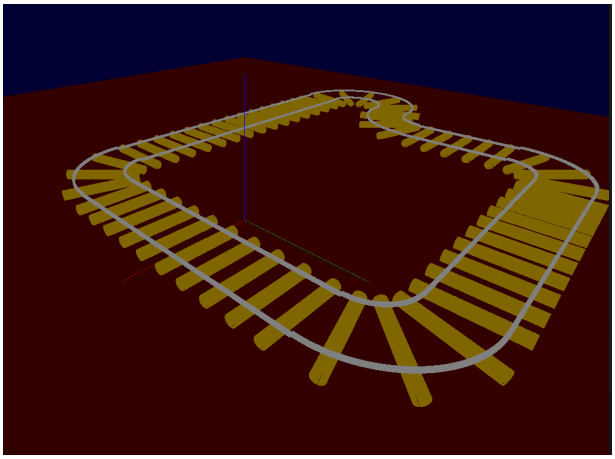
Synthèse d'images
Projet : A Railroad and a train

Il est temps de mettre en pratique les nouvelles notions obtenues pendant les cours de Synthèse d'Images. Pour se faire, vous devrez concevoir une toute petite simulation ferrovière. L'idée est de pouvoir représenter un chemin de fer défini dans un fichier de configuration (json)



Vous allez donc devoir concevoir votre version de cette application en `C/C++` avec OpenGL. Le but étant :

- De vous familiariser avec une modélisation de scène un peu plus complexe.
- De vous faire utiliser les fonctions de dessin 3D et appréhender les problématiques de construction 3D
- De vous faire implémenter une application interactive fonctionnelle.

Modalités de rendu

- **Nombre de personnes par projet :** 1
- **Livrable :**
  - Les sources `C/C++` ainsi qu'un système de compilation permettant de compiler et d'exécuter notamment sur l'environnement linux des machines de l'université. Pour ce faire, un système de type Git sera mis en oeuvre et mis à ma disposition avec accès aux différents comits. **Attention, un repo avec moins de 10 commits sera considérer comme du plagiat**.
  - Un **très court rapport** au format PDF décrivant votre projet : son mode d'installation (compilation et exécution), son fonctionnement (commandes utilisateurs), les résultats obtenus (fonctionnalités implémentées, capture d'écran, timing...), voire des détails techniques si vous souhaitez détailler une fonctionnalité ou un algorithme jugé original.
- **Date de rendu :** *officielle : 27 avril 2025*
- **Date de soutenance :** *pas de soutenance*

1 Eléments de modélisation de l'application

1.1 La scène

La scène est constituée d'un terrain plat sur le plan (x, y) décomposé en une grille de NxN cases (les cases ne sont pas forcément visibles mais le terrain, lui, l'est). Sur ces cases est représenté un circuit de chemin de fer. Quelque part sur ce circuit vous modéliserez un train et vous pouvez également ajouter des décors.

Les cases sont de taille 10x10 et la taille minimale N de la grille est de 10x10. Donc la scène représente un terrain d'au minimum 100x100.

1.2 Le circuit

Le circuit de chemin de fer est constitué de troncon de rails qui occupent chacun une case. Ce circuit est défini dans un fichier JSON qui a la forme suivante :

```
{
    "size_grid" : 10,
    "origin" : [-1,0],
    "path" : [
        [0,0],
        [0,1],
        [-1,1],
```

```
        [-2,1],
        [-3,1],
        [-3,-1],
        [-4,-1],
        [-4,-2],
        [-3,-2],
        [-2,-2],
        [-1,-2],
        [0,-2],
        [0,-1]
    ]
}
```

Dans ce fichier, `size_grid` correspond à la taille N de la grille. `origin` correspond à la position de la gare (cf. ci-dessous). Ces coordonnées correspondent au point inférieur gauche de la case. Ainsi la coordonnée $[0, 0]$ correspond à la case allant de 0 à 1 sur l'axe x et de 0 à 1 sur l'axe y.

Enfin `path` représente le circuit lui-même. Il sera constitué de tronçons de rails (cf. ci-dessous) créeant le circuit. Tout l'enjeu de l'application est donc de créer automatiquement ce circuit à l'aide de tronçons prédéfinis et des données présentes dans le fichier JSON.

Pour lire le fichier JSON, je vous suggère d'utiliser la bibliothèque de nlohmann qui est juste un fichier .hpp à ajouter à votre code et qui s'utilise très simplement.

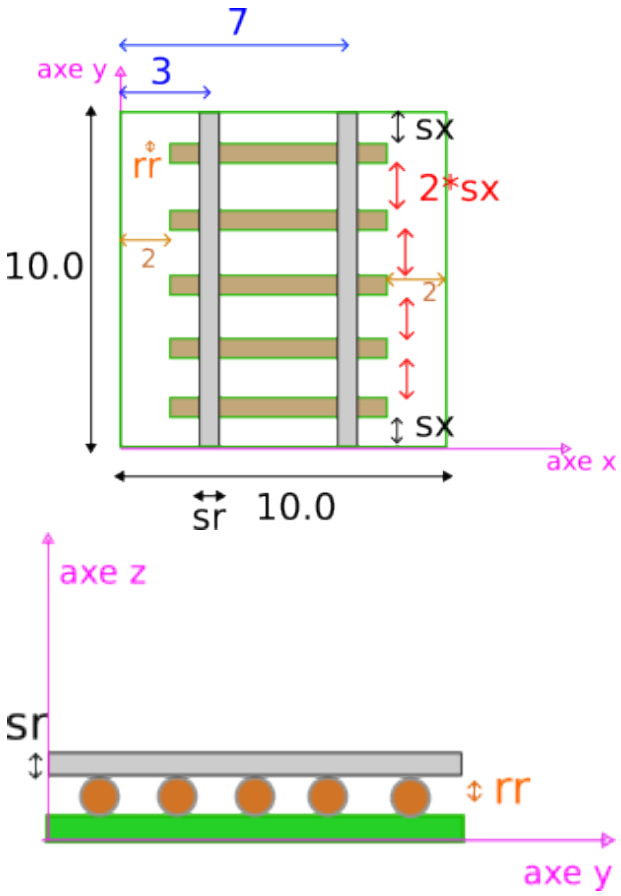
1.3 Les rails

Vous allez devoir concevoir deux, **et uniquement deux** types de rails : le "rail droit" et le "rail courbe".

