S3B

# Projet tutoré : 1ère itération

BERARD Valentin

OBERHAUSSER Lucas

SALTUTTI Johan

SOMMER Matthieu

**IUT Nancy Charlemagne – Département Informatique**

# Table des matières

# Description du projet

Il s'agit de réaliser une application en java, qui aura pour but d'initier des étudiants aux problématiques de localisation et de cheminement d'un robot dans un environnement discret. L'environnement dans lequel évoluera le robot sera, dans le cadre de ce projet, un labyrinthe et l'application sera utilisée pour des démonstrations dans des événements du type foire de la science.
Il est important de préciser que le robot ne sera pas un robot physique et que celui-ci ne sera géré que logiquement dans l'application, puisqu'il s'agit d'introduire simplement des notions clés du système de fonctionnement d'un robot. L'application n'est pas non plus un jeu comme nous le pensions au départ, mais un réel outil pédagogique permettant de montrer à des étudiants comment fonctionnent les systèmes de localisation et de déplacement d'un robot (avec néanmoins des interactions entre les étudiants et le logiciel).
L'application ne fonctionne que si un enseignant est présent avec l'étudiant pour pouvoir l'amener à se poser les bonnes questions et comprendre parfaitement les différents algorithmes.
De plus, cette application ne peut fonctionner que dans un environnement discret, en effet, nous n'avons, comme cela, pas à tenir compte d'un déplacement réel (en cm mais juste case par case), mais aussi tout dans l'environnement est plat, ce qui n'est pas réellement possible dans la réalité, avec par exemple une inclinaison du sol.

Cette application et ce projet seront structurés en quatre grandes parties : **Création de Labyrinthe**, **Localisation**, **Recherche de chemins** et **Vues**.

**• Création de labyrinthe :**

Le but de l'application étant de raisonner sur un environnement discret, donc fini, il semble intéressant de choisir les labyrinthes, étant, par définition, plus que propices aux problèmes de cheminement. C'est pourquoi il nous faut tout un panel de labyrinthes qui auront pour but d'exposer les limites et avantages de certains algorithmes de cheminement ou de localisation. L'application devra donc nous permettre de créer un labyrinthe (case par case, via des clics sur un JPanel) en ayant choisi sa taille, de le sauvegarder ou en charger un que l'utilisateur aura déjà construit (via un menu) ou encore d'en générer automatiquement (après avoir choisi l'algorithme de génération et cliqué sur un JButton).

# Description du projet

Il sera proposé deux types de génération : le premier permettra de créer un labyrinthe dit « parfait » (pour toutes cases vides x et y, il existe exactement un seul chemin reliant x et y) et repose sur l'algorithme de génération par fusion. La seconde permettra de créer des labyrinthes dits « imparfaits » et crée un labyrinthe parfait auquel on retire un certain nombre de murs.

C'est le module de l'application avec lequel le public aura le moins d'interaction, puisque nous nous intéressons plus au robot qu'à son environnement à proprement parler.

**• Localisation :**

Maintenant que nous avons un labyrinthe, nous pouvons nous intéresser au robot. L'intérêt de ce module est de faire interagir les étudiants avec l'application en les mettant dans des situations dans lesquelles ils peuvent ou non déplacer le robot pour qu'ils découvrent des méthodes pour se localiser. Ainsi, ils essaieront de déterminer l'emplacement du robot en fonction de ce que celui-ci voit (Vue 3D ou vue seulement des cases adjacentes au robot), pour en déduire des manières de déterminer sa position.
Le module affiche une vue globale du labyrinthe sans afficher le robot, et une vue locale qui correspond à ce que voit le robot (Vue 3D, vue cases adjacentes...). L'utilisateur de l'application pourra déplacer le robot et celle-ci devra mettre à jour les emplacements possibles du robot en fonction des déplacements et orientations passées.

On aura donc la possibilité de cacher ou d'afficher la position du robot dans le labyrinthe (via un JButton), d'alterner entre la vue en 3D et la vue des cases adjacentes (via un JRadioButton). On pourra également déplacer le robot dans le labyrinthe via les touches du clavier ou des JButton et on pourra changer l'emplacement initial du robot.

**• Recherche de chemins (optionnel) :**

Autre grand module de l'application : le module cheminement. À travers celui-ci, il sera question d'amener les étudiants à réfléchir sur les méthodes permettant de sortir d'un labyrinthe, en ayant connaissance de l'ensemble du labyrinthe ou seulement des cases adjacentes, et en empruntant le chemin le plus court ou le premier trouvé. De plus, les étudiants pourront créer leurs propres algorithmes via une interface et tester leurs solutions pour sortir du labyrinthe.

# Description du projet

Ce module va donc se découper en trois grandes parties : une partie dans laquelle l'utilisateur créera son propre comportement et pourra l'enregistrer et le charger, une autre dans laquelle il aura une vision seulement locale (cases adjacentes) et pourra choisir l'algorithme de son choix pour trouver la sortie (main droite, left-turn, Pledge, Trémeaux ou encore un comportement qu'il aura lui-même créé). Dans la troisième partie, l'utilisateur aura une vision globale du labyrinthe et pourra s'en servir pour raisonner et trouver la sortie. De la même manière, il pourra choisir l'algorithme de son choix (Dijkstra, A*, dead-end filling ou son propre comportement).

