Le développement de ce jeu de bataille navale a nécessité l'utilisation de divers concepts de programmation en C, tels que la programmation par socket pour la mise en réseau, les types de données struct pour le stockage des données pertinentes et la mémoire partagée pour la communication interprocessus. Une approche modulaire a été adoptée pour assurer la séparation des responsabilités et la maintenabilité. Dans les paragraphes suivants, j'aborderai les spécificités de ces choix.

Tout d'abord, la programmation réseau était une partie essentielle du projet. L'utilisation de sockets TCP/IP a été choisie en raison de leur fiabilité, ce qui est essentiel pour un jeu comme Battleship où chaque mouvement compte. La fonction create\_socket du serveur crée une socket, la lie à un port spécifique, puis écoute les connexions entrantes. Cette fonction accepte ensuite deux connexions, correspondant à deux joueurs, et stocke leurs descripteurs de fichiers de socket en vue d'une communication ultérieure. Des opérations de socket similaires sont utilisées du côté client, où il se connecte à l'adresse du serveur.

Le projet est divisé en plusieurs fichiers .h (header) afin de rendre le code modulaire et organisé. Cette séparation des modules est bénéfique pour la gestion du projet et l'évolutivité future. Par exemple, protocol.h définit les constantes et les commandes utilisées tout au long du projet, grids.h et message.h définissent les structures des grilles de jeu et des messages, respectivement. array.h contient une fonction permettant de copier des tableaux 2D, qui est utilisée à plusieurs reprises dans le code. Cette séparation signifie que si des modifications doivent être apportées à l'avenir, telles que l'ajout de nouvelles constantes, la modification de la structure des messages ou la modification de la fonction de copie des tableaux, elles peuvent être effectuées dans leurs fichiers respectifs, ce qui minimise le risque d'introduire des bogues dans un code non lié.

L'implémentation de la grille de jeu dans game\_grid.c reflète le choix d'encapsuler toutes les opérations liées à la grille, telles que l'initialisation de la grille, le placement des vaisseaux et la vérification des résultats des bombes. Elle maintient l'état du plateau et dicte la progression du jeu. Des fonctions sont créées pour générer un placement aléatoire des navires sur la grille au début du jeu.

Le moteur de jeu dans game\_engine.c est responsable de la gestion de la logique du jeu, comme la vérification de la condition de fin de jeu. Cette logique est distincte de celle de la grille, car elle concerne l'état du jeu dans son ensemble plutôt que l'état des cellules individuelles de la grille.

Un aspect important du projet est la mémoire partagée. Elle est utilisée pour partager les grilles entre les différents processus, leur permettant de lire et d'écrire sur la même grille. Elle utilise l'appel système mmap pour créer une région de mémoire partagée accessible par les deux processus.

L'une des difficultés rencontrées au cours du projet a été la gestion de plusieurs clients. Le serveur doit gérer correctement les communications des deux clients, ce qu'il fait à l'aide de la fonction manage\_client. Cette fonction gère la séquence d'envoi et de réception des messages de chaque client, y compris l'interprétation des commandes de déplacement et le renvoi des résultats.

Un autre défi consistait à garantir l'atomicité des opérations, ce qui est crucial pour les jeux au tour par tour comme Battleship. Le serveur assure un jeu au tour par tour en servant les clients de manière séquentielle et en gérant l'état du jeu de manière appropriée.

En termes d'améliorations possibles, le jeu pourrait bénéficier d'un mécanisme de traitement et de validation des erreurs plus robuste. Par exemple, la commande de déplacement des clients est actuellement considérée comme étant dans un format valide, mais pourrait potentiellement faire planter le serveur si elle était malformée. Il serait utile de mettre en œuvre des contrôles de validation plus stricts pour les entrées des clients.

En conclusion, ce projet de jeu Battleship a impliqué la conception d'un système client-serveur qui communique sur des sockets, maintient l'état du jeu et gère plusieurs clients. Il représente une application de divers concepts de programmation en C et présente plusieurs domaines d'amélioration future.