# Cahier des charges

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)

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.

# Interfaces utilisateur

## UIHome

Cette fenêtre est la fenêtre s’ouvrant lorsque l’application est exécutée. L’utilisateur a le choix entre créer un serveur et consulter la liste des serveurs. Seul un serveur peut être créé par application.



## 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

## UIListingServers

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 :



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 :



## UIWaitPlayers

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



## 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 :



## UIConfigUsername

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



## UICreateServer

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



## 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.



# 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
    1. Serveur
    2. 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 Oberveur-observé)
* Stockage de la base de données en **XML**. Manipulation XML avec **XPath** uniquement (pas de XML à la main) avec la **librairie 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