

Rapport du mini-projet Java : Space of War

Décembre 2018



Etudiants : David Albert, Matthieu Bellucci, Thibaud Clavel et Henri Durozay

à l'attention de : M.Vercouter

Table des matières

| | |
|--|-----------|
| 1 Cahier des charges | 2 |
| 1.1 Age of War | 2 |
| 1.2 Space of War | 3 |
| 1.2.1 Création d'une partie | 3 |
| 1.2.2 Partie réseau | 3 |
| 1.2.3 Fonctionnement global du jeu | 3 |
| 1.2.4 Règles de combat | 3 |
| 1.2.5 Système monétaire | 3 |
| 2 Modélisation du projet | 4 |
| 2.1 Cas d'utilisation | 4 |
| 2.1.1 Lobby | 4 |
| 2.1.2 Jeu | 4 |
| 2.2 Diagramme de composants | 5 |
| 2.3 Diagrammes de classes | 6 |
| 2.3.1 Lobby | 6 |
| 2.3.2 Jeu | 7 |
| 2.4 Diagrammes de séquences | 8 |
| 2.4.1 Lobby | 8 |
| 2.4.2 Serveur Partie | 10 |
| 2.4.3 Joueur Partie | 11 |
| 3 Exemple d'exécution du programme | 12 |
| 4 Difficultés rencontrées | 17 |
| 5 Répartition du travail | 18 |

Partie 1

Cahier des charges

Space of War est un jeu inspiré du célèbre jeu Flash Player Age of War.

1.1 Age of War

Age of War est un jeu de défense, de stratégie et de réflexion où l'on doit à travers cinq âges différents (de la préhistoire au futur) affronter une armée en essayant de détruire la base de l'adversaire. Pour se défendre, il est possible de déployer des unités mobiles qui se déploient sur une unique ligne de front. D'autres part, il y a également des unités fixes. En effet, il est possible de mettre des tourelles fixes sur sa propre base. Pour déployer des unités, il y a un système d'argent. Chaque unité coûte de l'argent. Au lancement du jeu, chaque joueur possède une certaine somme d'argent pour pouvoir déployer ses premières unités. Puis, le joueur gagne de l'argent en détruisant les unités de son adversaire.



FIGURE 1.1 – Screen d'une partie de Age of War

1.2 Space of War

1.2.1 Creation d'une partie

Au lancement du jeu, le joueur doit d'abord saisir son nom d'utilisateur. Ensuite, une fentre s'affiche, qui laisse la possibilit au joueur de rejoindre une partie ou de crer sa propre partie.

Si le joueur cre sa propre partie, il doit saisir le nom de la partie, le nombre de joueur ainsi qu'un mot de passe si il veut rendre la partie prive. Dans ce cas, le joueur est l'hte de la partie et il doit attendre que suffisamment de joueurs aient rejoint le salon de la partie pour qu'elle dmarre.

Si le joueur ne cre pas sa propre partie, il peut rejoindre le salon d'une partie dj cre. Et la partie se lance ds qu'il y a suffisamment de joueurs.

1.2.2 Partie rseau

Au lancement du jeu, chaque joueur (client) se connecte  un serveur centralis dans lequel sont rpertoris tous les salons de parties. Une fois qu'une partie est lanc, un nouveau serveur est lanc sur la machine du joueur qui a cre la partie. Ainsi, les autres joueurs se connectent  ce serveur.

1.2.3 Fonctionnement global du jeu

Notre version du jeu est relativement diffrente du jeu Age of War. En effet notre jeu peut se jouer de 2  8 joueurs. Et nous nous limitons  l'ge spatiale. Cette version est davantage stratgique puisqu'il est possible de dployer jusqu' 3 groupes d'units. Avec la possibilit de dplacer chaque groupe d'unit  l'endroit souhait.

1.2.4 Rgles de combat

Ds qu'une unit est  port d'une autre unit adverse, alors elle l'attaque. Sinon l'unit se dplace : vers l'objectif si l'unit est la premire de son groupe ou vers l'unit qui se situe devant elle. Concernant les dfenses, le fonctionnement est le mme que pour les units mobiles mais ces units ne se dplacent pas et se situent sur la base.

1.2.5 Systme montaire

Initialement tous les joueurs commencent avec la mme somme d'argent. Ensuite, la seule faon de gagner de l'argent est de tuer des units adverses. Cependant, tuer une unit avec une dfense rapporte moins d'argent que de la tuer avec une unit mobile. Ce systme favorise la creation d'units mobiles et incite donc les joueurs  ne pas baser leur jeu sur la dfense.

