ASD Tower Defense

Version 2.0 – En réseau

Rapport Intermédiaire

2010

GEN

HEIG-VD

01/01/2010

Aurélien Da Campo

Lazhar Farjallah

Pierre-Dominique Putallaz

Romain Poulain

Table des matières

[1. Introduction [OK] 2](#_Toc260998076)

[2. Règles du jeu [OK] 4](#_Toc260998077)

[Les créatures 4](#_Toc260998078)

[Les tours 4](#_Toc260998079)

[Les modes de jeu 4](#_Toc260998080)

[3. Etapes de lancement du jeu 5](#_Toc260998081)

[4. Cas d’utilisation 5](#_Toc260998082)

[5. Protocoles d’échange 5](#_Toc260998083)

[Serveur d’enregistrement 5](#_Toc260998084)

[Serveur de jeu 5](#_Toc260998085)

[6. Diagramme UML de classes [OK] 6](#_Toc260998086)

[Model du serveur 6](#_Toc260998087)

[Model du Client 7](#_Toc260998088)

[7. Charte graphique [OK] 8](#_Toc260998089)

[8. Rôle des participant au sein du groupe [OK] 10](#_Toc260998090)

[9. Itérations [OK] 11](#_Toc260998091)

[Itération 1 – Serveur d’enregistrement + Interface graphique 11](#_Toc260998092)

[Itération 2 – Serveur de Jeu + Architecture 12](#_Toc260998093)

[Itération 3 –Intégration du serveur de jeu + Interface du Jeu en réseau 12](#_Toc260998094)

[Itération 4 – Lifting de la GUI + Game Design + Amélioration Mode Solo 13](#_Toc260998095)

[Itération 5 – Serveur Web (facultatif) 13](#_Toc260998096)

[10. Conclusion [OK] 14](#_Toc260998097)

# Introduction [OK]

Ce projet prend place durant notre 4e semestre aux seins de la Haute Ecole d’Ingénierie et de Gestion du canton de Vaud (heig-vd). Ce cours de Génie logiciel (GEN) nous propose de mettre en pratique les notions théoriques acquises en créant une application de type Client/ Serveur.

Nous avons tout de suite pensée à l’amélioration d’un jeu que nous avions créé durant nos cours d’Algorithmes et Structures de Données (ASD2) suivi durant notre 3e semestre. En effet, nous avions réalisé un jeu et ce nouveau projet est pour nous l’opportunité d’étendre ce logiciel en lui fournissant des fonctionnalités réseau.

La gestion de ce projet se basera sur la méthode UP. Ce document vous présente le rapport intermédiaire du projet.

# Règles du jeu [OK]

Le principe du jeu est de survivre à diverses créatures dont la seule capacité est d’avancer le long du chemin le plus court pour rallier leur point de départ à la zone de fin. Lorsqu’une créature atteint la zone de fin, elle fait perdre une vie au joueur. Lorsque le joueur n’a plus de vie, il a perdu.

## Les créatures

Toutes les créatures prennent le chemin le plus court depuis leur emplacement jusqu’à la zone de fin. Les créatures terrestres doivent contourner les murs et les tours. Les créatures volantes peuvent survoler les tours.

## Les tours

Pour se défendre, le joueur peut acheter des tours qu’il place sur son plateau. Certaines tours font des dégâts (ciblés ou de zone) d’autres permettent seulement de ralentir les créatures. Certaines ne sont efficaces que sur un certain type de créature.

Les contraintes pour la construction d’une tour est de ne pas la créer là ou se trouve une créature à ce moment et il doit toujours y avoir un chemin entre la zone de départ et celle de fin.

## Les modes de jeu

Le jeu, dans sa première version, était uniquement local et donc proposait uniquement un mode solo ou le joueur survivait au vagues de créatures lancées par l’ordinateur. L’intégration de fonctionnalités réseau permettent son extension à un jeu multi-joueurs.

Une première analyse nous a permis de déterminer plusieurs types de partie dont voici quelques exemples :

* **Coopération**: plusieurs joueurs s’allient contre l’intelligence artificielle et jouent sur le même plateau.
* **Coopération  zone** : Idem que coopération mais chaque joueur possède une zone du plateau de jeu partagé et ne peut bâtir des tours que dans celle-ci.
* **Versus** : chaque joueur possède son propre plateau et joue seul contre tous les autres. Il gagne de l’argent périodiquement et en tuant des créatures. Pour augmenter son revenu périodique, il peut acheter des créatures qu’il envoie chez l’ennemi.
* **Domination** : un joueur est désigné pour se battre contre les autres. Le joueur seul a des caractéristiques et des bonus de meilleure qualité que les autres mais est handicapé par sa vitesse d’exécution des opérations réduite.

Nous avons retenu le mode versus uniquement car les délais ne permettent pas de les faire tous. Ce mode est inspiré d’une extension de Warcraft III (de Blizzard Entertainment) qui est la pionnière en matière de « tower defense ».

