Rapport projet Stratego

Introduction

Stratego est un jeu de stratégie qui se joue au tour par tour à deux sur un plateau de 10x10 cases. Chaque joueur doit placer 40 pions de différentes classes. Ces pions seront placés sur les 4 premières lignes de chaque côté par leur joueur face cachée en début de partie. Au milieu du plateau se trouve deux lacs par lesquels les pions ne peuvent pas passer. Le premier joueur à avoir trouvé le drapeau adversaire gagne la partie. Un joueur peut également gagner une partie si son adversaire ne peut plus déplacer aucun pion. Il peut donc avoir un match nul.

Participants :

- Vasiliev Alan (56037)
- Lejeune Rodrigues (56167)
- Tissir Yassine (56593)
- Soulimane AHDACH (54678)
- Blerim MATA (42938)
- Nicolas Gardeur (54317)

Composition

Pour l'implémentation de notre projet, nous avons décomposé le projet en 3 parties : Serveur / Client / Library. L'architecture est de type serveur/client. Cela permet de jouer à plusieurs au jeu simultanément.
Une possibilité de jeu solo a été implémentée (contre un bot). Dans ce cas, il est possible de sauvegarder une partie et de relancer celle-ci plus tard.

Un client peut décider de créer une partie privée. Dans ce cas un pop-up s'affiche avec l'id de partie et le second joueur peut introduire cet id dans le champ correspondant aux parties privées pour se joindre à la partie.
L'utilisation est plutôt simpliste : lancer le serveur via le main, lancer un nombre de clients souhaité en fonction du nombre de joueurs.
Un pseudonyme sera demandé à la connexion pour accéder à la page de lobbies. Cette page contient la liste de parties créées.

Rapport projet Stratego

Serveur

Dans ce projet, on va y retrouver l'ensemble des classes permettant la gestion du serveur (multi-threading, gestion des messages réceptionnés et émis, gestion de la base de données, …).
Server : classe où l'on va créer le serveur qui va gérer les connexions, déconnexions avec les clients, gérer la liaison avec la base de donnée …
Server_view : classe permettant d'avoir un visuel des utilisateurs connectés (ID, adresse IP, n° port).
connection_to_client : gestion des clients, instancié à chaque nouvelle connexion.
Base de données : La base de donnée possède 4 tables au total, la table « *User* » qui permet l'ajout et la vérification des utilisateurs, la table « *Stat* » qui va afficher le nombre de victoires réalisé par un joueur, la table « *History* » qui enregistre des informations de victoire sur une fin de partie et pour finir la table « *pause* » qui enregistre une partie contre un bot. La base de données est créée au niveau de la création du serveur. Si la base de données n'est pas présente, une nouvelle sera créée

Library

Il a fallu implémenter des classes pour la gestion des messages.
Effectivement il a été plus simple d'avoir des classes communes nommées **json generator et json interpreter**.
Json generator : classe créant des json en fonction d'un type de message.
Json interpreter : classe décomposant les json reçus.

Deuxièmement, les classes du modèle ainsi que le bot ont été placés dans cette partie du projet.

Bot

Au départ, nous avons commencé par créer un bot qui pouvait déplacer aléatoirement un pion dans une direction possible, en évitant les obstacles tels que les lacs et les bords du plateau. Nous avons également implémenté une règle qui fait en sorte que le bot engage une confrontation lorsqu'un pion ennemi se trouve autour d'un de ses pions. Pour varier les parties, nous avons créé 3 fichiers de placement différents des pions, qui se trouvent dans le dossier "placement" du projet "Stratego_Serveur", et qui permettent de jouer contre un bot qui n'a pas toujours les mêmes pions à la même place.

Ensuite, nous avons décidé de nous concentrer sur le pion spécifique qu'est le démineur. Nous avons commencé par implémenter la logique qui permet au bot de trouver et de désamorcer les bombes lorsqu'un de ses pions est tué par une bombe adverse. Cependant, au fil de l'avancée du développement, nous avons réalisé qu'il y avait beaucoup plus à implémenter pour gérer les démineurs que ce que nous l'imaginions au départ, et nous n'avions pas su terminer à temps. Nous avons donc préféré continuer avec un bot qui fonctionne correctement jusqu'à la fin du jeu, plutôt qu'un bot qui ne fonctionne pas correctement en plein milieu d'une partie.

