

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	18
Modèle de domaine	19
Base de données	20
Objectifs	20
Structure de la base de données.....	20
Interfaces Utilisateurs	21
UIHome.....	21
UIGameMain	22
UIServersListing.....	23
UIAskPass.....	23
UIWaitPlayers.....	23
UIManualConnect.....	24

UIUsernameConf	24
UIServerCreation	24
UIServerStats	25
Responsabilités et rôles de l'équipe.....	26
Iterations	27

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.

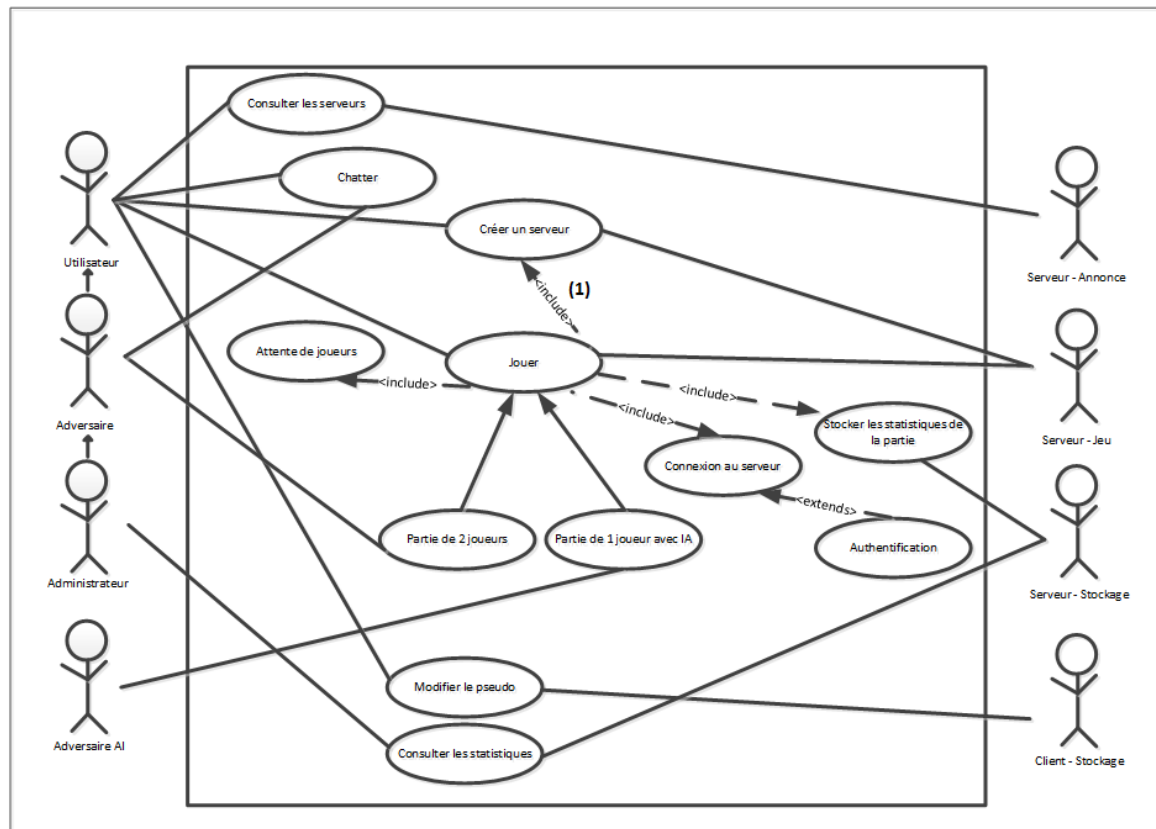
Ne connaît pas la position des bateaux des autres joueurs, sauf au moment d'un tir où le serveur indique si quelque chose a été touché chez l'adversaire ou non. Une fois la partie terminée, le premier joueur se déconnectant du serveur initie une déconnexion de tous les utilisateurs connectés, et le serveur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

Scenarios

Modifier le pseudo

Rédacteur : Parfait-plaisir-de-Pâques Noubissi

Chaque utilisateur doit avoir un pseudonyme pour pouvoir utiliser le programme. Ce pseudonyme est stocké dans un fichier de configuration.