Partie 2

Modélisation du projet

2.1 Cas d'utilisation

2.1.1 Lobby

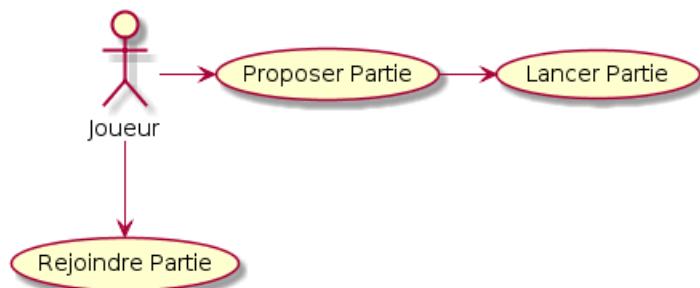


FIGURE 2.1 – Le lobby

2.1.2 Jeu

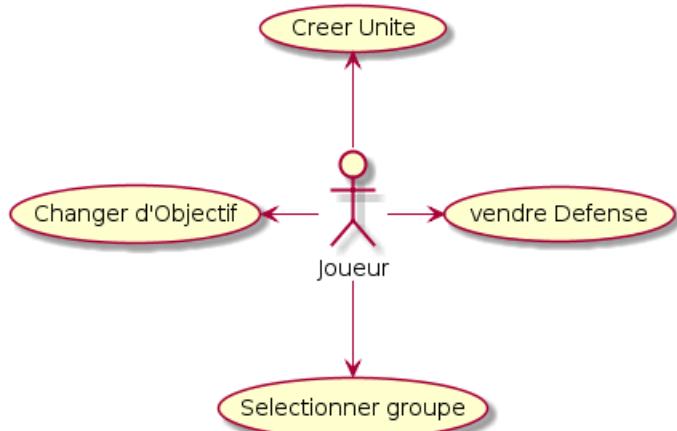


FIGURE 2.2 – Le jeu

2.2 Diagramme de composants

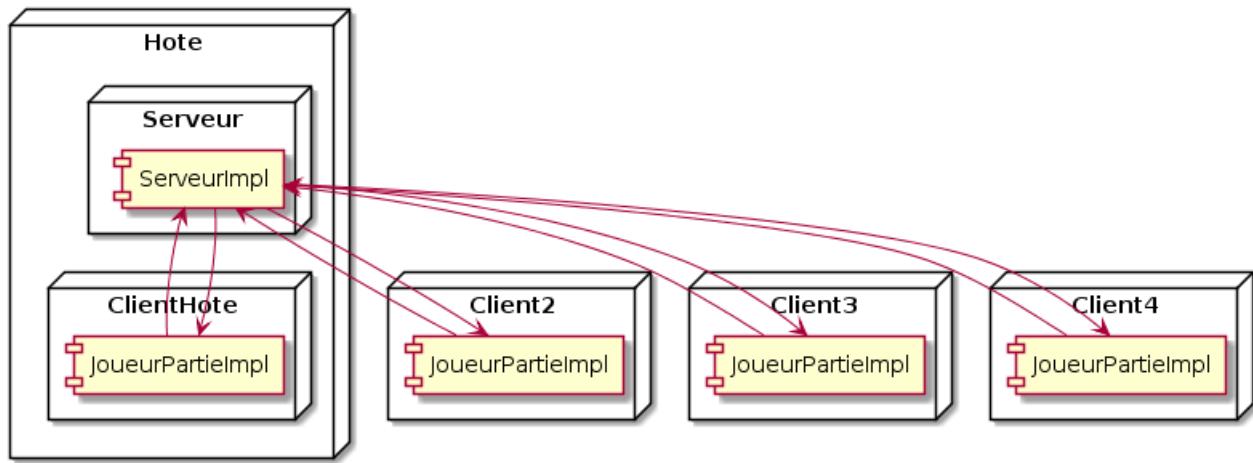


FIGURE 2.3 – Jeu

2.3 Diagrammes de classes

2.3.1 Lobby

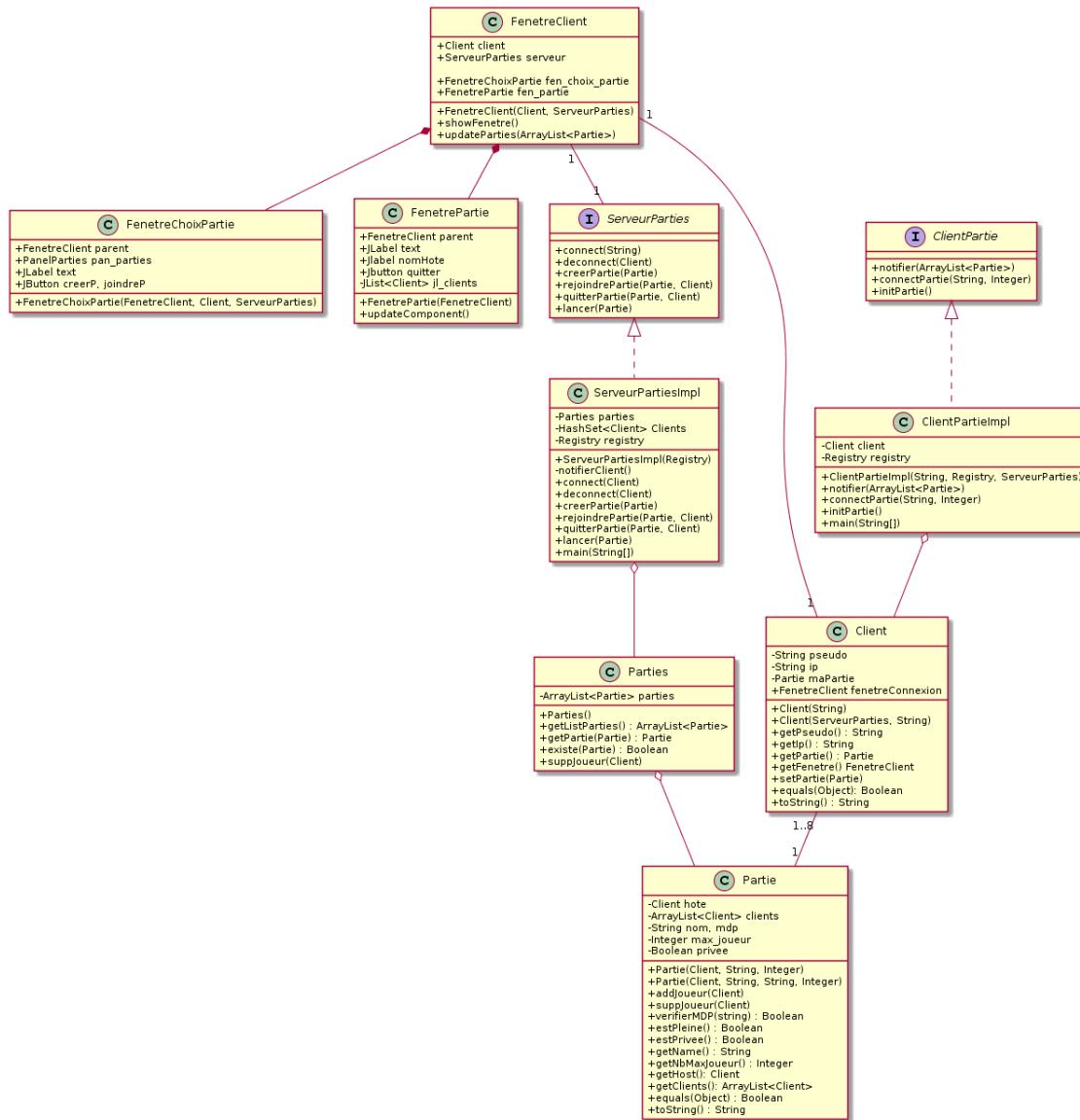


