The Maze

Bachelor 3
Développement logiciel
Mael Pena
Verner Boisson
Projet 2019-2020

Lien du projet The Maze

Sommaire

Introduction

Nous avions comme projet, de fin d'année de bachelor 3, à Ynov Bordeaux, d'effectuer un éditeur et un résolveur de labyrinthe en deux applications graphiques qui communiquent avec une API qui fait le pont avec une base de donnée. Tout cela hébergé en local.

1. Présentation du projet

1.1 Règles

Le labyrinthe est composé de murs qu'il est impossible de traverser, le niveau comporte également des obstacles, tels que des piège et des ralentissements . L'objectif principal du jeu est de trouver un point d'arrivé pour terminer la partie.

1.2 Comportement

L'utilisateur va pouvoir créer des niveaux, les modifier ou les supprimer dans un IHM. Ensuite dans un autre IHM, une IA va réaliser le niveau choisi avec pour objectif de trouver le point d'arrivé en trouvant le chemin nécessaire.

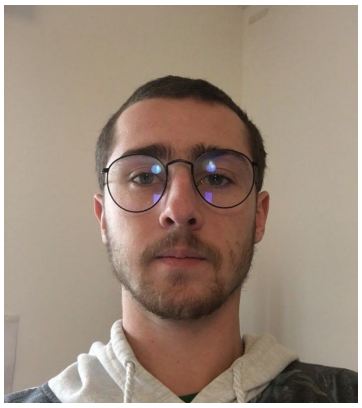
1.3 Condition de réussite

L'IA de résolution doit atteindre le point d'arriver.

2. Présentation de l’équipe

2.1 Mael Pena

Mael Pena est étudiant en Bachelor 3 à Ingésup - Ynov Bordeaux en spécialité développement logiciel.



2.2 Verner Boisson

Verner Boisson est étudiant en Bachelor 3 à Ingésup - Ynov Bordeaux en spécialité développement logiciel.

3. Labyrinthe

3.1 Taille

Le labyrinthe a une taille allant de 5x5 à 31x31.

3.2 Format

Le labyrinthe a une forme carré.

3.3 Composition

Le labyrinthe est défini par un point de départ et un point d'arrivé. Le labyrinthe comporte des murs qui délimite des chemins. Le labyrinthe peut comporter des pièges ou des obstacles.

3.4 Les pièges

Le piège ralenti la progression de l'IA.

3.5 Point de départ

Le point de départ est choisi par l'utilisateur.
Il doit y avoir obligatoirement un seul et unique point de départ.
Il s'agit du point de départ de l'IA.

3.6 Point d'arrivé

Le point d'arrivé est choisi par l'utilisateur.
Il doit y avoir obligatoirement un seul et unique point d'arrivé .
Il s'agit du point d'arrivé de l'IA.

3.7 l'IA

l'IA apparaît sur le point de départ et a pour objectif de trouver le chemin vers le point d'arrivé.

3.8 Story Time

Story time prévu à la conception du projet.

3.8.1 IHM Edition de niveau :

- En tant qu'utilisateur, je veux créer un niveau.
- En tant qu'utilisateur, je veux mettre un titre au niveau.
- En tant qu'utilisateur, je veux mettre un auteur au niveau.
- En tant qu'utilisateur, je veux modifier un niveau.
- En tant qu'utilisateur, je veux choisir un niveau.
- En tant qu'utilisateur, je veux voir les niveaux.
- En tant qu'utilisateur, je veux mettre des murs dans le niveau.
- En tant qu'utilisateur, je veux modifier l'emplacement d'arrivée.
- En tant qu'utilisateur, je veux modifier l'emplacement de départ.
- En tant qu'utilisateur, je veux mettre des obstacles.
- En tant qu'utilisateur, je veux choisir un obstacle.

3.8.2 IHM Résolution :

- En tant qu'utilisateur, je veux voir les niveaux.
- En tant qu'utilisateur, je veux choisir un niveau.
- En tant qu'utilisateur, je veux voir les mouvements de l'IA.

4. Technologies

4.1 Base de donnée

La base de donnée est en mongo, une base de donnée orienté document. Un choix adapté à notre projet où chaque document représente un labyrinthe.

4.2 Interface de programmation

L'interface de programmation ou API est en NodeJS. Un langage qui permet de faire des serveur web accessible grâce à des package tel que express, et la communication avec la base de donnée se fait via l'ORM mongoose.

4.3 Interface d'édition de labyrinthe

L'interface d'édition de labyrinthe est faite en Java, un langage orienté objet qui possède nativement Swing une librairie qui permet la gestion de l'interface. Il s'agit donc d'un langage adapté à la conception de cette interface.

