Sam FREDDI

CFPT Ecole Informatique

Maître d’apprentissage : Pascal Henauer

Résolution d’un labyrinthe par la méthode d’un arbre de recherche

DOCUMENTATION TECHNIQUE

Table des matières

Table des matières

[2 Introduction 3](#_Toc166594034)

[2.1 Contexte 3](#_Toc166594035)

[2.2 But du projet 3](#_Toc166594036)

[3 Rappel du cahier des charges 3](#_Toc166594037)

[3.1 Description de l’application 3](#_Toc166594038)

[3.2 Organisation du suivi 4](#_Toc166594039)

[3.3 Livrables 4](#_Toc166594040)

[3.4 Matériel et outils nécessaires 4](#_Toc166594041)

[4 Méthodologie de travail 6](#_Toc166594042)

[4.1 La méthode en 6 étapes 7](#_Toc166594043)

[1- S’informer 7](#_Toc166594044)

[2- Planifier 7](#_Toc166594045)

[3- Décider 7](#_Toc166594046)

[4- Réaliser 8](#_Toc166594047)

[5- Contrôler 8](#_Toc166594048)

[6- Évaluer 8](#_Toc166594049)

[5 Méthodologie de sauvegarde 8](#_Toc166594050)

[5.1 Méthode du 3-2-1 8](#_Toc166594051)

[6 Planification 8](#_Toc166594052)

[6.1 Planification prévisionnelle 9](#_Toc166594053)

[9](#_Toc166594054)

[6.2 Planification effective 10](#_Toc166594055)

[7 Analyse des fonctionnalités 11](#_Toc166594056)

[7.1.1 Interface utilisateur et fonctionnalité 12](#_Toc166594057)

[7.1.2 Menu principale de l’application. 12](#_Toc166594058)

[7.1.3 Menu des sauvegardes 13](#_Toc166594059)

[7.1.4 Page de simulation 14](#_Toc166594060)

[7.1.5 Panel de sauvegarde 15](#_Toc166594061)

[7.1.6 Panel choix du mode 15](#_Toc166594062)

[7.1.7 Sélectionner une entrée / sortie 16](#_Toc166594063)

[7.1.8 Panel d’information 17](#_Toc166594064)

[8 Analyse organique 18](#_Toc166594065)

[8.1 Structure des dossiers 18](#_Toc166594066)

[8.1.1 Dossier Asset 19](#_Toc166594067)

[8.1.2 Dossier Script 19](#_Toc166594068)

[8.1.3 Dossier Prefab 21](#_Toc166594069)

[8.2 Architecture 21](#_Toc166594070)

[8.3 Chemin des données 21](#_Toc166594071)

[8.3.1 Scène Menu 21](#_Toc166594072)

[8.3.2 Game Manager du main Menu 22](#_Toc166594073)

[Disposition du menu 23](#_Toc166594074)

[8.3.3 Main simulation 23](#_Toc166594075)

[8.4 Diagramme d’une méthode complexe 25](#_Toc166594076)

[8.4.1 Blocs 25](#_Toc166594077)

[25](#_Toc166594078)

[8.4.2 Système de connecteur 26](#_Toc166594079)

[8.4.3 Voici le diagramme simplifié de la génération du labyrinthe. 27](#_Toc166594080)

[27](#_Toc166594081)

[8.4.4 Comment le bot avance 28](#_Toc166594082)

[8.4.5 Système de RayCast 28](#_Toc166594083)

[8.4.6 Diagramme machine d’état 29](#_Toc166594084)

[8.5 Fonctionnement des modes de simulations 30](#_Toc166594085)

[8.5.1 Diagramme sur le mode manuel 31](#_Toc166594086)

[8.5.2 Diagramme sur le mode semi-automatique 31](#_Toc166594087)

[8.5.3 Diagramme sur le mode automatique 32](#_Toc166594088)

[8.6 Assets 33](#_Toc166594089)

[9 Les tests 33](#_Toc166594090)

[9.1 Périmètre 33](#_Toc166594091)

[9.2 Environnement 33](#_Toc166594092)

[9.3 Scénarios 34](#_Toc166594093)

[9.4 Suivi évolutif des tests 39](#_Toc166594094)

[9.5 Rapport de test 40](#_Toc166594095)

[10 Conclusion 40](#_Toc166594096)

[10.1 Difficultés rencontrés 40](#_Toc166594097)

[10.2 Amélioration possibles 40](#_Toc166594098)

[10.3 Bilan personnel 40](#_Toc166594099)

[10.4 Remerciements 40](#_Toc166594100)

[11 Bibliographie 40](#_Toc166594101)

[12 Glossaire 40](#_Toc166594102)

[13 Annexes 41](#_Toc166594103)

# Introduction

## Contexte

Afin de valider notre formation de CFC en Informatique au CFPT-I, il nous est demandé d’effectuer un travail pratique individuel en fin de 4e année. Pour ce travail, il m’est demandé de créer une application génération et solutions de labyrinthe en 3D.

## But du projet

Le but du projet est de pouvoir générer aléatoirement un labyrinthe et grâce à un algorithme de recherche de pouvoir voir et définir le chemin ou les chemins les plus courts jusqu’aux sorties possibles. L’application comportera aussi des sauvegardes manuelles et automatiques.

# Rappel du cahier des charges

## Description de l’application

Le projet du TPI décrit vise à créer un logiciel capable de générer un labyrinthe de manière aléatoire avec des blocs prédéfinis et de trouver le chemin le plus court entre une entrée et une sortie dans un labyrinthe.

Lorsqu’une génération de carte est terminée. L’utilisateur a la possibilité de sauvegarder la carte et de l’importer depuis le menu.

L'algorithme utilisé repose sur la recherche dans un arbre, avec un bot doté de capteurs et d'une trace du parcours. À chaque nœud, le bot est dupliqué dans toutes les directions possibles. Les bots qui rencontrent un mur meurent, laissant derrière eux une trace rouge affichée pendant une demi seconde. A la fin de l’algorithme seule ceux qui trouvent la ou les sorties seront afficher avec une trace verte et si aucune sortie n’est trouvé une trace jaune sera assigné a tracé le plus proche de la sortie. Les bots en cours de recherche auront une trace bleue.

L’algorithme de recherche est divisé en 3 modes distincts. Le mode manuel, semi-automatique et automatique.

Le mode manuel a pour but de trouver le chemin le plus court entre une entrée et une sortie donné par l’utilisateur. Si aucun chemin ne relie l'entrée et la sortie, le programme le fait savoir à l'utilisateur et lui propose 4 options. Changer de sortie. Changer de mode, relancer une génération ou importer une carte sauvegardée.

Le mode semi-automatique a pour but de trouver toutes les sorties possibles avec une entrée donnée par l’utilisateur. L’algorithme va tester tous les chemins possibles avec l'entrée donnée et nous retourner toutes les sorties possibles.

L'objectif du mode automatique est d'évaluer la performance dans le labyrinthe. L'algorithme explore chaque entrée du labyrinthe pour déterminer le nombre de sorties, le temps nécessaire pour achever le labyrinthe et la distance parcourue par les bots depuis cette entrée. Après avoir effectué le test sur chaque entrée, le programme nous retourne le score du labyrinthe. Un tableau avec le numéro de chaque entrée est visible. Quand l’utilisateur clique sur une entrée du tableau, le programme nous affiche le parcours effectué par les bots depuis cette entrée et nous donne toutes les valeurs liées à cette entrée.

Si le labyrinthe tester possède un score qui se trouve dans le top 10. Il est automatiquement sauvegardé.

Une interface utilisateur est envisagée pour le projet, avec la possibilité de sélectionner une entrée et une sortie, de visualiser le processus de recherche.

De plus, les prérequis incluent une partie de la construction d'un labyrinthe automatique, le développement d'un bot capable de se déplacer tout droit et de savoir dans quel type de nœud il se trouve ainsi que la classe de sauvegarde des labyrinthes.

Ces compétences sont décrites dans le plan de formation, acquises lors des ateliers New Technologie et TPI.