FIGURE 2.4 – Partie réseau

2.3.2 Jeu

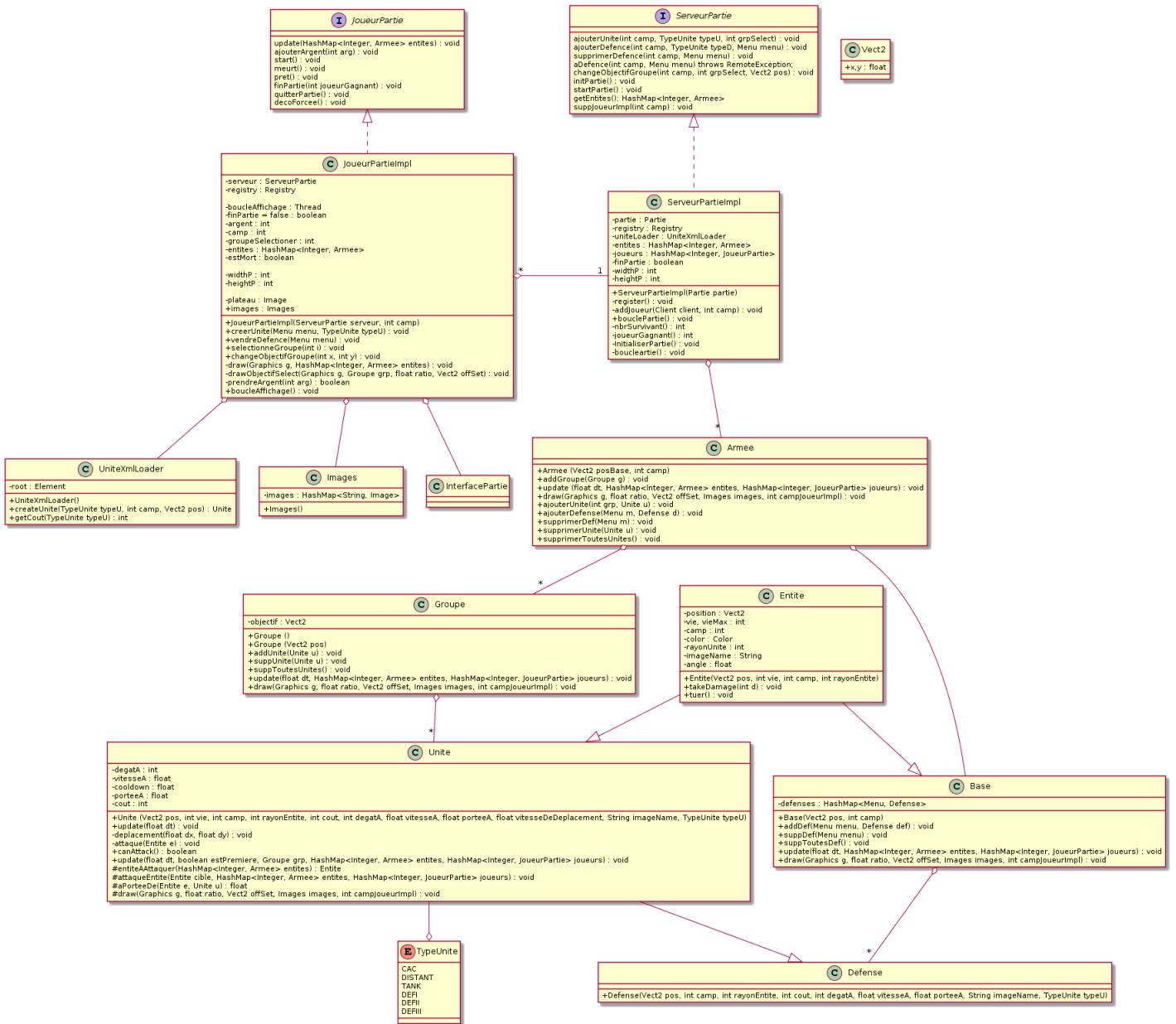


FIGURE 2.5 – Partie jeu

2.4 Diagrammes de séquences

2.4.1 Lobby

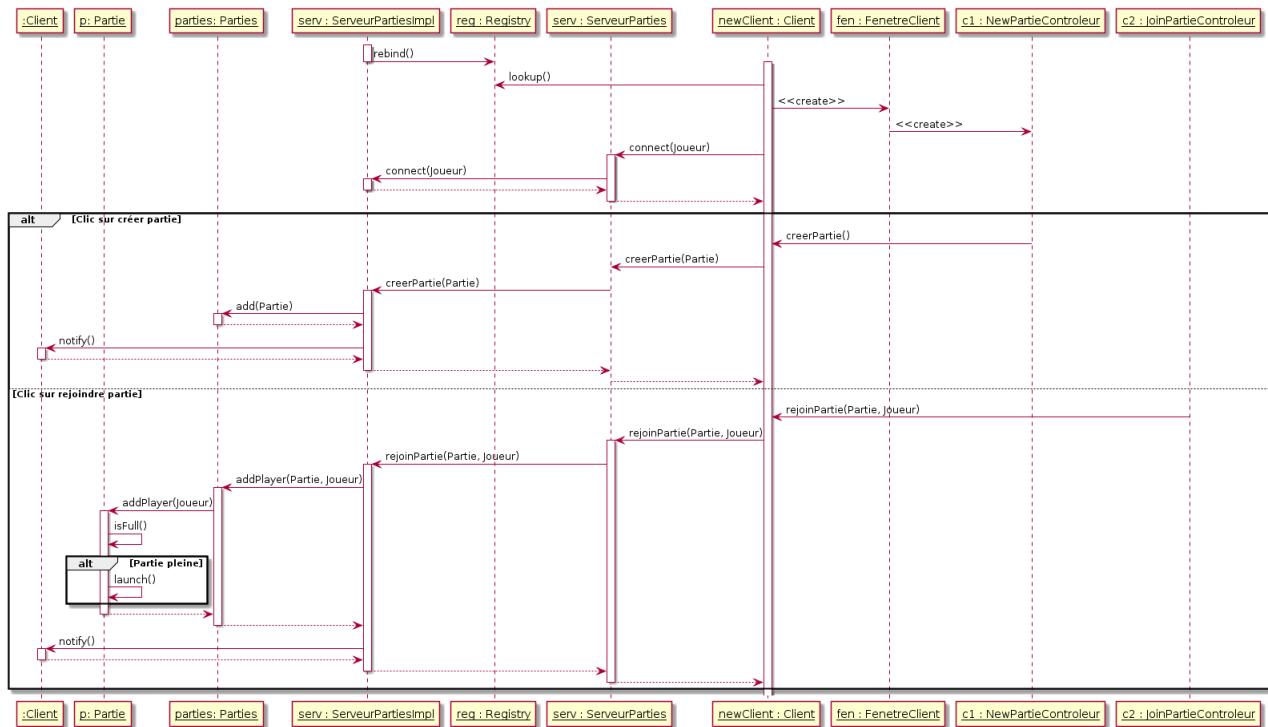


FIGURE 2.6 – Partie réseau

2.4.2 Serveur Partie

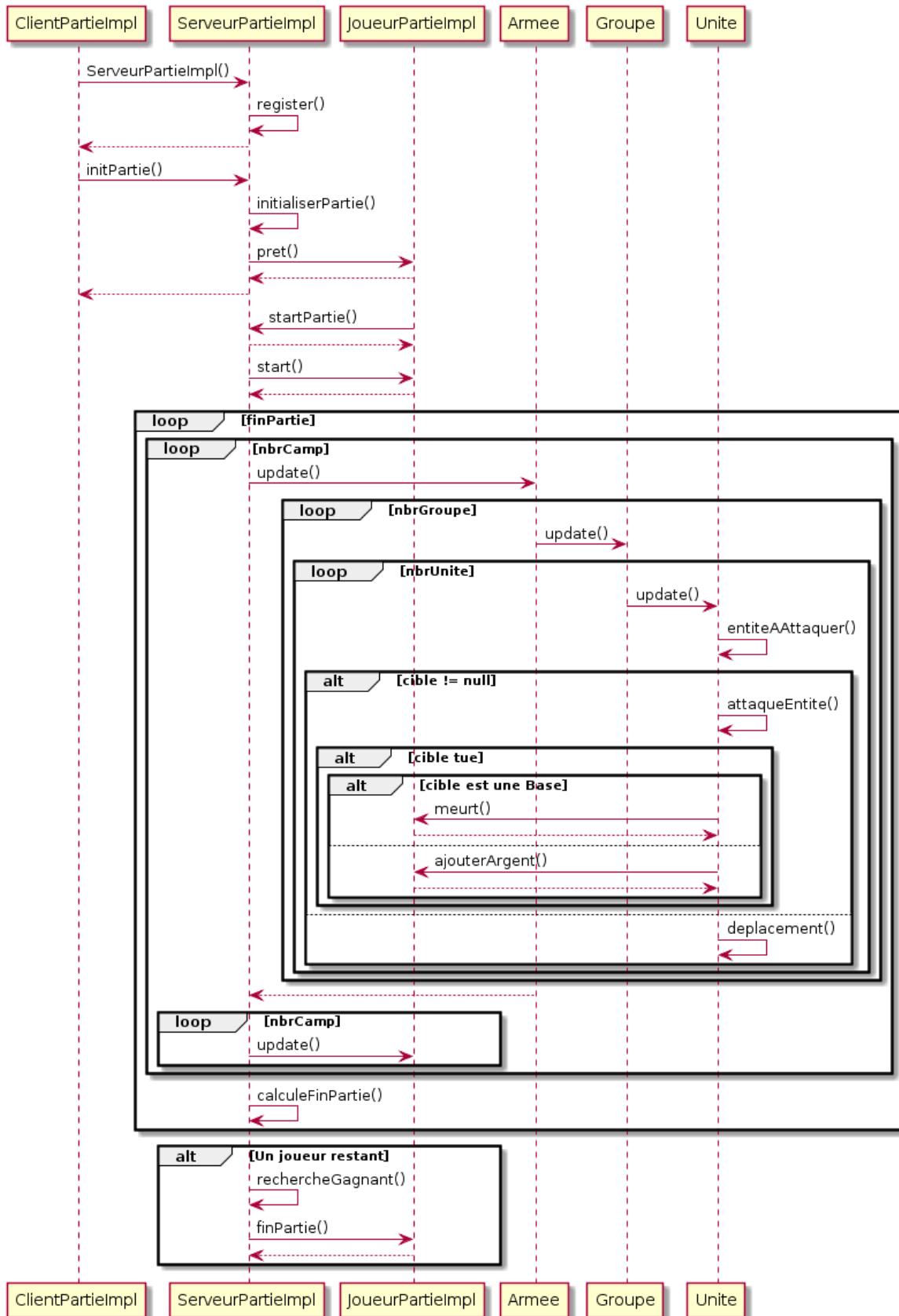


FIGURE 2.7 – Partie jeu

2.4.3 Joueur Partie

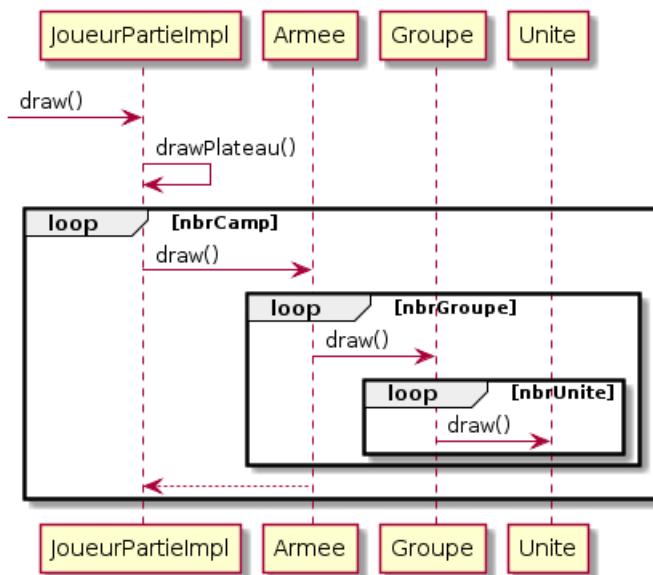


FIGURE 2.8 – Partie jeu

