Développement réseau/intelligence artificielle

1 Contexte et travail à réaliser

Le but de ce projet est de développer une intelligence artificielle (ou au moins un truc qui pourrait passer pour une intelligence) pour diriger un personnage dans un espace clos (que nous appellerons "labyrinthe"). Votre programme devra être capable d'atteindre le but suivant : récupérer un maximum de points en ramassant des moules disséminés dans le labyrinthe. Votre personnage ne sera peut être pas le seul dans cet espace. Il pourrait affronter l'intelligence d'un ou plusieurs autres étudiants en tour à tour. Afin de vous aider dans votre mission, divers bonus seront placés tout au long du parcours.

L'interface du jeu contient deux parties : le labyrinthe dans la partie gauche et les scores à droite.
Le labyrinthe peut se présenter sous diverses formes et complexités :

— La figure 1 montre un labyrinthe **parfait**.
— La figure 2 montre un labyrinthe assez ouvert comprenant quelques murs.

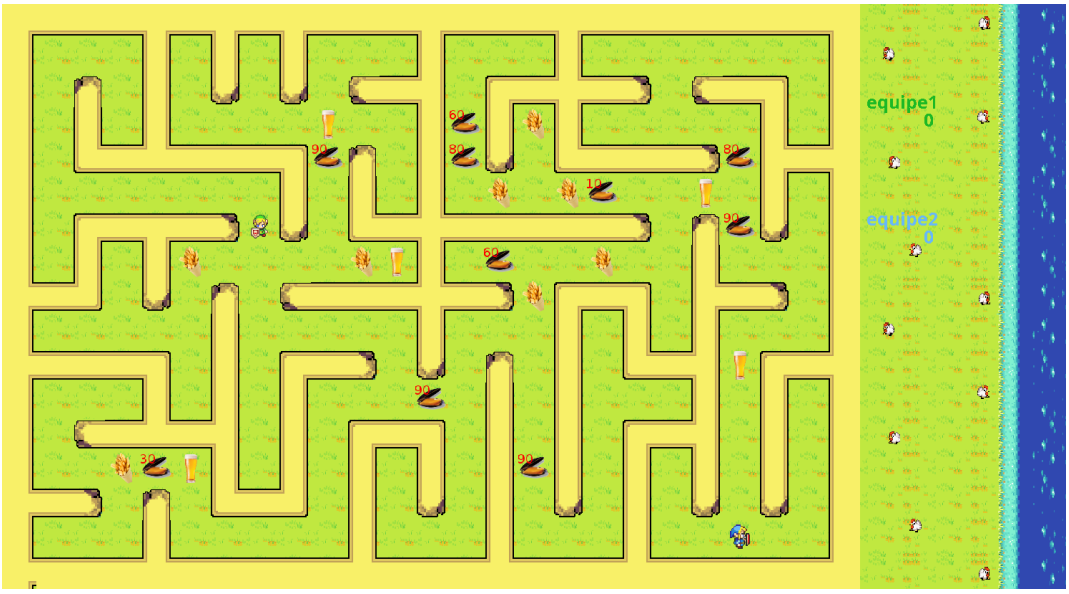

FIGURE 1 – Labyrinthe parfait.

Comme vous pouvez le voir sur les figures 1 et 2, les labyrinthes sont composés de blocs différents :

— de l'herbe sur laquelle vous pouvez marcher,
— les blocs jaunes représentent des murs quasi infranchissables donc des blocs sur lesquels vous ne pouvez pas vous positionner,
— des frites (bloc bonus) qui vous permettront d'avancer de deux cases dans une direction, et ce, même s'il y a un mur qui vous sépare du point d'arrivée. Lorsque vous utiliserez ce bonus, vous devrez bien faire attention à ce que le point d'arrivée soit une case sur laquelle vous pouvez marcher [1]. Dans le cas contraire, le personnage avancera d'une case dans la direction donnée si cela est possible. Évidemment, les bonus et/ou moules sur le passage seront ramassés.

