PRUVOST Jordan PAULIN Maxime

TP3 – Compte Rendu – Partie 2

VI – Création d'une maison

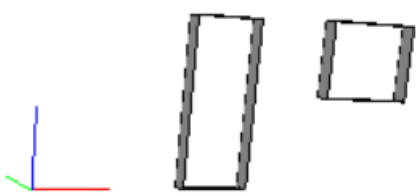
```
def Q4a():
    # Ecriture en utilisant des variables : A compléter
    wall1 = Wall({'position': [0, 0, 0],   'width':4, 'height':3, 'thickness': 0.2, 'orientation':0})
    wall2 = Wall({'position': [0, 6.8, 0], 'width':4, 'height':3, 'thickness': 0.2, 'orientation':0})
    wall3 = Wall({'position': [4, 0, 0],   'width':7, 'height':3, 'thickness': 0.2, 'orientation':90})
    wall4 = Wall({'position': [0, 0, 0],   'width':7, 'height':3, 'thickness': 0.2, 'orientation':90})

    house = House({'position': [-3, 1, 0], 'orientation':0})

    house.add(wall1).add(wall3).add(wall4).add(wall2)
    return Configuration().add(house)
```

La méthode **draw()** de la classe **Configuration** dessine les objets définis dans Wall avec différents paramètre. La rotation doit ici être prit en compte dans la création de la maison.
Dans Main.py, la création des quatre murs nécessite différents paramètres comme la position du mur, mais également sa rotation. Ici, il faut créer les quatre murs afin qu'il n'y ai pas de problème de « collision » entre eux.

VII – Création d'ouverture

```
def Q5a():
    # Ecriture avec mélange de variable et de chaînage
    opening1 = Opening({'position': [2, 0, 0], 'width':0.9, 'height':2.15, 'thickness':0.2, 'color': [0.7, 0.7, 0.7]})
    opening2 = Opening({'position': [4, 0, 1.2], 'width':1.25, 'height':1, 'thickness':0.2, 'color': [0.7, 0.7, 0.7]})
    return Configuration().add(opening1).add(opening2)
```

Après exécution du fichier Main.py, on obtient le dessin suivant, comme indiqué dans l'énonce



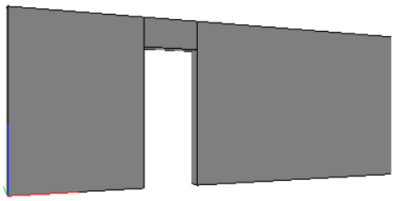
La modification de la classe **Section** avec la méthode **canCreateOpening(self,x)** permet de créer une ouverture dans une section de mur selon des paramètres que l'on fixe (ex : position, rotation, etc)

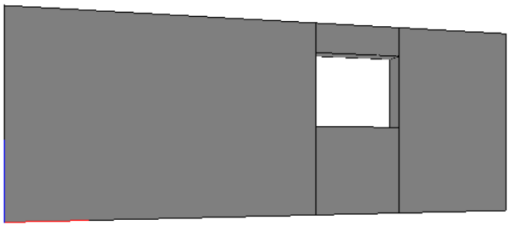
```
def Q5b():
    # Ecriture avec mélange de variable et de chaînage
    section = Section({'width':7, 'height':2.6})
    opening1 = Opening({'position': [2, 0, 0], 'width':0.9, 'height':2.15, 'thickness':0.2, 'color': [0.7, 0.7, 0.7]})
    opening2 = Opening({'position': [4, 0, 1.2], 'width':1.25, 'height':1, 'thickness':0.2, 'color': [0.7, 0.7, 0.7]})
    opening3 = Opening({'position': [4, 0, 1.7], 'width':1.25, 'height':1, 'thickness':0.2, 'color': [0.7, 0.7, 0.7]})

    print(section.canCreateOpening(opening1))
    print(section.canCreateOpening(opening2))
    print(section.canCreateOpening(opening3))
    return Configuration()
```

La création de la méthode **createNewSection(self, x)** de la classe **Section** permet de créer un nouvelle section sur un mur, selon des paramètres que l'on défini

```
def Q5c1():
    section = Section({'width':7, 'height':2.6})
    opening1 = Opening({'position': [2, 0, 0], 'width':0.9, 'height':2.15, 'thickness':0.2, 'color': [0.7, 0.7, 0.7]})
    sections = section.createNewSections(opening1)
    configuration = Configuration()
    for x in sections:
        configuration.add(x)
    return configuration

def Q5c2():
    section = Section({'width':7, 'height':2.6})
    opening2 = Opening({'position': [4, 0, 1.2], 'width':1.25, 'height':1, 'thickness':0.2, 'color': [0.7, 0.7, 0.7]})
    sections = section.createNewSections(opening2)
    configuration = Configuration()
    for section in sections:
        configuration.add(section)
    return configuration
```

Les deux sections ainsi obtenues sont les suivantes :





Enumerate est un compteur pour un objet, qui ici va ici retourner un itérable de la liste (c'est-à-dire la liste des objets).

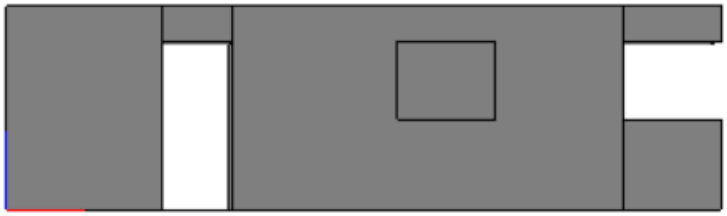
```
wall = Wall({'width':7, 'height':2.6,})
opening1 = Opening({'position': [2, 0, 0], 'width':0.9, 'height':2.15, 'thickness':0.2, 'color': [0.7, 0.7, 0.7]})
section = wall.findSection(opening1)
```

Ce code nous permet d'obtenir la section du mur « Wall » ou se trouve « opening1 »

La méthode **add(self, x)** retourne, une fois le fichier Main.py exécuté :

```
def Q5d():
    configuration = Configuration()
    wall = Wall({'position': [0, 0, 0],   'width':7, 'height':2.6, 'thickness': 0.2, 'orientation':0})
    opening1 = Opening({'position': [2, 0, 0], 'width':0.9, 'height':2.15, 'thickness':0.2, 'color': [0.7, 0.7, 0.7]})
    opening2 = Opening({'position': [4, 0, 1.2], 'width':1.25, 'height':1, 'thickness':0.2, 'color': [0.7, 0.7, 0.7]})

    wall.add(opening1)
    wall.add(opening2)

    configuration.add(wall)
    return configuration
```

Pour la suite, ainsi que dans la partie VIII nous avons obtenu des résultats « aberrants » qui ne correspondaient pas à ce que nous devions obtenir, ainsi, notre deuxième ouverture semblait « sortir » du mur comme ci-dessous



Conclusion

Ce TP nous a permit de comprendre l'utilisation d'OpenGL sous python ainsi que la mise en place de la modélisation 3D sous python. Malgré des problèmes croisés lors de notre programmation, nous avons tout de même appris quelques notions introduites dans ce TP.