Tp3

1)

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

D'abord on initialise pygame ce qui ouvre la fenêtre puis on quitte la fenêtre se referme.

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 ferme pygame si on appuie sur une touche grâce à la boucle while. En effet tant que continuer reste à vraie on laisse ouvert le programme et si on appuie sur le clavier continuer est à faux.

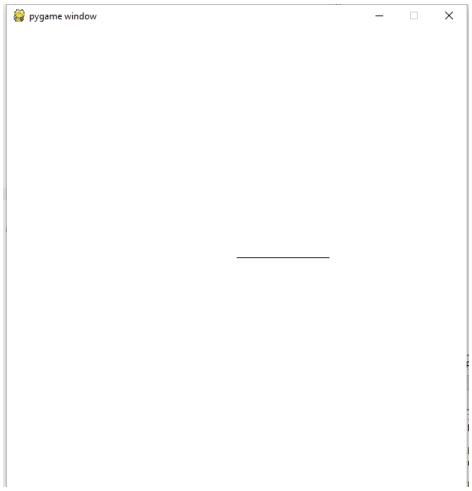
1)

```
import pygame
import OpenGL.GL as gl
import OpenGL.GLU as glu

if __name__ == '__main__':
    pygame.init()
    display=(600,600)
    pygame.display.set_mode(display, pygame.DOUBLEBUF | pygame.OPENGL)

    # Sets the screen color (white)
    gl.glClearColor(1, 1, 1, 1)
    # Clears the buffers and sets DEPTH_TEST to remove hidden surfaces
    gl.glClear(gl.GL_COLOR_BUFFER_BIT | gl.GL_DEPTH_BUFFER_BIT)
    gl.glEnable(gl.GL_DEPTH_TEST)

    # Placer ici l'utilisation de gluPerspective.

    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
```

```
            pygame.quit()
        exit()
glu.gluPerspective(45, (display[0] / display[1]), 0.1, 50.0)
```

2)

3)



1a)

1b)

C'est possible car display et setParamètre sont dans la même classe.
On effectue un traitement particulier car le paramètre screenPosition est utilisé pour initialiser la position de l'objet.

1c)

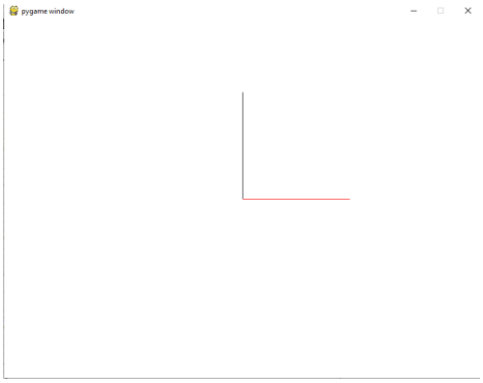
gl.glRotatef(-90,1,0,0)

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

1d)

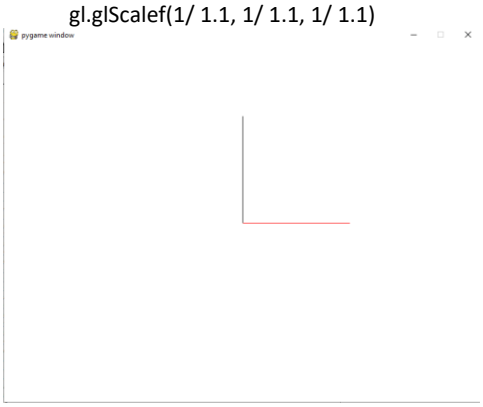
```
def processKeyDownEvent(self):
    # Rotates around the z-axis
    if self.event.dict['unicode'] == 'Z' or (self.event.mod & pygame.KMOD_SHIFT and self.event.key == pygame.K_z):
        gl.glRotate(-2.5, 0, 0, 1)
```