Création Unité

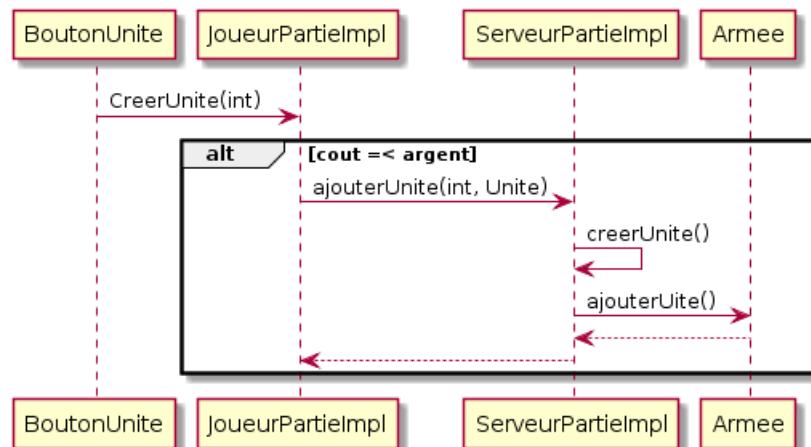


FIGURE 2.9 – Partie jeu

Partie 3

Exemple d'exécution du programme

Au lancement du jeu, l'utilisateur doit dans un premier temps rentrer l'adresse IP du serveur.

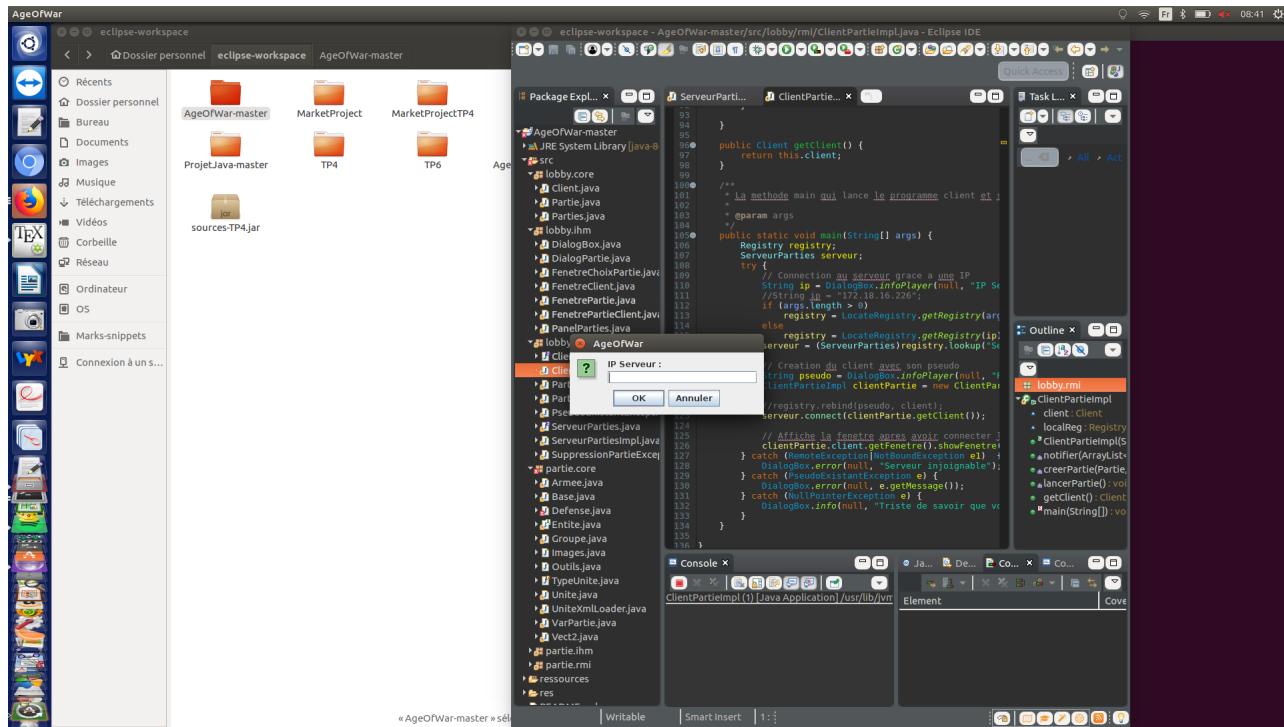


FIGURE 3.1 – Saisie de l'IP serveur

Ensuite il doit rentrer son nom d'utilisateur afin de pouvoir accéder à la partie "lobby" du jeu.

