

2014

Projet Génie Logiciel Bataille Navale



Julien Amacher

Bastian Gardel

Nguyen-Phuong Le

Parfait-plaisir-de-Pâques Noubissi

Yannick Widmer

À l'attention de M. Eric Lefrançois

2 Avril 2014

Table des matières

Table des matières	1
Introduction.....	3
Fonctionnement général.....	4
Objectifs	4
Utilisations.....	4
Règles du jeu	4
Contraintes	5
Spécifications.....	6
Fonctionnalités à implémenter	6
Système de fichiers.....	7
Technologies.....	7
Style	7
Partage des responsabilités entre le serveur et le client	8
Serveur :	8
Client :	8
Cas d'utilisation	9
Diagramme	9
Acteurs.....	9
Protocole Client/Serveur	11
Modèle de domaine	12
Base de données	13
Objectifs	13
Structure de la base de données.....	13
Interfaces Utilisateurs	14
UIHome.....	14
UIGameMain	15
UIServersListing.....	16
UIAskPass.....	16
UIWaitPlayers	16
UIManualConnect.....	17

UIUsernameConf	17
UIServerCreation	17
UIServerStats	18
Responsabilités et rôles de l'équipe.....	19
Iterations	20

Introduction

Ce projet s'inscrit dans le cadre du cours de Génie Logiciel dispensé par M. E. Lefrançois, à la Haute Ecole d'Ingénierie et de Gestion du canton de Vaud. Il s'étend sur un semestre entier et il est réalisé par une équipe de 5 étudiants : M. J. Amacher, M. B. Gardel, M. N.-P. Le, M. P. Noubissi et M. Y. Widmer. La méthode UP a été choisie pour la réalisation de ce projet.

Réaliser un projet d'un semestre entier en respectant une méthode rigoureuse peut s'avérer ardu. C'est néanmoins une expérience très enrichissante qui donne l'occasion de mettre en pratique la théorie vue précédemment en cours de Génie Logiciel et d'apprendre à travailler en équipe. C'est donc avec une grande avidité de connaissances et d'expériences que les membres de l'équipe ce projet ont entamé l'aventure.

Un brainstorming initial a pris place afin de décider du projet à réaliser. Après avoir écouté l'avis de chacun, il a été rapidement décidé de réaliser un jeu. Le côté ludique permet de prendre du plaisir lors de la conception et de l'implémentation et encore davantage lors des phases de tests. Il devient ainsi plus aisé de consacrer beaucoup de temps à la réalisation du projet et de le mener à bien.

Il a ensuite été nécessaire de trouver un jeu qui satisfasse à toutes les exigences requises pour le sujet du projet. Il fallait également que le jeu soit suffisamment connu de tous pour que chacun soit au clair et en accord sur les mécanismes à implémenter. Le choix s'est immédiatement porté sur un classique des jeux en réseau : la bataille navale.

Pour apporter une touche personnelle et suite aux conseils judicieux du professeur, l'équipe a décidé d'incorporer un système de bonus : Des bombes sous-marines qui permettraient de faire des réactions en chaîne et de renverser le cours d'une partie.

La suite de ce document décrit ainsi les spécifications, ainsi que le déroulement de ce projet.

Fonctionnement général

Objectifs

Le projet vise à développer un jeu de bataille navale permettant de jouer seul (utilisateur contre intelligence artificielle (AI) ou à deux (joueur 1 contre joueur 2)

Utilisations

Chaque personne joue grâce à sa propre machine, peut créer une partie et inviter d'autres joueurs à sa partie, ou à une autre partie.

Des bonus (ou malus), positionnés aléatoirement, peuvent être récoltés suite à un tir par le joueur ayant effectué ledit tir.

Les cartes de jeu peuvent être choisies. La flotte des joueurs s'adapte à la taille de la grille choisie.

Il doit être possible de choisir avec quelles personnes les personnes jouent la partie.

Lors d'une partie, il est possible de dialoguer avec son adversaire. Le client obtient une liste des parties qu'il peut joindre.

Règles du jeu

La Bataille Navale se joue à 2 joueurs. Soit 2 humains, soit 1 joueur humain et 1 intelligence artificielle.