## Organisation du suivi

|  |  |  |
| --- | --- | --- |
| Élève | Maître d’apprentissage | Experts |
| FREDDI Sam | Pascal HENAUER | Thomas TETART |
| / | / | Brian NYDEGGER |

## Livrables

* Rapport de projet
* Manuel utilisateur
* Résumé du rapport du TPI
* Journal de travail

## Matériel et outils nécessaires

|  |  |
| --- | --- |
| Matériel | Version / Model |
| Poste de travail (PC) avec un système d’exploitation Windows 10 Pro. | 10.0.19045 |
| Clé USB | 16 Go |
| SSD Samsung (Disque de travail) | 870 EVO - 250 Go |

|  |  |
| --- | --- |
| Outil | Version |
| Visual Studio Code | 1.82.0 |
| GitHub (Terminal) | 2.34.1 |
| Unity3D | 3.7.0 |
| Editor version | 2022.3.17f1 |
| Google Drive | *En ligne* |
| Word 2016 | 16.0 |
| Excel 2016 | 16.0 |

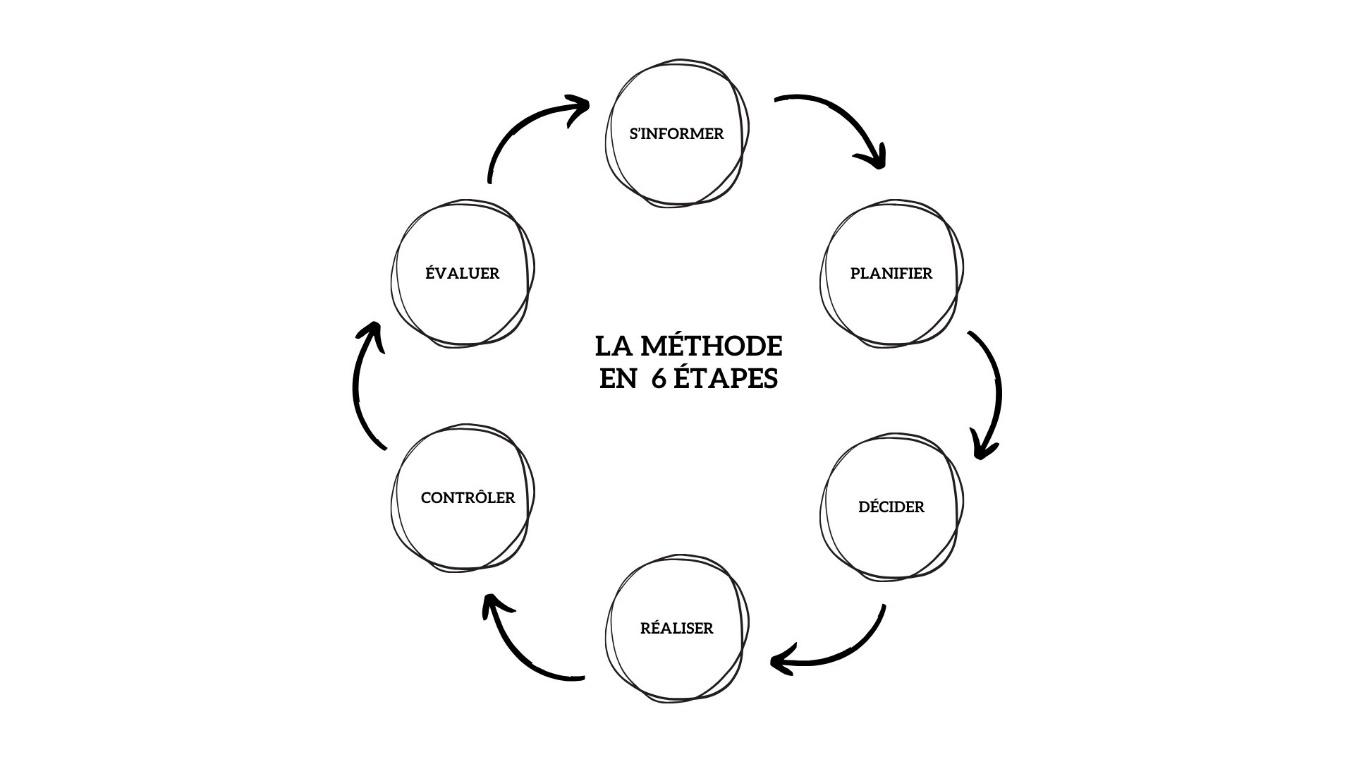
# Méthodologie de travail

Afin de réaliser ce travail de la manière la plus efficace possible j’ai choisi d’utiliser la méthode dite « en 6 étapes » qui se prête particulièrement bien aux travaux personnels comme le TPI.

Vous pourrez remarquer au long de la lecture que, dans certaines parties du document, il sera noté l’étape de la méthode correspondante.

Ci-après une explication simple de la méthode ainsi que chaque étape afin de comprendre la manière dont le travail a été réalisé.

## La méthode en 6 étapes

Cette méthode consiste à passer par plusieurs étapes pour réaliser un cahier des charges défini afin d’être le plus efficace possible. 

*Figure SEQ Figure \\* ARABIC 1 : Schéma de la méthode en 6 étapes*

Les étapes sont les suivantes :

### S’informer

Je dois comprendre et cerner exactement ce qu'on me demande avec toutes les contraintes qui me sont imposés. Dans le cas où il me manquerait une ou plusieurs informations pour la pleine compréhension de mon mandat, je vais chercher les informations et je vais m'informer afin que je sois en mesure de comprendre pleinement le travail demandé.

### Planifier

Il faut que je détermine les tâches nécessaires à la réalisation du mandat, que je les mette en ordre correctement en fonction de l'importance d'une tâche par rapport à une autre. Il faut également fixer la durée estimée de chaque tâche.

### Décider

Comment réaliser ce qui a été planifié. S'il y a plusieurs possibilités pour arriver au but du mandat, laquelle est la plus optimale ? C'est à cette étape que je dois prendre toutes les décisions de directions que le travail doit prendre pour arriver au but du mandat, le tout en respectant les contraintes imposées.

### Réaliser

Je dois me mettre à la tâche, réaliser ce qui est prévu dans ma planification en respectant les choix que j'ai fait lors de l'étape 3. Dans le cas où je dois faire autrement que ce qui a été prévu avant, avant de faire quoi que ce soit je dois me poser la question suivante : « Quelles sont les conséquences d’éventuelles dérogations à la planification ? »

### Contrôler

Contrôle du travail effectué, correspond aux tests à réaliser durant le travail. Le but est de contrôler que tout le travail effectué à l'étape 4 a correctement été fait et que le tout fonctionne parfaitement.

### Évaluer

On va chercher à savoir ce qui a bien fonctionné, ce qui pourrait être amélioré et dans le cas où il y a des choses qui ne fonctionnes pas, savoir pourquoi elles ne fonctionnent pas et ce qu'il faudrait faire et améliorer pour le faire fonctionner.

# Méthodologie de sauvegarde

Lors de la réalisation de ce travail il est de mon ressort de garantir les sauvegardes de l’avancement du projet, pour cela je vais utiliser la méthode du 3-2-1 qui garantit la sauvegarde du projet sur plusieurs supports.

## Méthode du 3-2-1

Cette méthode de sauvegarde est très simple même si son titre n’est pas très explicatif.

Pour faire simple, afin de respecter cette méthode il faut sélectionner 3 supports de sauvegarde différents.

Pour ma part les 3 supports seront les suivants : clé USB, cloud (Google Drive) et en local directement sur mon SSD de travail. J’ai sélectionné précisément ces 3 supports pour 1 seule raison, il peut arriver n’importe quoi lors de l’avancée du travail, j’aurais dans tous les cas un support contenant le projet et je minimise grandement les risques de perte d’avancement ou bien même de perte de projet.

# Planification

Dans le cadre du TPI, 2 planifications sont à effectuer. La planification prévisionnelle, qui est le reflet de la prévision des tâches, du temps passé sur chacune d’elles et les jalons. Puis, la planification effective qui est le reflet de comment s’est réellement déroulé le travail.

Ces 2 planifications sont importantes pour avoir une vue globale tout le long de l’avancement du projet pour savoir quand je suis en retard ou en avance sur le planning et pour pouvoir continuer le travail en conséquence. Mais également pour avoir une vue d’ensemble à la fin du projet pour en apprendre plus sur moi-même et pouvoir affiner mes planifications prévisionnelles futures.

Ci-dessous vous pouvez retrouver mes 2 planifications.

Cette partie correspond à l’étape 2 et 3 de la méthode en 6 étapes.

## Planification prévisionnelle

## 

## Planification effective

# Analyse des fonctionnalités

**Générer une carte aléatoire** : Cette fonctionnalité permet à l'utilisateur de créer des labyrinthes avec un algorithme de génération aléatoire.

**Choisir de placer une entrée et une sortie** : Permet à l'utilisateur de personnaliser le labyrinthe en décidant où se trouvent l'entrée et la sortie.

**Voir le chemin le plus court entre l'entrée et la sortie** : Un algorithme de recherche entre une entrer et une sortie. Le programme trouve la sortie s’il en existe une où nous donne le chemin le plus proche.

**Tester une entrée et regarder toutes les sorties possibles** : Permet aux utilisateurs de tester une entrer différente et de percevoir toute les sorties possibles.

**Tester tout le labyrinthe et regarder le score obtenu** : Fournit une évaluation de la performance globale du labyrinthe, en prenant en compte des facteurs tels que le temps et la distance parcourue et le nombre de sortie cumulé par sortie.

**Afficher la liste des entrées et donner les informations suivant l’entrée :** Cette fonctionnalité fournit des détails sur chaque entrée du labyrinthe. Nombre de sortie possible, distance parcouru, score de l’entrée.