Scénario au lancement du programme avec pseudonyme indéfini :

Si au moment de lancer le programme la valeur du pseudo est manquante dans le fichier de configuration, une fenêtre de création/modification de pseudonyme s'affichera à l'écran (cf. UIUsernameConf du chapitre Interfaces Utilisateurs).

1. La fenêtre UIUsernameConf s'affiche. Dans cette fenêtre, il y a un champ texte et un bouton « Valider ».
2. L'utilisateur entre son pseudonyme désiré dans le champ texte.
3. L'utilisateur clique sur le bouton « Valider »
4. Le pseudonyme est stocké dans le fichier de configuration.
5. L'utilisateur arrive sur la fenêtre d'accueil.

Scénario avec pseudonyme déjà défini :

L'utilisateur peut également accéder à la fenêtre de choix de pseudonyme à partir de la page d'accueil principale s'il désire modifier son pseudonyme.

1. L'utilisateur clique sur le bouton « Modifier son pseudonyme » sur la page d'accueil du programme.
2. La fenêtre UIUsernameConf s'affiche. Dans cette fenêtre, il y a un champ texte, un bouton « Modifier » et un bouton « Annuler ».
3. L'utilisateur entre son nouveau pseudonyme désiré dans le champ texte.
4. L'utilisateur clique sur le bouton « Modifier »
5. Le nouveau pseudonyme est remplacé dans le fichier de configuration
6. La fenêtre d'accueil principale se réaffiche.

Créer un serveur

Rédacteur : Nguyen-Phuong Le

Scénario ordinaire

1. L'utilisateur clique sur le bouton « Créer une partie » sur la page d'accueil principale.
2. La fenêtre UIServerCreation s'affiche à l'écran.
3. L'utilisateur choisit le mode de jeu « vs AI » ou « 2 joueurs » de la partie qui sera hébergée sur son serveur en cliquant sur l'un des deux boutons radios
4. Si l'utilisateur veut ajouter des bonus, il clique sur les checkbox correspondantes.
5. L'utilisateur entre le nom qu'il veut donner à son serveur hébergeant la partie

6. Si l'utilisateur veut ajouter un mot de passe à son serveur, il clique sur la checkbox correspondante.
7. Si l'utilisateur a cliqué sur la checkbox, il peut entrer le mot de passe désiré.
8. L'utilisateur clique sur le bouton « Créer »

Attente de joueurs

Rédacteur : Parfait-plaisir-de-Pâques Noubissi

Scénario ordinaire

1. L'utilisateur s'est connecté sur un serveur.
2. La fenêtre UIWaitPlayers s'affiche à l'écran indiquant l'attente.
3. Le joueur patiente.
4. Un autre joueur se connecte, la fenêtre UIWaitPlayers disparaît et la fenêtre de jeu s'affiche. Cf. Scénario « Jouer » pour la suite.

Consulter les serveurs

Rédacteur : Nguyen-Phuong Le

Scénario ordinaire

1. L'utilisateur clique sur « Afficher la liste des serveurs » dans la fenêtre d'accueil principale.
2. La fenêtre UIServersListing s'affiche à l'écran.
3. L'utilisateur peut voir quels serveurs hébergeant des parties existent, s'ils sont protégés par un mot de passe, le nombre de joueurs actuellement connectés à ces serveurs, et s'ils ont une place de libre.
4. Si l'utilisateur veut jouer, il choisit un serveur hébergeant une partie où il y a de la place en cliquant sur sa ligne.
5. L'utilisateur clique sur le bouton « Connecter ».