1. de l'herbe, un bonus ou une moule

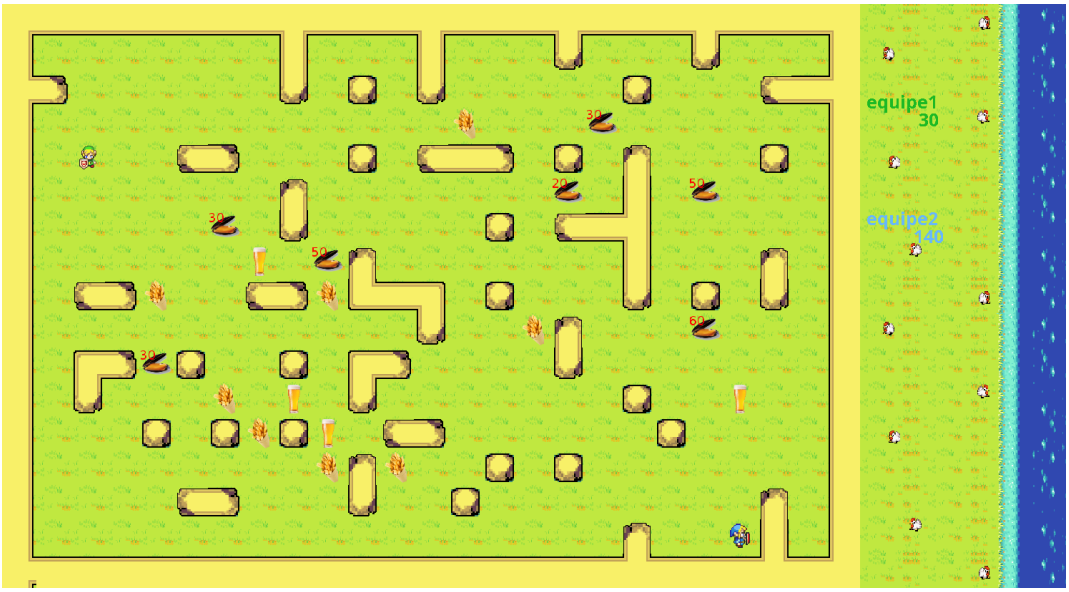

FIGURE 2 – Labyrinthe présentant une surface assez ouverte avec quelques murs.

— des bières (bloc bonus) qui vous permettent de jouer trois coups à la suite (mais ne permet pas de franchir les murs). Évidemment, les bonus et/ou moules sur le passage seront ramassés.
— des moules qui vous permettent d'engranger des points. Les moules ont un nombre de points aléatoires décidés en début de partie.
— plusieurs personnages dont un sera déplacé grâce à votre programme.

Les blocs "frites", "bières" et "moules" disparaissent lorsqu'un joueur passe dessus. Ils sont automatiquement transformés en herbe. Les frites et bières sont collectées et conservées pour pouvoir être utilisées lorsque le joueur le souhaite.

2 Dialogue avec le serveur

Dans un premier temps, vous allez devoir créer un socket qui doit ouvrir une connexion avec le serveur [2]. Une fois cette connexion ouverte, vous enverrez votre nom d'équipe et passerez en écoute pour récupérer votre numéro de joueur [3]. Lorsque tous les joueurs seront connectés et que ce sera votre tour, les informations complètes du labyrinthe vous seront envoyées. Il vous faudra ensuite décider quel coup jouer et l'envoyer au travers du socket.

Votre programme devra donc suivre cette architecture :

```
DEBUT DE PROGRAMME
  Création du socket
  Envoi de votre nom d'équipe
  Réception du numéro de joueur
  Boucle
    Réception des informations du labyrinthe
    Calcul du prochain coup à jouer (votre IA)
    Envoi du coup
  Fin De Boucle
FIN DE PROGRAMME
```

Le jeu s'arrête lorsque le serveur vous envoie le mot `FIN` à la place des informations sur le labyrinthe. Cela signifie que toutes les moules ont été ramassés, que le nombre de tours maximum a été dépassé ou que tous les joueurs ont passé leur tour.

N'oubliez pas de prévoir de pouvoir spécifier à votre programme l'adresse ip, le port pour la connexion au serveur ainsi qu'un nom d'équipe.

2. Par défaut, le port utilisé est le 1337.
3. Le joueur 0 se trouve en haut à gauche, le joueur 1 en bas à droite, le joueur 2 en haut à droite et le joueur 3 en bas à gauche.