Client

Pour finir, nous avons le client qui va se connecter au serveur et envoyer des messages au serveur. Par exemple lors de sa connexion un CONNECTION_TO_SERVER est envoyé et le client reçoit un REFRESH_LOBBY.

Rapport projet Stratego

Client : classe établissant la connexion avec notre serveur
GUI : gestion de la vue (voir images ci-dessous) + d'autres classes pour les différentes fenêtres (page d'accueil, page des lobbys)

Images

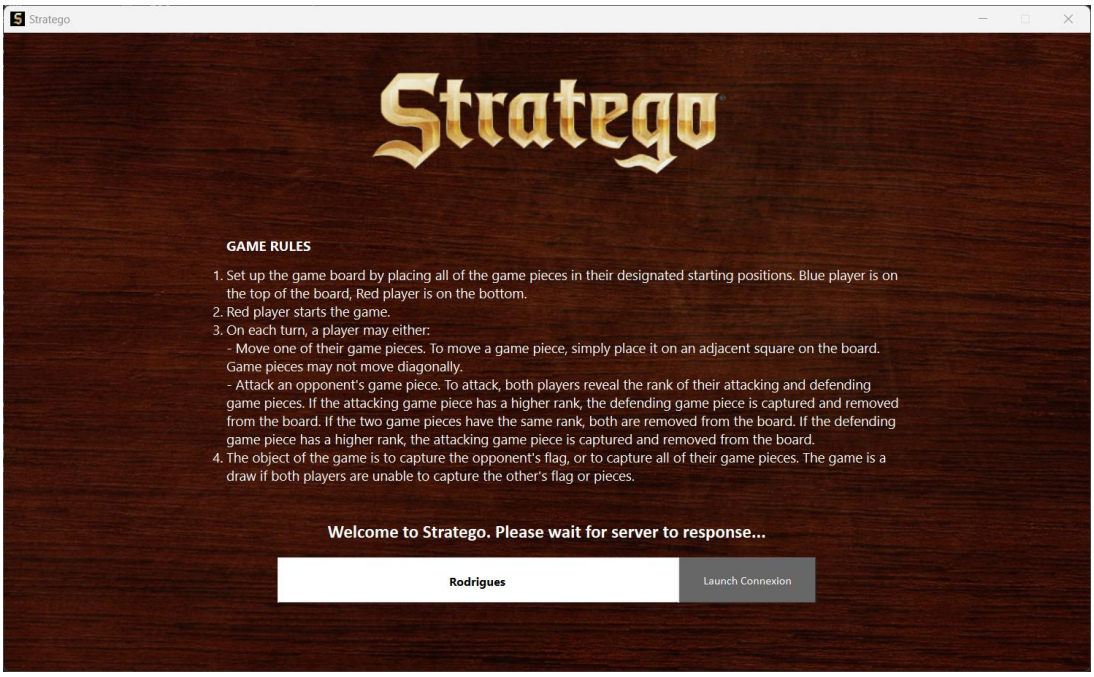

Figure 1 Connexion

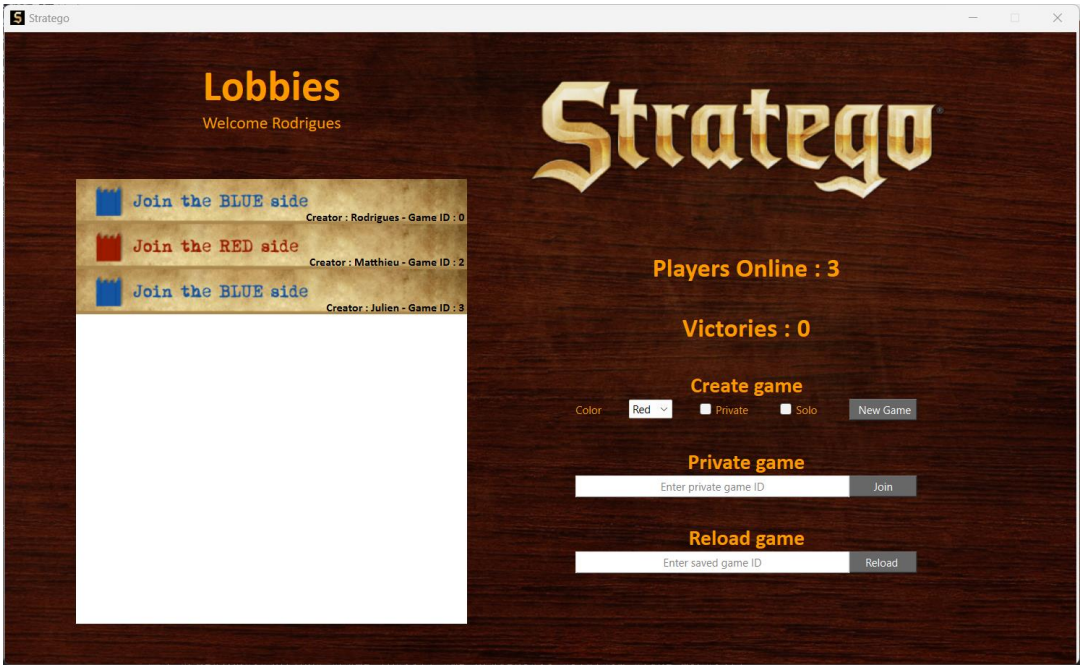

Figure 2 Menu

Rapport projet Stratego

Figure 3 Plateau de jeu

Fonctionnement

- Le joueur se connecte en entrant son pseudonyme et en cliquant sur « launch connection » (*Figure 1)*.
- La connexion est établie avec le serveur et la page des lobbies est affichée (*Figure 2)*. Sur cette page, le nombre de joueurs actuellement connectés et le nombre de parties qui ont été remportées par le joueur connecté sont affichés.
- Plusieurs choix s'offrent sur cette nouvelle page :
    - Se joindre à une partie créée (sur la gauche). La couleur qui sera attribué, ainsi que le créateur de la partie et l'ID de la partie sont affichés.
    - Créer une nouvelle partie. Il est possible de choisir la couleur que l'on veut arborer pendant cette partie. Il est aussi possible de choisir si cette partie sera privée, c'est-à-dire qui ne sera pas visible dans les parties en attente (à gauche). Une autre possibilité est de faire une partie privée contre un bot.