Au début du jeu, chaque joueur place ses bateaux sur son plan d'eau aux emplacements qu'il désire selon sa stratégie.

Le serveur décide aléatoirement quel joueur va commencer la partie.

Les joueurs effectuent ensuite des tirs à tour de rôle. Chacun son tour, les joueurs tirent sur le plan d'eau adverse en essayant de toucher les bateaux de l'adversaire. Les bateaux coulent lorsque qu'ils sont touchés sur l'entier des cases sur lesquelles ils sont placés.

Des bonus permettent de toucher et/ou couler plus d'un bateau à la fois. Lorsqu'une mine est touchée, ses dégâts sont propagés dans un certain périmètre. Le bonus satellite permet d'espionner pendant un cours instant une partie de la flotte adverse.

Le but est de couler tous les bateaux adverses. Une fois ceci fait, un gagnant est déclaré et la partie se termine.

Contraintes

Un seul serveur peut être créé par utilisateur.

Le nombre maximal de joueurs est 2 dans une même partie.

Spécifications

Fonctionnalités à implémenter

1. Intelligence artificielle
2. Fenêtres (UI)
3. Grille de jeu à dessiner
4. Positionnement des bateaux sur la grille de jeu
5. Design du protocole de communication
6. Implémentation du protocole de communication
7. Images des bateaux, images tiré/pas touché/touché/touché et coulé
8. Bruitages
9. Chat
10. Serveur (création, attente de clients, priorités, gestion des bonus, gestion des tirs, gestion du gagnant, stockage de la dernière configuration de création de serveur)
11. Client (logique entre les fenêtres, affichage des serveurs, connexion manuelle, jeu en lui-même, stockage du nom d'utilisateur)
12. Découverte des serveurs
13. Le processus de jeu en lui-même
 - a. Serveur
 - b. Client

L'administrateur du serveur peut, à tout instant (que le serveur soit démarré ou non), consulter les statistiques de ce dernier. L'on entend par statistiques le nombre de matches joué, le gagnant de chaque match (et le perdant), l'adresse IP du serveur, le mot de passe de connexion, et les noms des joueurs si une partie est en cours.

Système de fichiers

<Exécutable Java>

- client
 - config
client.xml : Contient le pseudo de l'utilisateur
client.dtd
- server
 - stats
matches.xml : Contient la liste des matchs et qui a gagné
matches.dtd
 - config
lastconfig.xml : Contient les paramètres du serveur lors de son dernière exécution
lastconfig.dtd

Technologies

- Fenêtres : **Swing** uniquement
- Langage : **Java 1.7** uniquement
- **Modèle MVC** (y compris Observateur-observé)
- Stockage de la base de données en **XML**. Manipulation XML avec **XPath** uniquement (pas de XML à la main) avec la **bibliothèque Java standard** uniquement
- Compatible au minimum avec **Windows 7**

Style

- Noms Java : camelCase
- Noms XML : snake_case
- Les XML ont chacun un DTD portant le même nom que le fichier XML et sont dans le même dossier que les XML
- Les interfaces commencent par un I (majuscule)
- Les classes représentant des interfaces graphiques commencent par UI (majuscules)
- Attention particulière quant à la réutilisabilité du code

Partage des responsabilités entre le serveur et le client

Serveur :

Il attend que 2 personnes se connectent. Une partie est ensuite créée comprenant tous les joueurs qui participeront à la partie.

Connaît la position des bateaux de tous les joueurs, ainsi que des bonus. Lors d'un tir, il indique à l'adversaire que sa carte a été attaquée, et où. Il indique également quels sont les dégâts à la personne ayant initié le tir. Lorsque tous les bateaux d'un des joueurs sont coulés, le serveur indique à tous les joueurs que la partie est terminée.

Client :

Attend que l'autre personne (l'adversaire) ait joué avant de demander la position de tir à l'utilisateur.

