# TP 3 : Représentation visuelle d’objets

## Introduction

Lors de cette séance de TP, nous allons nous intéresser à la représentation d’objets 3D à l’écran, à l’aide d’une fenêtre graphique. Nous allons construire progressivement des objets simples tels que des murs, portes, mais encore des fenêtres. Ce TP organisé sur deux séances, va mettre à l’épreuve nos capacités de programmation, notamment de méthodes et de classes, développées lors des cours et TD d’informatique.

## Préparation

### Utilisation de Pygame



En tapant le programme suivant :

On voit une fenêtre s’ouvrir et se refermer directement. Les lignes de code permettent d’initialiser la fenêtre, choisir ses dimensions, et la refermer.

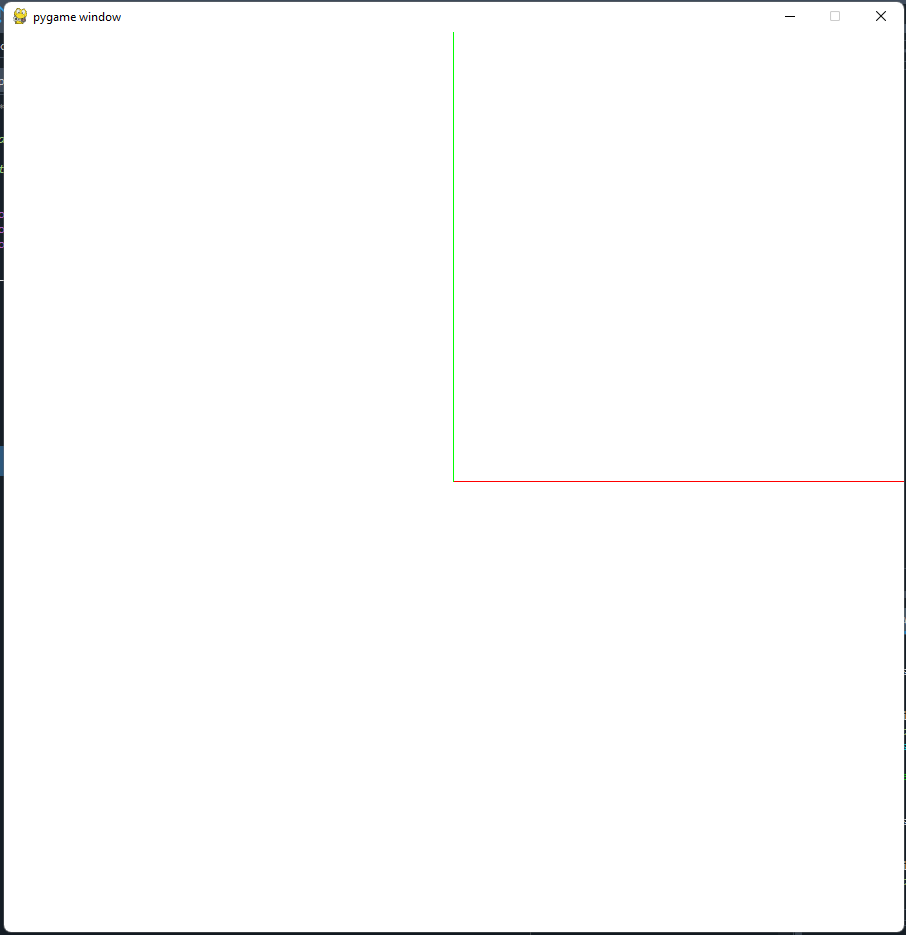
Une image contenant texte

Description générée automatiquement

Avec le second code :

Cette fois-ci, la fenêtre s’ouvre et ne se referme que lorsqu’une touche du clavier est appuyée. C’est ce qui est géré par la boucle while : on regarde si l’évènement d’une touche appuyée est détecté, si c’est le cas, le programme s’arrête.

### Utilisation de Pyopengl pour représenter des objets 3D



En utilisant OpenGL, on trace trois axes x, y et z respectivement de couleur rouge, vert et bleu. On a le résultat suivant :

NB : On ne voit pas l’axe z car il est perpendiculaire au plan.

On utilise ensuite les fonctions glRotatef et glTranslatef après la fonction gluPerspective afin de faire translater ou tourner l’image.