**Sauvegarder une carte avec un nom choisi** : Permet aux utilisateurs de sauvegarder leurs labyrinthes.

**Afficher la liste des cartes sauvegardées et du top 10** : Fournit un répertoire des cartes sauvegardées ainsi qu'un classement des meilleurs labyrinthes.

**Pouvoir charger une carte sauvegardée** : Permet aux utilisateurs de reprendre un labyrinthe intéressant.

### Interface utilisateur et fonctionnalité

Je vais détaille ici chaque interface et décrire les possibles que l’utilisateur peut avoir.

### Menu principale de l’application.



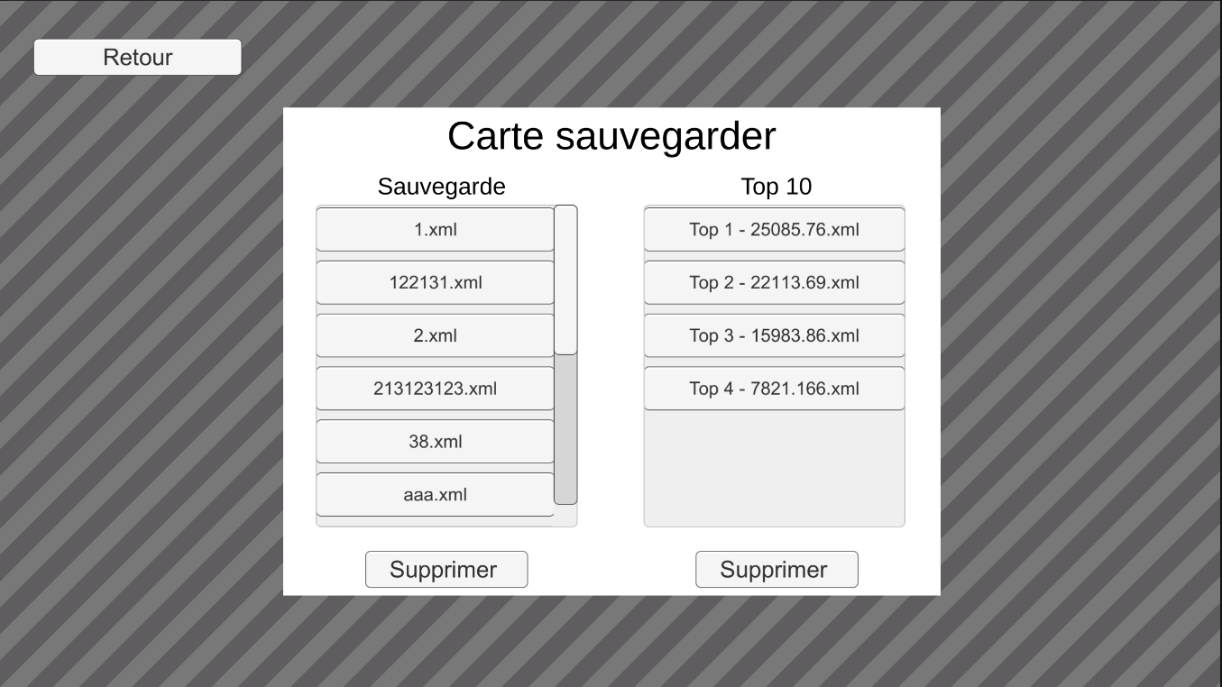
3

2

1

1. Le bouton généré labyrinthe permet à l’utilisateur de commencer une nouvelle génération d’un labyrinthe.
2. Le bouton Carte sauvegarder permets à l’utilisateur d’aller sur la partie du menu qui gère les sauvegardes. Il y verra la liste des cartes qu’il a sauvegarder ainsi que les sauvegardes du top 10.
3. Le bouton quitter sert à quitter l’application.

### Menu des sauvegardes



4

3

2

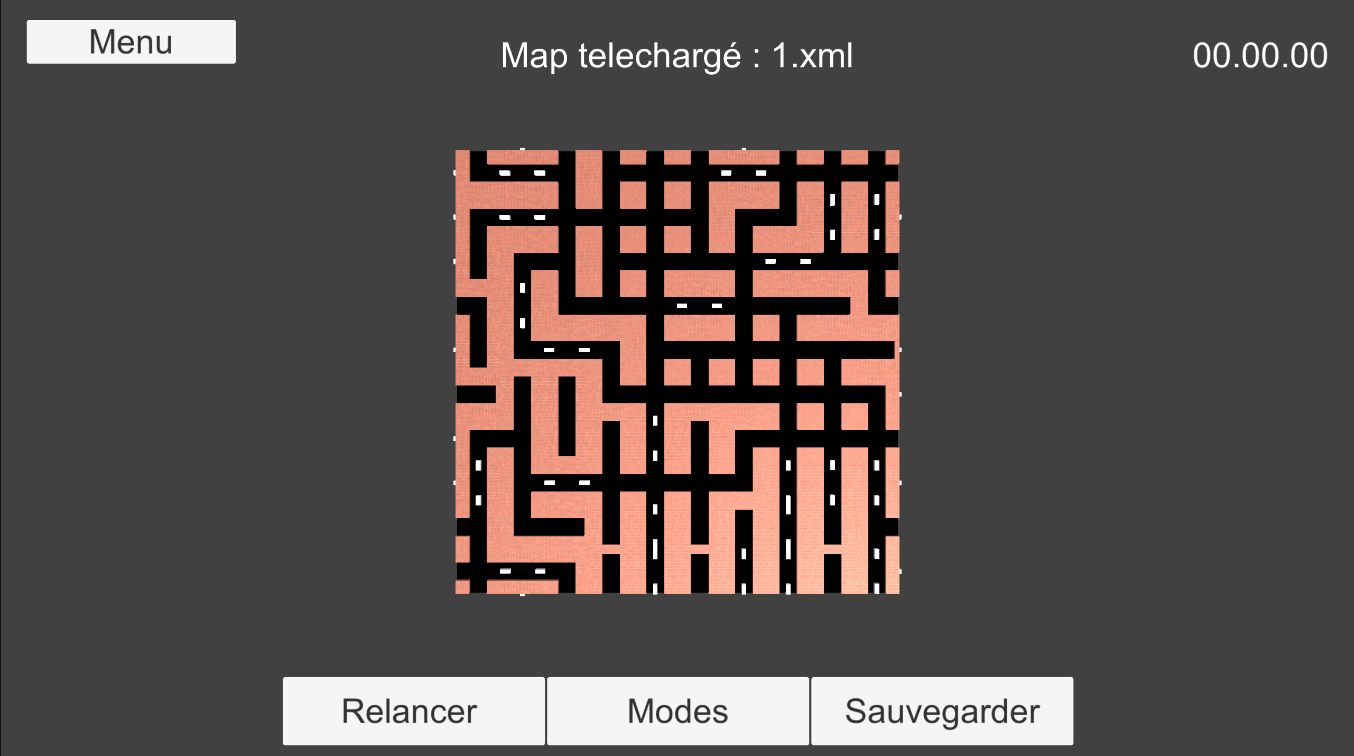
1

5

1. C’est la liste des labyrinthes que l’utilisateur à décider de sauvegarder. C’est une liste de bouton. S’il clique sur 1.xml. Le premier bouton de la liste. Le labyrinthe va être lancer la scène de simulation.
2. C’est la liste des 10 meilleurs labyrinthe que l’application à tester. C’est aussi une liste de bouton que nous pouvons charger en cliquant dessus.
3. Ce bouton sert à supprimer toute les labyrinthes que l’utilisateur à précédemment sauvegarder.
4. Ce bouton sert à supprimer tous les labyrinthes du top 10.
5. Le bouton retour sert à revenir sur l’affichage principale du menu.

### Page de simulation

Cette interface est affichée juste après une génération aléatoire ou qu’une carte ai été importer



3

4

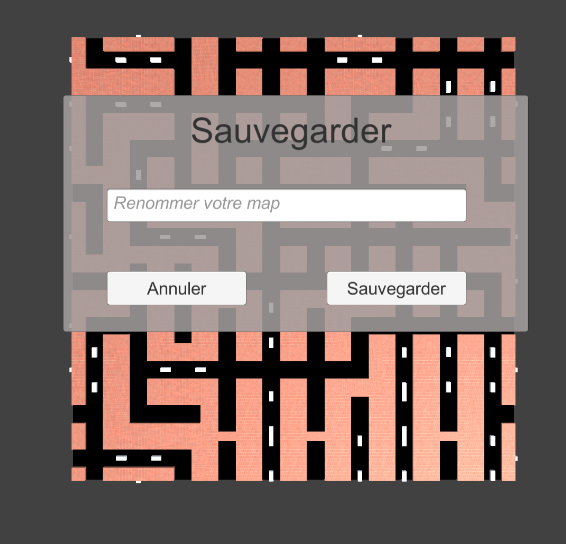
7

6

5

1. Le bouton Menu permets de revenir à n’importe qu’elle moment au menu principale.
2. Le texte numéro 2 permets d’afficher les informations sur ce que viens de faire le programme.
3. Le texte numéro 3 indique le chronomètre. Il est toujours a 0 et commence quand l’algorithme de recherche commence.
4. C’est le labyrinthe précédemment généré où importer.
5. Le bouton relancer permets à l’utilisateur de recommencer une génération aléatoire.
6. Le bouton Modes permets à l’utilisateur d’afficher le panel des modes. Il aura avec ce panel la possibilité de choisir le mode de recherche qu’il souhaite.
7. Le bouton sauvegarder permets à l’utilisateur d’afficher le panel de sauvegarde.