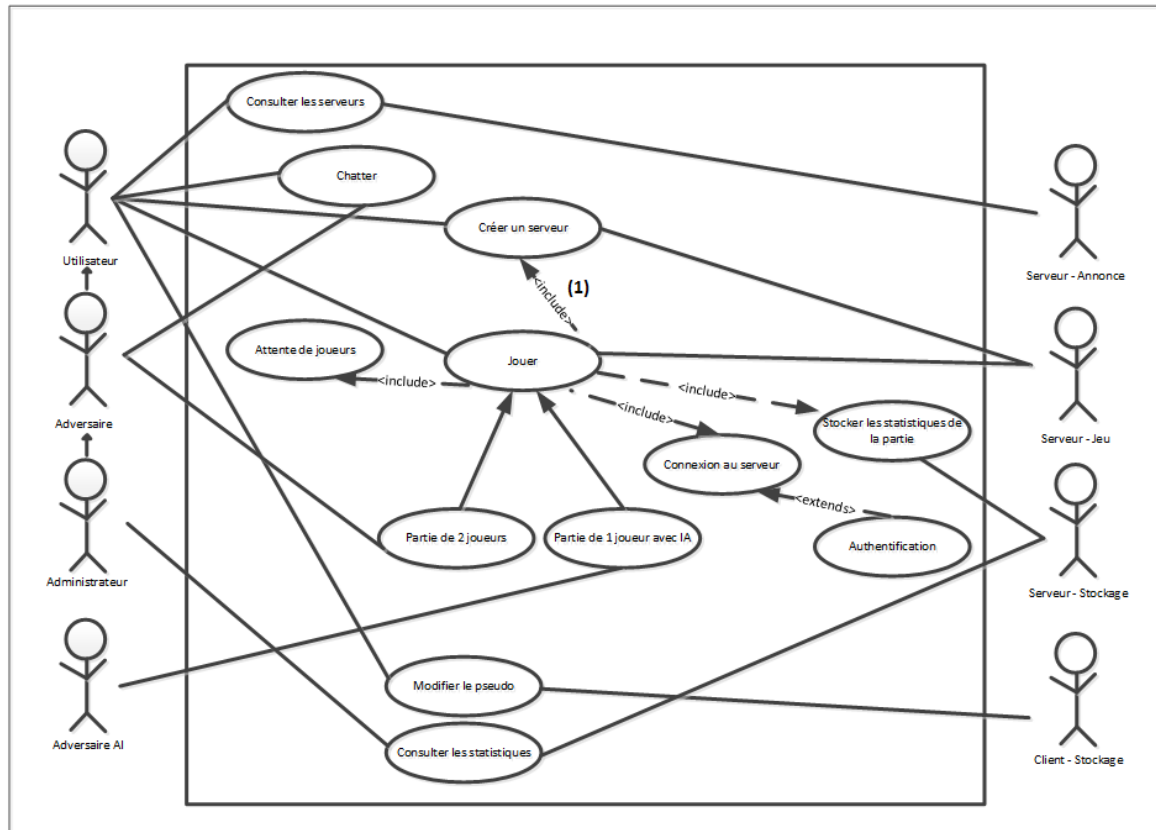
Ne connaît pas la position des bateaux des autres joueurs, sauf au moment d'un tir où le serveur indique si quelque chose a été touché chez l'adversaire ou non. Une fois la partie terminée, le premier joueur se déconnectant du serveur initie une déconnexion de tous les utilisateurs connectés, et le serveur est alors prêt à commencer une nouvelle partie sitôt deux utilisateurs connectés.

