ULCO - L3 Informatique

Rapport du projet "Peaceful Shadow Online"

Étudiants : CHRETIEN Octave, EHINGER Sylvain, MALLEVAEY Pierre, SPECQ Corentin

Encadrant : DEHOS Julien

Date de début : 29 mai 2017

Date de fin : 16 juin 2017

Objet du projet : Jeu de bataille navale en réseau

Table des matières

Présentation du projet

Analyse de la demande

Contexte du projet

Dans le cadre du projet de fin de la troisième année de licence informatique, nous avons développé un jeu en réseau sur une période de trois semaines à temps plein. Nous avons porté notre choix sur la réalisation d'un jeu de bataille navale, en y ajoutant un système de capacités spéciales et de pays.

Les règles de la bataille navale classique sont simples : les deux joueurs disposent chacun de deux grilles (une grille personnelle et une autre représentant la grille de leur adversaire). Le jeu se déroule de la façon suivante :
- Les deux joueurs placent leurs bateaux (de tailles 2,3,4 et 5 cases) sur leurs grilles respectives.
- Le premier joueur est désigné, et il choisit une case sur laquelle tirer à l'aveugle sur la grille représentant celle de son adversaire en indiquant ses coordonnées.
- Son adversaire lui indique s'il a touché un bateau ou bien tiré dans l'eau, et le joueur actif remplit la grille adverse pour répertorier son tir.
- Son tour se termine, et c'est à son adversaire de jouer.

Connaissant la taille et le nombre des bateaux, les joueurs peuvent adapter leurs coups en fonction de la grille adverse qui se remplit à mesure de la partie. Le premier joueur a avoir coulé la flotte ennemie gagne la partie.

Dans le cadre de notre projet, les deux joueurs disposent des deux grilles simulées sur une fenêtre de jeu, et les échanges se font à travers un serveur qui ordonne les tours ; cela permet de jouer à distance.

Nous avons décidé d'ajouter un aspect tactique supplémentaire au jeu en apportant la notion de pays différents à incarner, possédant chacun un pouvoir différent. Ces pouvoirs seront utilisables à intervalles réguliers et permettront d'influencer le jeu en modifiant la façon de tirer, ou en apportant des mécaniques supplémentaires. Ces pouvoirs sont très différents, allant de tirs particuliers à révélation de la grille adverse et à la réparation de bateaux, et sont indiqués au joueur lorsqu'il choisit sa nation.

Besoins fonctionnels du projet

Le besoin principal de ce projet est l'intégration des règles de la bataille navale dans un programme utilisable facilement. Ainsi, les mécaniques doivent être compréhensibles et l'interface intuitive, permettant un confort de jeu optimal. Par dessus les règles classiques du jeu de bataille navale, nous intégrons nos règles personnalisées (choix du pays et donc utilisation du pouvoir).

Le jeu nécessite la mise en place d'une architecture client/serveur afin de gérer les parties en ligne. Ce serveur est articulé autour d'un serveur principal gérant les nouvelles connexions et envoyant les joueurs deux par deux vers des nouvelles instances de serveurs gérant chacun une partie de bataille navale.

Besoins non fonctionnels du projet

Les besoins non fonctionnels de ce projet sont les technologies utilisées pour le mener à bien : le langage C++ déployé à travers la bibliothèque SFML qui permet de gérer la couche graphique et la couche réseau de notre application.

Priorités du projet

La priorité principale du projet est le système de jeu, fonctionnel et intuitif, permettant à deux joueurs d'utiliser notre application. Il s'agit donc de mettre en place la partie graphique reliée au système de jeu, et la partie réseau simple gérant les échanges entre les joueurs. En termes de priorité secondaire, la gestion des sous-serveurs permettant de jouer plusieurs parties indépendantes et simultanées.

