Rapport de projet

Semestre de printemps 2015

WarTanks

Destinataire : M. Lefrançois

**Armand Delessert**

Benoit Zuckschwerdt

Simon Baehler

Ngueukam Djeuda Wilfried Karel

Contenu

[Introduction 3](#_Toc421886468)

[Vue d’ensemble 4](#_Toc421886469)

[Base de donnée 5](#_Toc421886470)

[Communication Réseau 5](#_Toc421886471)

[Protocole 5](#_Toc421886472)

[Le client 5](#_Toc421886473)

[Le serveur 5](#_Toc421886474)

[Implémentation 7](#_Toc421886475)

[Client 8](#_Toc421886476)

[Architecture 8](#_Toc421886477)

[List des classes 8](#_Toc421886478)

[En jeux 11](#_Toc421886479)

[Bonus 12](#_Toc421886480)

[Bonus speed 12](#_Toc421886481)

[Mine 12](#_Toc421886482)

[UltraRapideFire (non implémenté) 12](#_Toc421886483)

[Laser 12](#_Toc421886484)

[Foudroiement (non implémenté) 12](#_Toc421886485)

[Volée de lame 12](#_Toc421886486)

[Bélier (non implémenté) 12](#_Toc421886487)

[Alpha strick 13](#_Toc421886488)

[Soin 13](#_Toc421886489)

[La Mort 13](#_Toc421886490)

[Fenetre 13](#_Toc421886491)

# Introduction

Le but de ce projet de génie logiciel est de réaliser un jeu de tank en deux dimensions en vue de dessus. Le jeu est décomposé en deux parties : la première est le client, soit le jeu lui-même et la seconde est le serveur, il permet de connecter les joueurs entre eux. Il n’y a pas de compte, l’utilisateur lance le jeu en spécifiant l’IP du serveur.

Ce projet a été réalisé dans le cadre du cours GEN dispensé en 2ème année de l’HEIG-VD d’une durée d’un semestre à raison recommandée/estimée de 6 heures de travail par personne et par semaine, par les étudiants suivants :

* **Armand Delesert**
* Benoit Zuckswert
* Baehler Simon
* Ngueukam Djeuda Wilfried Karel

Le nom du chef du groupe est en gras.

Les fonctionnalités réalisées sont les suivantes :

USER CASE

Coter serveur

* Initialement
  + Choix du nombre de joueurs
  + Choix de la carte
  + Choix du temps de la partie
* Finalement
  + Le nombre de joueur est fixe
    - Il est néanmoins possible de changer ça directement dans le code
  + La carte est fixe
    - Il est néanmoins possible de changer ça directement dans le code
  + Il n’y a pas de temps limite de jeux
    - Rien de mis en place à ce niveau la

Coter client

* Initialement
  + Choix le serveur sur lequel se connecter
    - En tant que joueur
    - En tant qu’observateur de la partie
  + Si joueur
    - Avant le début de partie
      * Choisir son tank ayant de caractéristiques différentes
        + Point de vie
        + Vitesse
        + Couleur
    - Durant la partie
      * Tirer
      * Se déplacer
      * Ramasser des bonus
      * Utiliser les bonus
    - A la fin de la partie
      * Consulter les scores (automatique)
      * Relancé la partie
* Finalement
  + Choix le serveur sur lequel se connecter
    - En tant que joueur
  + Si joueur
    - Avant le début de partie
      * -
    - Durant la partie
      * Tirer
      * Se déplacer
      * Ramasser des bonus
      * Utiliser les bonus
    - A la fin de la partie
      * Consulter les scores (automatique)
      * Relancé la partie

# Vue d’ensemble

L’application se compose d’un serveur et de plusieurs clients (un seul est représenté dans l’illustration qui suit).