Cas d'utilisation

Diagramme

Use cases

(1) Pas nécessairement effectué par la même entité



Acteurs

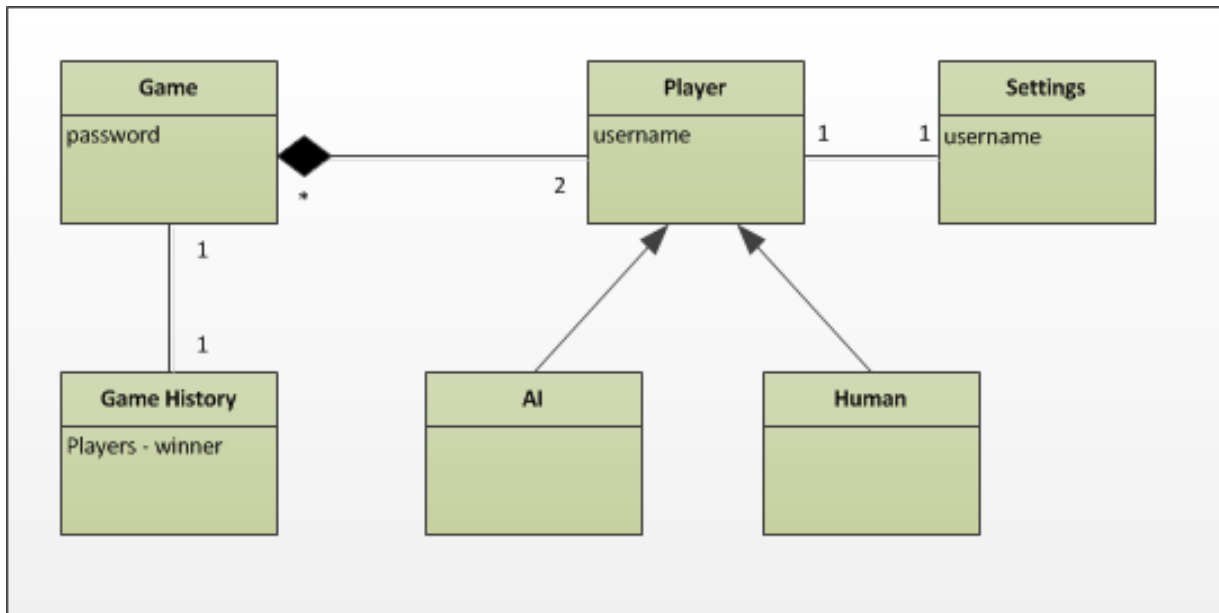
- **Utilisateur:** L'utilisateur est une personne physique qui sera en l'occurrence le joueur physique de la partie.
 - **Fonctions:**
 - L'utilisateur peut consulter la liste des serveurs à disposition.
 - Il peut aussi chatter avec son adversaire (que ça soit avec l'AI ou un autre joueur)
 - Il peut créer un serveur qui peut accepter un autre joueur ou un AI.
 - Il peut aussi jouer en rejoignant une partie.
 - Il peut aussi modifier son pseudo de joueur.
- **Adversaire:** L'adversaire est aussi est une personne physique, en l'occurrence ce sera le joueur de la partie.
 - **Fonctions:**
 - L'adversaire pourra consulter la liste des serveurs à disposition.

- Il peut aussi chatter avec son adversaire (que ça soit avec l'AI ou un autre joueur)
 - Il peut créer un serveur qui peut accepter un autre joueur ou un AI.
 - Il peut aussi jouer en rejoignant une partie.
 - Il peut aussi modifier son pseudo de joueur.
- **Administrateur:** C'est un adversaire ou un utilisateur, c'est donc une personne physique.
 - Fonctions :
 - L'administrateur peut consulter les statistiques de son serveur.
- **Adversaire AI:** C'est un adversaire non physique.
 - Fonctions :
 - Il peut jouer à la bataille navale contre un utilisateur en mode Humain vs IA
- **Serveur Annonce:** Le serveur s'annonce périodiquement sur le réseau local.
 - Fonctions :
 - Il va s'annoncer pour être aperçu dans la liste des serveurs à disposition pour un utilisateur.
- **Serveur Jeu:** Le serveur de jeu sera en interaction avec l'utilisateur.
 - Fonctions :
 - Il pourra héberger une partie de l'utilisateur pour que des utilisateurs puissent jouer.
- **Serveur Stockage:** Base de données XML des statistiques des parties.
 - Fonctions :
 - Elle peut stocker les historiques des parties dans un fichier XML
 - Elle peut stocker la dernière configuration du serveur.
- **Client Stockage:** Base de données XML du client.
 - Fonctions :
 - Stockage du pseudo de l'utilisateur (qui pourra éventuellement être modifié ultérieurement)