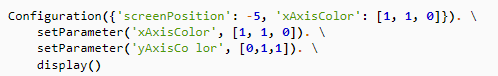
### Découverte de l’environnement du travail du TP

En observant les fichiers main.py et Configuration.py plus précisément et en faisant fonctionner le main, on voit que les axes sont tracés, comme vu précédemment. De plus, des évènements sont rajoutés selon la touche du clavier appuyé. Un appui sur la touche a arrête ou remet l’affichage des axes, alors que la touche z fait tourner les axes dans le sens trigonométrique autour de l’axe z, enfin, un appui sur la touche Z va faire tourner les axes selon le sens horaire autour de l’axe z.

Si nous remplaçons la commande Configuration() par la commande :



La partie screenPosition permet de gérer la distance de l’écran selon l’axe z. La commande xAxisColor sert à modifier la couleur de l’axe x (ici en jaune).



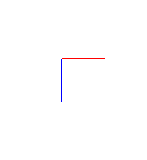
Ensuite, on utilise :

Le chaînage de l’appel des méthodes setParameter() et display() est possible car avant d’utiliser la méthode qui affiche ce qui est demandé, on peut modifier un ou plusieurs paramètres (par exemple la couleur des axes) à l’aide de la méthode setParameter().

Dans le « setter », un traitement particulier est effectué concernant le paramètre screenPosition. En effet si ce paramètre n’est pas spécifié, il doit être initialisé à une valeur, pour que l’affichage s’effectue correctement, car ce paramètre spécifie la distance entre l’objet à dessiner et l’écran.

Enfin, afin de représenter l’axe z verticalement, l’axe x horizontalement et l’axe y selon la profondeur, on tape la ligne de code suivante :



On obtient alors :

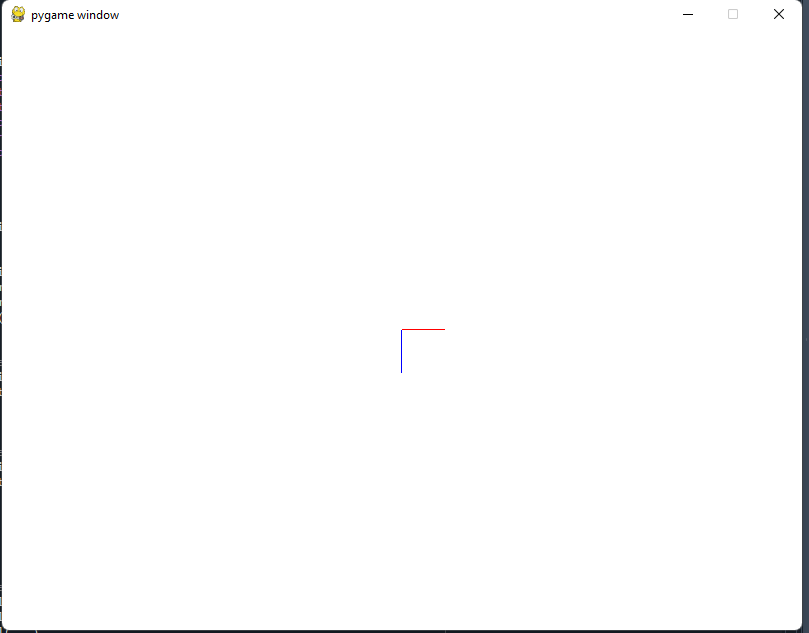
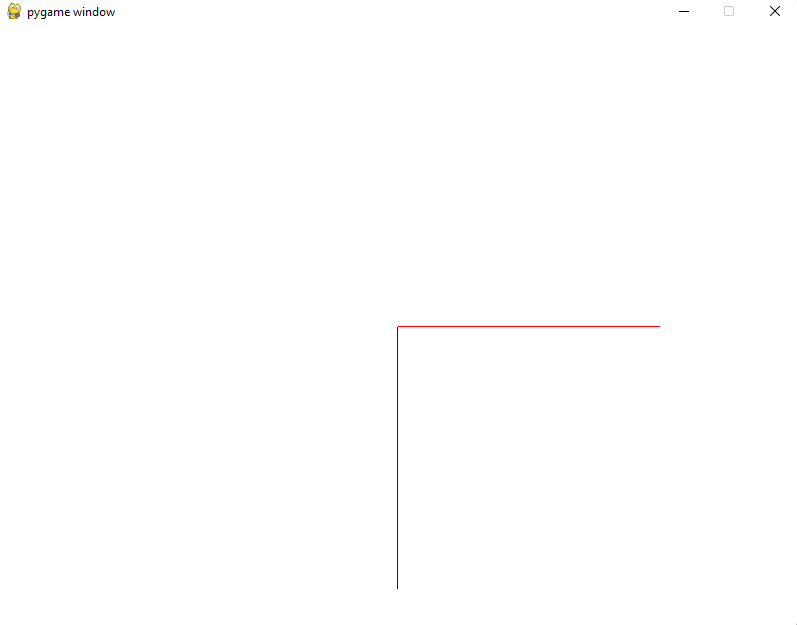
## Mise en place des interactions avec l’utilisateur avec Pygame