### Panel de sauvegarde

1. C’est un input texte qui va servir a donné un nom au labyrinthe qui va être sauvegarder.

3

2

1

1. Le bouton annuler arrête la tentative de sauvegarde et ferme le panel de sauvegarde.
2. Le bouton sauvegarder sauvegarde la carte avec le nom choisi précédemment.

### Panel choix du mode

4

3

2

1

1. Le bouton numéro 1 lance le mode manuel.
2. Le bouton numéro 2 lance le mode semi-automatique.
3. Le bouton numéro 3 lance le mode automatique de l’application.
4. Le bouton numéro 4 permets de cacher le panel des modes.

### Sélectionner une entrée / sortie

Sur chaque trou du labyrinthe un bouton est apparu. En cliquant sur un bouton l’utilisateur pour placé une entrée. Le système est le même s’il veut placer une sortie.

### Panel d’information

2

1

1

5

4

3

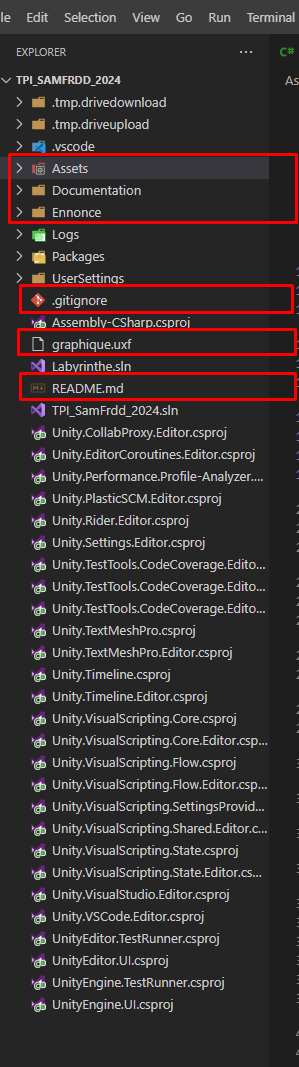
1. C’est un label qui ne donne l’information si la ou les sorties ont été trouvé.
2. C’est un label qui nous donne le temps que les bots ont fait pour trouver la solution en 1/100 secondes.
3. Le bouton numéro 3 donne à l’utilisateur la possibilité de changer de sortie.
4. Le bouton numéro 4 permets de ne plus afficher le panel information.
5. Le bouton numéro 5 permets à l’utilisateur de changer de mode tout en gardant le même labyrinthe.

# Analyse organique

L’analyse organique parle du fonctionnement interne. C’est donc dans cette partie que nous allons aborder la partie technique de ce travail en parlant de la structure de ma base de données, de l’architecture et de l’arborescence de mon application et le fonctionnement de mon application en passant par les librairies utilisées et les échanges internes de données.

## Structure des dossiers

Voici la structure du projet. Les dossiers et fichiers encadré sont ceux que je sauvegarde. Le reste sont des fichiers temporaires ou utilise à l’éditeur de Unity.



1. Dossier asset

Le dossier contient tout le contenu que j’utilise sur unity. Tous mes scripts y sont placés ainsi que les scènes, les UI, les préfabs et les matériaux.

1. Documentation

Ce dossier contient toute ma documentation concernant le projet.

1. Enoncé

Ce dossier contient l’énoncé du TPI ainsi que la grille d’évaluation.

1. Fichier. gitignore

Le fichier « .gitignore » est un fichier listant les fichiers et répertoires qui sont ignorés lors du push et du commit.

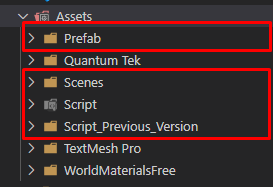
1. Fichier graphique.uxf

Le fichier « graphique.uxf » a été utilisé pour générer les diagrammes.

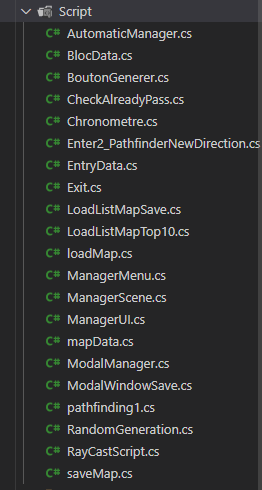
1. Fichier Readme.md

Ce fichier donne certaine indication sur la structure des fichiers.

### Dossier Asset

Le dossier 'Assets' est organisé comme suit : il comprend un dossier 'Prefab', un dossier 'Quantum Tek' contenant des actifs récupérés depuis l'Unity Asset Store, un dossier scène englobant toutes mes scènes, un dossier 'Script' comprenant tous les scripts que j'ai créés pour ce projet, un dossier 'Script\_Previous\_Version' renfermant les scripts auxquels je n'ai pas touché depuis les prérequis, ainsi que 'TextMesh Pro', une extension d'Unity regroupant des actifs UI, et 'WorldMaterialsFree', également issu de l'Unity Asset Store.

### Dossier Script



**AutomaticMangers.cs**

Le script automatique manager possède une classe qui gère tous la logique du mode automatique.

**BlocData.cs**

La classe blocData.cs est une classe serializable pour pouvoir stocké en XML la position, la rotation et le type de 1 blocs.

**CheckAlreadyPass.cs**

CheckAlreadyPass.cs est une classe qui se mets sur chaque bloc du labyrinthe. La classe note qu’elle bot est arrivé en premier sur le bloc.

**Chronometre.cs**

Chronometre.cs est une classe pour gérer le timer. Il s’occupe de commencer/reset et mettre au bon format le timer.

**Enter\_PathfinderNewDirection.cs**

Cette classe gère le départ du premier bot. Il va donner l’ordre de lui faire apparaître ainsi qu’à actualiser tous l’affichage.

**EntryData.cs**

EntryData est la classe qui me permets de stocké toute les informations de l’entré sous une forme objet.

**Exit.cs**

Le script Exit.cs gère quand le bot arrive à trouver la sortie. Il nous retourne qu’elle bot a trouvé la sortie.

**LoadListMapSave.cs**

LoadListMapSave me permets d’importer et de transformer en bouton tous les fichiers XML que l’utilisateur a voulu sauvegarder.

**LoadListMapTop10.cs**

LoadListMapSave me permets d’importer et de transformer en bouton les 10 meilleurs labyrinthe.

**LoadMap.cs**

LoadMap.cs me permets de pouvoir importer une carte dans le projet. Instancie la carte depuis un fichier XML.

**ManagerMenu.cs**

ManagerMenu gère l’affichage du menu. Il s’occupe d’afficher au lancement de l’application le bon panel.

**ManagerScene.cs**

ManagerScene.cs gère le changement de scène du projet.

**ManagerUi.cs**

ManagerUi gère tous l’affichage de la simulation. Il fait le point de passage entre le canevas et les scripts.

**MapData.cs**

MapData.cs est aussi une classe serializable. Elle stocke une liste de blockData ainsi que le score du labyrinthe.

**ModalManager.cs**

ModalManager.cs va gérer toute les fenêtres modales de la scène simulation. Il va faire en sorte que le panel s’affichage et que aucun autre panel puisse être afficher en même temps que lui.

**ModalWindowsSave.cs**

ModalWindowsSave.cs va gérer le panel de sauvegarde. Il va vérifier l’input de l’utilisateur. Et si tout est bon il envoie l’ordre de sauvegarder le labyrinthe.

**Pathfinding1.cs**

Le script Pathinding1 va gérer tous les déplacements du bot ainsi que toute la résolution du labyrinthe. Ce script se place directement sur chacun des bot.

**RandomGeneration.cs**

RandomGeneration.cs va gérer toute la génération de la carte. Que ca soit de la partie aléatoire ou de la partie utilisateur qui place les entrées/sorties.

**RayCastScript.cs**

RayCastScript.cs est la classe qui gérer les RayCast des bots.

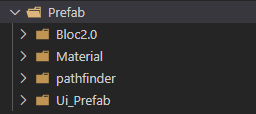
**SaveMap.cs**

La classe SaveMap.cs me permets de sauvegarder une classe en XML depuis un objet mapData

### Dossier Prefab

Voici le dossier Prefab qui contient 4 dossiers

* **Bloc2.0**

Le dossier Bloc 2.0 contient les blocs du labyrinthe pour la génération aléatoire

* **Meterial**

Le dossier Material contient tous les matériaux du projet.