Protocole Client/Serveur

- Connect
 - Se connecter au serveur, avec et sans authentification
- GetGameParameters
 - Récupère les informations serveur
- Beaconsing
 - Annonce des serveurs aux clients
- Disconnect
 - Déconnexion
- AttackCoordinate
 - Attaque l'adversaire
- AttackedCoordinate
 - L'adversaire attaque
- UserDisconnect
 - L'adversaire se déconnecte
- PositionMyBoats
 - Positionnement des bateaux sur la grille de jeu
- GameStart
 - Début de partie
- NOP
 - « Heartbeat »
- ChatSend
 - Envoi d'un message de chat
- ChatReceive
 - Réception d'un message de chat

Modèle de domaine



Base de données

La base de données est stockée en XML. Les manipulations sur la base de données se fait avec des expressions XPath uniquement (pas de XML à la main) et avec la librairie Java standard uniquement.

Objectifs

L'objectif de la base de données est de pouvoir stocker les noms d'utilisateurs, les dernières configurations utilisées des serveurs et l'historique des parties.

Structure de la base de données

Le stockage persistant du côté client se résume à stocker le nom d'utilisateur :

```
<?xml version="1.0"?>
<!DOCTYPE client_config SYSTEM "client_config.dtd">
<client_config>
  <username>My username</username>
</client_config>
```

Le stockage persistant du côté serveur se résume à stocker la dernière configuration :

```
<?xml version="1.0"?>
<!DOCTYPE server_config SYSTEM "server_config.dtd">
<server_config>
  <last_parameters> <!-- Parameters used when the server was last created -->
    <server_name>SuperServer</server_name> <!-- Server name that will be broadcasted -->
    <mode>2 versus AI</mode> <!-- Can be either "2 versus 2" or "1 versus AI" -->
    <authorization type="shared_secret" state="enabled"> <!-- type is fixed, state can be either enabled or disabled. When disabled, no authentication is required at all -->
      <password>This is the password</password> <!-- Password in cleartext -->
    </authorization>
    <specials>
      <special name="mines" state="enabled" /> <!-- mines are enabled -->
      <special name="satellite" state="disabled" /> <!-- satellite is disabled -->
    </specials>
  </last_parameters>
</server_config>
```

... ainsi que l'historique des parties :

```
<?xml version="1.0"?>
<!DOCTYPE games_history SYSTEM "games_history.dtd">
<games_history>
  <game>
    <date>2014-04-26 12:45:67</date>
    <players>
      <player winner="yes">Luc</player>
      <player winner="no">Marian</player>
    </players>
  </game>
  <game>
    <date>2014-04-25 12:14:15</date>
    <players>
      <player winner="yes">Marian</player>
      <player winner="no">Luc</player>
    </players>
  </game>
</games_history>
```

Interfaces Utilisateurs

UIHome

Cette fenêtre est la fenêtre s'ouvrant lorsque l'application est exécutée.

L'utilisateur a le choix entre créer un nouveau jeu qui sera stocké sur son serveur, afficher les statistiques de son serveur et consulter la liste des serveurs hébergeant une partie.