Cette partie vise à programmer les interactions possibles avec l’utilisateur, comme le zoom ou bien le déplacement des objets affichés.

### Zoom avec touches « Page Up » et « Page Down »

Afin de programmer ces actions, on programme deux réactions à des évènements de touches appuyées. Le premier est pour la touche « Page Up », il modifie l’échelle des 3 axes avec un facteur de 1.1 à chaque appui sur la touche. Pour la touche « Page Down », c’est sensiblement la même chose, sauf que le changement d’échelle s’effectue avec un facteur de 1/1.1.

Le programme nous permet alors de zoomer sur l’écran à l’aide des touches Page Up et Page Down :

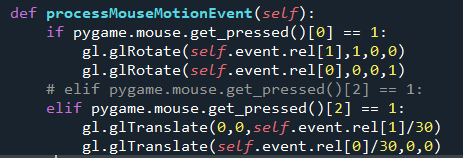


### Zoom avec la mollette de la souris

De même que pour la partie précédente, on programme une réaction à des évènements d’actionnement de la roulette de la souris. On note cependant un problème de fonctionnement du programme : il ne fonctionne correctement que lorsque Spyder est exécuté la première fois après son lancement. Après avoir eu l’avis de l’enseignant, nous décidons de passer ce problème et de se contenter des touches Page Up et Page Down, qui permettent de zoomer.

### Déplacement des objets avec la souris

Désormais, les clics gauche et droit de la souris permettent respectivement une rotation et une translation selon les axes x et z grâce à une structure à évènement qui sélectionne la fonction à appliquer selon le clic effectué. Nous avons modifié la fonction de translation afin de réduire la vitesse de déplacement des axes car celle-ci était trop rapide.



## Création d’une section

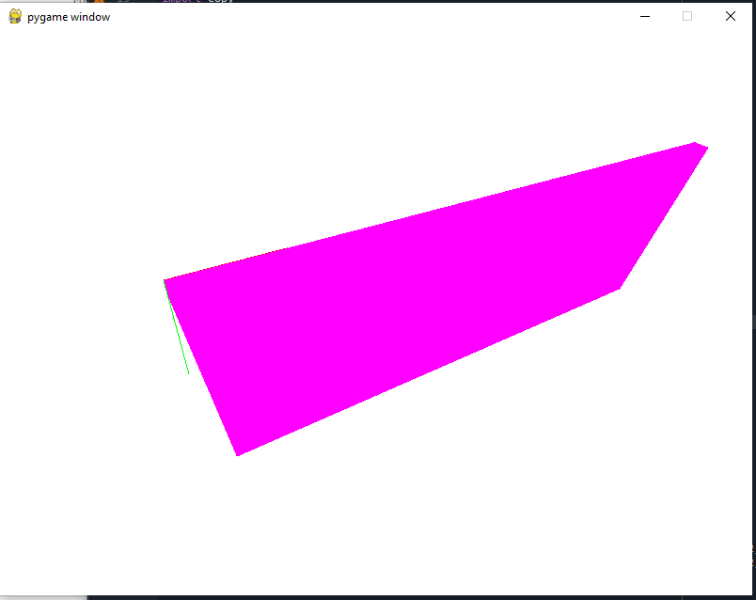
### Affichage des faces de la section

Dans la méthode generate de la classe Section, on construit les 8 sommets, qui permettent de faire les 6 faces, puis dans configuration, on ajouter la fonction qui permet de ne pas dessiner les faces non visibles. Ensuite, on crée la méthode draw, qui permet avec deux boucles for, de tracer les 6 faces de notre section. Elle prend aussi en compte la position demandée de la section en redéfinissant une matrice de projection.

