspective (45, 1, 0.1, 50)
gl.glTranslatef(0, 2, -5)
```

On obtient l'affichage qui suit :



On cherche ensuite, à faire une rotation de 90° autour de l'axe x. Voici comment nous avons utilisé glRotatef :

```
glu.gluPerspective (45, 1, 0.1, 50)
gl.glTranslatef(0, 2, -5)
gl.glRotatef(90, 1, 0, 0)
```

et on obtient :

Découverte de l'environnement du travail du TP

6. a) Ajouter à la fonction Q1a() la commande suivante :

La touche A fait disparaître le tracé.
La touche Z fait une rotation dans le sens anti-horaire.
On passe de à

La touche Z+MAJ fait une rotation dans le sens horaire :

Voici le code qui permet de réaliser cela dans configuration.py :

```
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 == pygame.K_z):
        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)
```

Pour la touche Z+MAJ : il a été défini que l'évènement d'appuie sur la touche Z est rattaché à la fonction glRotate qui fait une rotation autour de l'axe des x de -2.5° soit 2.5° dans le sens horaire.
Pour la touche Z : il a été défini que l'évènement d'appuie sur la touche Z est rattaché à la fonction glRotate qui fait une rotation autour de l'axe des x de 2.5° soit 2.5° dans le sens trigonométrique (anti-horaire).
Pour la touche a : il a été défini que l'évènement d'appuie sur la touche a renvoie l'inverse de l'état actuel du slef.parametre. Il disparait si les axes sont affichés, et inversement s'ils ne sont pas affichés.

b)

```
Configuration({'screenPosition': -5, 'xAxisColor': [1, 1, 0]}).display()
```

Analyser l'effet de cette modification.

Il y a simplement un changement de couleur de l'axe des x. A noter que .display() n'est pas nécessaire car il y a :

```
def main():
    # Enlever un des commentaires pour la question traitée

    #configuration = Q1a()
    configuration = Q1b_f()
    # configuration = Q2b()
    # configuration = Q2c()
    # configuration = Q3a()
    # configuration = Q4a()
    # configuration = Q5a()
    # configuration = Q5b()
    # configuration = Q5c1()
    # configuration = Q5c2()
    # configuration = Q5d()
    # configuration = Q6()
    configuration.display()
```

Pour le code suivant :

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

Cela nous affiche :

Pour le code suivant :

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

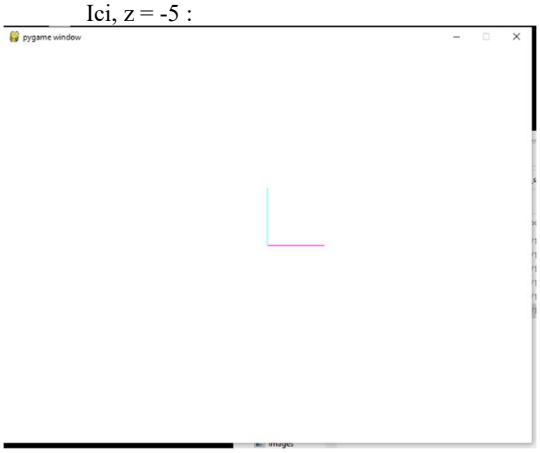
Cela nous affiche :

Expliquer pourquoi le chaînage de l'appel des méthodes setParameter() et display() est possible.

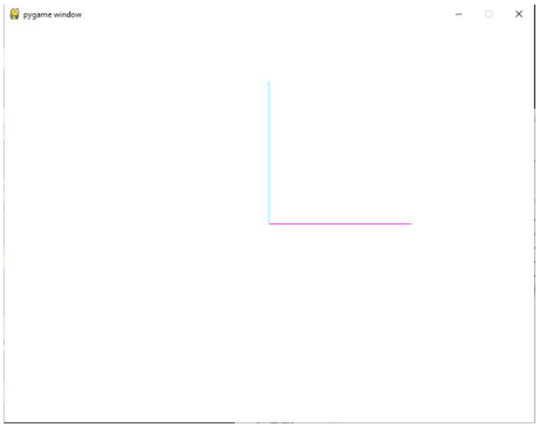
Le chaînage des méthodes setParameter() et display() est possible car ces méthodes retour self.

Un traitement particulier est effectué dans le « setter » pour le paramètre screenPosition. Expliquer pourquoi ce traitement particulier doit être effectué.

Ici, z = -5 :



z = -2



On note que plus z est grand dans les négatifs, plus la représentation est lointaine.
Ici, l'axe des x est rose, l'axe des y est cyan et l'axe des z est violets (que l'on ne voit pas).

c) Ajouter une seule instruction à la méthode initializeTransformationMatrix() pour que l'axe z soit représenté verticalement sur l'écran et que l'axe x soit représenté horizontalement.

On réalise une rotation autour de l'axe des x dans le sens horaire pour que l'axe des z devienne en position de l'axe des y à l'aide de la méthode glRotatef :

