Projet Individuel

Sujet 70 : Serveur de jeux au tour par tour

Etudiant : Yassine Badache

Encadrants : Yoann DUFRESNE, Gauvin MARQUET

Université Lille1
Sciences et Technologies

Table des matières

1 Introduction

L'an dernier, deux étudiants en Master 1 Informatique ont développé un *framework* permettant de développer, à l'aide d'automates, des jeux au tour par tour. Les jeux ainsi implémentés peuvent héberger une partie de manière indépendante, contenant les protocoles de communication entre elle-même et le client.

Le but principal de ce PJI était de concevoir un composant logiciel permettant d'accéder aux différents jeux implémentés, et de les instancier par ce biais. Ceci permettrait alors un accès simplifié aux jeux développés et une centralisation des ressources nécessaires à son bon fonctionnement.

Au-delà de l'accès public aux ressources, c'est notamment dans une vision plus 'interne' qu'a été proposé ce projet individuel. En effet, l'aboutissement final de ce composant logiciel serait d'héberger des intelligences artificelles, s'affrontant sur les supports développés via le *framework* de l'an dernier.

2 Conception

Le programme développé ici est supposé répondre à un besoin bien précis, c'est-à-dire l'accessibilité d'une ressource bien définie et son utilisation, que ce soit par un utilisateur humain ou par une intelligence artificielle développée spécifiquement pour le jeu souhaité. La première partie de ce projet a donc été dédiée à la conception de ce projet par les besoins d'un utilisateur quel qu'il soit.

2.1 Diagrammes de séquence

Ci-dessous sont détaillés les scénarios d'usage du logiciel, ainsi que leurs diagrammes de séquence associés. C'est l'ensemble de ces éléments qui nous a permis par la suite de cerner le cœur de ce qui est attendu à la fin de ce sujet.

Nous nommerons ici "entité" l'utilisateur : en effet, en plus d'un utilisateur humain, administrateur ou non, une intelligence artificielle devrait également être en mesure d'accéder à notre service afin d'instancier une partie ou rejoindre une partie existante, se confrontant alors à une autre intelligence artificielle ou à un joueur déjà présent en jeu.