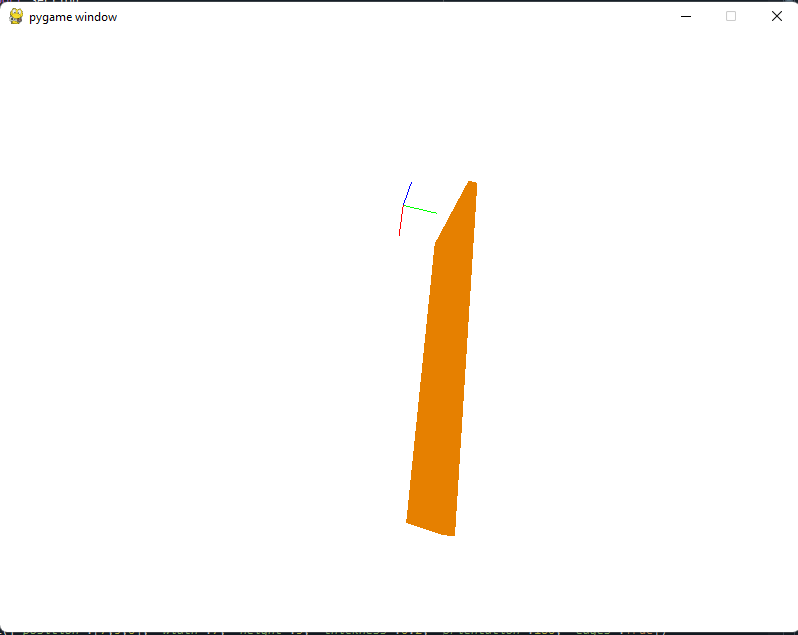
On obtient le résultat suivant (en changeant la couleur de la section) :

Une image contenant texte

Description générée automatiquement

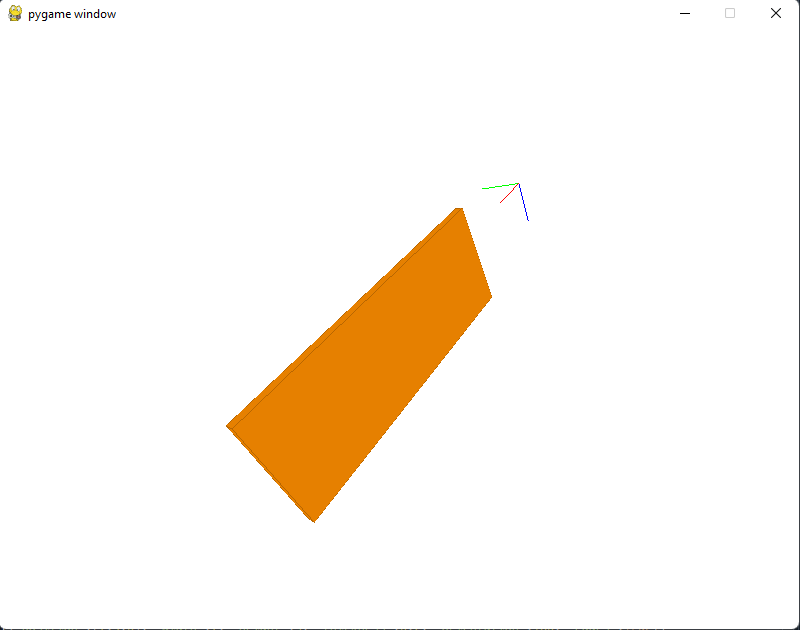


On peut maintenant choisir la position de la section :





### Affichage des arrêtes de la section



On code la classe drawEdges, qui permet l’affichage des arrêtes visibles de la section si le paramètre ‘edges’ est sur True. Les arrêtes ne sont affichées que si elles sont visibles et ont une couleur assombrie.

On a le résultat suivant :

## Création des murs

La création des murs se fait grâce à la classe Wall. Dans cette partie, nous avons programmé l’affichage d’un mur, qui est composé s’une section. On crée la méthode draw afin de pouvoir afficher le mur.