Spécifications

1. **Spécifications du système de jeu**
   - Mise en place de grilles simulant des grilles de bataille navale contenant les bateaux.
   - Positionnement dynamique des bateaux (rotation/changement de place)
   - Gestion des tirs en coordonnées x/y de la grille.
   - Déploiement des pays et affectation du pouvoir approprié.
   - Stockage de toutes les données utiles dans une classe représentant le joueur.
2. **Spécifications de l'interface graphique**
   - Une interface interactive répondant au clic de la souris et à certaines touches du clavier.
   - Une page d'accueil permettant de saisir son pseudo et de lancer la recherche d'une partie.
   - Une page d'attente (recherche de partie) pour faire patienter le joueur.
   - Une page de choix du pays à incarner explicitant les pouvoirs de chacun.
   - Une page de positionnement des bateaux permettant de remplir sa grille.
   - Une page d'attente en attendant que l'autre joueur soit également prêt à jouer.
   - Une page de jeu affichant les grilles et servant de plateau de jeu et affichant les pseudonymes et les coups joués.
   - Un bouton permettant d'activer son pouvoir et indiquant le temps avant réutilisation.
   - Une page "victoire" donnant au gagnant l'option de rejoindre le serveur principal ou de quitter.
   - Une page "défaite" permettant au perdant de faire de même.
3. **Spécifications du réseau**
   - Architecture client/serveur, sur deux machines séparées reliées par des sockets dédiées.
   - Déploiement d'un serveur principal gérant les connexions de nouveaux utilisateurs et capables de rediriger vers les parties.
   - Déploiement de sous-serveurs générés en multithreading gérant chacun une partie.
   - Transmission des grilles et ordonnancement des tours gérés par le serveur.

Enfin, comme convenu dans le cahier des charges, le logiciel fonctionne sur Linux avec les bibliothèques standards du C++ et la bibliothèque SFML.

Réalisation

Nous avons identifié trois grandes parties à la réalisation de notre projet : la partie du système de jeu, la couche graphique qui lui est superposée, et la partie réseau qui englobe le tout.

Interface graphique

Les fenêtres successives sont appelées par le joueur lui même, en local, et chaque fenêtre retourne le type de données nécessaire à la mise en place du jeu ; par exemple la fenêtre d'accueil renvoie le pseudo, la fenêtre "Positionnez vos bateaux" renvoie la grille du joueur.

Chaque bouton est géré de façon à n'être cliquable qu'au moment approprié, et un changement de couleur indique lorsqu'il devient cliquable.

L'interface a été bloquée en 1000x800 afin de pouvoir être lancée sur la plupart des machines.

L'interface graphique est principalement constituée d'une image recouvrant la fenêtre sur laquelle sont définies des zones cliquables telles que les boutons. Des rectangles de différentes tailles changent de couleur et de transparence afin de gérer la colorimétrie des boutons.

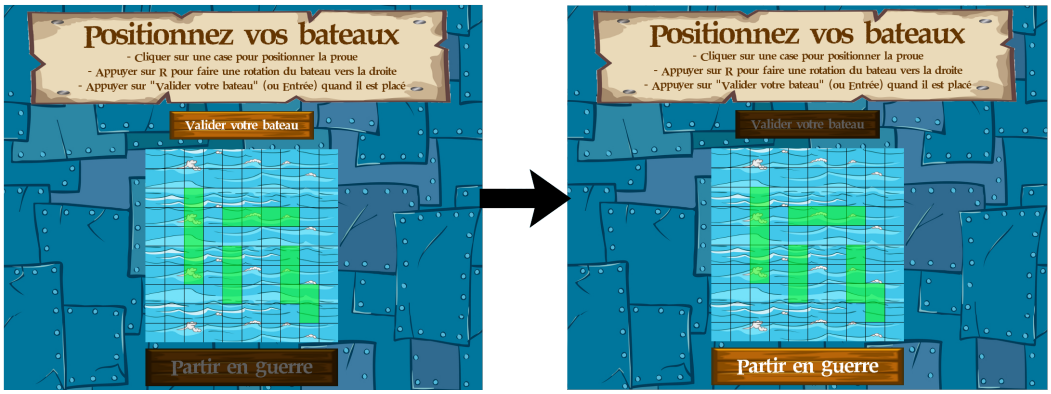

FIGURE 1 – Tant que l'on a pas validé tout ses bateaux, le bouton 'Partir en guerre' est inactif

De même, cette façon de faire permet d'empêcher le joueur dont ce n'est pas le tour de jouer, afin d'éviter tout problème de déroulement du jeu.