Connexion au serveur

Rédacteur : Bastian Gardel

Scénario ordinaire

1. Le client lance une connexion sur le serveur choisi en cliquant sur connecter dans la fenêtre de la liste des serveurs.
2. Le client attend que le serveur établisse la connexion.
3. Le serveur établit la connexion.
4. Si c'est un serveur qui nécessite authentification :
 - a. Cf. Scénario « Authentification »
5. La connexion est établie. (Cf. Scénario « Jouer »).

Scénario connexion manuelle

1. Le client clique « Spécifier manuellement » dans la fenêtre de la liste des serveurs.
2. La fenêtre UIManualConnect s'affiche

3. L'utilisateur entre l'adresse IP le nom d'hôte du serveur auquel il souhaite se connecter
4. L'utilisateur clique sur « Connecter »

Authentification

Rédacteur : Bastian Gardel

Scénario ordinaire

1. Le serveur demande au client de s'authentifier.
2. Le client demande à l'utilisateur de rentrer le mot de passe.
3. L'utilisateur a deux choix, soit il annule, soit il entre son mot de passe.
4. Si il annule :
 - a. Retour à la fenêtre d'accueil.
5. Le client envoie le mot de passe au serveur.
6. Le serveur vérifie les identifiants.
7. Si le mot de passe est incorrect :
 - a. Retour au point « a ».

Jouer & Chatter

Rédacteur : Julien Amacher

Scénario ordinaire :

La partie commence une fois deux entités différentes connectées au serveur (deux joueurs physiques ou un joueur physique et un joueur A.I.). Se référer au scénario **Attente de joueurs**

1. La fenêtre de jeu s'affiche
2. Les deux joueurs placent chacun leurs bateaux sur leur propre grille de jeu. Pour ce faire, chaque joueur envoie le message *PositionMyBoats* au serveur.
3. Si le bonus « mines » est activé :
 - a. Le serveur va positionner autant de bonus mines dans la grille de chacun des joueurs, toujours à des positions qui ne sont pas occupées par des bateauxSi le bonus « satellite espion » est activé :
 - a. Le serveur va positionner autant de bonus satellite espion dans la grille de chacun des joueurs, toujours à des positions qui ne sont pas occupées par des bateaux
4. Après l'envoi du message *PositionMyBoats*, les deux joueurs sont en attente du message *GameStart*.
5. Le serveur détermine quel joueur effectuera le premier coup
6. Le serveur envoie le message *GameStart* indiquant que toutes les informations sont disponibles pour commencer une partie. Ce message indique qui des deux joueurs commencera à tirer.
7. Le joueur qui commence à tirer demande à l'utilisateur de désigner une case sur laquelle le tir sera effectué. Le message *AttackCoordinate* est envoyé par ce joueur.
8. Le serveur vérifie si le joueur a touché un bateau ou un bonus.
(Rappel : seul le serveur connaît en tout temps la position des bateaux de chaque joueur.)

Si un bonus a été touché :

- a. Le bonus en question est activé à l'endroit où il est placé sur la carte

Si le bonus est une mine :

- a. La mine explose dans un rayon de 2 cases et endommage/coule les bateaux de l'adversaire se trouvant dans ce rayon.

Si le bonus est un satellite espion :

- a. Le satellite espion révèle à la personne ayant tiré un nombre aléatoire de cases de l'adversaire durant une brève période de temps.
9. Le serveur indique à la personne ayant tiré si son coup a touché et/ou endommagé et/ou déclenché un ou plusieurs bonus (une mine qui explose à proximité d'une autre mine la déclenche) ou n'a rien touché. Pour ce faire, une indication visuelle est présente sur sa grille de tir.
 10. Le serveur indique à la personne ayant subi le coup avec un message de type *AttackedCoordinate* quels sont les dommages subis (ou si rien n'a été touché)
 11. Le joueur ayant tiré est maintenant en attente du tir de l'adversaire (adversaire <= « joueur qui commence à tirer »)
 12. L'adversaire effectue maintenant les opérations 7 à 11.
 13. Les points 7 à 12 sont répétés tant que tous les bateaux ne sont pas tous coulés (les joueurs jouent effectivement à tour de rôle)
 14. Le serveur envoie le message *EndOfGame* à chaque joueur, indiquant quel joueur a gagné la partie.
 15. Les joueurs sont informés du gagnant.
 16. La partie est terminée.
- Si un des joueurs est un AI :
- a. il se déconnecte immédiatement.
17. Lorsqu'il ne reste qu'un seul joueur connecté, le serveur le déconnecte et est de nouveau prêt à recevoir deux nouveaux joueurs.
 18. Si un des joueurs est un AI :
 - a. il se reconnecte après la fin de la partie au même serveur.

