Alban Bernard
Thomas Blumet
Florian Burdairon
Thomas Halvick

POLYTECH LYON

RAPPORT PROJET JAVA :

Tic-Tac-Toe (Morpion)

Github : https://github.com/florianBurdairon/tic-tac-toe

I. Conception du projet

1) Objectifs du projet

Nous avons décidé de nous lancer dans la réalisation du morpion en souhaitant réaliser toutes les extensions. Cela permettrait de justifier la taille de notre groupe.

Afin de permettre un travail en groupe efficace, nous avons utilisé Github pour versionner et partager notre projet, cependant lors de session de développement en groupe, nous avons également beaucoup utilisé la fonction “Code With Me” de IntelliJ.

Notre jeu devait permettre de jouer contre un humain aussi bien que contre une intelligence artificielle, sur un seul ordinateur comme sur plusieurs lors de parties en réseau. La taille de la grille de jeu devait être variable (selon le choix de l’utilisateur). Les joueurs devaient pouvoir sauvegarder leur partie pour la reprendre plus tard. Et tout cela via une compilation automatisée grâce à Maven, réalisée après des tests nécessaires pour vérifier le bon fonctionnement du jeu.

Le jeu devait se jouer dans un invite de commande et afin de rendre le jeu plus clair, nous avons ajouté des couleurs aux textes. La plupart des invités les gère mais ce n’est pas nativement le cas des invites dans les logiciels de développement comme IntelliJ.

2) Architecture globale

Prenant en compte toutes ces contraintes, nous avons commencé par réaliser de rapides et simples diagrammes de classes et d'objets afin de nous donner une idée de l’architecture finale de notre projet.

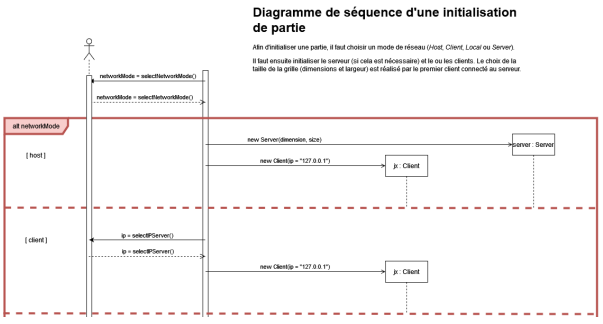
L’extension du jeu en réseau est l’extension qui a posé le plus de contraintes sur cette architecture. Dans le but de la faciliter au maximum, nous avons décidé de considérer toute partie comme une partie en réseau, avec la particularité pour un jeu en local que ce fait soit complètement invisible pour le ou les joueurs.

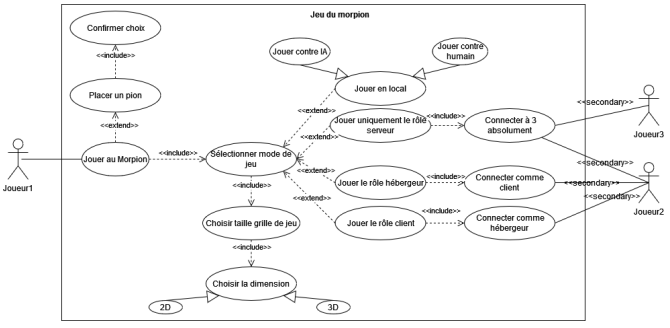
Pour les deux dimensions de grille (2D ou 3D), nous avons utilisé les règles du polymorphisme avec une interface Grid, la seule différence entre une partie en 2D et en 3D sera alors au niveau de l’instanciation de la grille de jeu..

L'intelligence artificielle contre laquelle il est possible de jouer utilise l’algorithme de minimax, il est déconseillé de lancer des parties avec une grille trop grande (surtout en 3D) afin de garder un temps de jeu agréable. Bonne chance pour la vaincre !

La sauvegarde convertit la partie en fichiers .json qui seront alors stockés sur l'ordinateur qui fait tourner le serveur. Ainsi, il est nécessaire de relancer le serveur sur la même machine dans le but de pouvoir continuer une partie.

De nombreux diagrammes (classes, séquence, cas d’utilisation) sont disponibles dans le répertoire sur Github (aperçu ci-dessous).





3) Informations générales

Le jeu est développé avec Java 17.0.6.

Le jeu est compatible avec Windows et Linux.

Le projet est disponible sur GitHub au lien https://github.com/florianBurdairon/tic-tac-toe.

L'archive comprend le code java dans le répertoire **src** dont les tests effectués et la javadoc dans le répertoire **javadocs**. Le répertoire **diagrams** comprend tous les diagrammes conçus lors de ce projet.

Pour jouer en réseau sur au moins deux ordinateurs différents, il est nécessaire que les deux soient connectés au même réseau local et que l'ordinateur faisant tourner le serveur ait ouvert son port **9876**.

