INFO 501
24/11/2021

Kaoutar Bourabi
Ouijdane Boumehraz

TP 3 : Représentation visuelle d'objets.

Introduction :

I- Préparation avant le TP :

1. Expliquer le programme :

```
import pygame
pygame.init()
ecran = pygame.display.set_mode((300, 200))
pygame.quit()
```

- pygame.init() : initialise tous les modules pygame importés.
- pygame.display.set_mode : crée une surface d'affichage,les valeurs passées en arguments représentent la hauteur et la largeur de la fenêtre.
- pygame.quit() : désinitialise le module pygame qui est été précédemment initialisés.

Après avoir exécuté le code ci-dessus une fenêtre s'ouvre et se ferme.

2.

```
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()
```

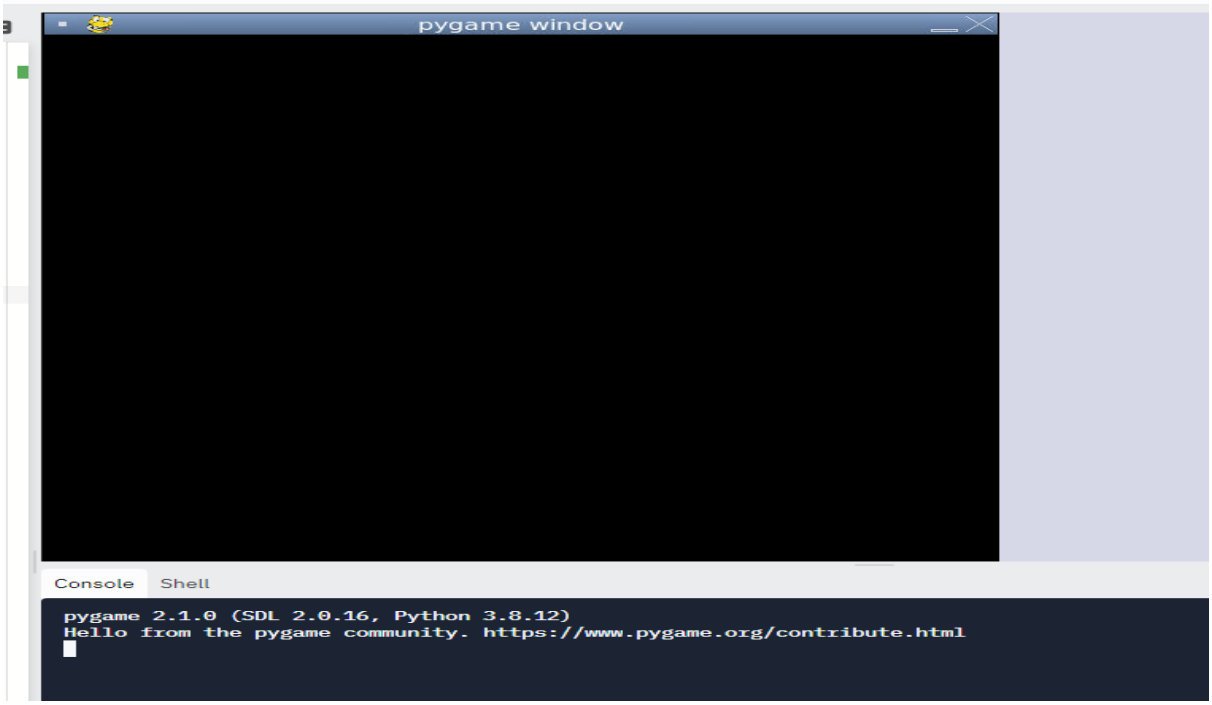
On récupère l'événement que pygame a capturé que l'on nomme event (il se peut qu'il n'y en ai pas, dans ce cas, le code n'est pas exécuté, et la boucle (qui contiendra tout le code d'affichage d'images, qui gérera les événements …) continue de boucler)

Dans le if maintenant : on regarde si le "type" de l'événement est une entrée clavier si oui, alors on met continuer à False, donc on quitte la boucle.