2.1 Réception des données

Pour ce qui est de la structure de la chaîne de caractères que vous allez recevoir, elle est composée comme ceci : `"taille/structureLabyrinthe/infosJoueurs"`.

Par exemple (voir figure 3) :

```
13x11/Mu-Mu-Mu-Mu-Mu-Mu-Mu-Mu-Mu-Mu-Mu-Mu-Mu-So-Mu-So-So-So-Mu-So-Mu-So-So-So-Mu-Mu-So-Mu-
Mu-Mu-So-Mu-So-Mu-So-Mu-So-Mu-Mu-So-So-So-Mu-50-30-So-90-60-Mu-So-Mu-Mu-So-Mu-Mu-Mu-Bp-Mu-Mu-
Mu-Mu-Mu-So-Mu-Mu-So-So-20-Bp-Bs-Mu-Bp-Mu-So-So-So-Mu-Mu-So-Mu-80-Mu-Mu-Mu-Bp-Mu-So-Mu-Mu-Mu-
Mu-So-Mu-So-40-Bp-10-So-Mu-So-So-So-Mu-Mu-So-Mu-Mu-Mu-So-Mu-So-Mu-Mu-Mu-So-Mu-Mu-So-Mu-So-So-
So-Mu-So-Mu-So-So-So-Mu-Mu-Mu-Mu-Mu-Mu-Mu-Mu-Mu-Mu-Mu-Mu-Mu-Mu-Mu/2-1,8-11,2
```

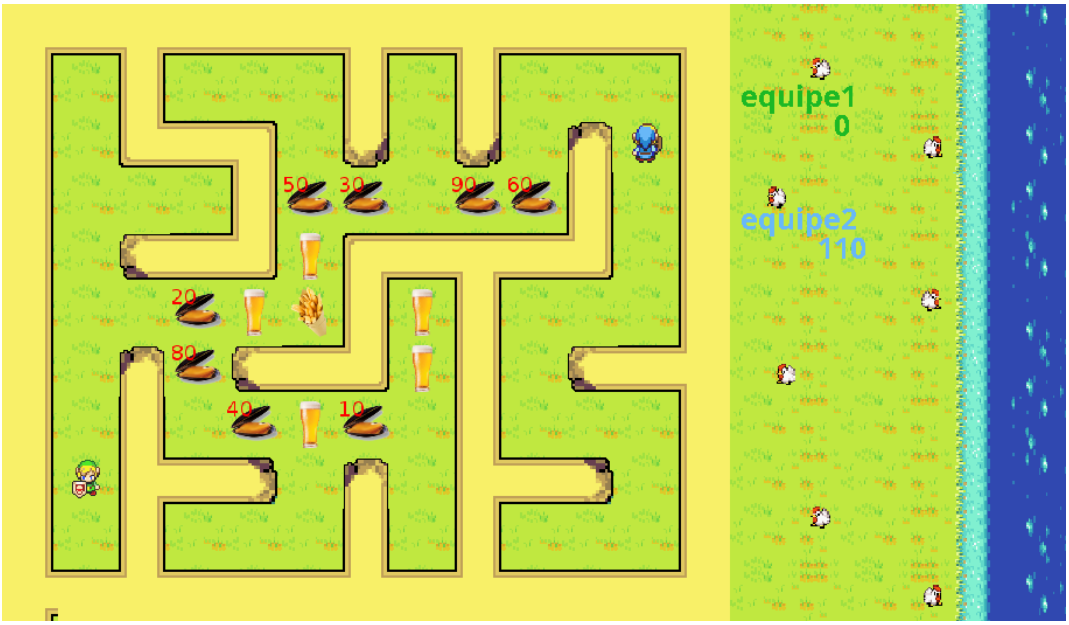

FIGURE 3 – Labyrinthe dont la structure est donnée sous forme de chaîne de caractères.

La taille du plateau correspond à la largeur et à la hauteur séparées par le caractère '`x`'.