La répartition des tâches s'est faite selon la liste suivante :

- Alban :
  - Développement du lanceur du jeu (choix du mode de jeu réseau)
  - Développement des fonctionnalités réseau
  - Développement des interactions utilisateur
  - Réalisation du diagramme de séquence
  - Rédaction du rapport
- Thomas B :
  - Développement du déroulé du jeu
  - Développement du système de sauvegarde
  - Développement des interactions utilisateur
  - Réalisation du diagramme de classe
  - Réalisation du diagramme d'objet
  - Rédaction du rapport
- Florian :
  - Manager du projet
  - Développement du déroulé du jeu
  - Développement du système de sauvegarde
  - Développement du lanceur du jeu (choix du mode de jeu réseau)
  - Développement des interactions utilisateur
  - Paramétrage de Maven, gestion des dépendances
- Thomas H :
  - Développement du système de grille
  - Développement de l'intelligence artificielle
  - Développement des tests unitaires

4) Exécution du jeu

Nous avons utilisé Maven comme structure pour le projet.
Pour compiler le projet dans une archive jar, il faut exécuter la commande : **mvn install**.
Cette commande va générer 2 fichiers jar dans le dossier target :

- Tic-Tac-Toe-1.0.jar : Fichier jar sans la dépendance GSON (nécessaire pour les sauvegardes)
- Tic-Tac-Toe-1.0-jar-with-dependencies.jar : Fichier jar possédant toutes les dépendances

Pour exécuter le jar, il faut exécuter la commande :
java -jar Tic-Tac-Toe-1.0-jar-with-dependencies.jar

Il y a également possibilité de saisir le mode réseau dans la ligne de commande pour pouvoir sauter le choix du mode réseau :
java -jar Tic-Tac-Toe-1.0-jar-with-dependencies.jar *[mode réseau]*

Modes réseau disponibles :

- local
- host
- client
- server

Pour exécuter uniquement les tests unitaires, il faut saisir la ligne : **mvn test**

II. Déroulé du jeu

1) Choisir le mode de jeu réseau

Lors du lancement de l'exécutable java, l'utilisateur est invité à choisir un mode de jeu réseau. Il a alors le choix entre 4 différents modes de jeu pour l'ordinateur actuel :

```
Veuillez choisir votre mode de réseau pour jouer.
 Disponible : 0 - Jeu en local (1vs1 ou 1vsIA)
              1 - Hébergeur
              2 - Client
              3 - Serveur uniquement
Votre choix :
```

- **LOCAL :** Le jeu sera exécuté en mode local, un seul ordinateur est nécessaire. Dans ce cas, le joueur devra refaire un choix pour le type de son adversaire : une intelligence artificielle s'il veut s'entraîner tout seul ou bien un autre joueur s'il veut jouer contre un ami sur le même ordinateur.

```
Veuillez choisir votre type d'adversaire :
  0 - Humain
  1 - Intelligence Artificielle
Votre choix :
```

- **HÉBERGEUR :** Le jeu sera exécuté en mode hébergeur, un second ordinateur sera nécessaire afin de jouer le second joueur. Le serveur sera hébergé sur cet ordinateur.
- **CLIENT :** Le jeu sera exécuté seulement côté joueur, sans lancer de serveur. Il est alors nécessaire de rentrer l'adresse IP du serveur préalablement lancé grâce aux autres options de mode de jeu réseau.
- **SERVEUR :** Le jeu sera exécuté uniquement côté serveur, sans aucun client connecté dessus. Il faudra alors que les joueurs se connectent via le mode "CLIENT" au serveur pour pouvoir jouer.

```
Lancement du serveur en mode serveur...
Adresse IP du serveur : 10.188.102.243
```

Une fois les deux joueurs connectés sur le serveur, le jeu est lancé !

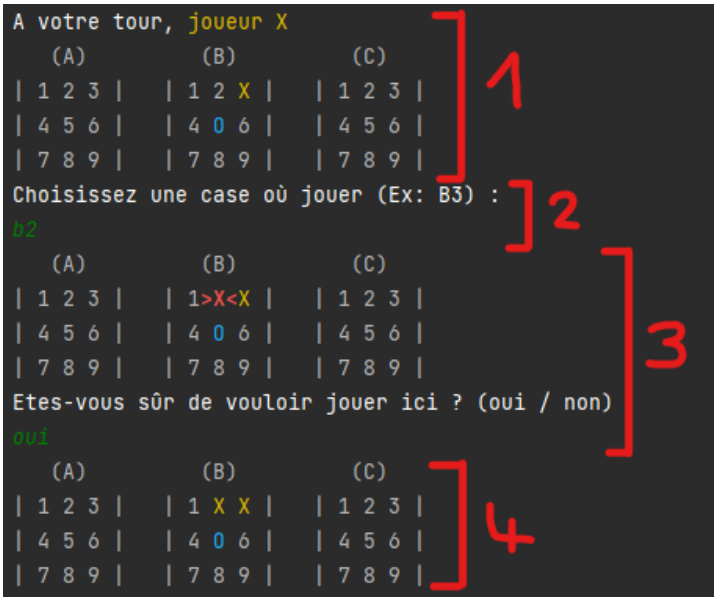
2) Déroulement de la partie

Le premier joueur à avoir été connecté sur le serveur choisit la taille de la grille de jeu et la dimension de celle-ci. Une fois ces données validées, la partie commence : chacun des joueurs se voit attribuer aléatoirement "X" ou "O", et le premier à jouer est le joueur "X".