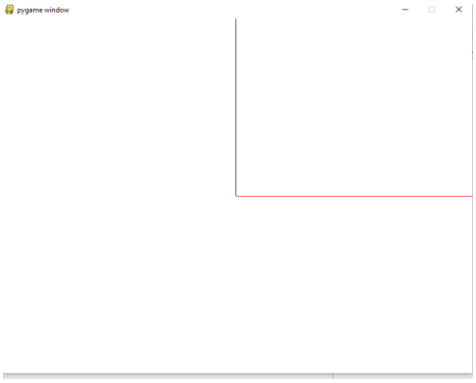
```
elif self.event.dict['unicode'] == 'z' or self.event.key == pygame.K_z:
    gl.glRotate(2.5, 0, 0, 1)
# Draws or suppresses the reference frame
elif self.event.dict['unicode'] == 'a' or self.event.key == pygame.K_a:
    self.parameters['axes'] = not self.parameters['axes']
    pygame.time.wait(300)
# grossir/reduire affichage
if 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)

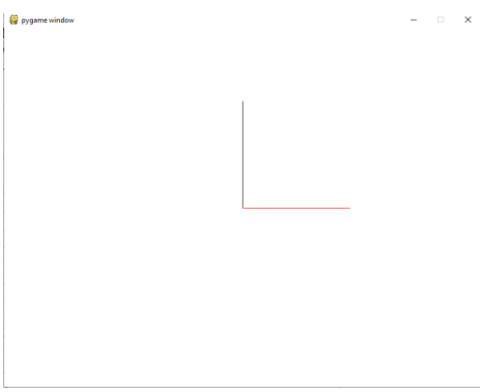
```
def processMouseButtonDownEvent(self):
    if self.event.button ==4:
        gl.glScalef(1.1, 1.1, 1.1)
    elif self.event.button == 5:
        gl.glScalef(1/ 1.1, 1/ 1.1, 1/ 1.1)
```

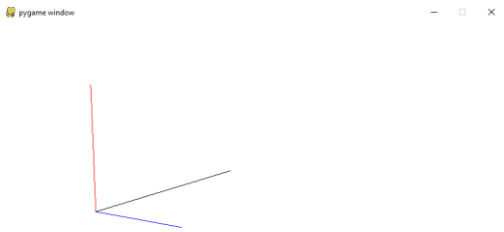




1f)

```
# Processes the MOUSEMOTION event
def processMouseMotionEvent(self):
    if pygame.mouse.get_pressed()[0]==1:
        gl.glRotatef(self.event.rel[1],1,0,0)
        gl.glRotatef(self.event.rel[0],0,0,1)
    if pygame.mouse.get_pressed()[2]==1:
        gl.glTranslatef(self.event.rel[0]/100,0,0)
        gl.glTranslatef(0,0,-self.event.rel[1]/100)
```





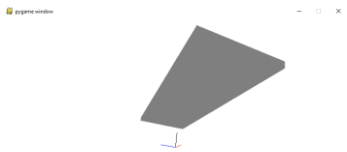
IV - Création d'une section

2a)

```
def generate(self):
    #definir les sommets
    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'],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],
            [0, 4, 5, 2],
            [4, 7, 6, 5],
            [7, 3, 2, 6],
            [1, 5, 6, 2]
            ]
```

2b)
On ajoute une section à configuration

```
def draw(self):
    gl.glPushMatrix()
    gl.glTranslate(self.parameters['position'][0],self.parameters['position'][1],self.parameters['position'][2])
    gl.glRotate(self.parameters['orientation'],1,1,1)
    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']) #
    for i in self.faces:
        for j in i:
            gl.glVertex3fv(self.vertices[j])

    gl.glEnd()
    gl.glPopMatrix()
```



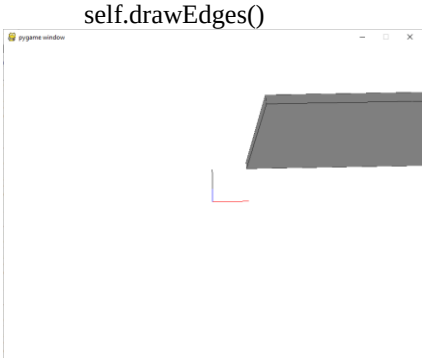
*

2c)