Dans tous les cas, on pourra soit se déplacer avec les touches du clavier soit lancer un algorithme de cheminement. Dans ce cas, on pourra agir sur son exécution (mettre en pause, accélérer, ralentir). On pourra également changer le point de départ, l'orientation du robot ainsi que sa case de sortie.

**• Vues :**

Pour pouvoir analyser l'environnement dans lequel se trouve le robot et tirer des enseignements, les étudiants auront à disposition différentes vues du labyrinthe. En fonction de celles-ci, il devra adopter différentes stratégies pour se repérer et sortir du labyrinthe. Le premier type de vue est le labyrinthe complet vu de dessus. Un autre type de vue est la vue des cases adjacentes au robot (devant et sur les côtés) avec laquelle on sait uniquement si ce sont des murs ou des cases vides.
L'application disposera également d'une vue en 3D, à la première personne, pour laquelle nous avons deux options : soit nous implémentons un moteur 3D, soit nous utilisons la librairie java3d. Pour finir, nous pourrons avoir une vue constituée uniquement d'informations sur des capteurs du robot, par exemple des lasers devant et sur les côtés qui donnent la distance séparant le robot du mur.

**• Autres spécifications techniques :**

Notre application et en particulier les vues devront se redimensionner dynamiquement en fonction du mode d'affichage (plein écran, fenêtré), pour que le logiciel ne soit pas dépendant de la résolution ni de l'écran qui l'affiche (puisque l'application est destinée à être affichée via des écrans de différentes tailles).

# Description du projet

# Etude de l’existant

### Étude de l'existant :

Il n'existe pas, à proprement parler, de logiciel à but pédagogique permettant d'introduire les problématiques de cheminement et de localisation. Néanmoins, il existe des applications qui reprennent certains points de notre projet.

On peut penser tout d'abord aux solveurs de labyrinthes qui permettent de résoudre des labyrinthes, mais ne montrent pas la logique derrière leurs algorithmes de cheminement, ce qui est le but de notre application.

Il existe également des jeux permettant de construire ses propres comportements, algorithmes. C'est le cas de lightbot (réf : lightbot.com ) qui est un jeu dans lequel on doit combiner des instructions pour amener un robot à allumer toutes les lampes d'un niveau. Plus on avance et plus le nombre d'instructions à utiliser se restreint, ce qui amène l'utilisateur à trouver des solutions de plus en plus complexes (boucles, écriture de fonctions, appels récursifs).

On peut aussi citer blue-bot, robot destiné aux classes de primaire, qui permet d'enregistrer une suite de déplacements qu'il exécutera ensuite. Il est souvent utilisé dans des parcours d'obstacles pour développer la logique des enfants.

# Etude technique

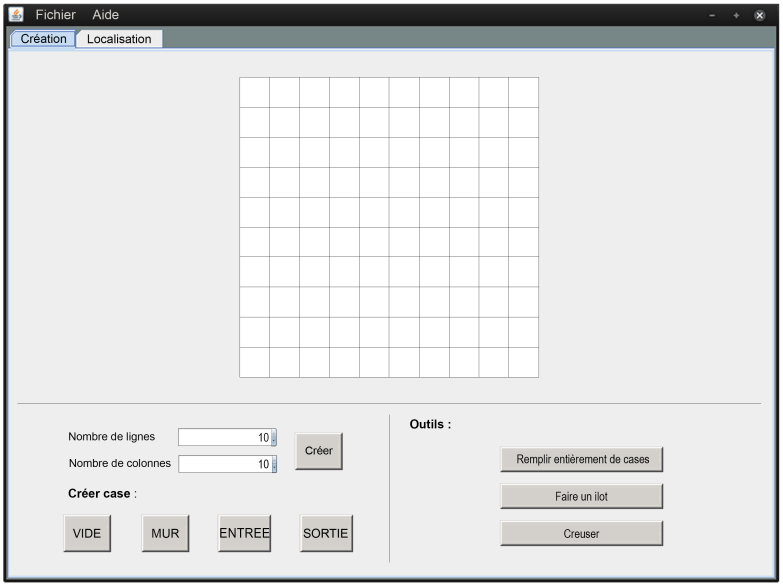
Les différents cas d'utilisations que nous avons déjà identifiés sont :