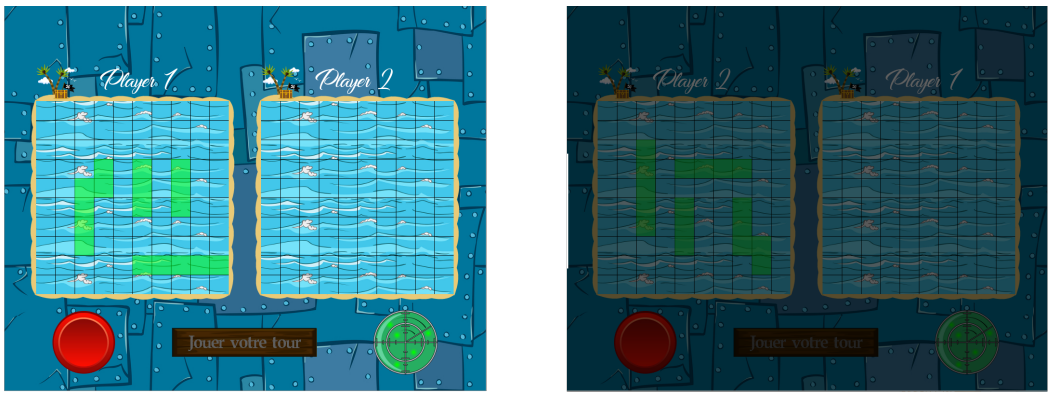

FIGURE 2 – La fenêtre du joueur actif est éclairée, celle de l'autre joueur est assombrie et inactive

A chaque positionnement, les cases concernées passent du type "mer" au type "boat". On peut alors recliquer pour supprimer le bateau actuel et le replacer ailleurs, et ce tant qu'on a pas validé notre

placement avec la touche Entrée ou bien un clic sur le bouton approprié. Si un bateau est placé ou positionné à un emplacement innaproprié, il affiche une case rouge à la place.