A tout moment (événements asynchrones) :

- a. Le joueur reçoit le message *ChatReceive* : il affiche **qui** parle et **ce que la personne dit** dans sa fenêtre de chat.
- b. Le joueur décide de parler à son adversaire :
 1. Le joueur clique dans la zone de texte du chat
 2. Le joueur appuie sur la touche Enter ou sur le bouton d'envoi
 3. A ce moment, le message *ChatSend* est envoyé au serveur
 4. Le serveur reçoit le message *ChatSend* et effectue deux opérations en parallèle :

1. Il confirme que le message a été reçu au joueur venant d'envoyer le message *ChatSend* [text=contenu du champ de texte de chat] en lui envoyant un message de réponse *ChatSend*.
2. Il envoie à tous les joueurs (c'est-à-dire y compris le joueur venant d'envoyer le message *ChatSend*) un message de type *ChatReceive* [userSpeaking=joueur ayant envoyé le message *ChatSend*, text=ce que le champ de texte de chat contient]
- c. Le joueur reçoit le message *UserDisconnect* : il affiche qu'un utilisateur s'est déconnecté.
Rappel : ce message n'indique pas que la partie est terminée. En effet, ce message a été prévu pour flexibiliser le jeu dans le cas éventuel d'une amélioration du jeu où il serait possible de jouer à deux contre deux ou plus.
- d. Le joueur reçoit le message *EndOfGame* **indiquant que le jeu ne peut plus continuer** : un message d'information indiquant que la partie est terminée s'affiche.
Note : le scénario ordinaire utilise ce message pour dire que la partie est terminée avec un gagnant.
- e. Le joueur reçoit le message *Nop* : il ne fait rien de spécial.
- f. Le joueur ferme la fenêtre *ou* son ordinateur s'éteint *ou* sa connexion Internet se déconnecte. Cette action ferme tout lien entre le joueur et le serveur. A ce moment rien de spécial n'est fait par le client du joueur ayant fermé la fenêtre. Le serveur, lui, détectera que le joueur s'est déconnecté et va alors émettre un message *UserDisconnect* puis, immédiatement après, un message *EndOfGame* au joueur se trouvant dans la partie. La partie est alors terminée.

Stocker les statistiques de parties

Rédacteur : Yannick Widmer

Scénario ordinaire :

Les statistiques sont stockées sur le serveur une fois la partie terminée. Se référer au scénario **Jouer**.

1. La fenêtre de fin de partie s'affiche.
2. La statistique va alors pouvoir être stocké sur le serveur. Pour cela il va devoir trouver le nom de l'adversaire. Si l'adversaire est un utilisateur physique :
 - a) Le serveur va appeler la fonction `getPseudoClient()`
 - b) Cette fonction envoie une requête au client de lui envoyer son pseudo.
 - c) Le client reçoit la requête d'envoyer son pseudo au serveur.
 - d) Le client récupère son nom d'utilisateur dans le fichier xml `client_config.xml`
 - e) Le client envoie son nom d'utilisateur au serveur.