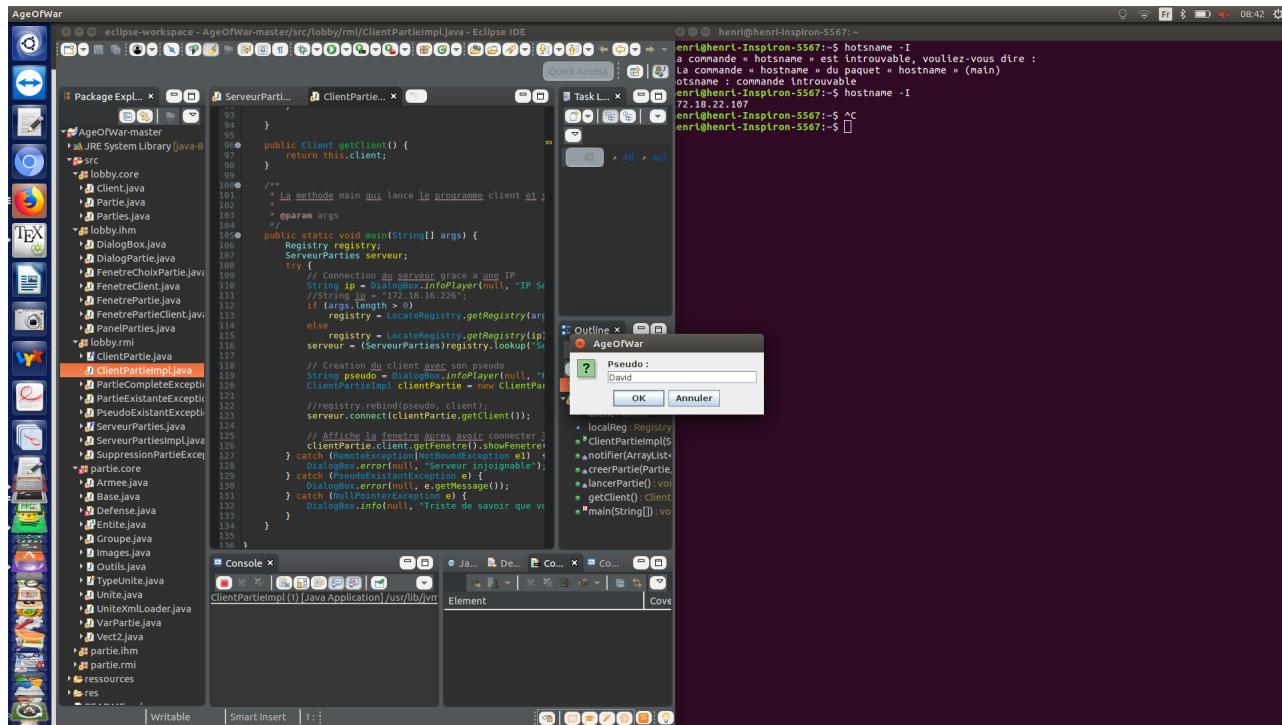


FIGURE 3.2 – Saisie du pseudo

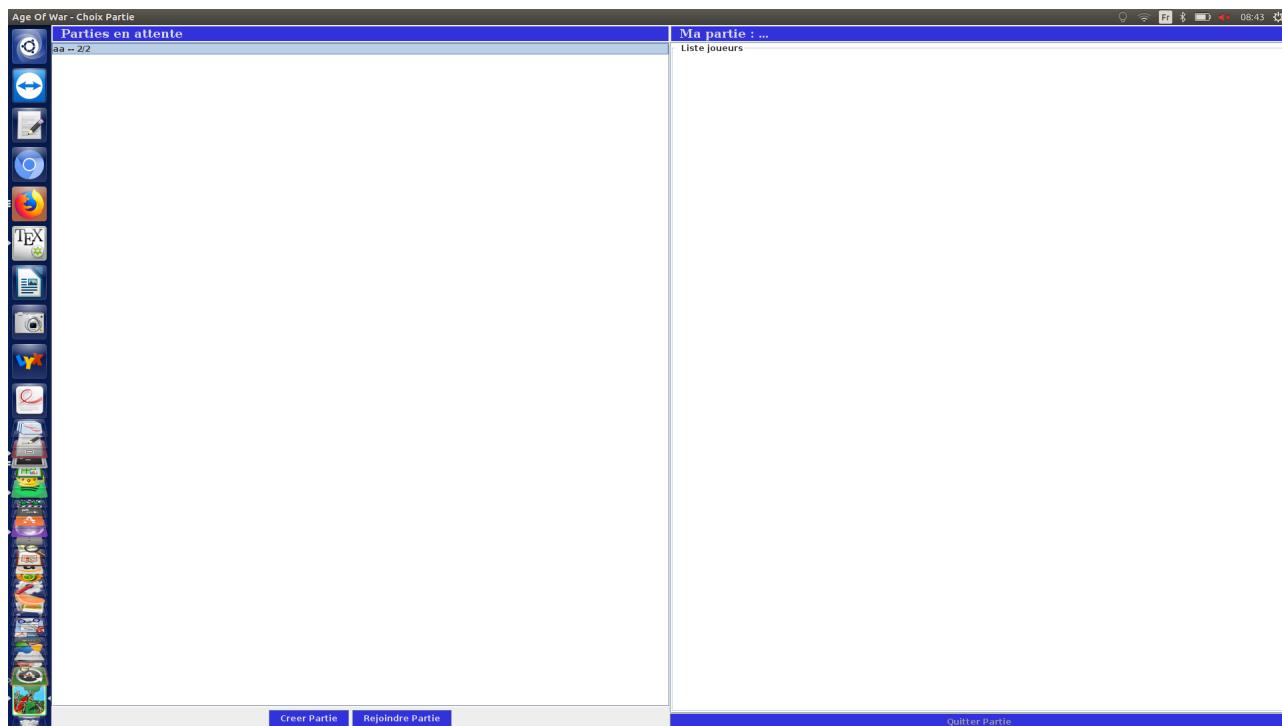


FIGURE 3.3 – Fenêtre du lobby

Une fois dans le lobby, le joueur a deux possibilités. Soit rejoindre une partie, soit créer sa propre partie. Dans cet exemple , seul la partie *aa* existe, mais le nombre maximal de joueur est déjà atteint. Donc si l'utilisateur veut jouer il à plutôt intérêt a créer sa propre partie ou bien attendre qu'une nouvelle soit créée. Ici, on va créer une nouvelle partie de deux joueurs, non privée (sans mot de passe).