2.1.1 Premier cas d'utilisation : créer une partie d'un jeu

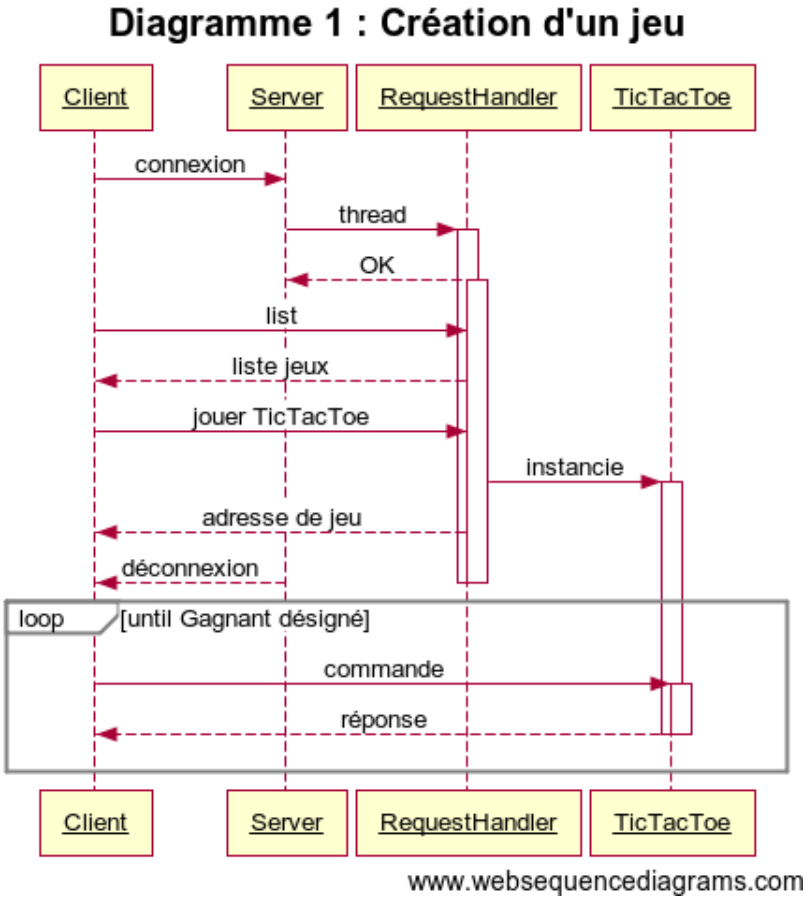

FIGURE 2.1 – Le premier cas d'utilisation, basique

Description

Une entité souhaite jouer à un jeu. Elle lance donc le client, qui lui permet de se connecter au serveur. Elle entre la commande "list" qui lui permet de voir quels jeux sont disponibles. Parmi eux, elle choisit le TicTacToe, qui est alors instancié du fait qu'aucune partie n'est active. Le serveur lui renvoie l'adresse de jeu, à laquelle l'entité se pourra se connecter que si suffisamment de joueurs souhaitent participer.

Ce cas d'utilisation pose les bases de l'utilisation de notre serveur : on l'utilise principalement pour jouer, et on attend que suffisamment de joueurs soient en attente pour ce jeu afin d'en instancier une.

2.1.2 Deuxième cas d'utilisation : rejoindre une partie existante

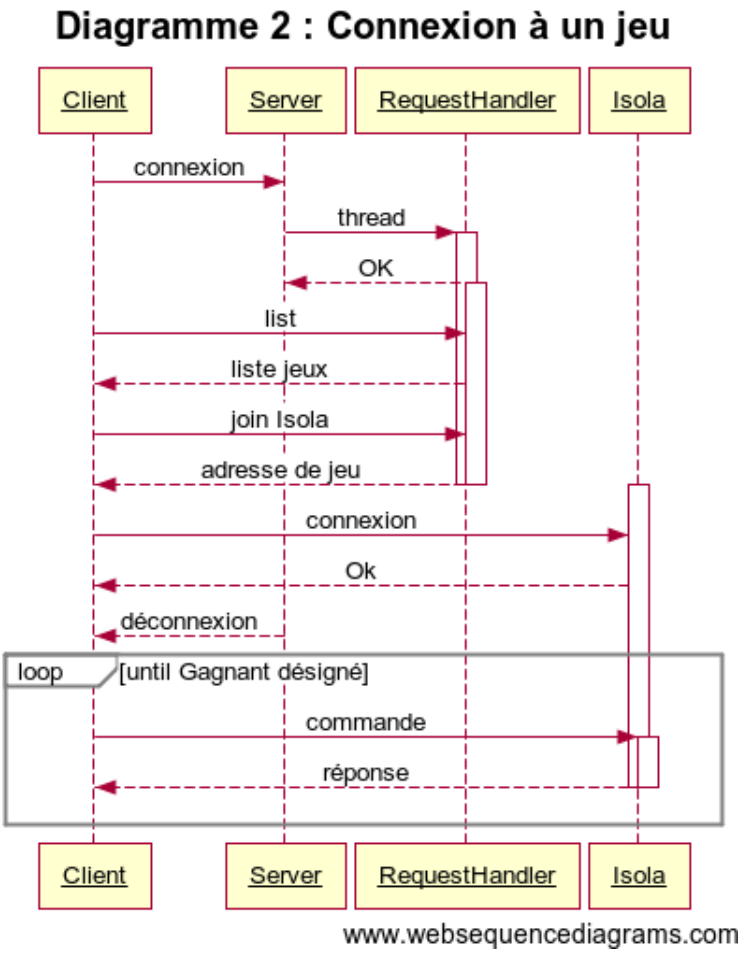

FIGURE 2.2 – Rejoindre une partie

Description

Une entité souhaite jouer à un jeu. Elle lance donc le client, qui lui permet de se connecter au serveur. Elle entre la commande "list" qui lui permet de voir quels jeux sont disponibles. Parmi eux, elle choisit de jouer à l'Isola. Des parties étant déjà en cours, le serveur lui renvoie l'adresse d'une des parties, à laquelle l'entité se connecte pour ensuite jouer à l'Isola.

Ce cas d'utilisation introduit la notion non plus de création d'une instance de jeu, mais bien de connexion à une partie existante. Il faut donc définir des moyens de contrôler et de retenir les jeux étant actuellement instanciés pour pouvoir effectuer la connexion entre le client et le jeu en question.