Utilisation de Pyopengl pour représenter des objets 3D

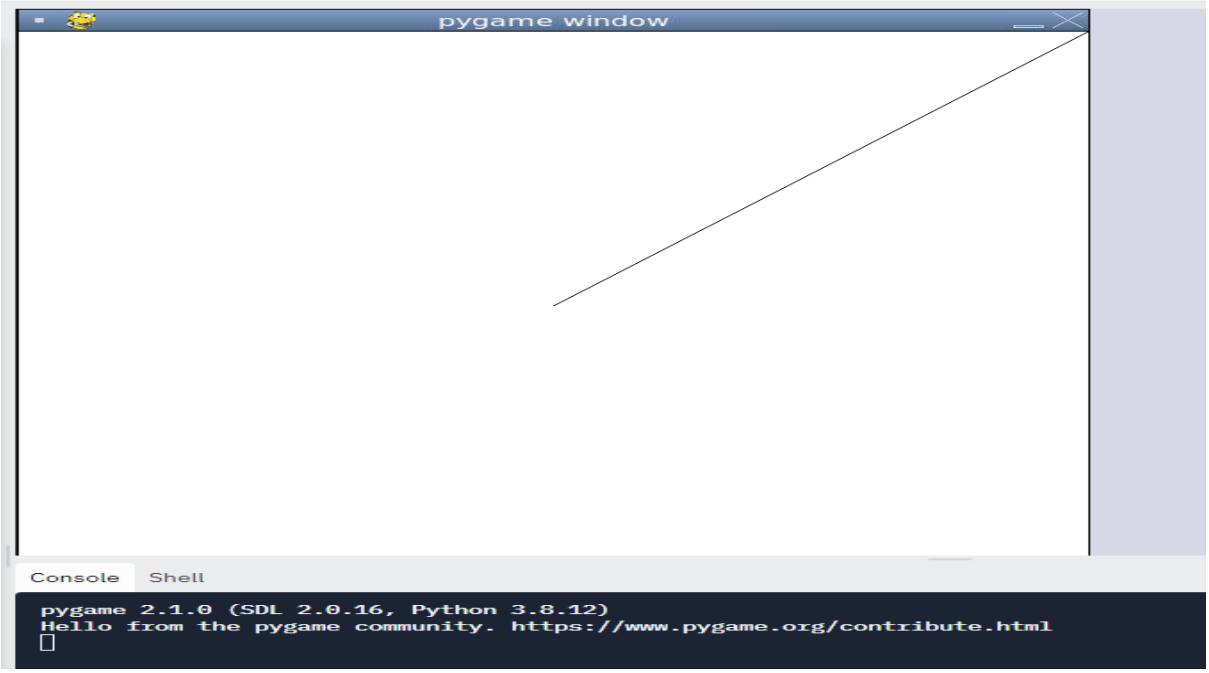
1. Utiliser la fonction gluPerspective :
   La fonction gluPerspective définit une matrice de projection de perspective. Ses paramètres sont :

   - Fovy : champ d'angle de vue, en degrés, dans l'axe y.
   - Aspect : proportions qui déterminent le champ de vue sur l'axe x. Les proportions sont le rapport entre x (largeur) et y (hauteur).
   - zNear : distance entre la visionneuse et le plan de découpage proche (toujours positive).
   - zFar : distance entre la visionneuse et le plan de découpage Far (toujours positive).



2. Tracer les axes en utilisant les fonctions **glBegin**, **glColor3fv**, **glVertex3fv et glEnd**:
   glColor3fv :permet de spécifier la couleur (paramètres dans l'ordre RGB) pour toutes les primitives graphiques qui vont suivre. Il est possible de rajouter un quatrième paramètre pour caractériser l'opacité (paramètre alpha qui vaut 1 par défaut).

   Les fonctions glBegin() et glEnd() ont pour rôle de délimiter une suite de sommets associés au tracé d'une ou plusieurs primitives de même nature. Cette suite de sommets pourra aussi bien définir des lignes brisées (contours) que des polygones (éléments surfaciques).
   Un sommet est défini par la fonction glVertex*().



3.

glTranslatef : produit la traduction spécifiée par (x, y, z). Le vecteur de translation est utilisé pour calculer une matrice de traduction 4x4.La matrice actuelle est multipliée par cette matrice de translation, le produit remplaçant la matrice actuelle. Autrement dit, si M est la matrice active et que T est la matrice de translation, M est remplacé par M T.

glRotatef calcule une matrice qui effectue une rotation dans le sens inverse des degrés d' angle sur le vecteur à partir de l'origine jusqu'au point (x, y, z).

La matrice actuelle est multipliée par cette matrice de rotation, le produit remplaçant la matrice actuelle. Alors, si M est la matrice actuelle et que R est la matrice de translation, M est remplacé par M R.

II- Découverte de l'environnement du travail du TP

1).a

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

1).b

- Expliquer pourquoi le chaînage de l'appel de méthodes setParameter() et display() est possible : SetParameter ajoute ou met à jour un paramètre unique dans une collection de paramètres interne.
-

1).c pour que l'axe *z* soit représenté verticalement sur l'écran et que l'axe *x* soit représenté horizontalement, on a ajouté : gl.glRotatef(-90,1,0,0) :

```
# 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)
```

II- Mise en place des interactions avec l'utilisateur avec Pygame

1).d pour effectuer un changement d'échelle, on a utilisé :

```
# Processes the KEYDOWN event
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)

    # zoom
    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)
```

1).e pour gérer l'effet de zoom avec la souris, on a utilisé cette méthode :

```
# Processes the MOUSEBUTTONDOWN event
def processMouseButtonDownEvent(self):
    if self.event.button == pygame.BUTTON_WHEELUP:
        gl.glScalef(1.1,1.1,1.1)
    elif self.event.button == pygame.BUTTON_WHEELDOWN:
        gl.glScalef(1/1.1,1/1.1,1/1.1)
```

1).f pour gérer le déplacement des objets, on a utilisé la méthode suivante :

```python
# Processes the MOUSEMOTION event
def processMouseMotionEvent(self):
    if pygame.mouse.get_pressed()[0]== 1 :
        x,z = self.event.rel
        gl.glRotate(x, 1, 0, 0)
        gl.glRotate(z,0,0,1)
    if pygame.mouse.get_pressed()[2]== 1 :
        x,z = self.event.rel
        gl.glTranslatef(x/25,0,0)
        gl.glTranslatef(0,0,z/25)
```