

Morpion

Comment jouer au jeu

- Ouvrir deux fois le exe MultijoueurGame présent dans le x64 a la racine de la solution.
- La première fenêtre, il faut écrire dans la console « server ».
- La deuxième fenêtre, il faut écrire dans la console « client ».
- Clic gauche sur une case pour placer son jeton.
- Espace pour restart la partie.

La fenêtre serveur est la première a joué.

Architecture du projet Solo (Sans Socket ni Thread)

Répartir la logique dans des classes bien distinctes et faire des fonctions assez lisibles et courte pour facilement intégrer le réseau et le multithread

Les classes importantes :

- Player controller
- GridManager
- GameManager (on a utilisé un pointer de fonction pour faire la passerelle avec le grid Manager)
- Tile

On a mis en place la possibilité de redémarrer la partie pour pouvoir tester et debug rapidement.

Aventure en Réseau

Approche UDP

On a commencé par implémentait les sockets en UDP mais on a eu des problèmes pour récupérer les messages envoyés. Et en plus, on s'est dit que TCP en plus ça collait plus vu qu'on est tour par tour et que c'était important pour nous de vérifier que les informations soient bien transmises et reçus. De plus, le morpion ne demande pas beaucoup de requête en temps réel.

Approche TCP

TCP Les class networks

Pour ces classes on a vraiment essayé de de mettre plein de condition pour check les erreurs et de découpé toutes les étapes de mises en place des sockets en fonctions.

Les classes importantes :

- Server network
- Client network

Problème N°1 Rencontré : Link entre les projets.

Architecture network un script client manager et un server manager et trois projets


L'approche initiale du projet était d'avoir un solo game qui contient toute la logique gameplay et avoir deux autres projets un pour le client et un pour le server.

On a donc envisagé d'avoir un manager pour le client et un autre pour le server.

La principale problématique qu'on a eu c'est qu'il fallait qu'on accède depuis les projets clients et serveurs au projet solo.

On a donc réussi à pouvoir inclure le game manager en linkant les additionnal directories au projet solo game et en utilisant les projets dependencies pour build en premier le solo game.

Malheureusement à partir du moment où on a voulu utiliser les méthodes du game manager.

 On a eu des erreurs linkers qu'on n'a pas réussi à résoudre. L'erreur était qu'il ne trouvait pas la classes GameManager.

On a trouvé deux solutions qui était soit de créer une static library à partir du solo game soit de créer un projet dll à partir du solo game.

Mais au vu de la méconnaissance du sujet et du temps qu'il nous rester on a préféré choisir une troisième approche.

Solution N°1 : Tout passer dans le même projet.

L'approche en question était de tout merge en un seul projet (multiplayerGame).

Donc ce qu'on a fait c'est qu'on a un seul point exécutable qui nous demande de choisir entre server et client. Cela va ensuite créer les class nécessaire en fonction. (Créer le ServerNetwork ou le ClientNetwork)

On est obligé de commencer à set up le server puis le client.

NetworkManager

On a décidé de créer un seul NetworkManager qui va venir encapsuler le game manager pour intégrer le réseau au gameplay.

On a un seul NetworkManager pour le client et pour le server.

Ce qui va changer c'est lors du constructeur, on va passer soit le socket du serveur soit celle du client et envoyer une bool qui check si on est le server ou pas. Ce qui permet de déterminer quel joueur on est et de savoir si on peut restart ou encore de renommer la fenêtre.

Les étapes importantes sont :

- Init
- Handle input
- Display

- Update
- BeginTurn

Data Envoyées

On envoie les données depuis deux endroits :

- GridManager => on envoie une structure ChangeTileStatus qui contient les informations sur quelle tile a été modifiée et par quel joueur
- NetworkManager -> HandleInput -> on envoie, une bool qui permet de déclencher le restart chez l'autre joueur.

Pour vérifier que l'information est bien envoyée, on a mesuré le nombre d'octet retournés par la fonction send. Pour le cas de la structure ChangeTileStatus, on a mesuré 8 octets car la structure est composée de deux ints.

Problème N°2 : le receive bloque la mise à jour de la windows (Implémentation Multithread)

On a dû utiliser du multithreading pour faire tourner et empêcher le blocage de la méthode receive.

En tout on a trois threads :

- Le thread principal qui gère mise à jour de la fenetre et les inputs
- Un thread pour le receive de l'update de la case
- Un thread pour le receive du restart

Pour créer les threads on a créé un pointer de fonction (BeginTurn) qui se déclenche à chaque nouveau tour. On crée et ferme les threads uniquement lorsqu'on a en besoin. Dans le cas du thread de placement de tile, on crée le thread uniquement lorsque c'est le tour de l'adversaire et lorsque c'est notre tour et qu'il a reçu son information, on le détruit.

On a créé une fonction pour chaque thread qui contient la logique et qui est appelé grâce au pointer du NetworkManager passée en paramètre de la fonction thread car elle est statique.

Problème n°3 : Les threads

Tout d'abord, on a eu une erreur de access violation qui close la console d'un des joueurs. C'était très compliqué à debug. Le crash se déclenchait lorsqu'on relançait la partie ou lorsqu'on posait des jetons à certains moments. Que cela soit le server ou le client. On pense que le problème est lié aux threads.

On a essayé d'implémenter des try-catch et des if dans les fonctions thread.

On a essayé d'ajouter un mutex pour empêcher l'écriture simultanée sur la grid par le thread restart et le thread changeTile.

Mais on avait encore des problèmes. On a rendu le restart déclenchable uniquement par le server car on s'est rendu compte qu'on ne pouvait que ReleaseMutex depuis le thread qui a verrouillé le mutex.

La solution finale qui a fixé le bug a été de mettre dans les threads uniquement des sets de variables et de bool pour qu'elles déclenchent, lorsqu'elles sont vrai, dans l'update du thread principal, les fonctionnalités demandées.

On a eu un autre problème lié à la sfml qui faisait des comportements déviants lorsqu'on met à jour l'ui ou les sprites dans les autres threads. La solution finale a fixé ce problème.