```
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)
```

On obtient l'affichage suivant :

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

1. Ajouter la gestion des touches « Page up » et « Page down ».

On va utiliser la méthode glScalef pour modifier l'échelle de 1.1 lorsque l'on veut zoomer positivement (la représentation va « grossir ») et de 1/1.1 pour zoomer négativement (l'image va s'éloigner) :

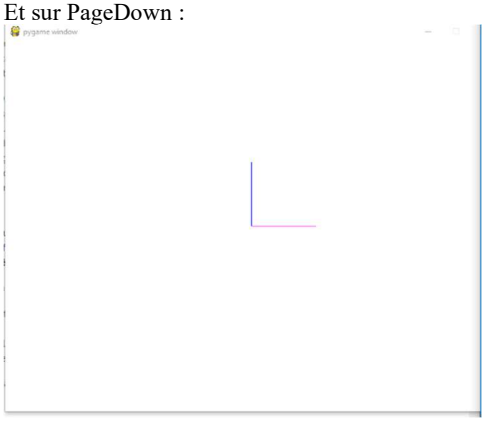
```
def processKeyDownEvent(self):
    # Rotates around the z-axis
    if self.event.dict['unicode'] == 'Z' or (self.event.mod & pygame.KMOD_SHIFT and self.ev
        gl.glRotate(-2.5, 0, 0, 1)
    elif self.event.dict['unicode'] == 'z' or self.event.key == pygame.K_z:
        gl.glRotate(2.5, 0, 0,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

    # Flèches haut et bas
    elif self.event.dict['unicode'] == 'PAGEUP' or self.event.key == pygame.K_PAGEUP:
        gl.glScalef(1.1, 1.1, 1.1)
    elif self.event.dict['unicode'] == 'PAGEDOWN' or self.event.key == pygame.K_PAGEDOWN:
        gl.glScalef(1/1.1, 1/1.1, 1/1.1)
```

Les keywords pour l'utilisation des touches PAGEUP et PAGEDOWN sont PAGEUP et PAGEDOWN.
Voici ce que l'on obtient en appuyant sur PageUP :

Et sur PageDown :



2. **Remplacer l'instruction pass dans la méthode processMouseButtonDownEvent() pour gérer l'effet de zoom avec la souris.**

On utilise l'attribue button associé aux valeurs de l'énoncé :

```
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)
```

3. **Remplacer l'instruction pass dans processMouseMotionEvent() pour gérer les déplacements des objets.**

Voici nos codes pour la rotation et la translation :

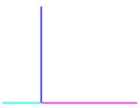
```
def processMouseMotionEvent(self):
    if pygame.mouse.get_pressed()[2]==1:
        gl.glTranslatef(self.event.rel[0]/10, 0, - self.event.rel[1]/10)

    elif pygame.mouse.get_pressed()[0]==1:
        gl.glRotatef(self.event.rel[1], 0, 1, 0)
        gl.glRotatef(self.event.rel[0], 1, 0, 0)
```

Pour la translation :
La condition est sur le clique droite. Donc, s'il est enfoncé, on translate l'axe x de la valeur du mouvement de la souris selon l'axe x et on translate l'axe z de l'opposé de la valeur du mouvement de la souris selon l'axe y. (car notre y est ici la profondeur et x l'axe horizontal et z à la vertical)

Pour la rotation :
La condition est sur le clique gauche. Donc, s'il est enfoncé, le mouvement de la souris selon y est utilisé pour l'angle de rotation autour l'axe y.
Et le mouvement de la souris selon x est utilisé pour l'angle de rotation autour l'axe x. La rotation selon z est composée de ces deux rotations.

Translation horizontale :



translation horizontale et verticale :

Exemple de rotation :

Nous avons réaliser les test en local et nous n'avons plus d'erreur dans test base :

```
Ran 7 tests in 0.687s

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

Et pareil pour test sur l'interface utilisateur :

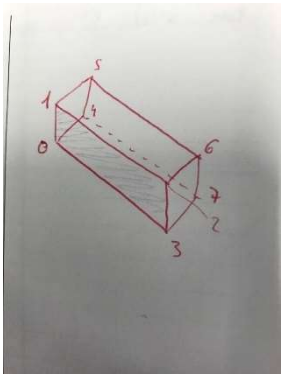
```
OK
An exception has occurred,
use %tb to see the full
traceback.

SystemExit: False
```

IV. Création d'une section

1. En s'inspirant du fichier Configuration.py, écrire la méthode generate(self) de la classe Section qui crée 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).

Nous devons tout d'abord compléter self.vertices où l'on rentre les 8 sommets du parallélépipède suivant :



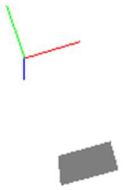
Chaque chiffre aux sommets correspond à l'index de la place dans la liste de chaque sommet dans self.vertices. Chaque sommet est représenté par 3 coordonnées et nous utilisons les attributs width, height et thickness. Ensuite, on rentre dans la liste self.faces l'ensemble des faces définit par les sommets des faces soient les vertices.
Voici le code :

```
# Defines the vertices and faces
def generate(self):
    self.vertices = [                   ##Définition des 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'], self.parameters['height']],
            [self.parameters['width'], self.parameters['thickness'], 0],
            ]
    self.faces = [                      ##Définition des faces
            [0, 3, 2, 1],
            [0, 1, 5, 4],
            [0, 4, 7, 3],
            [3, 2, 6, 7],
            [4, 7, 6, 5],
            [1, 5, 6, 2]
            ]
```