Ensuite, chacun des joueurs joue lorsque son tour vient jusqu'à la victoire de l'un des deux ou que la grille soit entièrement remplie (dans le cas d'une égalité).

Un tour se présente de cette façon :

1. Présentation de la grille au tour actuel.
2. Demande du prochain coup du joueur.
3. Demande de confirmation du joueur.
    a. Si le joueur confirme, on continue.
    b. Si le joueur refuse, on revient en 2.
4. Affichage de la grille après le nouveau coup.



3) Gestion des erreurs

Lors du déroulé de la partie, de nombreuses erreurs peuvent survenir, notamment lors des entrées utilisateur. Celles-ci sont simples à gérer, il suffit de demander de nouveau au joueur d'entrer une valeur.

Cependant, certaines erreurs peuvent être plus compliqués à résoudre comme lors d'une déconnexion involontaire. Nous gérons ce cas grâce au système de battements de coeurs implémenté entre le serveur et les clients. Lors d'une déconnexion d'un seul des deux joueurs, il est proposé à celui encore connecté s'il souhaite directement quitter la partie ou sauvegarder avant de quitter.

4) Gestion de la sauvegarde / interruption d'une partie

La possibilité est donnée, lors du tour d'un joueur, que celui-ci puisse saisir les mots-clés "**save**" ou "**quit**" lors de la saisie de la case à jouer.
Ces actions permettent respectivement :

- **soit de sauvegarder la partie en cours**: il est demandé à l'utilisateur de saisir un nom pour la sauvegarde, sauf si celle-ci existe déjà et dans ce cas la sauvegarde précédente est écrasée par la nouvelle,
- **soit de quitter volontairement une partie**: il est alors demandé à l'autre joueur restant de quitter aussi ou de sauvegarder avant de quitter.

Au redémarrage du jeu, une liste des sauvegardes doit apparaître après avoir saisi le mode de jeu. Le joueur peut alors choisir de reprendre une des parties ou simplement d'en démarrer une nouvelle.

Lors de la sauvegarde du jeu, 2 fichiers sont créés sur l'ordinateur du serveur. Ces deux fichiers sont créés dans un dossier sous un nom demandé au joueur dans les dossiers d'application de l'ordinateur (dans le répertoire AppData présent sur les postes avec un OS Windows, ou dans le répertoire "Home" pour ceux sous OS Linux). Ces répertoires ont été choisis, car ils sont disponibles depuis n'importe où sur l'ordinateur.

III. Spécificité du réseau

1) Messages du serveur pour les joueurs

Message du serveur pour les joueurs		
ID du message	Description	Paramètres
Select Dimensions	Demande à la cible de choisir la taille et la dimension de la grille de jeu.	
StartGame	Démarre le jeu ou reprend une partie sauvegardée.	Rôle du joueur ciblé (X ou O) + grille de jeu si sauvegarde
EndGame	Met fin au jeu.	Dernier coup + Joueur victorieux + est-ce une égalité
Play	Joue un tour d'un joueur.	Dernier coup du joueur adverse
Ask Confirmation	Demande de confirmation pour le tour du joueur.	
Validate	Valide le tour du joueur.	
Resume Game	Propose la liste des sauvegardes disponibles.	Liste des sauvegardes
Opponent Disconnected	Transmet la déconnexion du joueur adverse.	
Error	Rapporte une erreur trouvée.	Code de l'erreur
Quit	Demande au joueur de se déconnecter.	

2) Messages d'un joueur pour le serveur

Message d'un joueur pour le serveur		
ID du message	Description	Paramètres
Answer Dimensions	Répond à la demande du choix de la taille et de la dimension de la grille de jeu.	
Place	Définit le coup que le joueur souhaite jouer.	Coup du joueur
Wait Message	Rapporte l'attente du prochain message.	
Quit	Définit si le joueur souhaite sauvegarder ou directement quitter la partie.	Est ce que la partie doit être sauvegardée

Message d’un joueur pour le serveur		
ID du message	Description	Paramètres
Confirmation	Répond à la demande de confirmation du coup du joueur.	

3) Erreurs réseau

Lors de l’échec de la connexion ou d’une interruption brusque, chaque côté de la connexion à la capacité de détecter cette coupure et d’agir en conséquence. Cela se fait via le message d’ID “Network Error”. Côté serveur, ce message lancera “Opponent disconnected” pour l’autre joueur. Côté joueur, ce message arrêtera tout simplement le jeu.