Sinon c'est une intelligence artificielle :

- a) Le serveur utilisera le nom de AI pour le nom de l'adversaire.
3. Le serveur lit le fichier xml client_config.xml et récupère le nom de l'utilisateur qui héberge la partie.
4. Le serveur récupère la date et heure courante.
5. Le serveur récupère le pseudo du gagnant de la partie.
6. Si le serveur possède déjà un fichier xml game_history :
 - a) Le serveur ajoute à ce fichier une balise game qui contiendra la date et heure de la fin de partie, le pseudo du joueur qui héberge la partie, le pseudo de l'adversaire ainsi que un attribut qui définit le gagnant

Si le pseudo du gagnant correspond au pseudo du joueur qui héberge la partie

 - a) Le serveur insère dans l'attribut player_win un yes du joueur qui héberge la partie.
 - b) Le serveur insère un no dans l'attribut player de l'adversaire.

Sinon

 - a) Le serveur insère dans l'attribut player_win un no du joueur qui héberge la partie.
 - b) Le serveur insère un yes dans l'attribut player de l'adversaire.

Sinon
 - b) Le serveur crée un fichier xml game_history
 - c) Le serveur effectue l'opération 6.a.

Consulter les statistiques du serveur

Rédacteur : Nguyen-Phuong Le

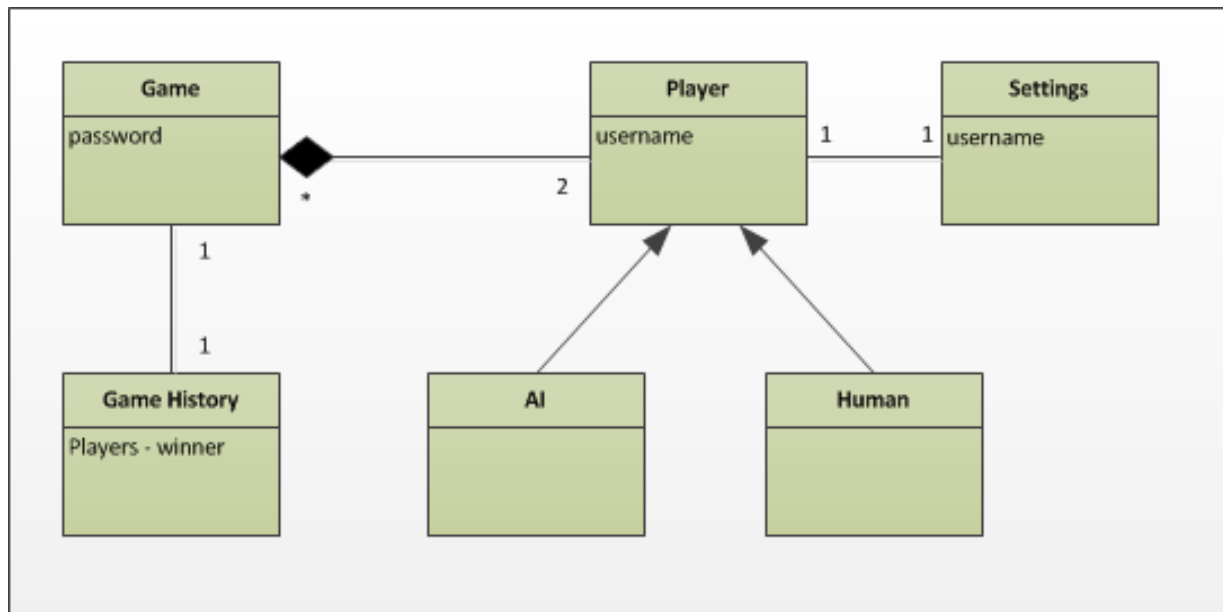
Scénario ordinaire

1. L'utilisateur clique sur « Statistiques du serveur » dans la fenêtre d'accueil principale.
2. La fenêtre UIServerStats s'affiche à l'écran.
3. L'utilisateur peut voir les IP locales et externes de son serveur, le mot de passe le protégeant, les joueurs se trouvant dans le serveur et l'historique des parties jouées.
4. L'utilisateur peut fermer la fenêtre en cliquant sur la croix en haut à droite.

