



---

## Rapport de projet Bataille Navale

---

*Auteur :*  
Charles ÉPONVILLE,  
Nicolas GILLE,  
Vincent METTON,  
Grégoire POMMIER,  
Sylvestre VIMARD

*Responsable :*  
M. Yannick GUESNET

27 avril 2016



## Table des matières

|   |          |
|---|----------|
| <b>Remerciements</b>                          | <b>1</b> |
| <b>Introduction</b>                           | <b>2</b> |
| <b>1 Modèle de jeu</b>                        | <b>3</b> |
| 1.1 Le plateau de jeu . . . . .               | 3        |
| 1.2 Les intelligences artificielles . . . . . | 3        |
| <b>2 Le modèle réseau</b>                     | <b>4</b> |
| 2.1 Protocoles . . . . .                      | 4        |
| 2.1.1 Protocole Réseau . . . . .              | 4        |
| 2.1.2 Protocole de l'application . . . . .    | 4        |
| 2.1.2.1 Connexion des joueurs . . . . .       | 4        |
| 2.1.2.2 Échanges des données . . . . .        | 5        |
| 2.2 Gestion des erreurs . . . . .             | 5        |
| <b>Conclusion</b>                             | <b>6</b> |

## Remerciements

Nous tenons à remercier blablabla

## Introduction

De nos jours, il est possible de jouer à des jeux de société même avec des personnes qui ne sont pas dans la même pièce grâce aux différents protocoles réseaux. De plus en plus de jeux sont donc implémentés et utilisent ces technologies pour permettre de jouer en réseau, c'est à dire, de pouvoir jouer entre deux machines distantes.

De cette constatation, le besoin est venu d'implémenter un classique des jeux de société : la bataille navale (ou "touché coulé") qui est un jeu se jouant à deux joueurs dont le but est de couler tous les bateaux de l'adversaire, disposés sur un plateau de jeu. La bataille navale pouvant se jouer avec plusieurs règles, nous avons opté pour "le tour par tour" : chaque joueur joue une et une seule fois puis donne la main à son adversaire.

Ce projet a pour objectif de disposer de deux modes de jeu : un en local permettant de jouer contre une intelligence artificielle et l'autre permettant de jouer contre adversaire physique en réseau.

Dans ce rapport, nous allons présenter les différents points de l'application, notamment en décrivant le modèle du jeu, ensuite la partie réseau, puis l'interface homme machine permettant à l'utilisateur de jouer dans des conditions correctes, enfin nous exposerons dans une dernière partie les différentes connaissances que ce projet a apporté, les difficultés rencontrées ainsi que l'évolution possible de ce projet.

# 1 Modèle de jeu

## 1.1 Le plateau de jeu

Dans le cadre de ce projet, nous avons choisi de donner aux joueurs la possibilité de jouer différemment à la bataille navale, notamment de jouer en trois dimensions.

Pour cela, nous avons réfléchi puis développé un modèle capable de s'adapter aux besoins des joueurs. En effet, celui-ci peut décider de toutes les tailles, longueur, largeur ou encore profondeur, ainsi qu'un nombre de dimensions quelconques.

Le plateau de jeu permet donc de jouer en une dimension(soit sur la longueur ou soit sur la largeur), en deux dimensions (longueur et largeur), en trois dimensions nous ajoutons la profondeur pour enfin pouvoir jouer en plus de trois dimensions.

## 1.2 Les intelligences artificielles

## 2 Le modèle réseau

### 2.1 Protocoles

#### 2.1.1 Protocole Réseau

Le protocole réseau utilisé pour que deux joueurs puisse se connecter et joueur en réseau est le protocole TCP<sup>1</sup> qui une un protocole en mode connecté mais également fiable quant à la connexion entre deux machines et pour la réception des paquets envoyés sur le réseau.

Nous avons préféré le protocole TCP au protocole UDP (<sup>2</sup> car, premièrement, son implémentation est bien plus aisée. En effet, nous n'avons pas à traité certains cas d'erreurs, notamment dans le cas d'un paquet qui soit mal formé ou incomplet car TCP assure que celui-ci sera totalement prêt avant de l'envoyer.

Deuxièmement, malgré le fait que le protocole UDP soit plus rapide dans l'envoi des données, utiliser le protocole TCP dans notre cadre n'a pas d'impact significatif puisque, comme il sera exposé un peu plus tard, chaque joueur doit attendre la réception de données avant de continuer son traitement. Ainsi, il n'est pas, en pratique, possible de recevoir des données qui ne sont pas attendus à des moments précis.

#### 2.1.2 Protocole de l'application

L'application doit suivre un certain protocole pour que les deux joueurs puissent communiquer correctement et sans problèmes majeurs. Il a donc été décidé de suivre un type de protocole classique qui est celui du question/réponse. Néanmoins, nous n'avons pas choisi d'implémenter de serveur d'hébergement centralisant toutes les parties et dirigeant toutes les actions des clients, nous avons opté pour un échange de type pair à pair, c'est à dire que les deux joueurs doivent jouer les deux rôles : le processus serveur et le processus client, chacun à des temps déterminés et non mutables.

Nous expliquerons dans cette partie la connexion entre les deux joueurs ainsi que comment se passe l'échange de données.

##### 2.1.2.1 Connexion des joueurs

Comme dit plus haut, chaque joueur doit jouer deux rôles bien distincts, celui du processus serveur comme celui du processus client. Dans ce cas-ci,

---

1. Transmission Control Protocol : <https://tools.ietf.org/html/rfc793>

2. User Datagram Protocol : <https://tools.ietf.org/html/rfc768>

il est donc bon de noter l'ordre des différentes opérations effectuées entre un joueur qui va accueillir la partie et celui qui va en être l'invité, nous allons donc expliciter quelles actions font parties du processus client et lesquelles font parties du processus serveur.

Les deux joueurs vont, en premier lieu, créer et lancer leur serveur respectif, en spécifiant l'adresse de la machine en format IPv4, un port d'écoute (nous vérifions qu'il soit compris entre 1024 et 65535), ainsi qu'un nombre de connexions autorisées qu'on nommera "backlog".

Ensuite, le joueur hôte met son serveur en attente d'une demande de connexion via sa socket d'écoute (il est donc considéré en tant que processus serveur) et, de son côté, le joueur invité (considéré comme le processus client à ce moment) va se connecter à l'adresse du joueur hôte. Cela aura pour effet, côté processus serveur, de créer une socket de service par laquelle le joueur passera pour communiquer avec le joueur invité.

À cette étape, il reste à faire la connexion inverse, c'est à dire que le joueur hôte (qui est maintenant le processus client) doit faire une demande de connexion à l'adresse du serveur du joueur invité (devenu le processus serveur) et, une fois la connexion acceptée par la socket d'écoute du processus serveur, une nouvelle socket de service est créée pour que le joueur invité puisse communiquer avec le joueur hôte.

Suite à ces différentes étapes, la connexion entre le joueur hôte et le joueur invité est établie, les échanges de données peuvent donc être effectués.

#### **2.1.2.2 Échanges des données**

### **2.2 Gestion des erreurs**



### **3 Interface Homme Machine**

## **4 Autres**

### **4.1 Idées d'améliorations**

### **4.2 Problèmes rencontrés**

### **4.3 Compétences acquises**

## Conclusion

Pour conclure, avec  $\text{\LaTeX}$  on obtient un rendu impeccable mais il faut s'investir pour le prendre en main.