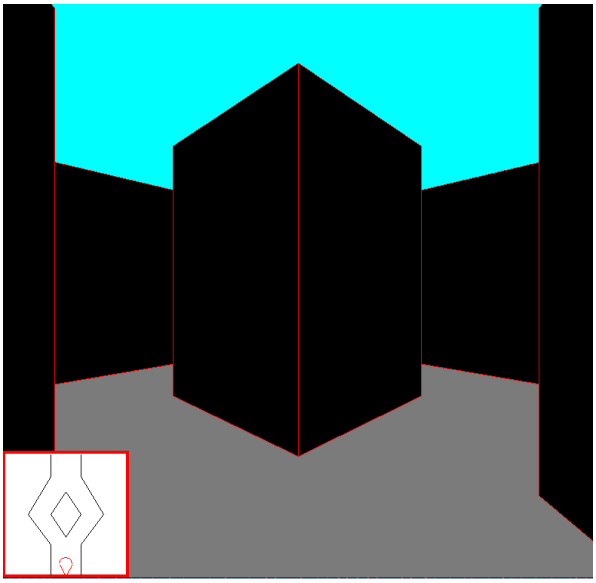


Moteur graphique

Rémi BUTET

Sommaire

Introduction

Le but de ce projet est de faire un moteur de jeu. Plus précisément d'implémenter le moteur 3D et le moteur physique.

La structure d'un moteur de jeu

Le moteur 3D 2	Arbres BSP* 2.1	Projections* 2.3	Clipping* 2.4	...
Le moteur physique 3	Déplacements*	Collisions*	Gravité	...
Le moteur de son	Musique	Ambiance		...
Autres	Gestion de l'IA	Gestion des textures		...

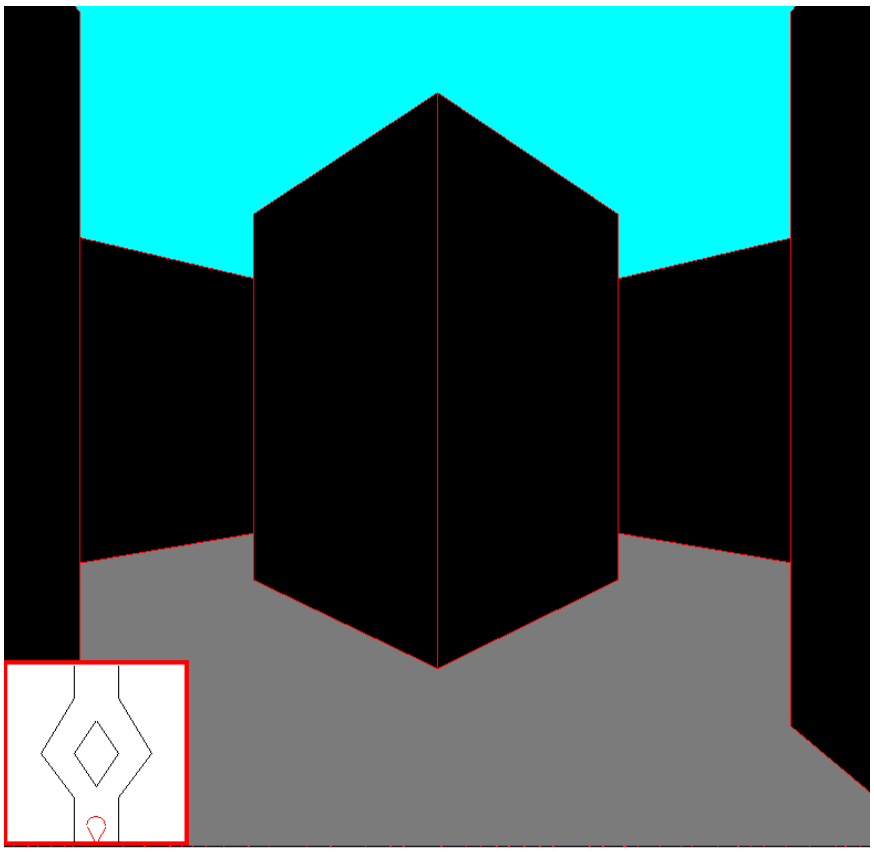


Figure 1 Ce que voit le joueur

1. Les arguments

Pour exécuter le programme il faut utiliser la commande :

```
$ ./projet_pfa <labyrinthe>
```