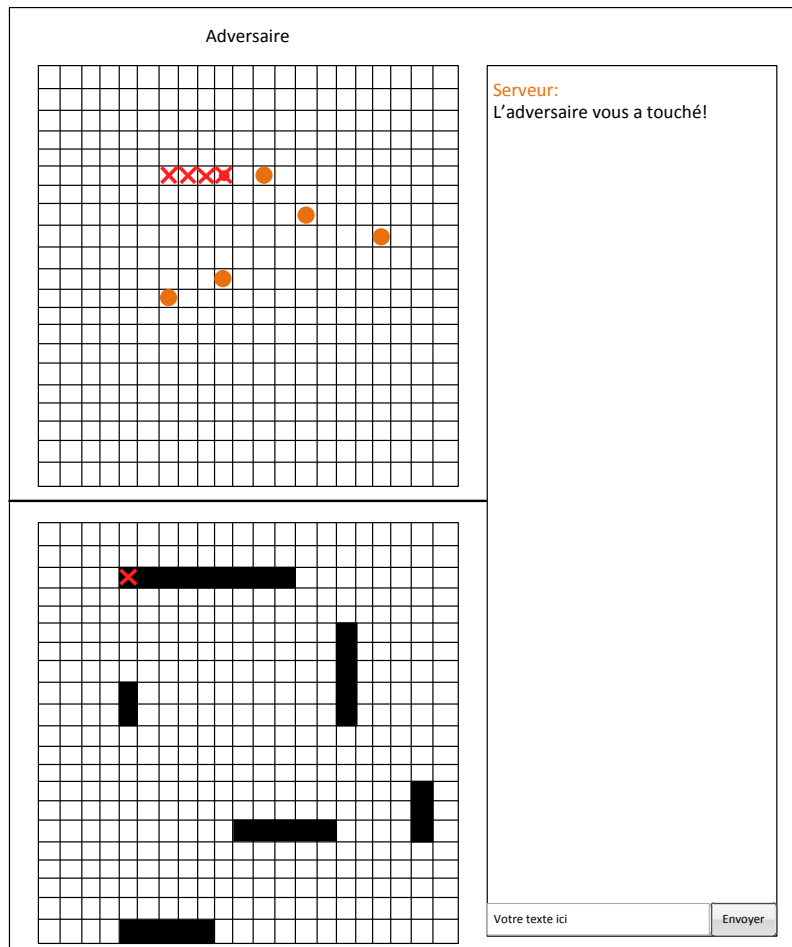
Le serveur de l'utilisateur ne pourra héberger qu'une seule partie à la fois. Si une partie a déjà été créée sur le serveur, alors il ne sera pas possible d'en créer une autre tant que la partie en cours n'est pas terminée. Si la création d'une nouvelle partie est impossible, alors le bouton correspondant n'apparaîtra pas.

Un bouton permettant de quitter l'application est également présent.



UIGameMain

Cette fenêtre s'occupe d'afficher l'état actuel de la partie.







1. Le jeu commence une fois deux joueurs connectés au serveur (L'AI compte pour un joueur)
2. Le joueur 1 place ses bateaux
3. En même temps, le joueur 2 place ses bateaux
4. Le serveur décide aléatoirement qui va commencer la partie
5. Au clic sur la case, un tir est effectué. Soit l'on a touché quelque chose, soit c'est un tir raté. L'on ne peut jouer sur une case précédemment jouée. La case jouée change d'apparence selon que le tir a réussi, le tir a réussi et a coulé un bateau ou le tir a échoué. Le chat indique le résultat du tir.
6. Le serveur annonce le gagnant (il ne peut y avoir qu'un seul gagnant)
7. Le jeu est terminé
8. Une fenêtre demande si l'on veut recommencer la partie

UIServersListing

Cette fenêtre a pour but de lister les serveurs actuellement présents en proximité et de permettre la connexion manuelle à un serveur éventuellement non listé :

Spécifier manuellement

Nom	Joueurs	Libre
 Luc's Server	2	
Hophop	1	
NoName	0	

Connecter

Une fois une ligne sélectionnée, un clic sur *Connecter* effectue la connexion au serveur choisi.

Les serveurs demandant une authentification se présentent avec une icône de cadenas.

UIAskPass

Cette fenêtre est affichée lorsqu'une tentative de connexion à un serveur demandant authentification est effectuée :

Ce serveur nécessite un mot de passe pour s'y connecter.

Mot de passe:

Connecter

Annuler

UIWaitPlayers

Une fois connecté au serveur (et authentifié, si nécessaire), la fenêtre suivante s'affiche :

Vous êtes connecté au serveur <nomServeur> sur <IP/Host>

Veuillez patienter. En attente de joueurs.

Déconnecter

UIManualConnect

Cette fenêtre permet de demander un nom d'hôte (game.heig-vd.ch) ou une IPv4 d'un serveur de jeu à l'utilisateur :

Adresse IP ou
nom d'hôte du serveur :

UIUsernameConf

Cette fenêtre a pour but de permettre à l'utilisateur de configurer son nom d'utilisateur.