```
def drawEdges(self):
    gl.glPushMatrix()
    gl.glTranslatef(self.parameters['position'][0],self.parameters['position'][1],self.parameters['position'][2])

    gl.glPolygonMode(gl.GL_FRONT_AND_BACK, gl.GL_LINE) # on trace les faces : GL_FILL
    for i in self.faces:
        gl.glBegin(gl.GL_QUADS) # Tracé d'un quadrilatère
        gl.glColor3fv([self.parameters['color'][0]*0.8,self.parameters['color'][1]*0.8,self.parameters['color'][2]*0.8])
        for j in i:
            gl.glVertex3fv(self.vertices[j])
        gl.glEnd()
    gl.glPopMatrix()
```

on ajoute à draw() :

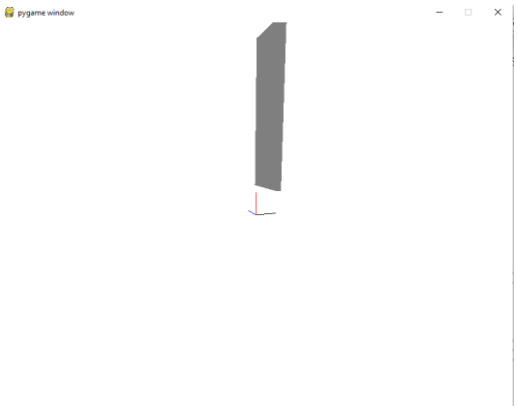
```
if self.parameters['edges']:
    self.drawEdges()
```



V - Création des murs

3a)
La class wall corresponds à un mur : les différents paramètres sont la positon du mur, l'épaisseur, l'orientation, la couleur.

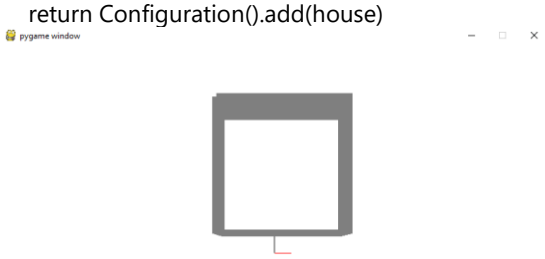
```
def draw(self):
    # A compléter en remplaçant pass par votre code
    gl.glPushMatrix()
    #gl.glRotate(self.parameters['orientation'],0,0,1)
    for i in self.objects:
        i.draw()
    gl.glPopMatrix()


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



VI - Création d'une maison

4a)

```
    def draw(self):
        # A compléter en remplaçant pass par votre code
        gl.glPushMatrix()
        gl.glRotate(self.parameters['orientation'],0,0,1)
        for i in self.objects:
            i.draw()
        gl.glPopMatrix()

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



5a)

```
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'], 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 = [
            [0,4,5,1],
            [3,7,6,2],
            [0,4,7,3],
            [1,5,6,2]
            ]


    # Draws the faces
    def draw(self):
        # A compléter en remplaçant pass par votre code
        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)
        for i in self.faces:
            gl.glBegin(gl.GL_QUADS)
            gl.glColor3fv(self.parameters['color'])
            for j in i:
                gl.glVertex3fv(self.vertices[j])
            gl.glEnd()
        gl.glPopMatrix()
```

5b)

```
def canCreateOpening(self, x):
        return (self.parameters['height']>=x.getParameter('height')+x.getParameter('position')[2]-self.parameters['position'][2]
            and x.getParameter('position')[2]>=self.parameters['position'][2]
            and self.parameters['width']>=x.getParameter('width')+x.getParameter('position')[0]-self.parameters['position'][0]
            and x.getParameter('position')[0]>=self.parameters['position'][0]
            )
```

```
In [7]: runfile('E:/Info501/tp3/main.py', wdir='E:/Info501/tp3')
True
True
False
```

5c)