2.1.3 Troisième cas d'utilisation : ajouter un jeu sur le serveur

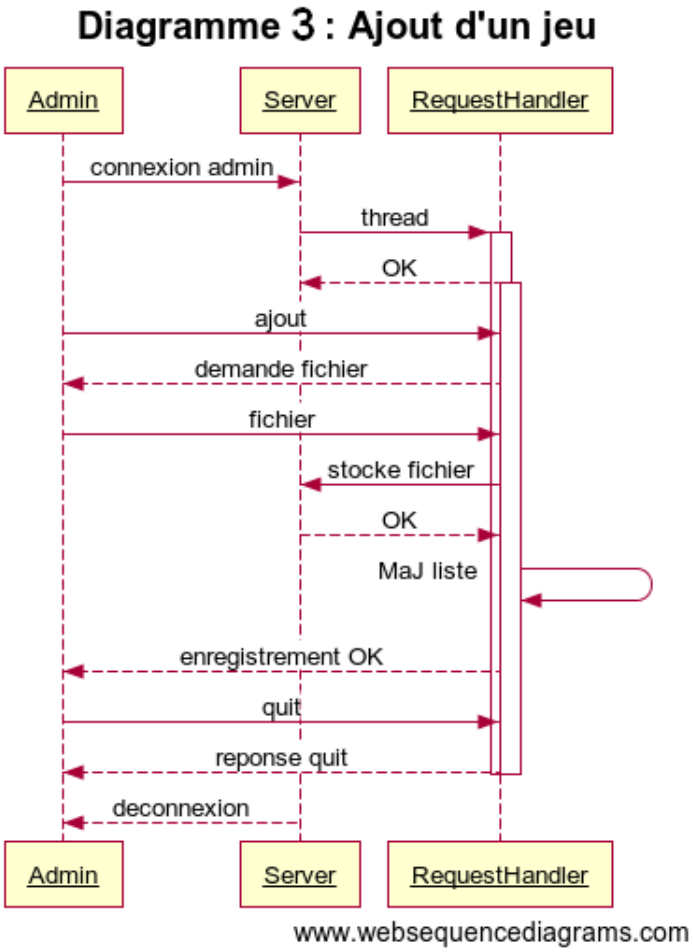

FIGURE 2.3 – Ajout d'un jeu

Description

Un administrateur se connecte au service en ligne, via un pseudonyme et mot de passe associés. Il ajoute ensuite un jeu via la commande associée, et le fichier du jeu lui est demandé : il le choisir, et valide son choix. Il choisit un nom pour le jeu choisi, et le serveur stocke ledit fichier dans ses données, mettant à jour la liste des jeux disponibles. Une fois l'enregistrement effectué, l'administrateur entre la commande 'quit' et se voit déconnecté du serveur.

Ce troisième cas d'utilisation introduit un autre acteur possible, soit l'administrateur. Entrent en compte des problèmes de sécurité liés au statut privilégié, l'enregistrement de fichiers et la saisie de celui-ci.