Couche serveur

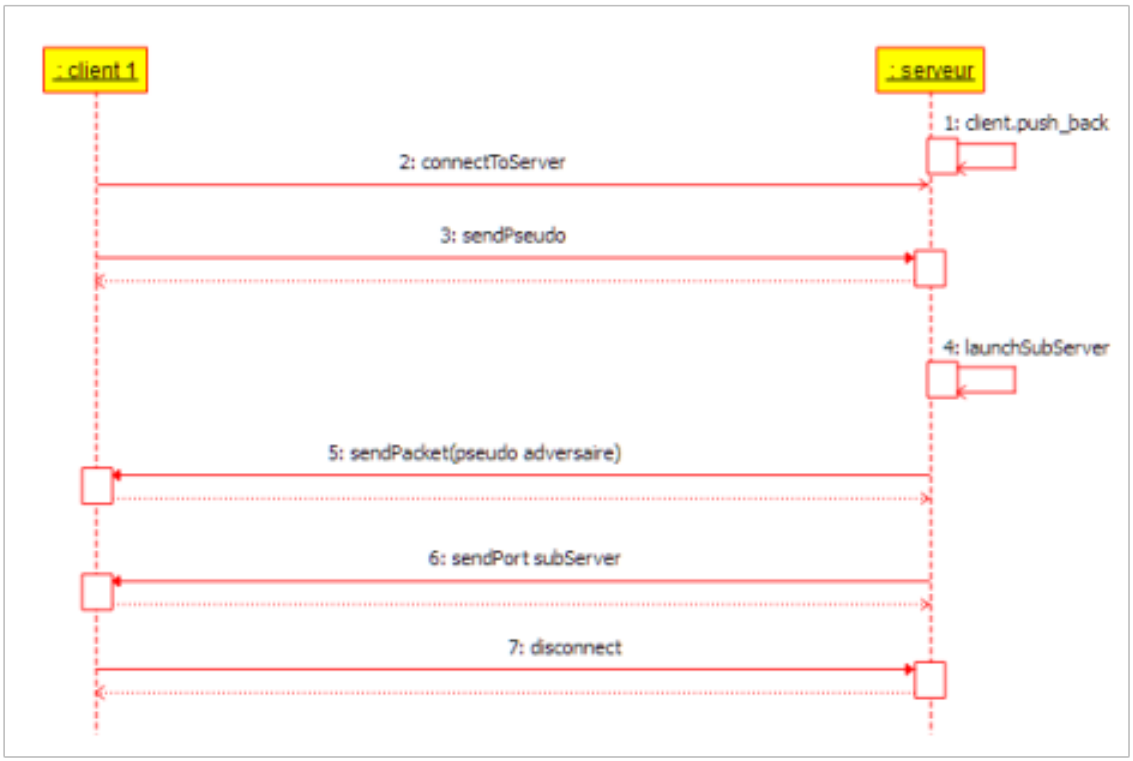

FIGURE 3 – Diagramme de séquence de la connexion d'un client au serveur principal, en recherche d'une partie

Le serveur de jeu est le premier auquel les joueurs se connectent. Il lie deux joueurs et les guide vers un sous-serveur disponible sur lequel ils pourront lancer une partie ensemble. Il gère la disponibilité des sous-serveurs et récupère les joueurs qui le souhaitent à la fin de leur partie.

Le sous-serveur gère principalement deux choses : l'ordonnancement des tours de jeu et l'attribution de la possibilité de jouer à chaque joueur, mais également la transmission des grilles entre ces derniers.

Pour la mise en place du serveur, nous avons considéré plusieurs choix, par exemple celui de l'architecture ou du type de socket.

— Le protocole de transmission : nous avons opté pour le protocole TCP, en raison de sa fiabilité. Un client n'ayant besoin de communiquer qu'avec un seul port à la fois, l'UDP n'est pas nécessaire et nous avons préféré nous assurer de la bonne transmission des paquets.

— Le multithreading mis en place permet de gérer un nombres de sous-serveurs simultanés, permettant au serveur principal de continuer à écouter de nouvelles connexions pendant que des joueurs sont en partie. De cette façon, plusieurs parties indépendantes peuvent être jouées en simultané.

— L'architecture client/serveur. Nous avons hésité entre une architecture 'on cloud', dans laquelle l'intégralité des calculs serait gérée par le serveur et les clients se contenteraient d'affichage ; et une architecture distribuée dans laquelle le serveur gérerait l'ordre des tours et la transmissions

et les clients gèreraient eux même les calculs. Comme l'objectif était de mettre le serveur sur un RaspBerry Pi, nous avons opté pour une architecture distribuée afin d'alléger la charge du serveur.