On a le résultat suivant :

Une image contenant texte, carte de visite

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

## Création d’une maison

Une image contenant texte

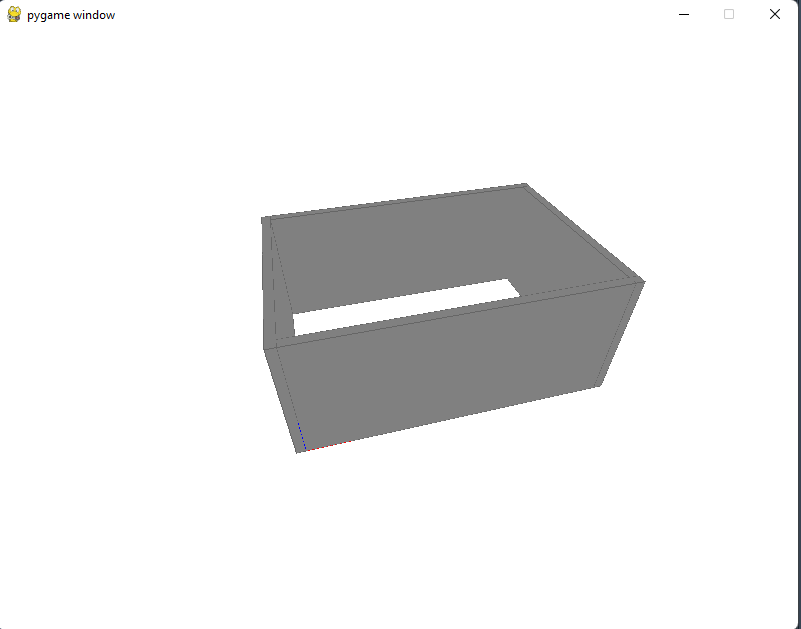
Description générée automatiquementCette partie vise à combiner plusieurs murs afin de créer les quatre murs d’une maison. On crée la méthode draw de la classe House, qui dessine les murs demandés en adaptant une certaine orientation et position demandée.

La méthode draw est :

En modifiant la partie Q4a() dans le main, on obtient le résultat suivant :

Une image contenant texte

Description générée automatiquement



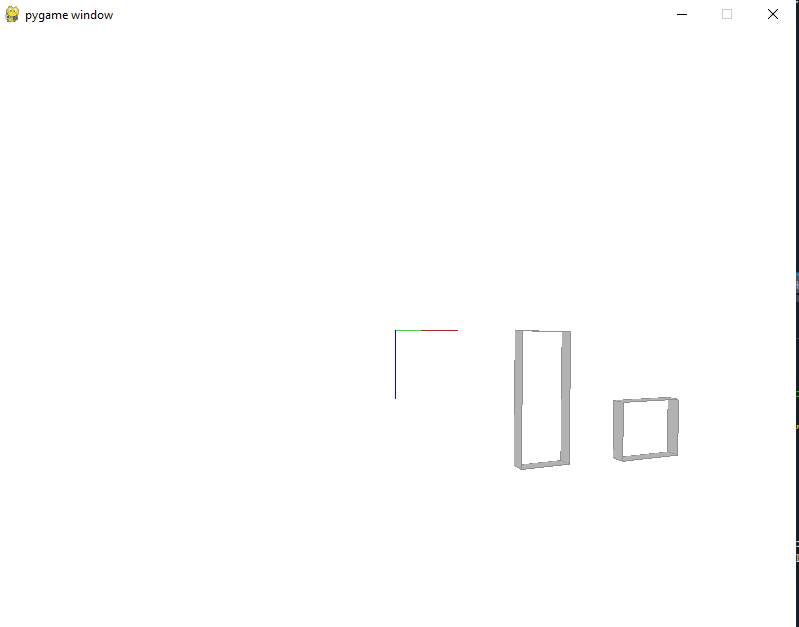
Si nous demandons une certaine position et/ou une certaine orientation, on a bien une maison identique qui est déplacée selon les conditions :

