TAKAHASHI Vincent
RICHARD Valentin
Date du TP : 24/11/2021

Rapport de TP3 (séance 1) – Représentation visuelle d'objets

I. Présentation du TP

Le but de ce TP est d'afficher des images en 3 dimensions.

II. Préparation du TP

Utilisation de Pygame

1. Question (1)

Nous avons implémenté le code fourni dans le sujet.
Le première ligne sert à importer la librairie Pygame, la seconde permet d'initialiser tout ce qui a été importé et de signifier que l'on va utiliser la librairie. Avec la troisième ligne on crée une fenêtre graphique (ecran), d'une taille 300*200. Cette dernière est pour le moment vide. Enfin, la dernière ligne permet de signifier que nous avons fini de travailler avec la librairie et de quitter la fenêtre que nous avons créé.
En effet, lorsque l'on exécute le programme, une fenêtre s'ouvre et se ferme presque instantanément.

2. Question (2)

Nous avons implémenté le nouveau code.
Cette fois-ci une fenêtre s'ouvre mais ne se ferme que lorsque l'on appuie sur une touche du clavier.
Ceci est possible grâce à la boucle while : tant qu'il n'y a pas d'événement de type : pygame.KEYDOWN (appui sur une touche du clavier), la fenêtre reste ouverte.

Utilisation de Pyopengl pour représenter des objets 3D

1. Question (1)

Nous avons exécuté le code.
Une fenêtre blanche s'ouvre. Elle ne se ferme plus quand on appuie sur une touche du clavier, mais quand on clique sur la croix avec la souris.
On a rajouté la fonction gluPerspective. Cela ne change rien au programme pour le moment.

2. Question (2)

Nous avons complété notre code avec ce que fourni le sujet.
Nous obtenons un axe noir dans la fenêtre lorsque nous exécutons le programme. Nous apportons des modifications pour faire apparaitre les 3 axes de 3 couleurs différentes. L'origine de nos axes est en (0, 0, -2).

3. Question (3)

Nous avons ensuite ajouté les fonctions glTranslatef et glRotatef.
Avec ces fonctions, nous obtenons une nouvelle fenêtre ou les axes sont translatés et/ou tournés par rapport à la fenêtre originale.

III. Découverte de l'environnement du TP

1. Question (1)

La ligne de commande donnée permet de faire tourner les axes créés précédemment selon l'axe z lorsque l'on appuie sur une touche (en l'occurrence la touche z pour tourner dans un sens et shift + z pour tourner dans l'autre sens). Nous pouvons aussi faire disparaître les axes en appuyant sur la touche a

C'est la fonction processKeyDownEvent(self) qui permet de définir un événement contenu dans la librairie pygame d'effectuer une action. Ici par exemple, l'événement event.key est un appui de touche de clavier.

2. Question (2)

La ligne de code proposée dans le sujet permet deux choses. Premièrement elle permet de réduire la taille de la figure dans la fenêtre grâce au 'sceenPosition'. Ensuite elle permet de colorer l'axe de x (dans le cas présent en mélangeant du rouge et du vert, ce qui donne du jaune).

Il est possible de chainer les méthodes 'setParameter' et 'display' car la méthode 'setParameter' ne fait que modifier les paramètres de figure. Paramètres qui ont de toute façon déjà une valeur par défaut et qui sont nécessaire pour l'afficher la figure à l'écran. La méthode 'setParameter' n'influence pas le fonctionnement de 'display' (à ceci près qu'elle affiche une figure différente en fonction des paramètres).

Dans la méthode setParameter, un traitement particulier est donné à 'sceenPosition'. Ce traitement est nécessaire car la méthode fonctionne avec un dictionnaire. Lorsque l'on rentre un paramètre, cela revient à créer une clé et à lui associer la valeur que l'on souhaite pour ce paramètre. En ce qui concerne 'sceenPosition', lorsque l'on rentre une nouvelle valeur, on ne tient pas compte de valeur précédente et on souhaite repartir de zéro pour effectuer la modification. On a donc besoin d'initialiser la position de la figure avant d'appliquer les nouveaux changements, d'où l'utilisation de la méthode 'initializeTransformationMatrix'.

3. Question (3)

On remarque que pour que l'axe z soit vertical et l'axe x horizontal, il suffit juste de « placer l'axe z à la place de l'axe x », c'est-à-dire faire une rotation de 90° de l'ensemble des axes selon l'axe y.

```
# Initializes the tranformation matrix
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)
# Getter
```

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

1. Question (1)

On utilise la même méthode que celle que l'on voit dans l'exemple donné avec les touches z et a. On crée une condition if qui est validée que lorsque l'événement « appui sur la touche K_PAGEUP » (respectivement K_PAGEDOWN) est validé. On change toutefois l'action à réaliser qui n'est plus une translation ou une rotation mais un zoom avec la fonction Scalef. On utilise pour le zoom les valeurs données dans le sujet.

```
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)
```

2. Question (2)

On garde encore la même structure que précédemment. Cette fois, la condition à valider est un coup de molette de souris (BUTTON_WHEELUP/BUTTON_WHEEL_DOWN). On crée deux conditions, une pour le zoom et une pour le dé-zoom.

```
if self.event.button == pygame.BUTTON_WHEELUP :
  gl.glScalef(1/1.1, 1/1.1, 1/1.1)

if self.event.button == pygame.BUTTON_WHEELDOWN :
  gl.glScalef(1.1, 1.1, 1.1)
```

3. Question (3)

On cherche tout d'abord à tourner le repère selon les axes x et z avec les mouvements de la souris lorsque le clic droit est enfoncé. Pour cela, on crée tout d'abord la condition sur le clic droit, l'objet ne tournera pas sans un appui du clic droit. Pour le faire tourner en bougeant la souris, on utilise comme vitesse de rotation la méthode rel qui nous permet d'avoir la vitesse de la souris. Ainsi, le repère tournera avec la souris.