2. Tracer les faces du parallélépipède.

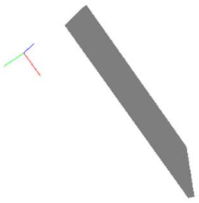
Analyse fonction Q)2b : elle ajoute un objet de type Section à la configuration actuelle.
Instrction Configuration().add(section).display() : cette instruction dessine les axes et les objets (ceux prédéfinis + ceux ajouté par la fonction add).
Pour afficher la première face à l'aide du code fournit en exemple, on doit rajouter avant glPushMatrix() dans le code pour récupérer une matrice et après glPopMatrix() qui permet de la sauvegarder. On ajoute une ligne de code par rapport à une translation pour reculer car la face se trouve en 0,0,0 et 'l'œil' est situé en ce point. Voici le code :

```python
# Draws the faces
def draw(self):
    # A compléter en remplaçant pass par votre code
    gl.glPushMatrix()
    gl.glTranslatef(0, -2, 0)
    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
    gl.glVertex3fv([0, 0, 0])
    gl.glVertex3fv([1, 0, 0])
    gl.glVertex3fv([1, 0, 1])
    gl.glVertex3fv([0, 0, 1])
    gl.glEnd()
    gl.glPopMatrix()
```



A présent, nous voulons afficher les 6 faces. Pour cela, nous allons utiliser self.faces et on va aller chaque élément de cette liste et prendre l'indice de cette liste qui correspond à des coordonnées d'une vertice que l'on donne dans les gl.glVertx3fv comme suit :

```python
# Draws the faces
def draw(self):
    # A compléter en remplaçant pass par votre code
    for face in self.faces:
        gl.glPushMatrix()
        gl.glTranslatef(0, -2, 0)
        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
        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()
```

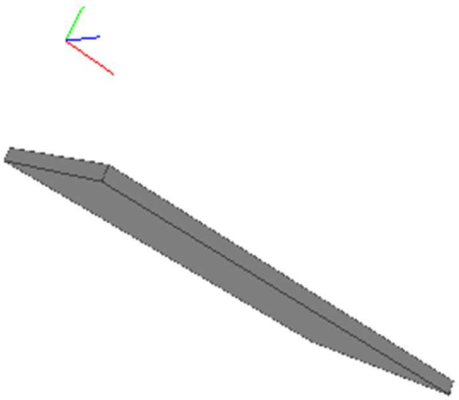


3. Dessiner les arêtes de la section.

Pour la méthode drawEdge() :
On reprend la méthode draw mais en rajoutant glPolygonMode :

```python
# Draws the edges
def drawEdges(self):
    # A compléter en remplaçant pass par votre code
    for face in self.faces:
        gl.glPushMatrix()
        gl.glTranslatef(0, -2, 0)
        gl.glPolygonMode(gl.GL_FRONT_AND_BACK, gl.GL_LINE) # on trace les faces : GL_FILL
        gl.glBegin(gl.GL_QUADS) # Tracé d'un quadrilatère
        gl.glColor3fv([0.5*0.5, 0.5*0.5, 0.5*0.5]) # Couleur gris moyen
        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()

# Draws the faces
def draw(self):
    # A compléter en remplaçant pass par votre code
    if self.parameters['edges']==True:
        self.drawEdges()
    for face in self.faces:
        gl.glPushMatrix()
        gl.glTranslatef(0, -2, 0)
        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
        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()
```



On a rajouté if self.parameters['edge'] == True : pour éviter que les traits se tracent 2 fois.

V. Création des murs

```
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)
```

Analyse de Wall.py :
Des exceptions sont prérentrées pour faire ‘planter’ le programme si des paramètres ne sont pas rentrés.
Voici notre code :

```
def draw(self):
    # A compléter en remplaçant pass par votre code
    for x in self.objects:
        if (isinstance(x,Section)==True):
            gl.glPushMatrix()
            gl.glRotatef(self.parameters['orientation'], 0, 0, 1)
            x.draw()
            gl.glPopMatrix()
        elif():
            x.draw()
```

Avec :

```
def Q3a():
    return Configuration().add(Wall({'position':[1,1,0], 'width':7, 'height':2.6, 'orientation':45}))
```

Explication :
On itère dans une liste d’objet avec une condition : isinstance vérifie que x est bien une Section et si la condition est vérifiée, il renvoie l’objet qui a effectué une rotation d’angle ‘orientation’.
Pour la fonction Q3a(), elle ajoute un objet de type mur à la configuration actuelle.
On obtient à l’écran ceci :

.
On note bien une orientation de 45°.

VI. Création d’une maison

VII. Création d’ouvertures

VIII. Conclusion

Durant ce tp, nous avons pu réaliser sur Spyder une représentation en 3D d'objets.
Nous n'avons pas pu malheureusement aller au bout de ce TP car nous nous sommes retrouvés bloqué avec une erreur.
L'orientation fonctionne mais elle est décalée.