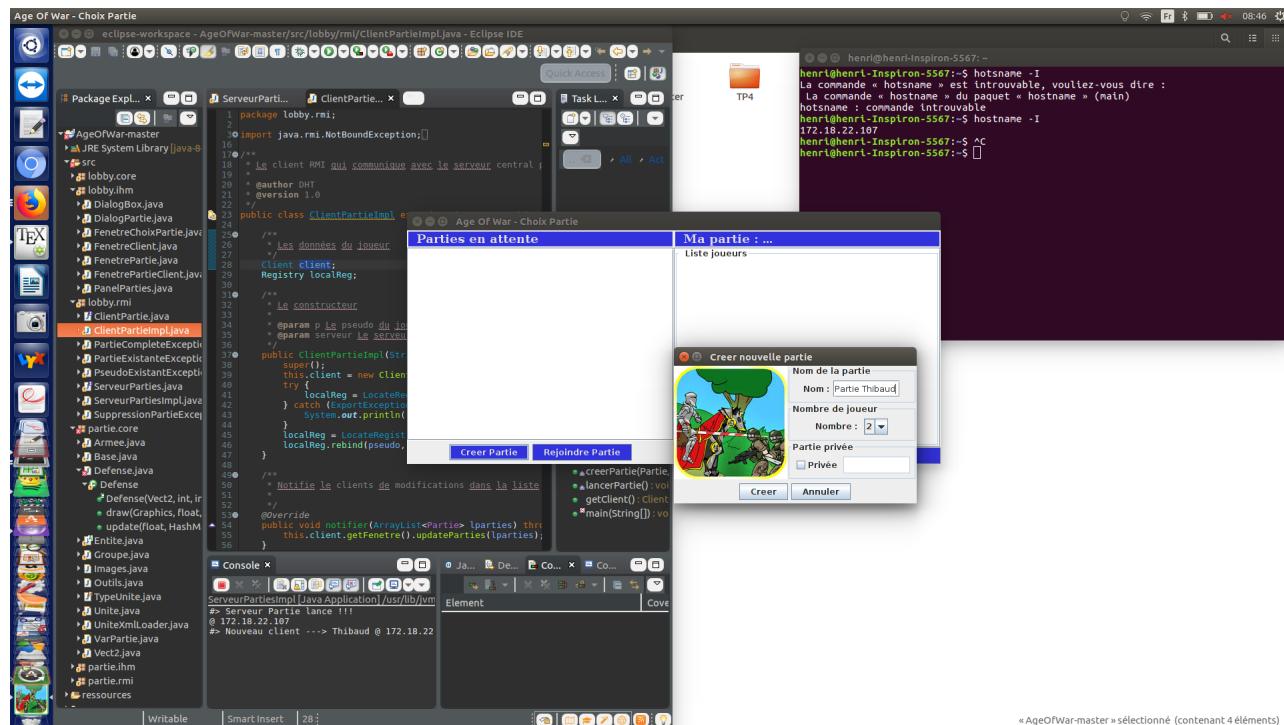


FIGURE 3.4 – Fenêtre de création de partie

Ainsi, si un nouveau client lance le jeu, il aura la possibilité de se connecter à cette partie.

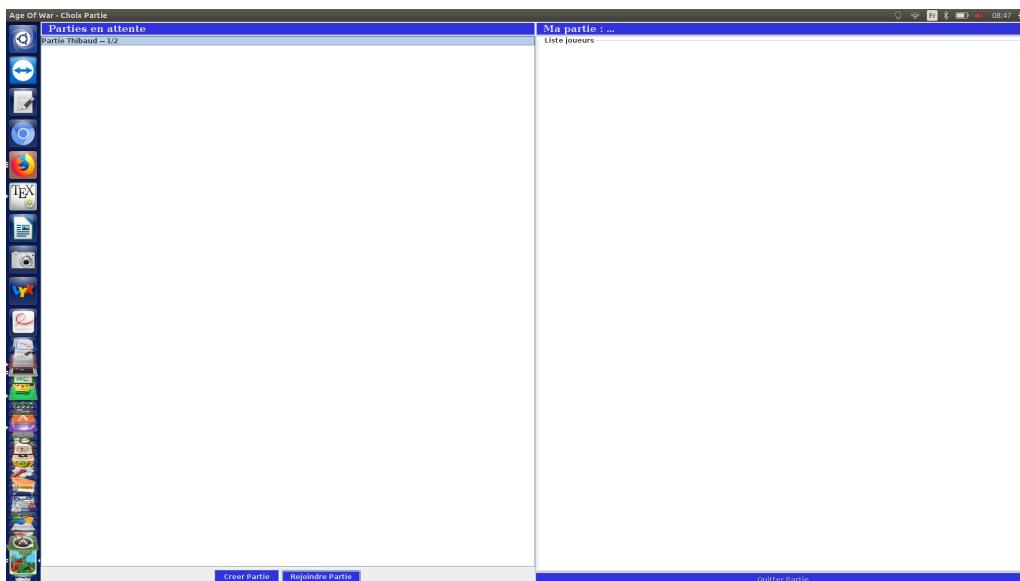


FIGURE 3.5 – Fenêtre du lobby avec une partie non complète

Une fois que le lobby est rempli, la partie est lancé.