2.1.4 Quatrième cas d'utilisation : quitter une partie en cours

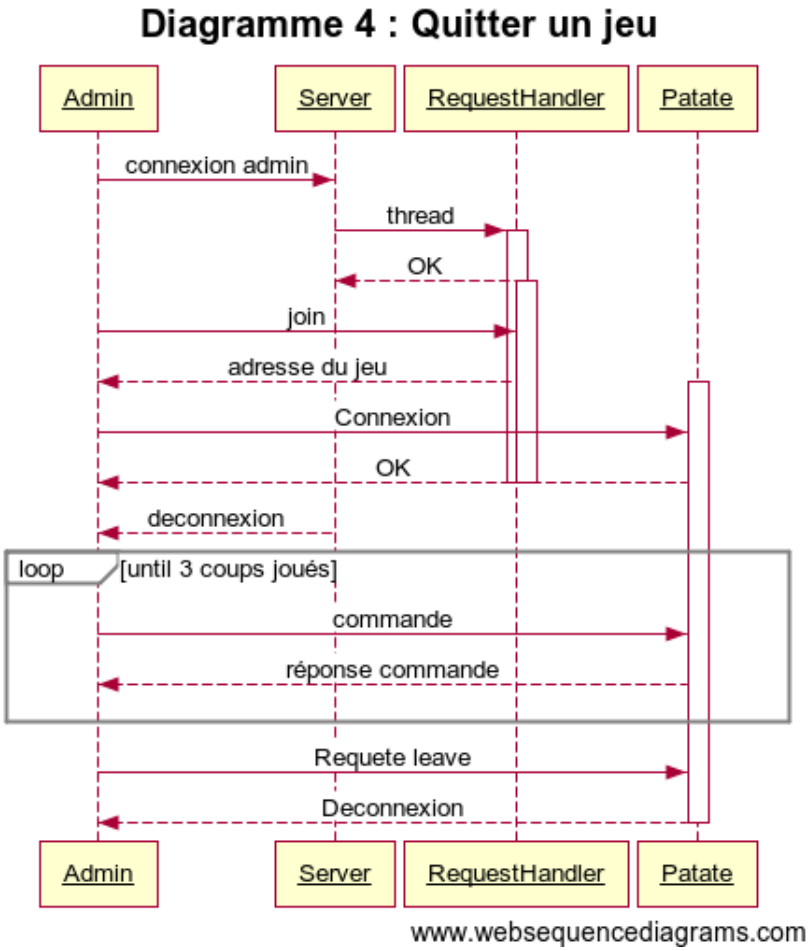

FIGURE 2.4 – Quitter une partie

Description

Une entité souhaite participer à une partie. Elle entre donc la commande 'join Patate', qui lui permettra de se connecter à une partie actuellement en cours de la Patate chaude. L'adresse du jeu lui est renvoyée et elle s'y connecte. Cependant, au bout de trois coups joués, elle se rend compte qu'elle ne souhaite pas continuer à jouer. Elle entre donc la commande 'leave', et se voit immédiatement déconnectée du jeu en cours.

Lorsque l'on joue à un jeu, il est aussi bénéfique de savoir comment le quitter, pour des raisons diverses. Cependant, ce cas d'utilisation ne concerne pas vraiment le serveur développé par nos soins : en effet, une fois connecté au jeu, l'interaction avec le *software* devient nulle, et la commande permettant de quitter une partie devrait alors être partie intégrante du système de jeu.

2.2 Automate à états finis en découlant

De ces diagrammes de séquence découlent quelques questions primaires qui ont dû être traités en tout premier lieu, avant encore d'entamer la conception de notre programme, à commencer par sa représentation 'graphique'. Le schéma suivant montre le modèle final de notre programme ainsi que tous ses états et transitions possibles.

