Roboc Game

Le jeu de Roboc est un jeu multijoueur à jouer en réseau.

NB: Pour l'instant, il ne fonctionne qu'en local.

Déroulement d'une partie

Début de la partie

Lorsque la partie débute, tous les joueurs sont positionnés aléatoirement sur le labyrinthe choisi.

Un tour du jeu

Tour à tour, chaque joueur est invité à saisir une commande depuis l'application client. Lorsque c'est son tour, chaque joueur a différentes possibilités:

- *Se déplacer*
  Pour se déplacer d'un pas dans une direction donnée, saisir N (Nord), S (Sud), E (Est) ou O (Ouest).

- *Se déplacer de plusieurs pas*
  Pour se déplacer plusieurs fois dans une direction donnée, saisir la lettre de la direction suivie d'un chiffre.
  Par exemple, pour aller de deux pas vers l'Est, saisir E2.
  Un déplacement de plusieurs cases sera réparti sur les tours à venir à raison d'un pas par tour.

- *Créer une porte*
  Pour créer une porte dans un mur, saisir P et la direction du mur à percer.
  Par exemple, PN pour percer un trou dans le mur situé juste au-dessus du joueur.

- *Murer une porte*
  Pour murer une porte, saisir M et la direction de la porte à murer.
  Par exemple, MN pour murer la porte située juste au-dessus du joueur.

Fin d'une partie

Une partie s'achève lorsqu'un joueur est arrivé sur la porte de sortie du labyrinthe.

Labyrinthes

Les labyrinthes disponibles se trouvent par défaut dans le dossier _game_maps situé à la racine de l'arborescence

du projet.

Ajout de nouveaux labyrinthes

De nouveaux labyrinthes peuvent être ajoutés dans ce répertoire dans des fichiers .txt. Ces labyrinthes doivent respecter les contraintes suivantes:

- Ils doivent être rectangulaires.
- Ils ne doivent comporter que des symboles autorisés.
  Ces symboles sont listés dans la variable valid_map_items du module parameters.py dans le package parameters.
- Ils ne doivent contenir qu'une seule case de sortie (un unique caractère U).
- Ils doivent respecter les contraintes de taille listées dans les variables map_min_size et map_max_size du module parameters.py dans le package parameters.

Signification des symboles

Symboles statiques présents sur la carte:

- Mur: **O**
- Porte: **.**
- Sortie: **U**

Durant le jeu, les joueurs sont représentés sur la carte par leur identifiant numérique.

- Emplacement du joueur 1: **1**
- Emplacement du joueur 2: **2**
- ...
- Emplacement du joueur n: **n**

Pour l'instant, comme les joueurs sont représentés par des entiers sur la carte, il est impossible d'avoir plus de 9 joueurs connectés.

Déroulement d'une session

Une session correspond à une exécution de l'application server.py. Elle peut comporter plusieurs jeux impliquant chacun différents joueurs.

Etape 1: Lancement de l'application serveur

L'application serveur se lance en exécutant le fichier server.py qui se trouve à la racine de l'arborescence du projet.

Etape 2: Choix du labyrinthe

On doit ensuite choisir depuis l'application serveur le labyrinthe qui sera utilisé pour la partie à venir.

Etape 3: Lancement des applications client

Une fois le labyrinthe choisi, on peut exécuter le fichier client.py une fois par joueur.

Etape 4: Lancement d'une partie

Une fois que tous les joueurs sont connectés, une partie commence lorsqu'un joueur saisit la commande C.

Etape 5: Nouvelle partie ou clôture de la session

A la fin de chaque partie, chaque joueur est invité à dire s'il souhaite continuer à jouer.

- S'il reste au moins un joueur de connecté, on revient à l'étape 2 pour relancer une partie.
- Si tous les joueurs se sont déconnectés, on met fin à la session.

Architecture du projet

Le projet est structuré de la manière suivante.

Package sessions

Il contient toute la gestion des sessions client et utilisateur.

Sessions

La classe *MainSession*, qui permet de gérer une session serveur depuis l'exécution de server.py jusqu'à la déconnexion du serveur. Elle hérite de la classe Session.

La classe *ClientSession*, qui permet de gérer la session client. Elle hérite de la classe *Session*.

Interacteurs

Les trois principaux types d'interacteurs sont *ShellInteractor*, *DistantInteractor* et *DeafInteractor*. Ces classes sont utilisées par les sessions client et serveur pour communiquer entre elles et avec les utilisateurs.

ShellInteractor: Utilisée pour communiquer à travers la console avec l'utilisateur.
Les messages envoyés à l'utilisateur sont affichés dans la console, et les messages envoyés par l'utilisateur sont saisis par input dans la console.
En fonctionnement normal, c'est l'interacteur utilisé par le serveur pour communiquer avec l'utilisateur qui choisit le labyrinthe.
Il est aussi utilisé par les sessions clientes pour communiquer avec les utilisateurs saisissant les commandes à envoyer.

DistantInteractor: Les classes *ClientInteractor* et *ServerInteractor* héritent de *DistantInteractor*, et sont utilisées pour communiquer via des sockets. Les messages envoyés à l'utilisateur sont envoyés au socket de communication, et les messages envoyés par l'utilisateur sont récupérés par un recv du socket.

En fonctionnement normal, *ClientInteractor* est l'interacteur utilisé par le serveur pour communiquer avec les sessions clientes, et *ServerInteractor* est l'interacteur utilisé par les sessions clientes pour communiquer avec le serveur.

DeafInteractor: Utilisée pour simuler un interlocuteur. Lors de la création de l'interacteur, on lui passe déjà tous les messages qu'il devra envoyer lorsqu'il sera sollicité. Pour le moment, on n'utilise cet interacteur qu'à des fins de test. A terme, on pourrait faire évoluer cet interacteur pour simuler un joueur lors d'une partie qui incluerait aussi de vrais joueurs. On pourrait ainsi "jouer contre l'ordinateur".

ClientInteractorFactory et *DeafInteractorFactory*: Permettent de créer respectivement des DeafInteractors et des ClientInteractors. On passe l'une d'elles à la classe *MainSession* pour créer les interacteurs des joueurs qui se connectent.

Package game_logic

Il contient les classes Game et Player qui permettent de gérer la logique d'une partie.

Package graphical_layout

Il contient la classe Map qui permet de charger un labyrinthe.

Package test

Il contient les tests des classes MainSession, Game et Player. Ces tests peuvent être lancés via la commande python -m unittest à la racine du projet.

Reste à faire

Pistes d'amélioration:

- Faire en sorte que le serveur ne soit pas forcément en localhost.
- Ajouter une meilleure interface graphique.
- Ajouter la gestion d'un chat entre joueurs.