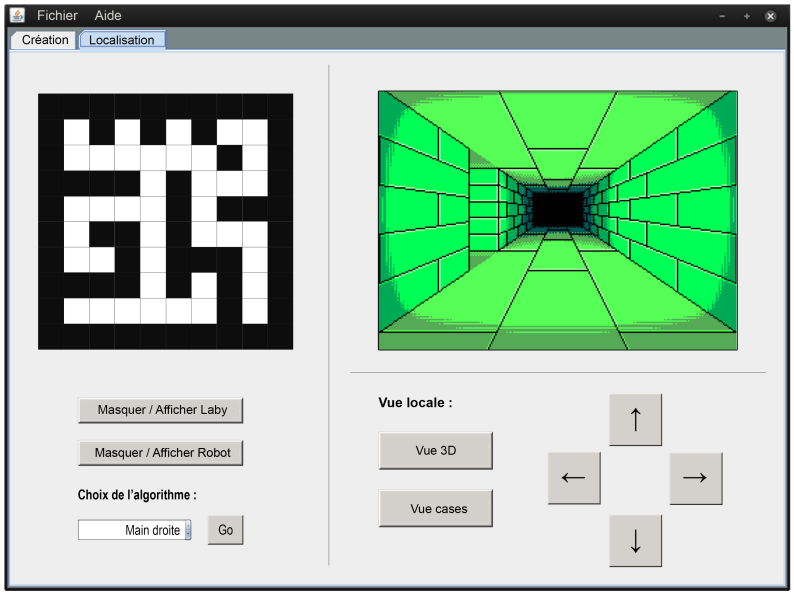
- La création du labyrinthe
  - Créer une case « Mur »
  - Créer une case « Vide »
  - Créer une case « Entrée »
  - Créer une case « Sortie »
  - Placer le robot
  - Charger un Labyrinthe
  - Sauvegarder un Labyrinthe
- La génération du labyrinthe
- Afficher / Masquer la position du robot
- Afficher / Masquer la vue 3D
- Afficher / Masquer cases adjacentes
- Déplacer le robot
- Changer emplacement initial
- Créer le comportement du Robot
- Utilisation de boucles, d'actions de déplacement…
- Sauvegarder le comportement
- Exécuter l'algorithme

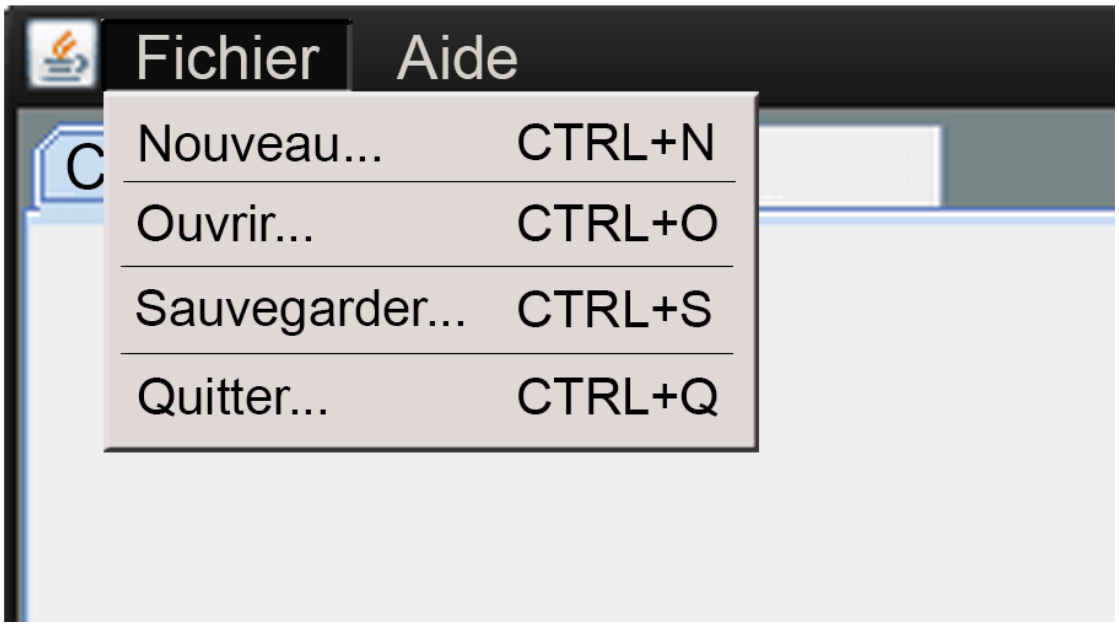
Nous envisageons de représenter le labyrinthe par un tableau à deux dimensions. Celui-ci contiendra des éléments de type Case qui pourront être soit des murs soit des chemins.

Pour la définition de comportements par l'utilisateur, nous pensons utiliser ce que l'on appelle un "behaviour tree" (qui s'apparente à un automate (cf. Machine de Turing)) qui peut prendre la forme d'un arbre que l'on explore ou non selon les conditions qu'il y a au nœud courant. Pour cela, nous utiliserons sûrement le pattern Composite (dérivé de Décorateur).

**• Maquette :**


Fichier
Aide
Création
Localisation
Nombre de lignes
10
Nombre de colonnes
10
Créer
Créer case :
VIDE
MUR
ENTREE
SORTIE
Outils :
Remplir entièrement de cases
Faire un ilot
Creuser


Fichier
Aide
Création
Localisation
Masquer / Afficher Laby
Masquer / Afficher Robot
Choix de l'algorithme :
Main droite
Go
Vue locale :
Vue 3D
Vue cases




L'application devrait être structurée en différentes sections qui respectent les modules énoncés précédemment :

- Section création/génération de labyrinthes
- Section localisation
- Section création de comportement personnalisé
- Section cheminement avec vue relative
- Section cheminement avec vue globale