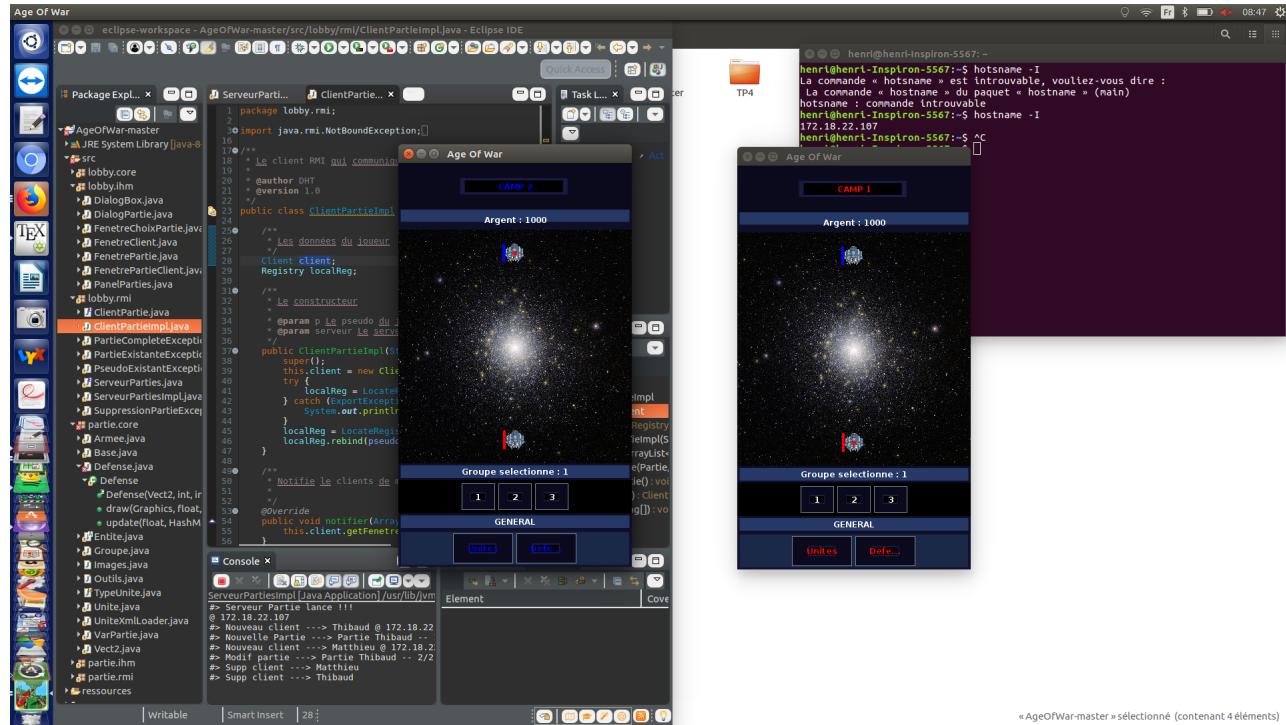


FIGURE 3.6 – Début du jeu

Le joueur peut maintenant déployer des unités mobiles, qui se déplacent vers un objectif défini par le joueur.

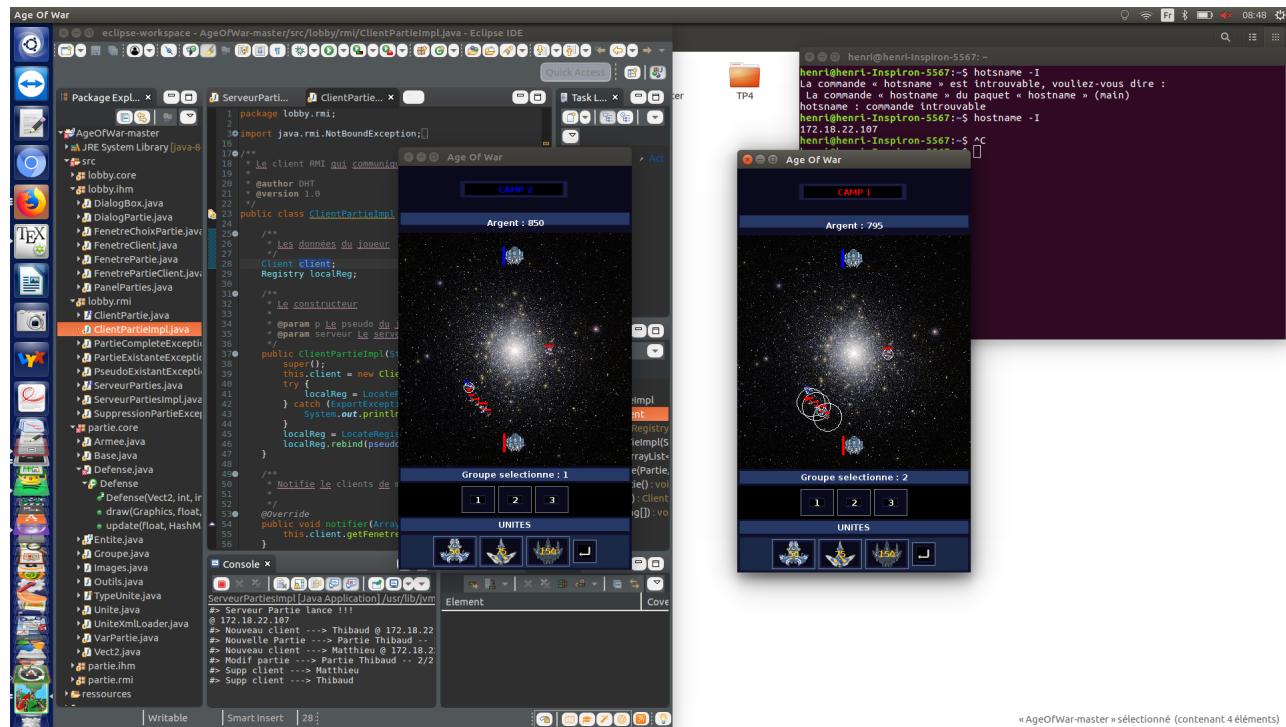


FIGURE 3.7 – Crédit et déplacement d’unités

Il peut également déployer des tourelles (unités fixes).

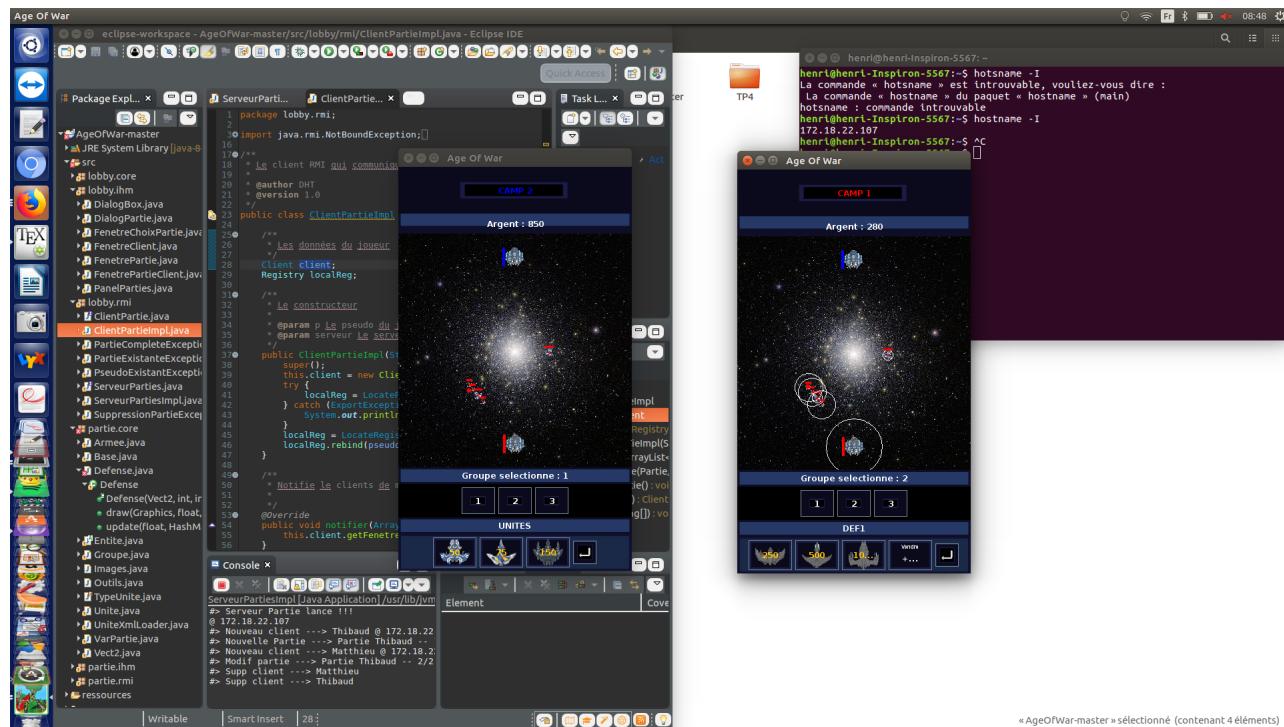


FIGURE 3.8 – Déroulement du jeu

Le joueur peut déployer jusqu'à 3 groupes d'unités mobiles.