Pour la structure du labyrinthe, l'ensemble des blocs est donné sous forme d'une ligne de lettres ou de nombres séparés par des tirets. Cette ligne représente le labyrinthe entier en partant du bloc tout en haut à gauche [4] jusqu'en bas à droite (de gauche à droite et de haut en bas). Les codes utilisés sont les suivants :

- La lettre "Mu" pour un mur.
- La lettre "So" pour de le sol.
- La lettre "Bs" pour un bonus frite (bonus de saut).
- La lettre "Bp" pour un bonus bière (bonus des 3 pas).
- Un nombre pour une moule. Le nombre représente le nombre de points que vous gagnerez en ramassant la moule.

Remarque : Le labyrinthe est **toujours** entouré de murs. Il n'y a pas de sortie. Dans les informations de structure, les murs entourant le labyrinthe sont présentes. C'est pourquoi la structure commencera toujours par autant de murs que la largeur du labyrinthe (+1 pour le mur tout à gauche sur la première ligne jouable).

Les informations concernant les joueurs contiennent : Le nombre de joueurs dans le labyrinthe suivi d'autant de couples X,Y séparés par des tirets donnant leur position.

L'exemple précédent représente un labyrinthe de 13 cases horizontalement par 11 cases verticalement. Les 143 (13×11) codes suivants représentent la structure du labyrinthe. Finalement, il est indiqué qu'il y a 2 joueurs. Le joueur 0 se trouve en $(1, 8)$ et le joueur 1 se trouve en $(11, 2)$. **Attention !** Les indices commencent à 0.

4. Bloc se trouvant en (0,0)

2.2 Envoi des données

Les coups que vous allez devoir envoyer seront représentés par une lettre simple pour les mouvements simples et un ensemble de plusieurs lettres séparées par des "-" lorsque vous utiliserez des bonus.

Les messages simples sont les suivants :
— "N" pour aller au Nord (en haut)
— "S" pour aller au Sud (en bas)
— "E" pour aller à l'Est (à droite)
— "O" pour aller à l'Ouest (à gauche)
— "C" pour rester là où vous êtes (Centre). À utiliser si vous voulez passer votre tour.

Les messages composés (si vous utilisez des bonus que vous auriez récoltés) sont les suivants :
— pour utiliser une frite qui vous permet d'avancer de deux cases dans une direction même si un mur vous sépare de la position d'arrivée : envoyez le code "Bs" suivi d'un "-" et de la direction dans laquelle vous voulez aller. Par exemple, si vous envoyez le code "Bs-S", vous allez avancer de deux cases vers le sud même si un mur se trouve juste en dessous de vous. Rappel : ceci n'est possible que si la position d'arrivée est une case sur laquelle vous pouvez marcher.
— pour utiliser une bière qui vous permet de faire trois déplacements au lieu d'un seul : envoyez le code "Bp" suivi d'un "-" et de trois directions (et exactement trois) séparés également par des "-". Par exemple, si vous envoyez le code "Bp-E-E-S", vous allez effectuer les trois déplacements (Est, Est puis Sud) en un seul tour. Rappel : Seuls les coups valables seront effectués.

Attention à bien utiliser vos bonus lorsque c'est possible seulement sinon ils seront perdus.

A noter : Deux joueurs peuvent se trouver sur la même case au même moment.

3 Fichiers fournis

Dans ce dépôt git, vous pourrez trouver plusieurs répertoires :
— `Serveur` → répertoire contenant les fichiers sources pour compiler puis exécuter le serveur.
Pour compiler le serveur, exécutez :
`javac PecheAuxMoules.java`
Pour exécuter le programme serveur, il vous suffit d'exécuter la classe principale :
`java PecheAuxMoules`
Ajoutez l'option -h pour prendre connaissance des différentes options possibles.
— `IA` → Répertoire contenant trois IA fournies en exemple.
`alea` : IA purement aléatoire qui choisit une direction au hasard.
`aleaQuiNeSeCognePas` : IA aléatoire qui choisit une direction au hasard parmi celles possibles.
`parcoursComplet` : IA qui va parcourir le labyrinthe en passant sur toutes les cases.

Pour exécuter un client, compilez et exéctuez le (exemple avec alea) :

```
javac ClientAlea.java
java ClientAlea 127.0.0.1 1337 nomDeMonIA
```

Pour rappel, les paramètres suivants le nom de la classe du client sont : l'adresse ip du serveur et le port à utiliser et le nom de l'IA.