Pour la translation, le résultat est similaire. On prend le clic gauche en condition du if et les mouvements de la souris comme vitesse de translation. Ici nous avons divisé la vitesse de la souris par 100 car le repère étant très petit, le moindre petit coup de souris faisait sortir nos axes de l'écran. C'est un problème que nous n'avons pas eu avec la rotation, les angles ne dépendant pas du « zoom » du repère.

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

  if pygame.mouse.get_pressed()[0] == 1 :
    gl.glRotatef(10, self.event.rel[0], 0, self.event.rel[1])

  if pygame.mouse.get_pressed()[2] == 1 :
    gl.glTranslatef(self.event.rel[0]/100, 0, self.event.rel[1]/100)
```

V. Création d'une section

1. Question (1)

Nous nous sommes penchés ici sur la création d'une section dans la fenêtre de pygame. Nous avons d'abord pris la fonction exemple en tant que référence et complété celle-ci afin d'obtenir la liste des points de notre futur parallélépipède. Le schéma sur le sujet de TP nous a aidé à rentrer les coordonnées de nos points.

Pour les faces, il suffisait de rentrer les numéros des points par ordre d'implémentation et de les relier entre eux.

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

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

2. Question (2)

La partie de remplissage des faces du parallélépipède nous a posé problème. En effet, la méthode demandée dans le sujet est de remplir une face avec le code donné et utiliser des matrices de transposition pour transformer cette face afin de pouvoir remplir toutes les faces de notre rectangle. Nous n'avons pas bien compris cette consigne. Nous avons donc, comme le code qui définissait une face était donné, défini les 5 autres faces de la même manière, en adaptant le code avec les bonnes coordonnées.

```
gl.glBegin(gl.GL_QUADS) # Tracé d'un quadrilatère
gl.glColor3fv([0.5, 0.5, 0.5]) # Couleur gris moyen
gl.glVertex3fv(self.vertices[0])
gl.glVertex3fv(self.vertices[1])
gl.glVertex3fv(self.vertices[2])
gl.glVertex3fv(self.vertices[3])

gl.glVertex3fv(self.vertices[4])
gl.glVertex3fv(self.vertices[5])
gl.glVertex3fv(self.vertices[6])
gl.glVertex3fv(self.vertices[7])

gl.glVertex3fv(self.vertices[0])
gl.glVertex3fv(self.vertices[1])
gl.glVertex3fv(self.vertices[5])
gl.glVertex3fv(self.vertices[4])

gl.glVertex3fv(self.vertices[0])
gl.glVertex3fv(self.vertices[3])
gl.glVertex3fv(self.vertices[7])
gl.glVertex3fv(self.vertices[4])

gl.glVertex3fv(self.vertices[2])
gl.glVertex3fv(self.vertices[3])
gl.glVertex3fv(self.vertices[7])
gl.glVertex3fv(self.vertices[6])

gl.glVertex3fv(self.vertices[1])
gl.glVertex3fv(self.vertices[2])
gl.glVertex3fv(self.vertices[6])
gl.glVertex3fv(self.vertices[5])

gl.glEnd()
```

3. Question (3)

Pour les arrêtes, on utilise la même méthode en définissant le point de départ et le point d'arrivée de chaque arrête. Pour la couleur on met du noir (0,0,0) pour pouvoir différencier facilement les arrêtes des faces.

```
# A compléter en remplaçant pass par votre code

gl.glPolygonMode(gl.GL_FRONT_AND_BACK, gl.GL_LINE)
gl.glBegin(gl.GL_QUADS)
gl.glColor3fv([0.2,0.2,0.2])

gl.glVertex3fv(self.vertices[0])
gl.glVertex3fv(self.vertices[1])

gl.glVertex3fv(self.vertices[0])
gl.glVertex3fv(self.vertices[4])

gl.glVertex3fv(self.vertices[0])
gl.glVertex3fv(self.vertices[3])

gl.glVertex3fv(self.vertices[1])
gl.glVertex3fv(self.vertices[2])

gl.glVertex3fv(self.vertices[1])
gl.glVertex3fv(self.vertices[5])

gl.glVertex3fv(self.vertices[2])
gl.glVertex3fv(self.vertices[3])

gl.glVertex3fv(self.vertices[2])
gl.glVertex3fv(self.vertices[6])

gl.glVertex3fv(self.vertices[3])
gl.glVertex3fv(self.vertices[7])

gl.glVertex3fv(self.vertices[4])
gl.glVertex3fv(self.vertices[5])

gl.glVertex3fv(self.vertices[5])
gl.glVertex3fv(self.vertices[6])

gl.glVertex3fv(self.vertices[6])
gl.glVertex3fv(self.vertices[7])

gl.glVertex3fv(self.vertices[4])
gl.glVertex3fv(self.vertices[7])

gl.glEnd()
```

VI. Conclusion de cette séance

Il nous semble avoir bien progressé pendant cette séance. Nous avons avancé sans nous préoccuper des tests, nous penserons à consacrer à temps pour tester nos programmes en séance 2.