Protocole Client/Serveur

- Connect
 - Se connecter au serveur, avec et sans authentification
- GetGameParameters
 - Récupère les informations serveur
- Beacons
 - 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.

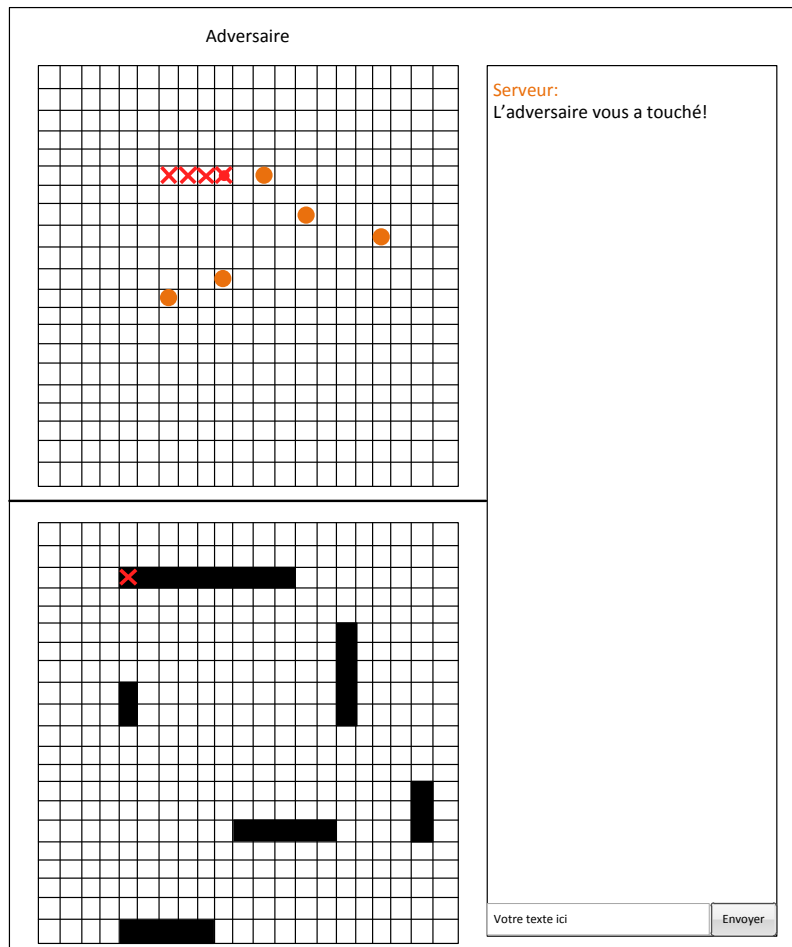
Le serveur de l'utilisateur ne pourra héberger qu'une seule partie à la fois. Si une partie a déjà été créée sur le serveur, alors il ne sera pas possible d'en créer une autre tant que la partie en cours n'est pas terminée. Si la création d'une nouvelle partie est impossible, alors le bouton correspondant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

Rappel des différents cas d'utilisation :

1. Jouer : Jeu principal. 2 façons de jouer :
 - a. Partie de 2 joueurs
 - b. Partie de 1 joueur vs IA
2. Connexion au serveur : Connexion au serveur de jeu pour jouer
3. Authentification : Authentification pour la connexion à un serveur protégé par mot de passe
4. Créer une partie sur un serveur : Créer une partie joignable sur un serveur.
5. Consulter les serveurs hébergeant une partie : Afficher la liste des serveurs.
6. Consulter les statistiques du serveur : Afficher les IP, mot de passe, joueurs actuellement connectés et historique des parties effectuées sur ce serveur
7. Attente de joueurs : Attente du premier joueur ou de l'adversaire
8. Modifier le pseudo : Définir ou modifier le pseudonyme de l'utilisateur
9. Chatter : Pour communiquer avec son adversaire
10. Stocker les statistiques de la partie : Stockage des statistiques

Plan d'itération initial :