Le diagramme bloc présentant les principaux composants utilisés dans le cadre de ce projet est le suivant :

FAIRE UN AUTRE DIAGRAMME



Le serveur se compose des principaux éléments suivants :

L’application cliente se compose des principaux éléments suivants :

* La JVM exécutant le Java Runtime Environment 8 (JRE8) permettant d’exécuter l’application cliente écrite en Java
* L’application cliente
* Les sons et les images du jeu

# Base de donnée

# Communication Réseau

## Protocole

Lorsque l’on crée un protocole réseau, il est particulièrement nécessaire de faire attention à ce que toutes les implémentations de ce dernier soient interopérables. Nous avons donc porté un soin particulier à celui-ci en expliquant chaque cas possible et en donnant des exemples.

Il a été décidé de créer un protocole simple, permettant également le transport de données sous format xml.

Le client et le serveur gèrent tous deux :

* L’envoi de messages (avec contenu binaire additionnel éventuel)
* La réception de messages (avec contenu binaire additionnel éventuel)
* La détection de *timeout* réseau
* La déconnexion

## Le client

* La connexion au serveur
* L’envoie des actions du joueurs (pression sur les touches)
  + Déplacement
  + Tire
* L’envoie du désir de refaire une partie
* Reçoit du serveur les mises à jour du plan de jeu et les affiche à l’écran
  + Position des ennemis
  + Tire des ennemis
* Reçoit du serveur l’état du jeu
  + Points de vie des autres joueurs
* Reçoit du serveur le tableau des scores à la fin de la partie.

## Le serveur

* La connexion des clients
* La notification si les 2 joueurs sont connectés
* L’état des autres joueurs
  + Points de vie
* Reçoit du client les commandes du joueur
  + Déplacement -> Position
  + Tire
* La transmission des actions des joueurs
  + Déplacement -> Position
  + Tire
* L’envoie de score en fin de partie
* La réception du désir de relancé une partie
  + Si les deux joueurs veulent rejouer on relance une partie
  + Si non on attends deux joueurs

Chaque message échangé entre le serveur et le client :

* Peut se sérialiser en XML
* Peut se désérialiser depuis un contenu XML
* Spécifie quel est le contenu binaire supplémentaire à envoyer et comment le recevoir

## Implémentation

L’envoi d’un message débute par convertir celui-ci en sa représentation XML par sérialisation. Le message est ensuite envoyé au serveur. Une fois le message envoyé, le contenu binaire additionnel est envoyé au serveur.

# Client

Notre implémentation possède deux facettes : la partie du client et la partie du serveur. Pour le client nous avons utilisé le moteur graphique 2D nommée skick2D ainsi qu’une foison d’images, tuiles et sprites, la charge de travail pour la création d’image par notre main jugée trop conséquente. Les différentes images ainsi que leurs source et décrite plus bas dans la documentation.

## Architecture

Le diagramme UML de notre projet se d’un point de vue client ce présente de la manière suivante. Il n’est démonté ici que la partie sans infrastructure réseau.

### List des classes

**Game**

Il s’agit de la classe principale qui va lier tous les éléments du jeu. Sa tâche est principalement de créer les objets et de les initialiser. Elle gère aussi toute la partie collision, la pression de touche, et le mouvement de la caméra.

Elle s’occupe de créer et d’initialiser les autres classes comme

* Au travers de la factory
  + Le joueur
  + Les ennemies du joueur
  + Les bonus
* Les scores
* La map
* La HUD du joueur
* L’affichage de fin
  + Victoire
  + Défaite

**End**

Cette classe est la super-classe des deux sous classe qui sont appelé en fin de partie. Il s’agit d’une classe relativement simple ayant pour but d’afficher l’encan de défait ou de victoire. Et si le temps l’avait permis l’affichage des score. Ses sous classe sont

* Vitctory
* Defeat

**HUD**

Cette classe est l’observateur de la classe Player, cette dernière est notifiée quand le joueur passe sur un bonus afin de mettre à jour la barre d’action du joueur. Elle aurait aussi du afficher les point de vie du joueurs, mais faute de temps cette possibilité a été abandonnée au profit d’autre tache

**Factory**

Cette classe est utilisée par la classe Game, elle est chargée de créer les objets de jeux comme les joueurs, les ennemies et les bonus.