1.3.1 Le rail droit

Le "rail droit" est défini avec 5 balasts et 2 rails. Chaque rail est centré sur l axe x au position 3 et 7 (ou au position `POS_X_RAIL1` et `POS_X_RAIL2` si vous voulez permettre une configuration plus souple). Chaque rail est un parallélépidède de section sr x sr et de longueur 10. sr devra être une constante modifiable dans votre code.

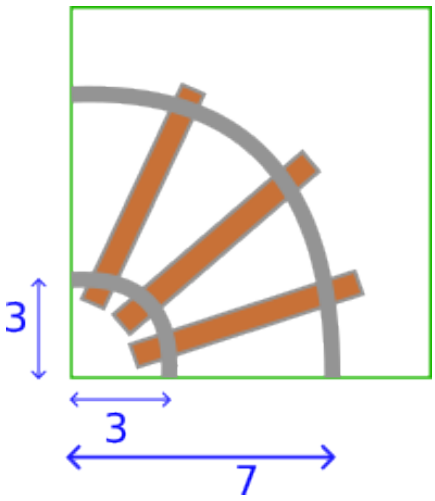
Chaque balast est un cylindre (si possible fermé aux 2 bouts mais ce n'est pas obligé) de **rayon** rr (toujours une constante modifiable dans le code) et de hauteur 6 (voir schéma), partant de x=2 à x=8. Là encore, les constantes 2 et 8 pourraient être modifiable. Les 5 balasts sont répartis uniformément le long du rail. Le premier est à sx en y puis il y a une distance de $2 * sx$ entre chaque balast (**balast non comprise**).



1.3.2 Le rail courbe

Le "rail courbe" est défini avec 2 rails courbés, un intérieur et un extérieur, et 3 balasts. Le rail intérieur est un troncon de carré de section *sr* x *sr* mais qui part sur l'axe x de la position 3 et arrive sur l'axe y en position 3 (ou là encore aux positions `POS_X_RAIL1` et `POS_X_RAIL2`). Le rail extérieur part de 7 en x et arrive en 7 en y. Pour construire ces éléments vous devrez exploiter la structure `convex_2D_shape`.

Les 3 balasts sont réparties uniformément sur les 90°. Le premier rail est à l'angle $\pi/12$. Ce sont toujours des cylindres de rayon *rr* et de taille 6.



1.4 Les autres éléments

Vous devrez rajouter à votre scène deux autres éléments graphiques que vous constituerez à l'aide des formes de base à votre disposition où encore de la structure `convex_2D_shape`. L'originalité et la complexité de ces formes seront pris en compte.

1.4.1 La gare

Le placement de cette gare est défini dans le fichier JSON évoqué précédemment. La gare peut être comprise dans la case indiquée (donc d'une taille de 10x10) mais vous pouvez faire le choix de la créer plus grande. Si cela recouvre une partie du circuit ce n'est pas grave.

1.4.2 Le train

Par ailleurs vous pouvez positionner où vous le souhaitez un train qui lui sera "posé" sur les rails du circuit et occupera une case (pas plus).

2 Interface Homme-Machine et l'application

Votre application sera lançable en ligne de commande et prendra en argument le nom du fichier JSON à charger.

Ensuite, une touche permettra de quitter l'application. Une autre touche activera ou désactivera l'éclairage (voir ci-après).

Enfin contrairement à ce que vous pouvez avoir dans les TD, vous devrez permettre à l'utilisateur de piloter une caméra de type FPS. Cette caméra n'aura pas besoin de regarder vers le haut

ou vers le bas mais devra pouvoir se déplacer librement dans la scène et regarder à 360° dans le plan (x, y).

3 Illumination

Vous devrez proposer les deux modes d'illumination tels que proposés dans le framework de vos TD. Vous aurez ainsi un mode "flat" (celui par défaut) et un mode "réaliste"

Pour l'éclairage "réaliste", vous devrez positionner au moins une source de lumière ponctuelle au niveau de l'avant de votre train. Par ailleurs au moins un élément de votre application devra être texturé !

4 Eléments d'évaluation

4.1 Critères d'évaluation

Votre projet sera évalué sur les éléments suivants :
— La fidélité de la modélisation des tronçons de rails.
— La capacité à automatiquement construire le circuit en fonction du fichier JSON.
— La qualité et la fluidité de l'IHM.
— L'originalité et la complexité de la modélisation de la gare et du train
— La cohérence et la qualité de l'illumination.

4.2 Pour aller plus loin

Le respect des spécifications ci-dessus vous permettra d'avoir une note tout à fait correcte (15 ou 16). Pour aller plus loin dans le fun et la note, vous pouvez par exemple :
— Animer le train et en particulier lui faire suivre le circuit.
— Proposer plus de lumières.
— Rajouter des décors.
— Animer certains objets (la gare, le train, les décors)
— Ajouter des éléments dans le fichier de configuration JSON (mais attention à bien mentionner toute modification de ces spécifications dans votre rapport).

Tout autre ajout sera valorisé de toutes façons.