4.4 Interface de résolution

L'interface de résolution de labyrinthe est aussi fait en Java pour les mêmes raison que l'interface d'édition de labyrinthe.

5. La base de donnée

5.1 Mongo

Nous avons choisi MongoDB pour ce projet car nous avons estimé qu'une base de donnée noSQL orienté document était plus adapté au stockage de labyrinthe. Nous avons choisi MongoDB comme base de donnée orienté document car c'est la plus populaire et qu'elle possède une documentation à jour, de plus nous l'avons étudié en cours.

5.2 ORM

Nous avons choisi Mongoose comme ORM car c'est un ORM en NodeJS, le langage de l'API. C'est un ORM populaire et qui a une documentation à jour et nous avons aussi étudié cette ORM en cours.

5.3 Schema

```
{
  id: {type: Number, required: true, unique: true},
  title: {type:String},
  author: {type:String},
  maze: [[{type:String}]],
  createdAt: {type: Date, default: new Date()},
  timer: {type: Number},
  movement: {type: Number},
}
```

5.4 Types

- id : L'id est un nombre qui permet d'identifier le labyrinthe de façon unique, un id est obligatoire pour chaque labyrinthe.
- title : Le titre est une chaîne de caractère.
- author : L'auteur est une chaîne de caractère.
- createdAt : La date de création est une date.
- timer : Le temps est un nombre en milliseconde qui représente le temps que met l'IA à résoudre le labyrinthe.
- movement : Le mouvement correspond au nombre de déplacement que met l'IA pour résoudre le labyrinthe.
- maze : Le labyrinthe est un tableau de tableau de chaîne de caractère. Il est utilisé comme un tableau de tableau de caractères mais Mongo n'a pas ce type.

À chaque case du tableau, une lettre correspond au type :

- 'F' : Free (correspond à une case vide)
- 'W' : Wall (correspond à un mur du labyrinthe)
- 'S' : Start (correspond au point de depart)
- 'G' : Goal (correspond au point d'arrivée)
- 'M' : Mud (correspond à un piège)

5.5 Justification

Nous avons utilisé ce schéma de donné car il permet d'avoir un document simple et facilement utilisable ce qui limite le traitement des requête en java qui n'est pas très intuitif.

6. L’interface de programmation (API)

6.1 NodeJS

Le NodeJS permet de faire des API et des web-serveur de façon simple avec Express. La communauté NodeJS étant très active, il y a beaucoup de package sur NPM et qui sont à jours. Nous avons eu l’occasion de faire des API en NodeJS plusieurs fois et nous avons également utilisé le package Mongoose ce qui a facilité ce choix.

6.2 Fonctionnalité

Notre API est un CRUD classique qui fait le pont entre nos IHM et la BDD. Elle permet la lecture, l’écriture, la modification et la suppression de labyrinthe dans la base de donnée. Nous n’avons pas besoin d’interface graphique côté web donc l’API renvoie uniquement du JSON.

7. L’interface d’édition

7.1 Java

Java nous paraissait être le meilleur choix pour faire une interface Homme-Machine. C’est un langage orienté objet, ce qui nous a paru indispensable pour ce projet. C’est aussi un langage qui possède une documentation à jour ainsi qu’une communauté très active. Nous avons étudié ce langage dans la conception de programme utilisant la librairie SWING qui permet de gérer les interfaces graphique.