* **Pathfinder**

Le dossier Pathfinder contre la prefab du bot

* **Ui\_Prefab**

Le dossier Ui\_Prefab contient tous les prefabs des Ui. Prefab des boutons etc…

## Architecture

Pour réaliser ce projet Unity3D, j’ai choisi de suivre les méthodes d'Objet que Unity nous donne à disposition. Pour l’écriture du code j’ai décidé de suivre la méthode DEMETER que nous avons appris lors de notre formation.

## Chemin des données

Structure des donnés dans les scènes Unity

### Scène Menu

La scène Main\_Menu est organiser de la manière suivante.

3

1

2

4

5

1. **Main Camera**

La main Camera est le point de vue de l’utilisateur. Elle est placée dans la scène et projette ce qu’on verra à l’écran.

1. **Directional Light**

La Directional Light sert à émettre de la lumière dans la scène. Les objets placés ne se verront pas si nous y mettons pas de la lumière.

1. **Simple Main Menu**

Simple Main Menu est l’objet parent de mon menu. Il contient les panels pour le menu et pour la liste des cartes sauvegarder. Tous les objets de type bouton ou label sont à l’intérieur de ce Game Object.

1. **Game Manager**

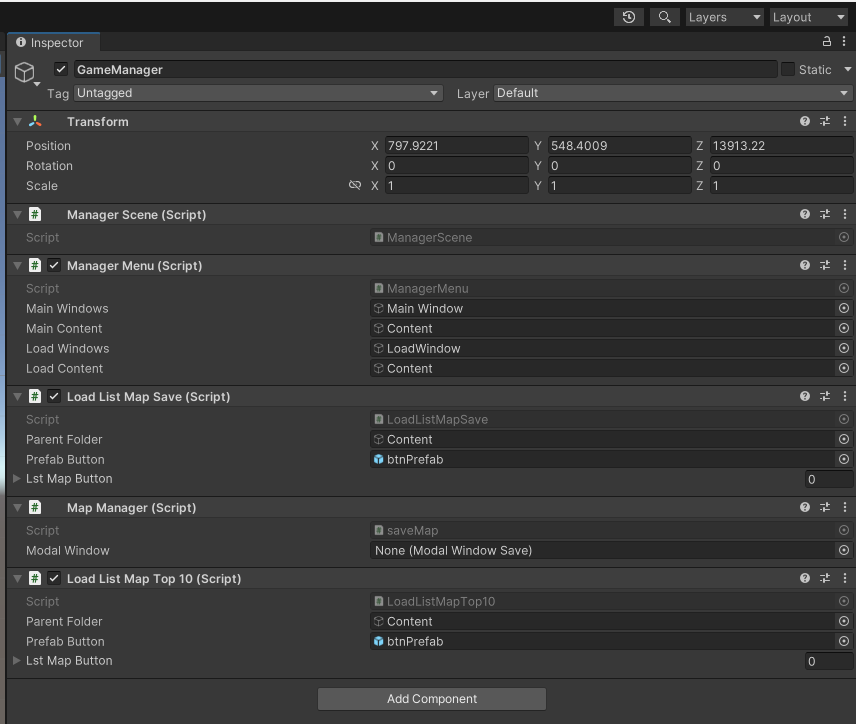
Le Game Manager est le « Manager » de la scène. Il contient de tous les scripts liés à la scène. Tous le bon fonctionnement de la scène passe par lui.

1. **EventSystem**le EventSystem est un élément central pour la gestion des interactions utilisateur dans Unity, en permettant une prise en charge efficace et flexible des événements d'entrée.

### Game Manager du main Menu

Voici comme est structuré le Game Manager de la scène menu

L’objet Game Manager contient dans le jargon unity des Component.

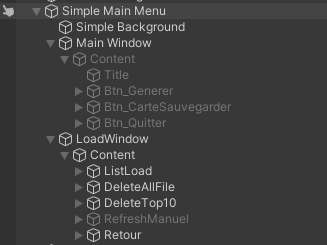
Un composant (ou component) dans Unity est une unité modulaire de fonctionnalité ou de comportement attachée à un GameObject. Les composants sont les blocs de construction de base qui définissent le comportement et les propriétés d'un GameObject dans une scène Unity.

Le Game Manager du Menu contient 6 composent.

* Transform
* ManagerScene
* ManagerMenu
* LoadListMap
* MapManager
* LoadListMapTop10

### Disposition du menu

Lorsqu'on lance Simple Main Menu, on remarque deux éléments principaux : la Main Window et Load Window. Main Window héberge les trois boutons

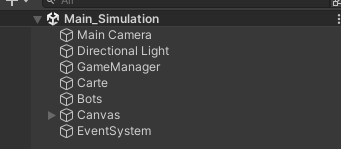
* Btb\_Generer
* Btn\_CarteSauvegarder
* Btn\_Quitter

La fenêtre LoadWindow contient

* La liste des cartes sauvegarder
* La liste des cartes du top 10
* Un bouton delete les cartes sauvegarder
* Un bouton delete le top 10
* Un bouton retour

### Main simulation

La scène Main simulation est utilisé pour l’algorithme de création et de recherche. Cette scène contient tous l’affichage nécessaire au bon fonctionnement de l’algorithme.



* Game Manager

Le Game manager de cette scène contient tous les scripts généraux. Et au bon déroulement de l’application.

Il contient les scripts suivant :

* Transform
* MapManager
* LoadMap
* ManagerUi
* RandomGeneration
* ModalWndowSave
* Chronometre
* ModalManager
* AutomaticManager
* Carte

Le gameObject « Carte » va servir à organiser tous les blocs du labyrinthe instancier dans la scène. Tous les blocs instancier vont le GameObject « Carte » comme parent.

* Bots

Le gameObject « Bots » va servir comme la carte. Il va servir à organiser tous les bots instancier dans la scène.

* Canvas

L’objet Canvas va servir à organiser tous les UI de la scène.

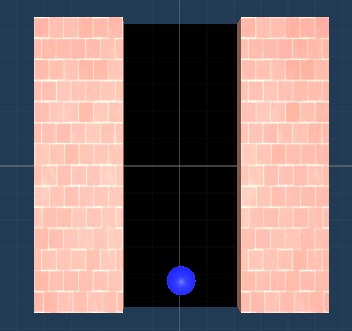
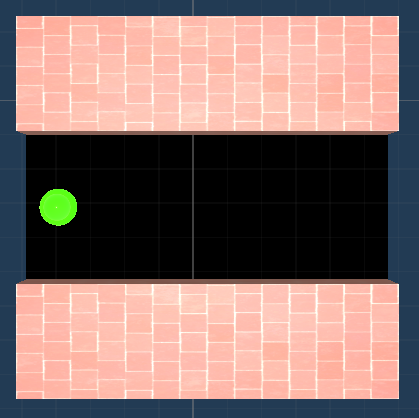
## Diagramme d’une méthode complexe

Cette partie est un bout de l’étape 4 de la méthode en 6 étapes.

### Blocs

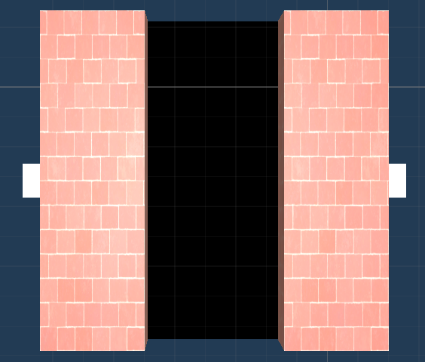
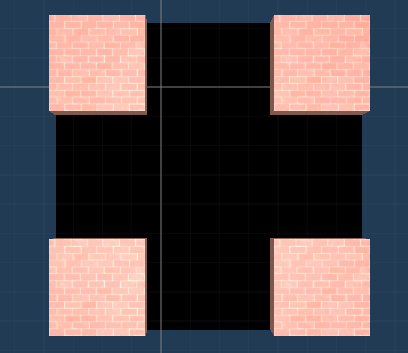
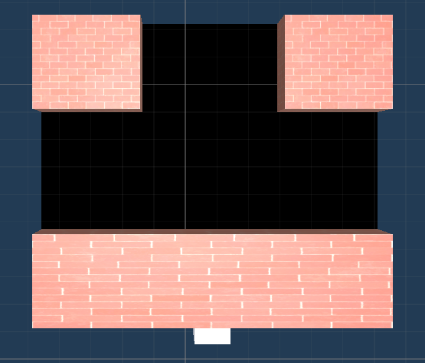
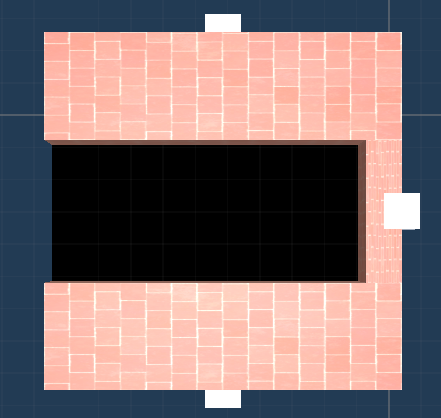
Voici les 8 blocs que j’ai utilisés pour générer mon labyrinthe.