Iteration	Dates	Objectives	En charge	Heures prévues	Heures effectives
1	29 Avril – 5 Mai	<u>Tâche 1 : Chatter</u> : Interface graphique (uniquement l'apparence) de UIGameMain, partie chat seule est réalisée. Lorsque l'on clique sur le bouton « Envoyer », ou que l'on tape « Enter », le message « Bouton Envoyé cliqué » s'affiche dans la console.	NLE YWR	3	
		<u>Tâche 2 : Créer un serveur</u> : Interface graphique (uniquement l'apparence) de UIServerCreation est réalisée. La fenêtre est déplaçable mais n'a pas de barre de titre. On peut cliquer sur le mode de jeu, les boutons radios réagissent et un message « Bouton radio [x] sélectionné » s'affiche dans la console. On peut cliquer	NLE	5	

		sur les checkbox, les checkbox réagissent et un message « Checkbox [x] sélectionnée/désélectionnée » s'affiche dans la console. On peut cliquer sur le bouton « Créer » ou taper « Enter » et un message « Bouton Créer cliqué » s'affiche dans la console.			
		<u>Tâche 3 : Attente de joueurs</u> : Interface graphique (uniquement l'apparence) de UIWaitPlayers est réalisée. La fenêtre est déplaçable mais n'a pas de barre de titre. Lorsque l'on clique sur le bouton « Déconnecter », le message « Bouton Déconnecter cliqué » s'affiche dans la console.	NLE	3	4
		<u>Tâche 4 : Authentification</u> : Interface graphique (uniquement l'apparence) de UIAskPass est réalisée. La fenêtre est déplaçable mais n'a pas de barre de titre. Lorsque l'on clique sur le bouton « Connecter » ou que l'on tape « Enter », le message « Bouton Connecter cliqué » s'affiche dans la console. Lorsque l'on clique sur le bouton « Annuler », le message « Bouton Annuler cliqué » s'affiche dans la console.	NLE	3	4
		<u>Tâche 5 : Créer un serveur, Consulter les statistiques du serveur, Afficher la liste des serveurs</u> : Interface graphique (uniquement l'apparence) de UIHome est réalisée. La fenêtre est déplaçable mais n'a pas de barre de titre. Lorsque l'on clique sur un bouton, le message « Bouton [x] cliqué » s'affiche dans la console.	NLE	3	5
		<u>Tâche 6 : Connexion au serveurs</u> : Interface graphique (uniquement l'apparence) de UIManualConnect est réalisée. La fenêtre est déplaçable mais n'a pas de barre de titre. Lorsque l'on clique sur un bouton, le message « Bouton [x] cliqué » s'affiche dans la console.	NLE	3	4
		<u>Tâche 7 : Modifier le pseudo</u> : Interface graphique (uniquement l'apparence) de UIUsernameConf est réalisée. La fenêtre est déplaçable mais n'a pas de barre de titre. Lorsque l'on clique sur un bouton, le message « Bouton [x] cliqué » s'affiche dans la console.	NLE	3	4
2	6 Mai	<u>Tâche 8 : Jouer, Chatter</u> : le Protocole	JAR	10	