**Bonus**

Cette classe permet la création de bonus sur la map. Une instance de cette objet et crée quand on l’ajout dans la map mais également quand un joueur roule sur le bonus une instance de cette objet est créé dans la classe Player.

**IFightable**

L’interface Ifightable implémente nos protagonistes joueurs et ennemy, ils utilisent tous les deux une classe Score qui contient leurs score respectif. Globalement ces deux classe sont pareil à la différence près que Player est observé par la HUD crée par Game et que le joueur et représenté en bleu et Ennemy en rouge. Ce deux classe créent un objet Explosion a leurs mort.

**Bullet**

La classe bullet est la super-classe des trois différents types de munition/projectile qui sont :

* AlphaStrick
* Mine
* Laser

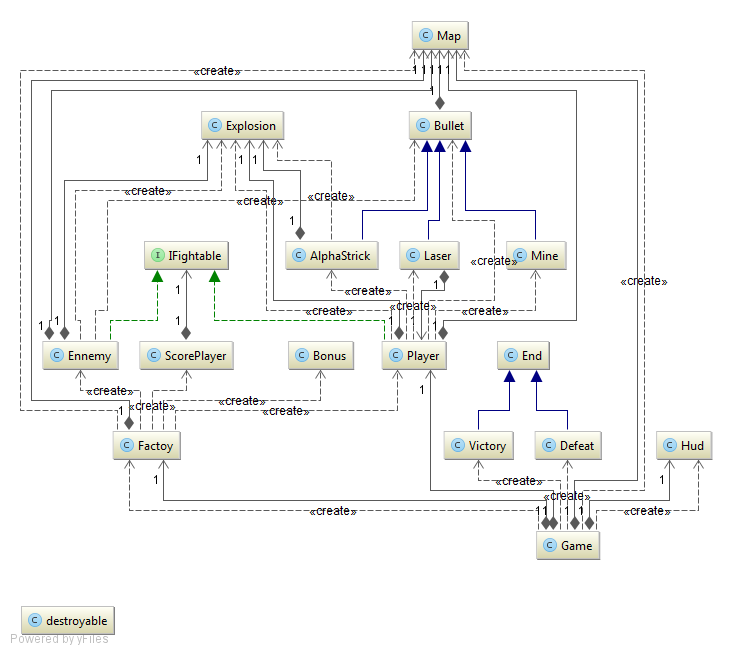
Ses projectiles sont décrits plus loin dans la section dédiée au bonus.

**Explosion**

Il s’agit d’une classe relativement simple, elle n’a pour tâche que de gérer l’animation des explosions qui sont appelée quand un joueur meurt ou quand le bonus AlphaStrick est lancé

**Map**

Cette classe gère la génération de la map avec les collisions entre les bords de maps et les object-obstacles

FAIRE UN DIAGRAME D’ACTIVITE

# En jeux

Une fois un serveur rejoint et le nombre de joueurs demandés pour commencer une partie attient, la partie peut commencer. La fenêtre de jeux dispose des éléments suivants

* Le tank du joueur en bleu
* Le/les tanks ennemi en rouge
  + En noir si détruit
* La HUD
  + Bloc en haut à gauche
  + Barre de bonus
* Les bonus disponible sur le sol
* Les projectiles (décrit plus loin)
  + Projectile simple
  + Mine
  + Laser
  + Attaque alpha



# Bonus

Dans notre jeux le joueur peux ramasser des objets, dans le cas où ils ces bonus sont disponible (munition supérieur à 0) il peut les utiliser. Voici un listing des différents bonus.