Le bloc avec le point bleu signifie que c’est une entré et le point vert signifie que c’est une sortie.

 Bloc Entré Bloc sortie

### 

### 

Bloc Virage Bloc Plein

Bloc ligne droite Bloc croisement

Bloc en T Bloc sens unique

### Système de connecteur

Le système de connecteur marche comme un puzzle. Chaque côté du bloc possède son connecteur. Si ce côté possède une route il porte un tag unity (route), si ce côté est bloqué il porte le tag (bloque). Grâce à Unity on peut utiliser la fonction OnTriggerEnter(collider other). Cette fonction nous retourne le collider d’un objet qui est entré en contact avec le connecteur.

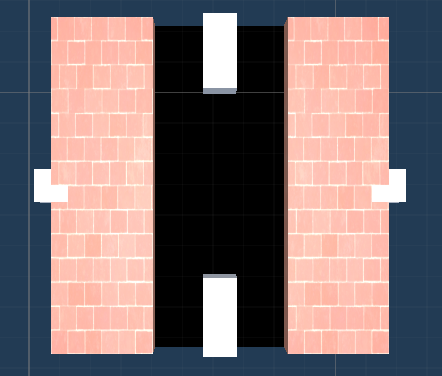
Depuis cette fonction, on vérifie si de quel type est le connecteur rencontré et on lui donne son état.

Les 3 états possibles sont :

“PasConnecte”

“MauvaiseConnection”

“Connecte”

Avec l’état du connecteur on peut déterminer dans l’algorithme si le bloc qu’on vient de placer est légitime d’être à l'endroit où il est.

2

4

3

1

Voici les 4 connecteurs du bloc « ligne droite » Le bloc numéro 1 et 3 sont avec le tag « route » et 2 et 4 « bloque ».

### Voici le diagramme simplifié de la génération du labyrinthe.

### 

### Comment le bot avance

Le bot avance grâce au 2 lignes de code suivante.

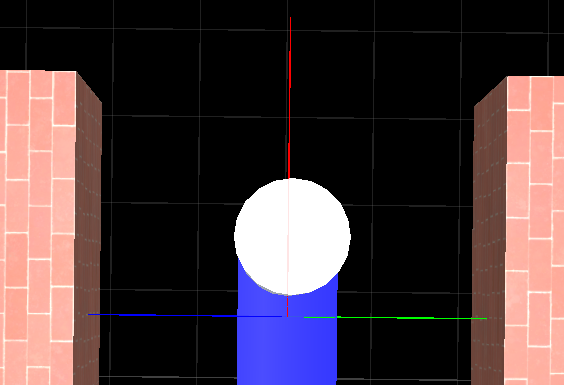
  Vector3 movement = Vector3.forward \* CurrentSpeed \* Time.deltaTime;

  transform.Translate(movement);

* Vector3 movement = Vector3.forward \* CurrentSpeed \* Time.deltaTime;
* Calcul du déplacement (movement) en avant de l'objet.
* Vector3.forward représente la direction avant de l'objet.
* CurrentSpeed détermine la vitesse de déplacement.
* Time.deltaTime garantit un mouvement fluide, indépendamment du taux de rafraîchissement de l'écran.
* transform.Translate(movement);
* Déplacement de l'objet selon le vecteur de déplacement calculé (movement).
* Cela assure que l'objet se déplace dans la direction avant avec une vitesse constante, adaptée au framerate.

### Système de RayCast

Les RayCast sur Unity sont des rayons invisibles émis depuis un point dans une direction spécifique pour détecter des collisions avec des objets dans le monde virtuel. Ils retournent des informations telles que la distance à l'objet touché et son identifiant.

Dans notre configuration, le bot est équipé de trois RayCast : l'un est dirigé vers l'avant, le second est orienté à 90 degrés vers la droite, et le troisième est orienté à 90 degrés vers la gauche.

3

2

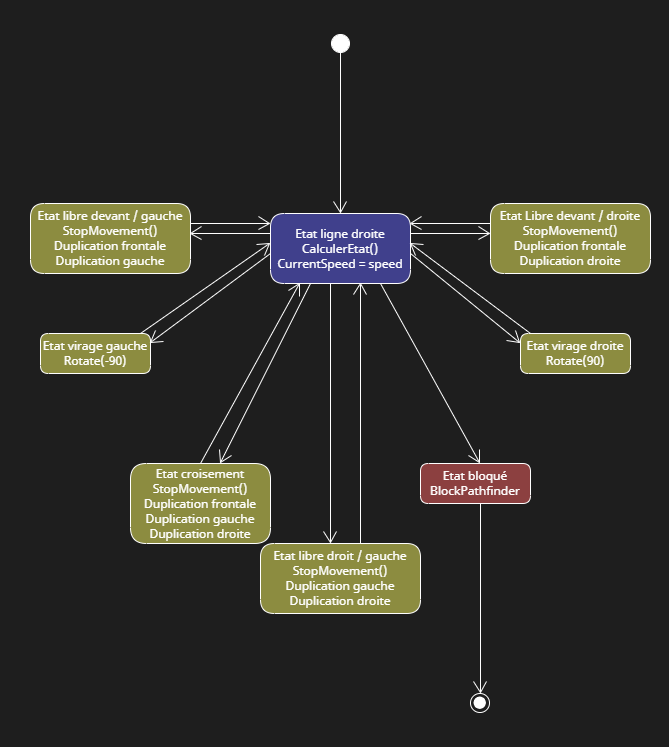
1

1. Frontale (rouge)
2. Gauche (bleu)
3. Droite (vert)

Les trois RayCasts que nous utilisons sont capables de détecter jusqu'à une distance de X mètres. Si le RayCast entre en contact avec un mur, il renvoie les valeurs suivantes : 1 pour le RayCast frontal, 2 pour celui de gauche et 4 pour celui de droite. En l'absence de collision avec un mur, le RayCast renvoie la valeur 0. Cette méthode est utilisée pour attribuer un état à chaque configuration.

Pour obtenir l'état du bot, nous devons additionner les valeurs des trois RayCasts : RayCast frontal + RayCast gauche + RayCast droite. Ce calcul nous donne un résultat compris entre 0 et 7, définissant ainsi nos 8 états possibles. Si aucun RayCast ne détecte de mur, le résultat est 0. Si tous les RayCasts détectent un mur, le résultat est 7.

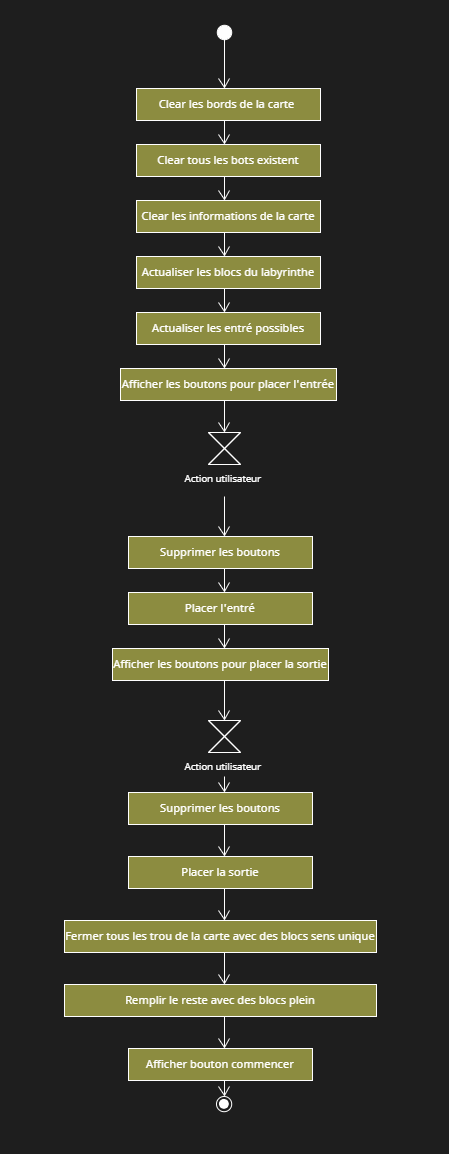
### Diagramme machine d’état

Voici la machine d’état du bot. A chaque frame on calcul l’état. Il est représenté ici dans état ligne droite.

Cette machine d’état permet de déplacer le bot partout dans le labyrinthe. Maintenant pour avoir un bot complétement fonctionnel il faut maintenant gérer les trainés.