Pseudo :

UIServerCreation

Cette fenêtre demande à l'utilisateur quels paramètres utiliser pour l'instanciation du serveur :

Mode: ☐ AI ☒ 2 joueurs

Bonus: ☒ Satellite espion
☒ Mines

Nom:
Max. 20 caractères

☒ Accès restreint




Mot de passe:

UIServerStats

Cette fenêtre affiche les adresses IP internes et externes sur lequel le serveur écoute, le mot de passe, les joueurs dans la partie en cours ainsi que l'historique des parties (joueur ayant gagné affiché à gauche avec une icône, joueur ayant perdu au centre et date de fin de partie à droite).

Adresse IP locale : 192.168.6.9
Adresse IP externe : 189.53.1.34
Mot de passe: <aucun>
Partie en cours : luc
marian

Historique :

	luc	paul	2014-02-01 08:36:44
	marian	adrien	2014-01-01 09:19:06
	sophie	adrien	2014-04-04 10:31:09

Responsabilités et rôles de l'équipe

Amacher Julien	Noubissi Parfait-plaisir-de-Pâques	Le Nguyen-Phuong	Gardel Bastian	Widmer Yannick
<u>Software Architect</u> Designer Implementer End user Customer Analyst Tester	<u>Process Engineer</u> Implementer End user Customer Analyst Tester	<u>Project Manager</u> Designer Implementer End user Customer Analyst Tester	<u>Deployment Manager</u> Designer Implementer End user Customer Analyst Tester	<u>Configuration Manager</u> Designer Implementer End user Customer Analyst Tester
Communication protocol design		User Interface implementation	Game mechanics processes implementation	Artificial intelligence design and implementation
Communication protocol implementation		Game grid design		instant messaging (chat) design and implementation
Servers discovery processes design and implementation		Ships positioning design and implementation		
		Illustrations (ships, event icons, etc.)		
		Sound effects		
<u>SERVER</u> conception and implementation : Creation, waiting for clients, priorities, bonus management, shooting management, winning management, last server configuration storage				
<u>CLIENT</u> conception and implementation : Logic between windows, servers display, manual connection, game progress, username storage				

Iterations

Designations:

Julien Amacher : JAR
 Bastian Gardel : BGL
 Nguyen-Phuong Le : NLE
 Parfait-plaisir-de-Pâques Noubissi : PNI
 Yannick Widmer : YWR

Iteration	Dates	Objectives	En charge	Heures prévues	Heures effectives
1	29 Mars – 5 Mai	Consulter les serveurs : UIServersListing	NLE	4	
		Chatter: UIGameMain partie chat seule	NLE YWR	2	
		Créer un serveur: UIServerCreation	NLE	5	
		Attente de joueurs : UIWaitPlayers	NLE	2	
		Grille de jeu: UIGameMain	NLE	6	
		Authentification: UIAskPass	NLE	2	
		Fenêtre d'accueil: UIHome	NLE	2	
		Connexion au serveurs : UIManualConnect	NLE	2	
		Modifier le pseudo : UIUsernameConf	NLE	2	
2	6 Mars – 12 Mai	Architecture: Protocole réseau	JAR	10	
		Chatter: UIGameMain partie chat : intégration à la fenêtre principale	NLE YWR	2	
3	13 Mai – 19 Mai	Architecture: Interactions fenêtres & synchronisations	JAR PNI	10	
		Chatter: Implémentation sens unique	NLE YWR	10	
4	20 Mai – 26 Mai	Placement en dur des bateaux	BGL JAR NLE PNI	10	
		Chatter: Implémentation deux sens	YWR	6	
5	27 Mai – 2 Juin	Jeu : 1 joueur solo tire sans adversaire	BGL JAR PNI YWR	16	
6	3 Juin – 9 Juin	Humain vs Humain : implementation	BGL JAR PNI YWR	16	
7	10 Juin – 16 Juin	Humain vs Humain : Protocoles de tests	BGL	6	
		IA : tirs sur les 2 diagonales	JAR PNI YWR	6	

		Intégration des effets sonores	NLE	6	
8	17 Juin – 24 Juin	IA : Implementation finale	JAR PNI YWR	10	
		Humain vs IA : Protocoles de tests	YWR	6	
		Rédaction de la documentation	NLE	2	