Une image contenant texte, carte de visite, graphiques vectoriels

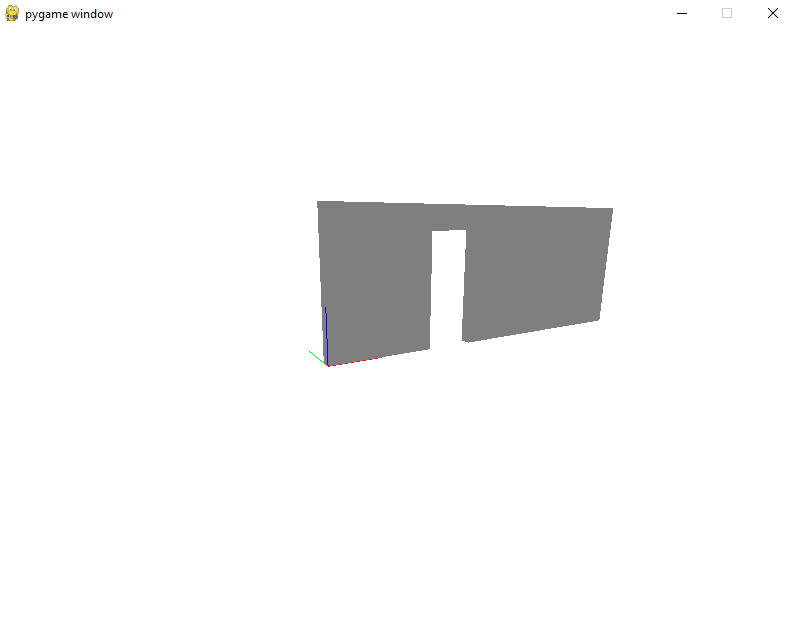
Description générée automatiquement

## Création d’ouvertures

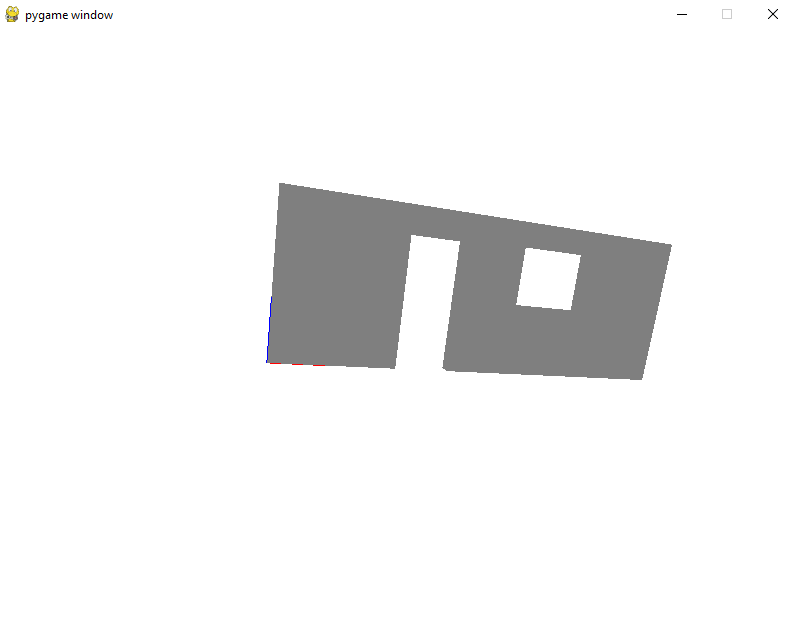
Dans un premier temps, nous avons créé une ouverture, pour cela il fallait supprimer l’affichage de deux faces, tout en gardant le contour et les arrêtes.

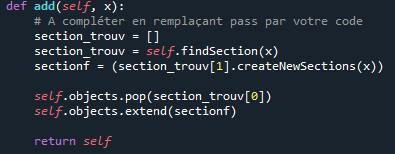


Puis, nous l’avons inséré dans un mur. Pour cela, il a fallu décomposer le mur en plusieurs sections pour reconstituer le mur avec l’ouverture.



Enfin, nous avons voulu insérer plusieurs ouvertures dans le mur. Pour cela, il a fallu gérer plusieurs contraintes de taille et de nombre de sections à découper.





## Conclusion

Durant ce TP, nous avons pu créer un environnement en 3D, selon différents axes, ainsi que les fonctionnalités permettant de se déplacer dans cet environnement. Puis, nous avons codé des formes géométriques s’affichant dans cet environnement, dans le but de concevoir une maison. Ces différentes étapes ont suscité plusieurs fonctions permettant notamment de créer un mur, des ouvertures, l’orientation et la position des pièces. Pour cela, nous avons réutilisé les notions de liste, classes et méthodes.