Pour cela chaque bot. Connais le nom de son parent. Et le parent de ses enfants. Quand un bot meurt, on met sa trace en rouge. On va ensuite vérifier sur son parent si tous ses enfants sont morts. Si tous ses enfants sont morts en mets sa trace en rouge. Et exécute ca jusqu’au premier bot. Le premier bot porte un variable « isOriginel » a true. Si le bot originel reçoit l’ordre de son enfant de passé a rouge on sait que aucune sortie n’a été trouvé. Au contraire si un bot donne l’ordre « Sortie trouver » On remonte jusqu’au parent le chemin pour arriver à la sortie. Si on est dans le mode manuel dès que l’originel reçois un ordre de passé a rouge ou a vert si la sortie est trouvée on stop l’algorithme et on appelle la fonction ExitFound ou ExitNotFound dans la classe ManagerUI.

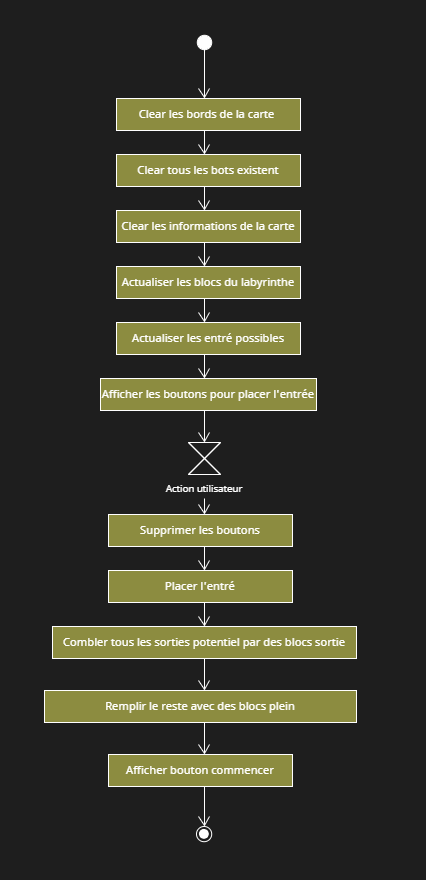
## Fonctionnement des modes de simulations

Le mode manuel a pour but de donner le choix à l’utilisateur de placer l'entrée et la sortie. Le bot fera par la suite l’examination du labyrinthe pour y trouver la sortie. Si aucune sortie n’est trouvé le bot affichera “Sortie non trouvé” sur le panel de fin.

### Diagramme sur le mode manuel

Le mode semi-automatique a pour but de donné toute les sorties possibles par rapport à une entré donnée par l’utilisateur.

### Diagramme sur le mode semi-automatique



Le mode automatique a pour but de tester tous le labyrinthe. Le programme va tester toute les entrées les uns après les autres et stocké toute les informations. A fin de l’algorithme le programme nous donne toute les données et fait le cumule du score de chaque sorite.

### Diagramme sur le mode automatique

## Assets

Pour le TPI j’ai utilisé 2 asset provenant du unity asset store.

* Simple menu

<https://assetstore.unity.com/packages/tools/gui/simple-menu-154642#description>

Dans cette asset j’ai seulement pris leur image de background. Le reste de l’asset a été enlevé du projet.

* World Materials Free

<https://assetstore.unity.com/packages/2d/textures-materials/world-materials-free-150182>

World Materials free donne accès a énormément de matériaux. J’ai extrait de cette asset 4 matériaux que je trouvais intéressant pour le TPI.

# Les tests

Cette partie corresponds à l’étape 5 de la méthode en 6 étapes.

## Périmètre

Les objectifs du plan de test sont les suivants :

* Valider les fonctionnalités du cahier des charges
* Réaliser cette validation à la fin du cahier des charges

## Environnement

Les outils constituants mon environnement pour réaliser les différents tests sont les suivants :

* Windows 10 Pro.
* UnityEditor 2022.3.17f1
* Carte graphique X
* Processeur i7-8700
* 32 GO de ram

## Scénarios

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No.** | **User Story** | **Scénario** | **Données** | **Résultat attendu** |
| T001 | En tant qu’utilisateur je veux pouvoir lancer une génération aléatoire depuis le menu | 1 Depuis le menu l’utilisateur clic sur le bouton générer | Clic sur le bouton | L’application lance le début d’une génération de carte |
| T002 | En tant qu’utilisateur je veux pouvoir voir l’évolution de la génération de la carte | 1 L’utilisateur a cliqué lancé la génération du labyrinthe |  | L’application généré une carte aléatoirement en posant les blocs les uns après les autres sans problèmes de connexion entre les routes |
| T003 | En tant qu’utilisateur je veux pouvoir placer une entré sur mon labyrinthe | 1 Quand l’utilisateur choisit le mode 1 il doit pouvoir placer une entré  2 L’utilisateur clique sur l’entre numéro 2 | Clic sur le bouton d’entré N.2 | Un bloc entré est correctement placé à l’endroit où l’utilisateur a cliquée |
| T004 | En tant qu’utilisateur je veux pouvoir placer une sortie sur mon labyrinthe | 1 Quand l’utilisateur choisit le mode 1 il doit pouvoir placer une entré  2 L’utilisateur clique sur la sortie numéro 4 | Clic sur le bouton d’entré N.4 | Un bloc sortie est correctement placé à l’endroit où l’utilisateur a cliquée et toute les autres sortie potentiel se retrouve bloqué avec un bloc sens unique |
| T005 | En tant qu’utilisateur je veux pouvoir voir le programme qui ferme totalement toute les sortie de mon labyrinthe avec les blocs adéquats au mode choisit | 1 Après que l’utilisateur ai placé une entré dans le mode 2  2 le programme complète automatiquement le labyrinthe |  | Toute les trou sur la carte sont bien comblé par des blocs à sens unique dans la bonne orientation |
| T006 | En tant qu’utilisateur je veux pouvoir ouvrir un panel pour ouvrir le choix des mode a disposition | L’utilisateur ouvre un panel avec le bouton « afficher mode » | Clic sur le bouton affiche mode | Nous ouvre un panel avec à l’intérieur le choix des 3 modes |
| **Test No.** | **User Story** | **Scénario** | **Données** | **Résultat attendu** |
| T007 | En tant qu’utilisateur je veux pouvoir lancer le mode numéro 3 et que le programme lance directement les entrés et sortie nécessaire au bon fonctionnement de ce mode | 1 L’utilisateur clique sur le bouton mode 3 après la génération | Clic sur le bouton mode 3 | L’application place l’entré au bloc numéro 1 et toute les autres possibilité seront des sorties |
| T008 | En tant qu’utilisateur je veux pouvoir voir le bot se dupliquer lors d’un croisement | L’utilisateur débute l’algorithme et nous voulons voir le bot qui se duplique en X fois lors de ce premier croisement | Clique sur le bouton commencer l’algorithme | Le bot « originel » reste au milieu et 2 « enfant / close » vont dans les X possibilité |
| T009 | En tant qu’utilisateur je veux pouvoir voir une trace bleu dernière le bot lorsqu’il effectue la recherche de la sortie | 1 L’utilisateur lance le mode 1 | Clique sur le mode 1 | Il y’a bien un trainé bleu lorsque le bot a toujours la possibilité de trouvé une sortie |
| T010 | En tant qu’utilisateur je veux pouvoir voir une trace verte derrière le bot qui a trouvé une sortie | 1 L’utilisateur lance le mode 1 | Clique sur le mode  Une carte avec une entrée et sortie possible1 | Il y’a bien la trainé en vert pour le bot qui a trouvé la sortie. La trainé est commencé du début jusqu’à la fin du parcours |
| T011 | En tant qu’utilisateur je veux pouvoir voir une trace rouge qui disparait au bout de 0,5 seconde lorsque mon bot rentre dans une section sens unique | 1 L’utilisateur lance le mode 1 | Clique sur le mode | Il y’a bien une trace rouge qui disparait au bout de 0,5 seconde lorsque un bot se rentre en collision avec un bloc a sens unique |
| T012 | En tant qu’utilisateur je veux pouvoir voir une trace jaune derrière le bot qui est le plus proche de la sortie | 1 L’utilisateur lance le mode 1 | Clique sur le mode  Une carte avec une entré et une sortie pas possible | On s’attend a voir une trace jaune derrière le bot qui est le plus proche de la sortie donné. Le plus proche en terme de distance |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No.** | **User Story** | **Scénario** | **Données** | **Résultat attendu** |
| T013 | En tant qu’utilisateur je veux pouvoir voir si le bot ne trouve pas de sortie | 1 L’utilisateur clique sur le bouton mode 1 après la génération | Clic sur le bouton mode 1 | L’application m’informe sous forme de texte que aucune sortie n’a été trouvé avec l’entré la sortie choisi |
| T014 | En tant qu’utilisateur je veux pouvoir voir dans une liste toute les carte que j’ai pu sauvegarder | 1 L’utilisateur clique depuis le menu sur le bouton « charger une carte sauvegarder » | Clique sur le bouton | Nous retourne une liste de bouton cliquable qui représente chaque carte que nous avons sauvegarder |
| T015 | En tant qu’utilisateur je veux pouvoir avoir la possibilité de sauvegarder un labyrinthe que j’apprécie avec un nom que je choisi | L’utilisateur lance une génération aléatoire et clique sur le bouton sauvegarder | Clique sur le bouton sauvegarder et rentre la valeur « Test » comme nom de carte | Nous ferme la fenêtre modal de sauvegarde et nous affiche un message de succès. Nous sauvegarde dans fichier Test.xml dans l’arborescence de fichier |
| T016 | En tant qu’utilisateur je veux pouvoir ne pas choisir de nom à la carte que je sauvegarde | L’utilisateur lance une génération aléatoire et clique sur le bouton sauvegarder | Confirme la sauvegarde sans mettre de nom | Nous ferme la fenêtre modal de sauvegarde et nous affiche un message de succès. Nous sauvegarde dans fichier NomUnique.xml dans l’arborescence de fichier |
| T017 | En tant qu’utilisateur je veux pouvoir recommencer une génération d’un nouveau labyrinthe juste après avoir terminé une génération précédente | L’utilisateur clique sur le bouton, Relancer après la génération automatique de la carte | Clique sur le bouton relancer | Nous voulons le début d’une nouvelle génération. L’affichage est remis a 0 et l’ancienne carte est supprimé de la scène |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No.** | **User Story** | **Scénario** | **Données** | **Résultat attendu** |
| T018 | En tant qu’utilisateur je veux pouvoir lancer l’algorithme du mode 1 et voir mon chemin jusqu’à la sortie | L’utilisateur clique sur le mode 1 après une génération aléatoire et choisi une entré et une sortie. | Bouton mode 1  Choisi entré n.2  Choisi entré n.5 | Le bot commence et effectue correctement l’algorithme de recherche. Le programme nous retourne toute les donner relative à l’algorithme lancé. |
| T019 | En tant qu’utilisateur je veux pouvoir lancer le mode 2 et regarder toute les sorties possible avec l’entrer que je choisi | L’utilisateur clique sur le mode 2 après une génération aléatoire et choisi une entré | Bouton mode 1  Choisi entré n.2 | Le bot commence et effectue correctement l’algorithme de recherche. Le programme nous retourne toute les donner relative à l’algorithme lancé. |
| T020 | En tant qu’utilisateur je veux pouvoir voir la fitness de mon labyrinthe en cliquant sur le mode 3 | L’utilisateur clique sur le mode 3. |  | Chaque entré est tester 1 par 1. Et a la fin le programme nous retourne un tableau avec chaque entré et son fitness classé par ordre décroissant |
| T021 | En tant qu’utilisateur je veux pouvoir cliqué sur une des entré dans le tableau et voir son parcours jusqu’à chaque sortie | L’utilisateur attend que l’algorithme du mode 3 se finisse pour ensuite cliquer sur la première ligne du tableau retourné | Clique sur une des entré présenté dans a la fin du mode 3 | Affiche le trace exact depuis l’entré choisi jusqu’à chaque sorti possible |
| T022 | En tant qu’utilisateur je veux pouvoir voir les 10 meilleurs labyrinthe stocké sur mon application en fonctions de leur fitness | L’utilisateur clique sur le bouton « carte sauvegarder » depuis le menu |  | Nous affiche les 10 meilleurs carte sauvegarder automatiquement en fonction de leur fitness |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No.** | **User Story** | **Scénario** | **Données** | **Résultat attendu** |
| T023 | En tant qu’utilisateur je veux pouvoir quitter l’application depuis le bouton quitter du menu | L’utilisateur clique sur le bouton quitter depuis le menu | Clique sur le bouton menu | L’application se ferme directement |
| T024 | En tant qu’utilisateur je veux pouvoir supprimer toute mes carte sauvegarder | L’utilisateur clique sur le bouton « Supprimer » en dessous de la liste de carte que nous avons sauvegarder | Clique sur le bouton supprimer correspondant | Nous supprime toute les cartes que nous avons sauvegarder et l’affichage est update automatiquement |