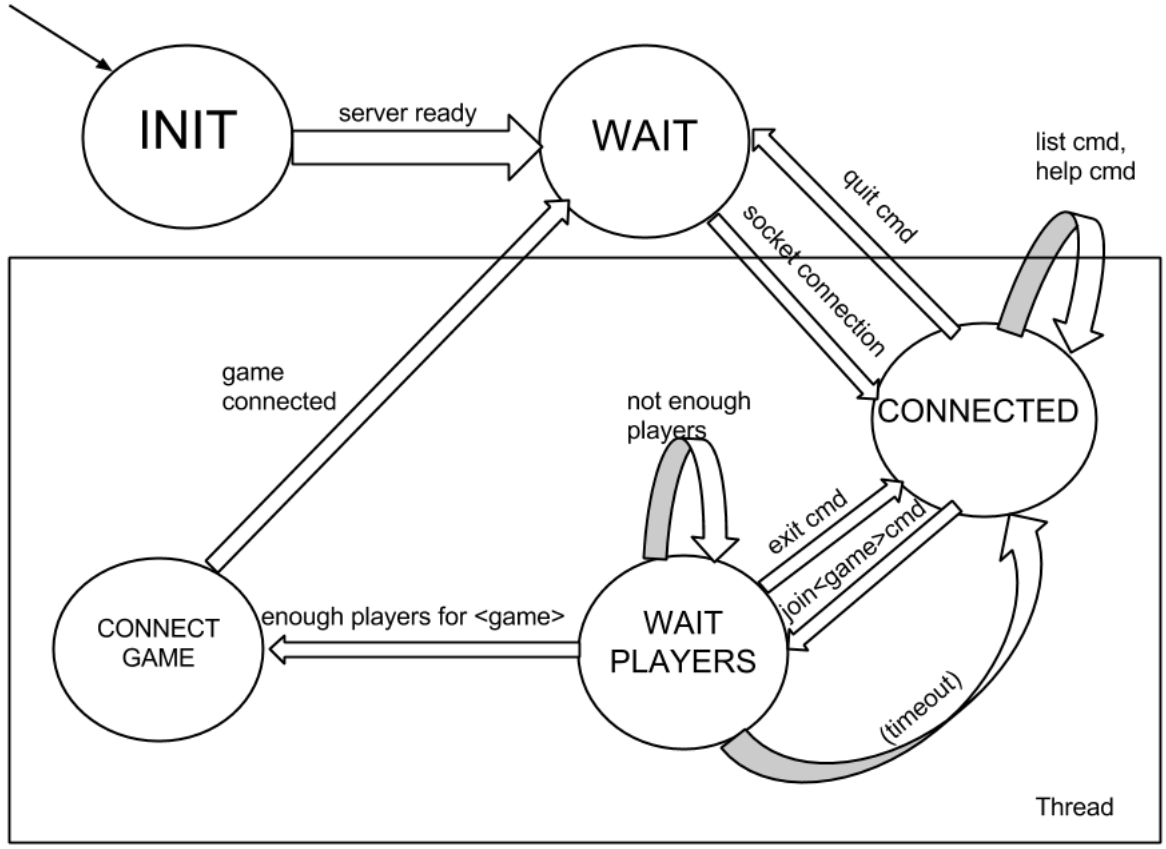

FIGURE 2.5 – L'automate décrivant notre serveur

Explication

- **INIT**
  La phase d'initialisation du serveur, qui devrait être plus ou moins instantanée.

- **WAIT**
  Une fois l'initialisation terminée et le serveur prêt, celui-ci entre dans une phase d'attente, où il attendra la connexion d'un client. Dans le cadre de ce projet, le client est une simple application développée en parallèle, mais à l'avenir, le logiciel est prévu pour soutenir n'importe quel type de client.

- **CONNECTED**
  Le client, une fois connecté, est en mesure d'envoyer des commandes et en recevoir les réponses. Le format utilisé est le JSON, pour la compréhension sur multiples supports et son universalité. Si l'utilisateur entre les commandes 'list' ou 'help', il reste dans l'état courant, CONNECTED. Il ne changera d'état qu'à deux conditions :

  – La commande 'quit', pour se déconnecter du service
  – La commande 'join (game)' afin de d'accéder au service principal du serveur, soit la connexion à un jeu.

- **WAIT PLAYERS**
  Cet état est un état intermédiaire entre l'envoi de la commande et la connexion effective au jeu demandé. En effet, on ne peut se connecter à jeu que sous certaines conditions :

  – Il y a déjà une ou plusieurs instances du jeu demandé en cours, et il y a des places restantes pour y jouer -au cas où ils ont plus de deux joueurs-.
  – Il y a assez de joueurs demandant le jeu pour instancier une nouvelle partie.

  En dehors de ces deux conditions, le client restera dans cet état. Si l'attente est trop longue, le client est renvoyé dans l'état précédent, pouvant alors demander un autre jeu ou retenter une connexion / une instanciation du jeu existant.

- **CONNECT GAME**
  Une fois que l'une ou l'autre de ces conditions sont réunies, le client peut enfin se voir connecter à l'instance d'une partie ou la créer avec un autre joueur, et ainsi y jouer. Dès lors, le rôle du serveur est terminé, et il retourne dans l'état **WAIT** pour qu'un autre client s'y connecte.

Ainsi défini, nous avons un cadre clair et des états définis auxquels s'en tenir lors de notre développement : notre ligne de conduite est désormais claire pour le développement de ce serveur, nous pouvons nous pencher sur les aspects plus techniques de ce projet, posant les problématiques et y donnant des éléments de réponse à intégrer dans notre logiciel.

3 Problématiques soulevées

L'ensemble des diagrammes proposés plus haut nous démontrent qu'un certain nombre de questions méritent d'être posées, qu'elles soient techniques ou intuitives, et l'automate à états finis nous interrogent sur les fonctionnalités techniques du sujet. Cette partie les résume et y apporte un élément de réponse, afin d'expliquer au mieux le fonctionnement de notre serveur.