Cependant il est aussi possible de rajouter des arguments :

-mode <2D|3D> : Permet de choisir entre la 2D et la 3D

-fov <int> : Permet de choisir angle de vision

- dims : Permet de choisir la taille de la fenêtre lors de l'ouverture du programme

-scale <int> : Permet de choisir l'échelle de la Mini-Map

-map : Permet d'activer la Mini-Map

-step <int> : Permet de choisir la distance entre deux pas

-xmin <int> : Permet de choisir la distance minimale pour voir un objet

-xmax <int> : Permet de choisir la distance maximale pour voir un objet

-debug : Permet de faire afficher les informations dans la console

-yaw <float> : Permet de choisir la sensibilité lors du déplacement de la souris

Exemple :

```
$ ./projet_pfa labyrinthes/labyrinthe1.lab -mode 3D -map
```

2. Les touches

Une fois le programme exécuté, il est possible d'interagir avec en appuyant sur des touches du clavier. Cependant il n'est possible d'appuyer que sur une touche à la fois.

Voici la liste des touches :

Z : Pour avancer

S : Pour reculer

Q : Pour aller à gauche

D : Pour aller à droite

A : Pour tourner la tête à gauche

E : Pour tourner la tête à droite

R : Afficher ou cacher la Mini-Map

F : Changer de mode de vue (2D ou 3D)

W : S'accroupir ou se lever

X : Courir ou arrêter de courir

Espace : Pour sauter

Echap : Pour arrêter le programme

Mouvement de la souris :

Gauche : Pour tourner la tête à gauche

Droite : Pour tourner la tête à droite

3. Le programmes

a. Listener

Le module Listener permet d'écouter les mouvements de la souris et des touches du clavier. Puis lance une exception qui est gérée par le module main pour réaliser l'action.

b. Render

Le module Render permet de modifier le mode vu (2D ou 3D) et d'exécuter le Render choisi.

c. Render2D

Permet de faire afficher la map en 2D et de faire afficher la Mini-Map.

d. Render3D

Permet de faire afficher la map en 3D, pour simplifier la compréhension du code, je l'ai divisé en 3 fonctions distinctes.

e. Physic

Permet de gérer la collision du joueur avec un mur. En vérifie que le segment du joueur ne rentre pas en collision avec le segment d'un mur.

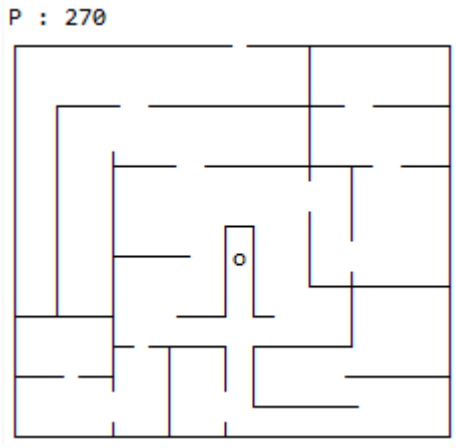
f. Parse_lab

Permet de charger le fichier du labyrinthe. En plus de la fonction initiale j'ai implémenté une autre fonction qui permet de créer beaucoup plus simplement un labyrinthe.

4. Le fichier labyrinthe

Pour simplifier la création d'un labyrinthe j'ai implémenté une fonction qui permet dans créer un à partir de caractères : `─`, `│`, `┌`, `┐`, `└`, `┘`, `├`, `┤`, `┬`, `┴`, **`o`** (La position du joueur), **`P : <angle>`**

Exemple : **labyrinthe1.lab**




```
$ ./projet_pfa labyrinthes/labyrinthe1.lab -mode 3D -map
```

Conclusion

Ce projet m'a permis de me rendre compte de la simplicité de la création d'un moteur graphique.

J'ai aussi appris beaucoup de choses sur le langage Ocaml cependant je préfère tous de même le langage Java. Qui depuis la version 8 implémente toutes les fonctionnalités de la programmation fonctionnelle.