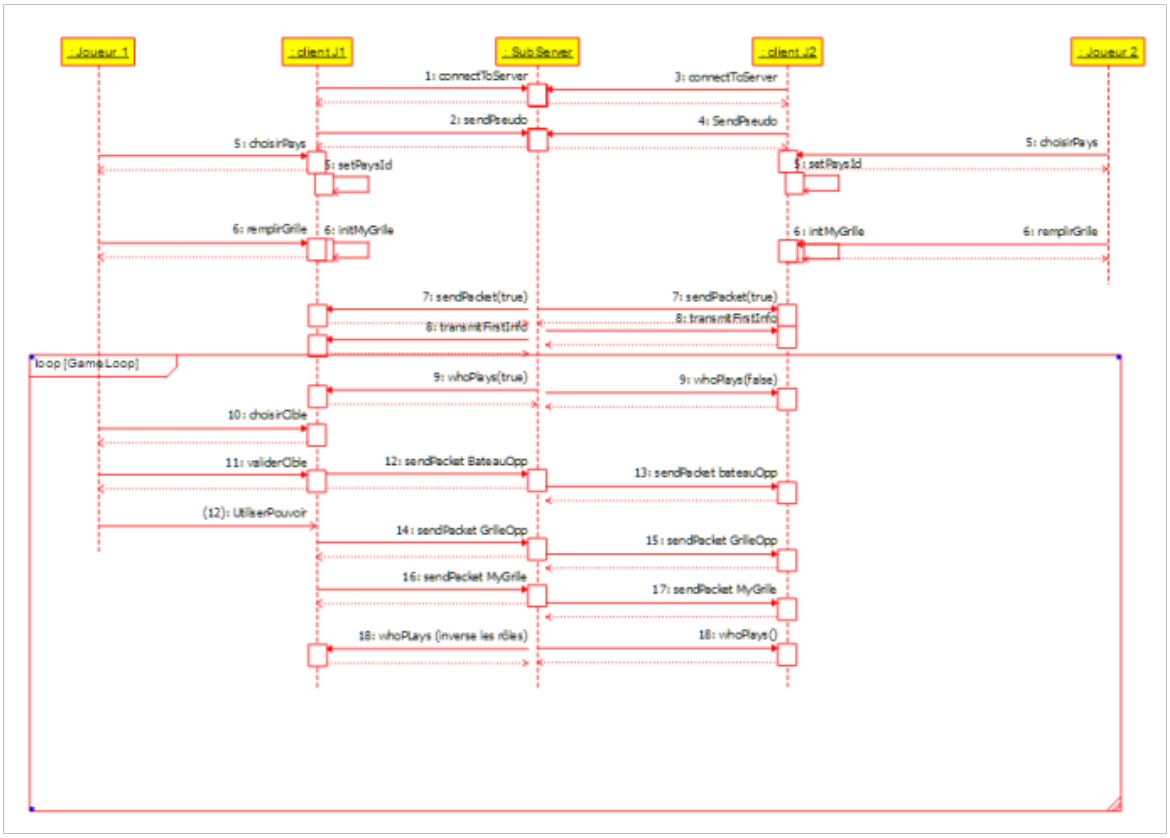

FIGURE 4 – Diagramme de séquence de la connexion de deux clients au serveur de jeu, et échanges en cours de partie

Système de jeu

Les fonctions du système de jeu sont généralement intégrées directement dans les différentes fenêtres, déclenchées par les pressions sur les boutons appropriés. Ces différentes fenêtres sont appelées par le client lui même.

L'architecture client/serveur distribuée faisant gérer les calculs par le client, nous avons développé une classe Player qui stocke toutes les informations utiles au déroulement de la partie en local (les grilles, les pseudonymes...) La classe Player envoie les informations au serveur qui se contente de les envoyer à son adversaire. Il contient également les entiers indiquant le nombre de cases de bateaux restantes sur chaque grille, qui sont utilisés pour les conditions de victoire/défaite.

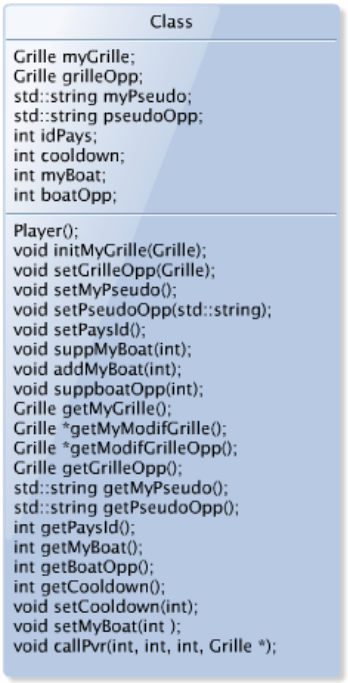

FIGURE 5 – Structure de la classe Player

Comme les sf : :Packet ne peuvent stocker que des types de donnée élémentaires, on a fait en sorte de simplifier au maximum les échanges. Ainsi, nous avons réduit les bateaux à un entier contenu dans l'attribut type d'une Case, permettant de passer les cases contenant un bateau de façon claire et simple.