## Bonus speed

* Image :C:\Users\Simon\Documents\2em\GEN\projet\WarTanks\WarTanks\WarTanksGUI\src\ressources\UI\bonus\1.bmp
* Quantité : 3
* Capacité offerte : Augmente la vitesse de déplacement du tank pour une durée de 10 secondes

## Mine

* Image : C:\Users\Simon\Documents\2em\GEN\projet\WarTanks\WarTanks\WarTanksGUI\src\ressources\UI\bonus\2.bmp
* Quantité : 3
* Capacité offert : dépose une mine sur la position courante représentée par l’image suivante C:\Users\Simon\Documents\2em\GEN\projet\WarTanks\WarTanks\WarTanksGUI\src\ressources\bullet\2.png

## Ultra Rapide Fire (non implémenté)

* Image : C:\Users\Simon\Documents\2em\GEN\projet\WarTanks\WarTanks\WarTanksGUI\src\ressources\UI\bonus\3.bmp
* Quantité 3
* Capacité offert : Permet de tirer plus rapidement. De basse le tire simple ne doit pas être enchainée

## Laser

* Image :C:\Users\Simon\Documents\2em\GEN\projet\WarTanks\WarTanks\WarTanksGUI\src\ressources\UI\bonus\4.bmp
* Quantité : 3
* Capacité offerte : Créer un laser d’une taille de 32x200 devant le joueur blessant tous les joueurs pris dans le laser. Le laser est représenté par l’image suivante :C:\Users\Simon\Documents\2em\GEN\projet\WarTanks\WarTanks\WarTanksGUI\src\ressources\bullet\laser.png

## Foudroiement (non implémenté)

* Image : C:\Users\Simon\Documents\2em\GEN\projet\WarTanks\WarTanks\WarTanksGUI\src\ressources\UI\bonus\5.bmp
* Quantité : 3
* Capacité offerte : foudroie une zone d’un rayon de 32px autour du joueur toutes les secondes durant 3 secondes, blessant les joueurs pris dans la zone

## Volée de lame

* Image :C:\Users\Simon\Documents\2em\GEN\projet\WarTanks\WarTanks\WarTanksGUI\src\ressources\UI\bonus\6.bmp
* Quantité : 3
* Capacité offerte : tire un bullet dans les quatre directions

## Bélier (non implémenté)

* Image :C:\Users\Simon\Documents\2em\GEN\projet\WarTanks\WarTanks\WarTanksGUI\src\ressources\UI\bonus\7.bmp
* Quantité : 3
* Capacité offerte : Fixe de piques sur l’avant du tank, si le joueur entre en collision avec l’avant de son tank dans les 10 secondes qui suivent l’activation du bonus, le tank adverse subit un point de dégât.

## Alpha Strick

* Image : C:\Users\Simon\Documents\2em\GEN\projet\WarTanks\WarTanks\WarTanksGUI\src\ressources\UI\bonus\8.bmp
* Quantité : 3
* Capacité offerte : Effectue une attaque sur une zone de 64x64 devant le tank, les joueurs dans la zone prennent un point de dégât. Une animation est déclenchée à l’impact.

C:\Users\Simon\Documents\2em\GEN\projet\WarTanks\WarTanks\WarTanksGUI\src\ressources\bullet\explosion1.png

## Soin

* Image : C:\Users\Simon\Documents\2em\GEN\projet\WarTanks\WarTanks\WarTanksGUI\src\ressources\UI\bonus\9.bmp
* Quantité : 3
* Capacité offerte : ajoute un point de vie au joueur

## La Mort

* Image : C:\Users\Simon\Documents\2em\GEN\projet\WarTanks\WarTanks\WarTanksGUI\src\ressources\UI\bonus\10.bmp
* Quantité : 3
* Capacité offerte : inflige un point de dégât à un joueur au hasard

# Fenêtre

Décrire ici la différente fenêtre et ce qu’elles permettent de faire