	– 12 Mai	réseau (documentation complète) est défini dans un document texte décrivant son fonctionnement.			
		<u>Tâche 9 : Jouer</u> : Interface graphique (uniquement l'apparence) de UGameMain, partie Grille de jeu uniquement est réalisée. La fenêtre est déplaçable mais n'a pas de barre de titre.	NLE	5	
		<u>Tâche 10 : Consulter les serveurs</u> : Interface graphique (uniquement l'apparence) de UIServersListing est réalisée. La fenêtre est déplaçable mais n'a pas de barre de titre. Lorsque l'on clique sur un bouton, le message « Bouton [x] cliqué » s'affiche dans la console.	NLE	4	
		<u>Tâche 11 : Chatter</u> : Interface graphique (uniquement l'apparence) de la partie chat est intégrée à la fenêtre UGameMain. La fenêtre est déplaçable mais n'a pas de barre de titre. Lorsque l'on clique sur le bouton « Envoyer », ou que l'on tape « Enter », le message « Bouton Envoyé cliqué » s'affiche dans la console.	NLE YWR	3	
3	13 Mai – 19 Mai	<u>Tâche 12 : Jouer, Connexion au serveur, Authentification, Créer une partie sur un serveur, Consulter les serveurs hébergeant une partie, Consulter les statistiques du serveur, Attente de joueurs, Modifier le pseudo, Chatter, Stocker les statistiques de la partie</u> : Interactions des fenêtres et synchronisations. Les boutons ouvrant une fenêtre particulière fonctionnent et lorsqu'une fenêtre est refermée, on revient sur la bonne fenêtre.	JAR PNI	12	
		<u>Tâche 13 : Chatter</u> : Implémentation à sens unique. Un utilisateur peut envoyer des messages à un autre et l'autre recevra les messages qui s'afficheront dans la fenêtre de chat sans pouvoir répondre pour l'instant.	NLE YWR	10	
4	20 Mai – 26 Mai	<u>Tâche 14 : Jouer</u> : Placement en dur des bateaux. Les bateaux sont placés dans certaines cases précises définies en dur dans le programme. On voit que les bateaux sont placés sur la grille en fonction des paramètres indiqués dans le code.	BGL JAR NLE PNI	16	
		<u>Tâche 15 : Chatter</u> : Implémentation à deux sens. Le chat est complètement fonctionnel. Deux utilisateurs peuvent	YWR	6	

Projet Génie Logiciel Bataille Navale

		communiquer.			
5	27 Mai – 2 Juin	<u>Tâche 16 : Jouer</u> : 1 joueur solo tire tout seul sans adversaire et un autre client reçoit les coups. Les coups joués sont affichés dans la console.	BGL JAR PNI YWR	22	
6	3 Juin – 9 Juin	<u>Tâche 17 : Jouer</u> : Humain vs Humain : implementation complète. Le jeu à deux joueurs fonctionne en faisant des essais non structurés de parties.	BGL JAR PNI YWR	22	
7	10 Juin – 16 Juin	<u>Tâche 18 : Jouer</u> : Humain vs Humain : Protocoles de tests pour vérifier de manière systématique tous les cas de figure pour être certain que le jeu fonctionne correctement. Implémentation des Bonus terminés.	BGL	10	
		<u>Tâche 19 : Jouer</u> : vs IA : L'IA tire sur les 2 diagonales. Elle commence par tirer en haut à gauche, puis elle descend en diagonale jusqu'à en bas à droite et ensuite, elle recommence avec la diagonale qui commence en haut à droite pour finir en bas à gauche. Nous pouvons voir que les tirs arrivent correctement et dans le bon ordre.	JAR PNI YWR	6	
		<u>Tâche 20 : Jouer</u> : Intégration des effets sonores. Les tirs manqués font un son d'éclaboussure, les tirs réussis font un son d'explosion, un bateau complètement coulé fait un bruit de naufrage.	NLE	6	
8	17 Juin – 24 Juin	<u>Tâche 21 : Jouer</u> : vs IA : Implémentation finale. L'IA fonctionne et est intelligente et réagit correctement	JAR PNI YWR	10	
		<u>Tâche 22 : Jouer</u> : vs IA : Protocoles de tests pour vérifier de manière systématique tous les cas de figure pour être certain que le jeu fonctionne correctement.	YWR	6	
		<u>Tâche 23 : Rédaction de la documentation finale terminée. Le présent document est terminé et prêt à être rendu.</u>	NLE	6	

Historique des modifications des planifications :

À l'issue de l'itération	Modifications apportées	Motivations
1	Les tâches 1 et 2 ont été déplacées à l'itération 2	Le temps a été sous-estimé pour les tâches de l'itération 1. Il a donc manqué du temps pour terminer ces tâches. Elles ont donc été déplacées à l'itération suivante.
2		
3		
4		
5		
6		
7		
8		