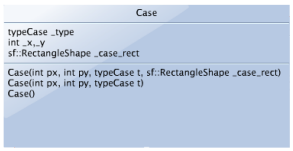

FIGURE 6 – Contenu de la classe Case - Coordonnées, contenu de la case

Les tirs sont effectués en local, et l'actualisation des grilles se fait à travers les différents getters et setters. Les pouvoirs sont gérés par des fonctions externes à la classe Player, afin de faciliter leur accession suite à des problèmes de développement.

Bilan

Déroulement du projet

Différences par rapport aux prévisions :

— Réadaptation du planning constante.

— Fluctuation de la backlog très importante.

— Pour la V2, l'organisation s'est faite de façon inverse par rapport aux prévisions.

— Au niveau de la maquette, des ajouts de fenêtres (fentres d'attente) non prévues au départ.

— La plannification et la priorité des différentes fonctionnalités à développer ont évolués au cours du temps, à cause de certains bugs important à corriger.

— Suite à de multiples problèmes lors de l'intégration des mécaniques de gameplay, la structure du code a évoluée afin que le rendu corresponde au cahier des charges. Ainsi, certaines classes ont été remplacées par une gestion différente des événements.

Problèmes rencontrés :

— Serveur : au départ, problème de typage des paquets, à impliquer une redéfinition des opérateurs d'envoi et de reception des paquets

— Serveur : Implémentation du Multithreading laborieux.

— Intégration du système de jeu difficile du fait d'un manque de communication au debut du projet. Ce qui a necessité une réécriture complète du code à plusieurs reprises.

— Gestion de tous les cas de placement des bateaux difficile que ce soit au niveau de système de jeu ou de l'affichage dans la grille.

Réalisation des objectifs

Objectifs réseau

fonctionnalité	réalisation
connexion au serveur	complète
déconnexion au serveur	complète
transmission/réception des paquets	complète
multithreading	complète
lancement d'une partie	complète
stockage d'infos sur serveur	complète
gestion des tours	complète
gestion de la fin de partie	complète

Objectifs graphiques

fonctionnalité	réalisation
interactions clavier/souris (boutons)	complète
différentes fenêtres	partielle
affichage des bateaux	partielle
affichage des tirs	complète
choix du pays	complète
récupérations d'infos (pseudo, grille...)	complète
dynamisation des fenêtres	partielle
musique	partielle

Objectifs du système de jeu

fonctionnalité	réalisation
intégration des pays	partielle
intégration des pouvoirs	complète
intégration des joueurs	complète
placement des bateaux	complète
stockage des informations	complète
effectuer un tir valide	complète

Pour la fonctionnalité d'affichage des bateaux, celle ci n'est que partielle car ceux ci sont représentés par des cases colorées et non des sprites unitaires de bateaux.

Conclusion pour les projets futurs

Les améliorations possibles à notre projet

— L'ajout de sprites de bateaux. Notre gestion actuelle de la grille rend la chose difficile, mais en modifiant la structure de nos données cela rendrait leur mise en place gérable.

— L'équilibrage des pouvoirs et des pays n'a pas été poussé à son maximum, puisque cela nécéssiterait une phase de parties test. L'ajout de nouveaux pays est également possible.

— Développer notre bande son pour qu'elle couvre des bruitages pendant la partie.

Ce qui a bien marché :

— La mise en place de plusieurs outils de communication et de partage de travail (github, discord, trello) ont permis d'être efficaces.

— Face aux différentes problématiques rencontrées, nous avons su nous adapter rapidement pour ne pas prendre trop de retard.

Les erreurs à ne plus commettre :

— Dissocier dans les premiers jours du projet les différentes parties de l'application, notamment le système de jeu et son affichage graphique, dont la réunion nous a par la suite causé des problèmes.