## Suivi évolutif des tests

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test No.** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** |
| T001 | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| T002 | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| T003 | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| T004 | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| T005 | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| T006 | ❌ | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| T007 | ❌ | ❌ | ❌ | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| T008 | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| T009 | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| T010 | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| T011 | ❌ | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| T012 | ❌ | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| T013 | ❌ | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| T014 | ❌ | ❌ | ❌ | ❌ | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| T015 | ❌ | ❌ | ❌ | ❌ | ❌ | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ |
| T016 | ❌ | ❌ | ❌ | ❌ | ❌ | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ |
| T017 | ❌ | ❌ | ❌ | ❌ | ❌ | ❌ | ❌ | ✅ | ✅ | ✅ | ✅ |
| T018 | ❌ | ❌ | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| T019 | ❌ | ❌ | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| T020 | ❌ | ❌ | ❌ | ❌ | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| T021 | ❌ | ❌ | ❌ | ❌ | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| T022 | ❌ | ❌ | ❌ | ❌ | ❌ | ❌ | ❌ | ✅ | ✅ | ✅ | ✅ |
| T023 | ❌ | ❌ | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| T024 | ❌ | ❌ | ❌ | ❌ | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |

# Conclusion

Cette partie corresponds à l’étape 6 de la méthode en 6 étapes.

## Difficultés rencontrés

Je n’ai pas rencontré de grandes difficultés, mais je dois admettre que j’ai eu certain souci avec le changement de sortie. J’ai eu sur quel que jour pas mal de problème. Cependant les soucis ont été pu être régler. J’ai également eu des difficultés à bien repartir et séparé mon code. Je trouve que le code que j’ai fourni n’est pas très compacte et tir en longueur parfois. Il y’a certain parti du code que j’ai essayé de factoriser sans gros succès. L’affichage a été compliqué. J’ai réussi à créer un affichage simple. Mais quand j’ai voulu essayé de l’améliorer je n’arrivais pas à mettre en place mes idées.

La documentions technique a également été dur à produire. Les Diagramme d’activé ont refait a plusieurs reprise sans un résultat suffisant.

## Amélioration possibles

Pour les améliorations possibles il y’a en premier lieu un meilleur tableau pour la liste des entrées. Par la suite on aurait pu imaginer afficher le numéro de l’entrée à la fin du mode automatique. Pour le moment l’utilisateur ne sait pas sûr qu’elle entrée il clique.

L’affichage et le design général de l’application pourrait également être améliorer.

Il y’a certain parti du code qui ne sont pas du tout factoriser. Si j’avais eu plus de temps j’aurais aimé prendre plus de temps à factoriser certain parti et ainsi améliorer la lisibilité du code.

La génération de la carte n’est pas optimale. Une amélioration possible est de généré la carte sans instancier les blocs. Pour gagner en vitesse. Dans la solution actuelle on attend que le bloc soit instancié dans la scène pour être vérifier et ensuite on le laisse ou on le retire. Généré le labyrinthe sans passé par cette étape serait infiniment plus rapide.

## Bilan personnel

[A REMPLIR]

## Remerciements

[A REMPLIR]

# Bibliographie

Source d’information méthode en 6 étapes : <https://www.afci-ju.ch/fichiers/Planification-par-la-mthode-des-6-tapes-15.pdf>

Logo de PHP : <https://fr.m.wikipedia.org/wiki/Fichier:PHP-logo.svg>

Logo de Bootstrap studio : <https://en.wikipedia.org/wiki/File:Bootstrap_Studio_Logo.png>

Logo de DBeaver : <https://www.crunchbase.com/organization/dbeaver/technology>

Logo de GitLab : <https://logowik.com/gitlab-vector-logo-4667.html>

Logo d’apache : <https://commons.wikimedia.org/wiki/File:Apache_HTTP_server_logo_%282019-present%29.svg>

Logo de MariaDB : <https://linuxnatives.net/2015/10-reasons-to-migrate-to-mariadb-if-still-using-mysql>

# Glossaire

|  |  |
| --- | --- |
| CRUD | Acronyme anglais pour « Create, Read, Update and Delete ». Il s’agit des actions les plus fréquentes réalisées sur une base de données. |
| User Story | Il s’agit d’un élément issu de du développement agile. Le but d’une « User Story » est de décrire une fonctionnalité sous la forme d’une petite phrase vue par l’utilisateur/client. |
| MVC | Il s’agit d’un modèle qui permet de séparer le plus possibles les différentes parties du code, qui notamment utilisé dans les applications web. « MVC » est l’acronyme de « Model, Vue, Controller ». |

# Annexes

L’ensemble des livrables définis au début de ce document se trouvent dans le même répertoire.