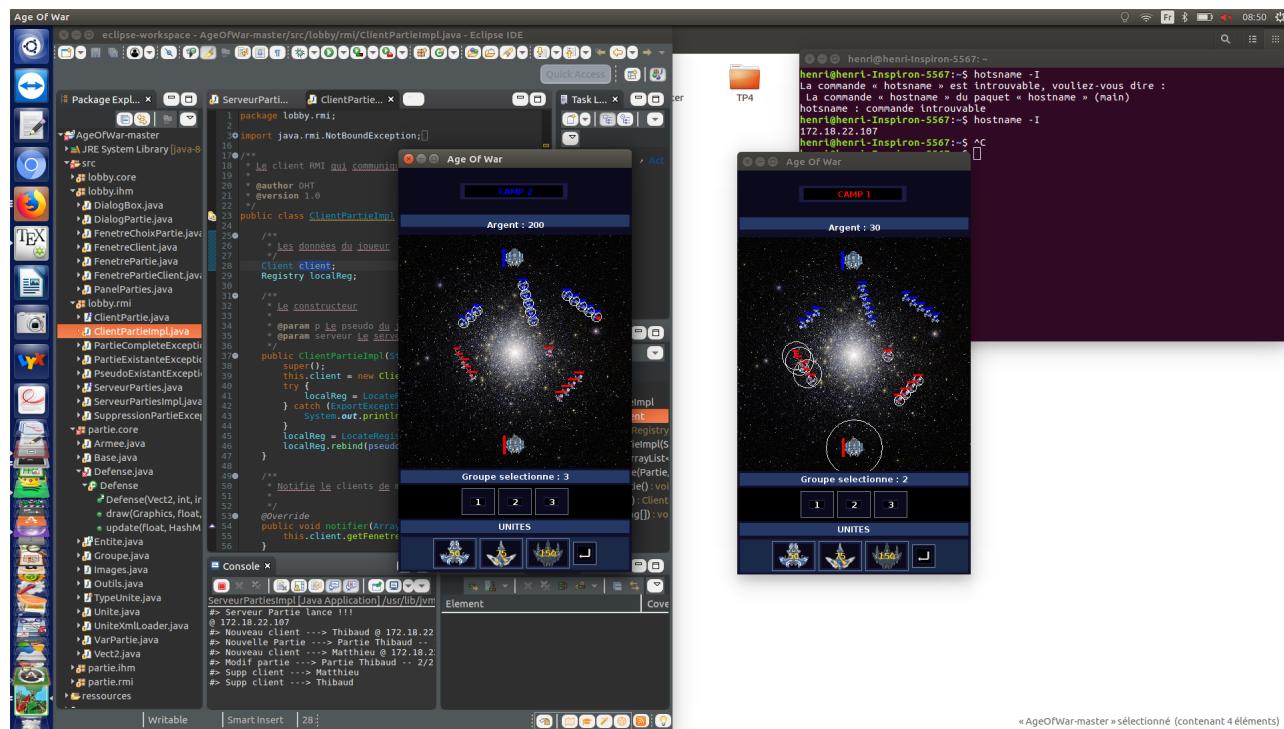


FIGURE 3.9 – 3 groupes d'unités s'affrontent

La partie se termine lorsqu'il ne reste plus qu'une base en vie.

Partie 4

Difficultés rencontrées

Pour la création du jeu, nous avons divisé le travail en deux, une partie *lobby* et une partie *jeu*, la répartition est détaillée dans la prochaine partie. Cette répartition a fait que lorsque nous avons essayé de créer la liaison entre le *lobby* et le *jeu*, nous avons rencontré de nombreux problèmes, certains liés à la liaison réseau, et d'autres plus obscurs dont nous n'avons toujours pas compris la cause mais trouvé un moyen de le régler.

Problème 1

La première difficulté a été de faire communiquer les clients et l'hôte après le lancement de la partie, car nous pensions qu'un seul registre RMI suffirait, hébergé sur la machine de l'hôte. Mais nous avons découvert qu'il fallait créer un registre RMI sur chaque machine, et faire en sorte que l'hôte ait accès aux registres des clients. Pour cela, il a fallu trouver un moyen de récupérer l'adresse IP de tous les joueurs. Nous avons décidé que dès la connexion au *lobby*, l'IP serait stocké dans la classe Client. La méthode pour récupérer l'IP a été fortement inspirée par la solution à un problème similaire trouvée sur *StackOverflow*, et implique l'usage des sockets. Une fois toutes les IP récupérées, il ne restait plus qu'à les communiquer au serveur de jeu pour qu'il stocke le pointeur sur chaque client.

Problème 2

La seconde difficulté, et la plus importante, a été un bug qui est apparu à la toute fin de notre projet, à savoir le mercredi 19/12. Au lancement de la partie *jeu*, un serveur de jeu est créé, puis celui-ci crée une instance de l'objet JoueurPartieImpl pour tous les clients (dont l'hôte), qui génère une interface Swing en local, c'est-à-dire sans aucune requête au serveur. Au démarrage de la boucle de jeu, seul le dernier client généré voyait son interface disparaître, à l'exception du Panel central où on voit les unités. Nous pensions au départ avoir affaire avec un simple problème d'itération sur la boucle qui lance les clients, mais au final, le problème s'est avéré bien plus complexe. En effet, après de nombreux tests, nous avons découvert que l'interface se lançait au début, puis disparaissait au démarrage de la boucle de jeu (la méthode *bouclePartie()* du serveur), avant de réapparaître une fois que l'hôte quittait la partie. Nous avons également vérifié qu'il ne s'agissait pas d'un problème réseau, car même sans aucune information provenant du serveur, le même bug se produisait.

La résolution de ce problème a été de créer un Thread sur le serveur pour exécuter la méthode *bouclePartie()*. Cette méthode semblait être la chose qui provoquait ce bug, bien que certains tests excluaient cette possibilité. C'est probablement un problème de timing entre le serveur et le dernier client, qui n'avait pas le temps de communiquer avec le serveur avant le démarrage de la boucle, mais nous ne sommes sûrs de rien.

Ces 2 problèmes ont été les plus importants pour nous, car il a fallu bien comprendre nos morceaux de codes respectifs pour arriver à analyser la source de chaque problème.

Partie 5

Répartition du travail

| | Lobby | | | Jeu | | | | |
|----------|-------|-----|--------|-----|-----|--------|-----|---------|
| | ihm | rmi | metier | IHM | rmi | metier | UML | Rapport |
| David | + | + | ++ | | | | + | + |
| Henri | + | + | | | | | + | ++ |
| Matthieu | | | | + | ++ | + | + | + |
| Thibaud | | | | + | + | ++ | + | + |

FIGURE 5.1 – tableau de la répartition des tâches