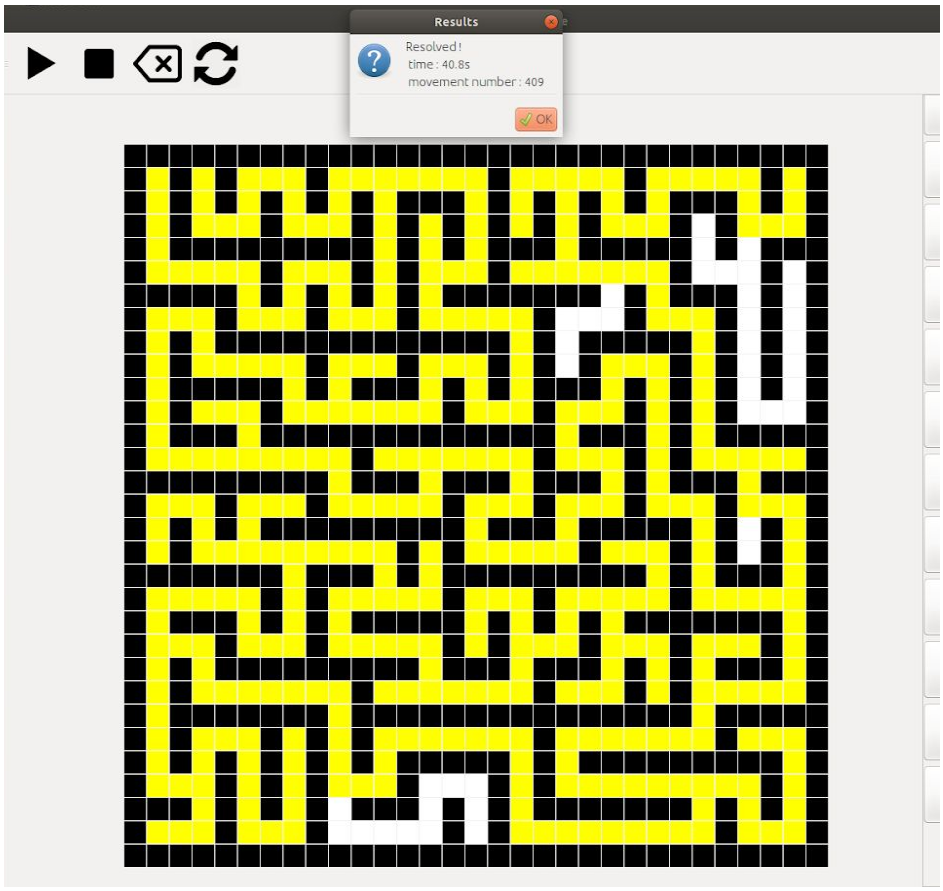
7.2 Fonctionnalité

- Choisir la taille
- Positionner un départ
- Positionner une arrivée
- Positionner un obstacle
- Positionner un piège
- Positionner un mur
- Générer un labyrinthe aléatoire
- Effacer une position
- Effacer tout le labyrinthe
- Remplir tout le labyrinthe de mur
- Enregistrer le labyrinthe
- Modifier un labyrinthe
- Supprimer un labyrinthe
- Mettre un titre
- Mettre un auteur

8. L’interface de résolution

8.1 Java

Cet interface Homme-Machine était initialement prévu en C++, mais nous avons finalement choisi de la faire également en java car de nombreuse fonctionnalitées était similaire à l’interface d'édition.



8.2 Fonctionnalité

- Choisir un labyrinthe
- Rafraîchir la liste de labyrinthe
- Lancer la resolution
- Enregistrer le résultat
- Récupère tous les labyrinthes
- Arrêter la résolution

- Effacer la résolution

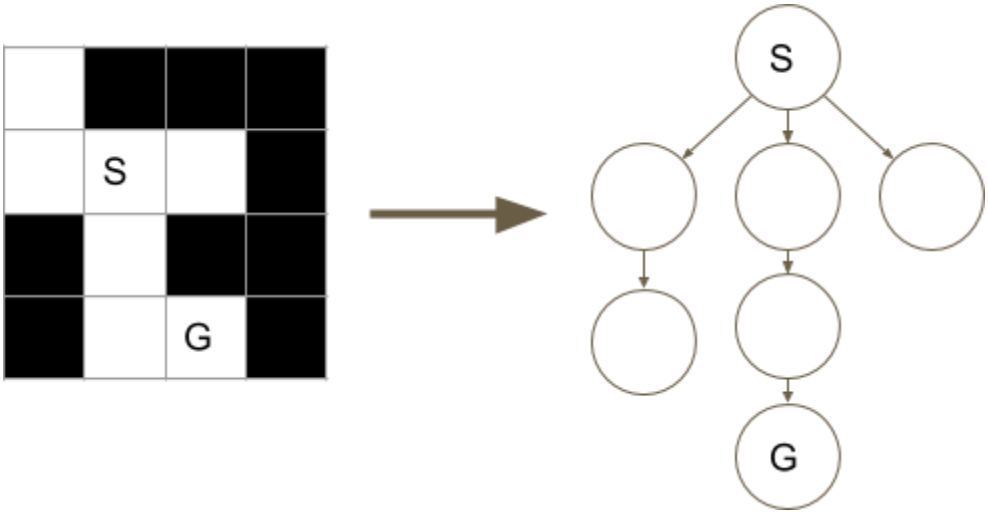
9. Algorithme

9.1 Introduction

Nous avons choisi l'algorithme DFS (Depth-First Search) l’algorithme de parcours en profondeur. Nous avons trouvé que c’était un algorithme adapté à la résolution et à la génération aléatoire de labyrinthe.