    - Se joindre à une partie privée en entrant l'identifiant d'une partie privée préalablement créée (cet identifiant doit être partagé par le créateur de la partie par un canal tierce).
    - Reprendre une partie en cours contre un bot, en entrant l'identifiant de cette partie.
- La partie lancée, l'écran de jeux est affiché (*Figure 3*). Dans un premier temps, chaque joueur place ses pièces. Une fois toutes les pièces placées, le joueur rouge démarre son tour de jeu et effectue son mouvement. Pour finaliser son mouvement, il clique sur le bouton de droite signalant « to next player ».

  Quand un des deux joueurs trouve le drapeau de l'adversaire ou que plus aucun de mouvement n'est possible pour l'un des joueurs, la partie se termine, le vainqueur est affiché et les joueurs sont redirigés vers la page des lobbys.

Rapport projet Stratego

S'il s'agit d'une partie contre un bot, un bouton « pause » supplémentaire est affiché et permet de sauvegarder et quitter la partie en cours. L'ID de la partie sera fournie au joueur pour récupérer sa partie sauvegardée.

Répartition

Client : Alan / Rodrigues / Nicolas
Serveur : Alan / Rodrigues / Nicolas
Bot : Yassine / Soulimane
Base de données : Blerim / Alan

Git

Voici le lien du git contenant le projet

https://git.esi-bru.be/54317-56037-56167-56593-54678-42938/stratego

Table des matières