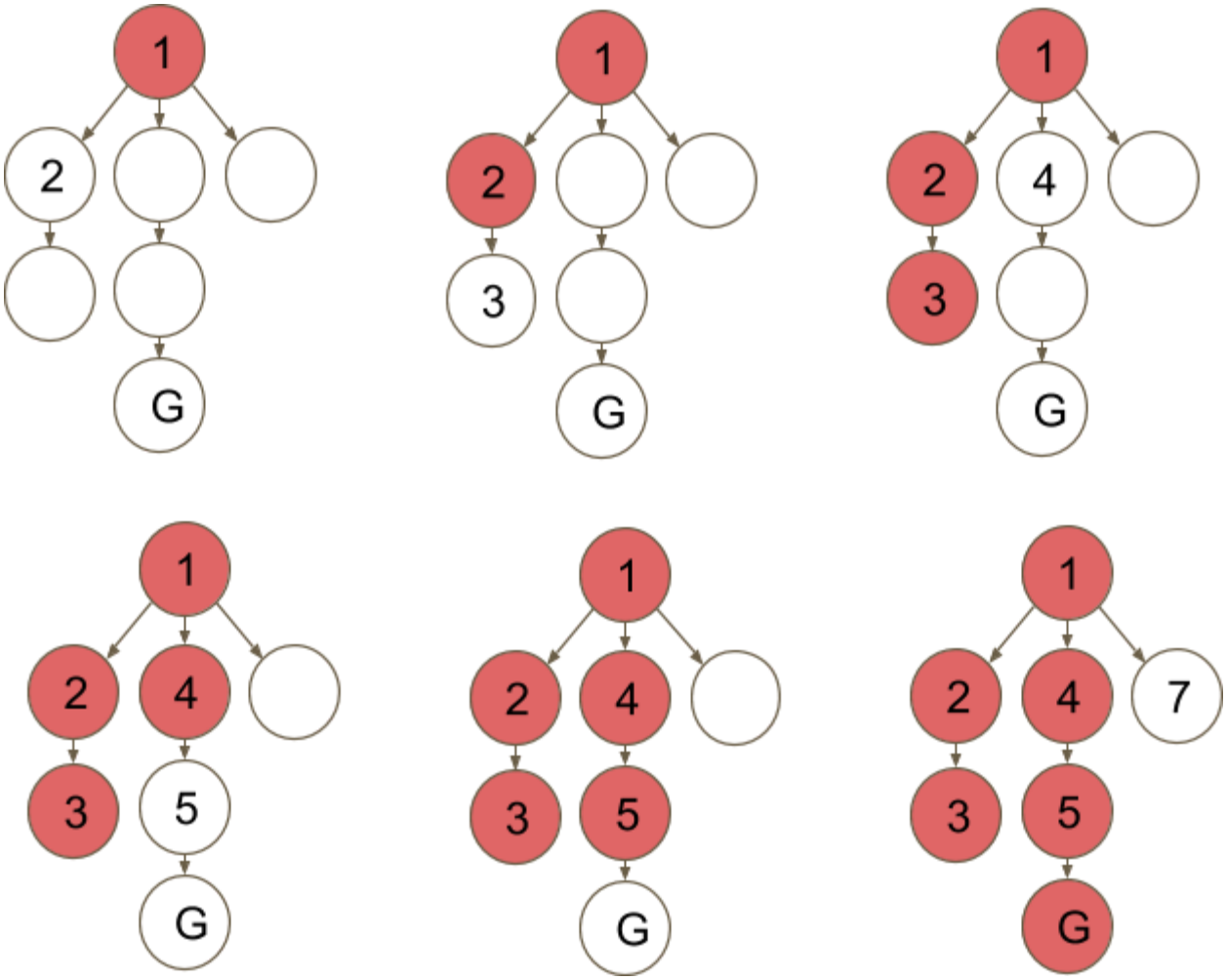
9.2 Représentation en graphe

Un labyrinthe peut se représenter sous la forme d’un graphe avec le départ comme racine.



9.3 Principe

Le principe du DFS :



10. Conclusion

C’était un projet intéressant, qui nous a permit de nous confronter à des problématiques d’algorithmie complexes. Mais nous avons su être à la hauteur et mener à bien nos objectifs.

Ce projet nous a permis de consolider nos compétence en java ce qui nous a aider comprendre la philosophie de la programmation orienté objet. Il nous a également permis d’utiliser des librairie comme Swing pour nous familiariser avec les interfaces graphiques ou HttpURLConnection afin de communiquer avec une base de donnée via une api. Nous avons également appris à utiliser des Threads, des fonctions récursives ainsi que le fonctionnement des graphes.

En résumé, ce projet nous a apporté énormément de compétences et de capacitées. Nous avons vraiment apprécié développer ce projet et étions enjoué a chaque nouvelle fonctionnalité apporté. C'était pour moi un des plus gros projet de ma scolarité et j’ai hâte de commencer le prochain.

Cependant pour nos prochains projet, nous pouvons améliorer notre gestion du temps.

11. Suggestion d’amélioration du projet

- Visualisation des scores des labyrinthes
- Avoir différent algorithme de résolution
- De Nouveaux obstacles
- De pouvoir mettre la résolution en pause