# Etapes de lancement du jeu

# Cas d’utilisation

# Protocoles d’échange

## Serveur d’enregistrement

PETIT TEXTE DE PRESENTATION

*Vous découvrirez le protocole du serveur d’enregistrement en annexe.*

## Serveur de jeu

Le protocole du serveur de jeu décrit les échanges entre le client et le serveur concernant les interactions de jeu. Le client va envoyer des changements d’états ainsi que des demandes de création d’objet, demandes auxquelles le serveur va donner une confirmation en cas de réussite ou une erreur en cas d’échec.

Le serveur quant à lui va envoyer à tous les clients l’état courant de l’ensemble des objets, qu’ils aient été crées par le client cible ou d’autre clients.

Le schéma de protocole ainsi que les scénarios d’utilisations associés sont présentés en annexe « Serveur de jeu, dialogue client/serveur JSON ». Une checklist ainsi qu’un bref descriptif des services est également inclus dans le document.

# Diagramme UML de classes [OK]

Nous avons récupérer nos anciens diagrammes de classes pour les améliorer considérablement afin qu’ils introduisent les particularités de ce travail. Nous pensons que les schémas les plus intéressants dans ce projet sont les Models (MVC), c’est pourquoi nous dans ce rapport intermédiaire, nous vous présentons uniquement ceux-ci. Les Vues et Contrôleurs seront présenté dans la version final du document.

## Model du serveur

Voici le diagramme de classes représentant la partie model (M) du serveur en termes d’architecture MVC.

|  |  |
| --- | --- |
| C:\Users\Dark\Desktop\Class Model - Server.png  Ce schéma de plus grande taille est fourni en annexe. | La classe **Jeu\_Serveur** est la classe maîtresse de ce model. Elle contient et fourni trois gestionnaires et le terrain.  Chaque gestionnaire encapsule et gère toutes les entités d’un type d’élément du terrain (**Tour**, **Créature**, **Animation**). Une tâche sera créée spécialement pour la gestion de ses éléments.  Une **tour** appartient à un joueur. Un **joueur** construit des tours dans un **emplacement** et celui-ci appartient à une **équipe**.  C’est le **terrain** qui fourni les **équipes** initiales et les **vague de créature**s du mode solo. Le terrain contient le **maillage** pour les déplacements des créatures. Ce dernier est détaillé dans la version 1.0 du jeu. (c.f. rapport ASD-TD v1\_0).  Une **créature** à été lancée par une équipe via une **vague de créatures**. En mode versus les vagues de créatures seront construites avec l’interface graphique.  Les **animations** sont des éléments divers pouvant être affiché sur le terrain. Elles sont par exemple créées par la mort d’une créature pour indiquer le gain d’argent. Les **attaques** sont des animations créées par les tours (flèche, boulet, etc.). C’est elles qui blessent les **créatures**.  La **fenêtre de jeu** implémentera les diverses interfaces de se model pour se tenir au courant des modifications du model. |

## Model du Client

Voici le diagramme de classes représentant la partie model (M) du client en termes d’architecture MVC.

|  |  |
| --- | --- |
| Class Model - Client.png | Le rôle du client et très léger, il s’occupe simplement d’afficher et de maintenir les informations reçues du serveur de jeu. De plus, les éléments qu’il possède lui permettent d’empêcher le client de faire des demandes inutiles au serveur. Par exemple, la demande de pose d’une tour sur un mur ne sera pas envoyée au serveur mais le client empêchera directement cette action (via l’interface) grâce aux informations qu’il contient. De toute façon toutes les requêtes reçus par le serveur seront contrôlées avant de modifier le model du serveur. |

# Charte graphique [OK]

Nous présentons ici nos premières idées de l’élaboration des interfaces utilisateurs de notre jeu. Nous nous concentrerons principalement sur la partie réseau.

|  |  |
| --- | --- |
| D:\Developpement\Java\WorkSpace\ASDTowerDefense\documents\v2.0\charteGraphique2.png | **Menu principal**  Premier menu lors du lancement du jeu. Trois choix sont offerts au joueur :  **1. Partie solo**  **2. Créer une partie multijoueurs**  **3. Rejoindre une partie multijoueurs** |
| **1. Mode solo**  Cette fenêtre permet au joueur de faire une partie seul. 4 terrains s’offre à lui. Il en sélectionne un et la partie commence.  Note : La fenêtre de jeu solo ne sera pas présentée dans se rapport | D:\Developpement\Java\WorkSpace\ASDTowerDefense\documents\v2.0\interface_ModeSolo.png |
|  | **2. Créer une partie réseau**  Le joueur introduit les caractéristiques du jeu (terrain, équipes aléatoires, etc.) et créer une partie réseau. |
| **2.1 Attente de joueurs**  La partie est créée, il faut maintenant attendre les joueurs. Le joueur qui créer la partie est considéré comme un modérateur de la partie en question. Il peut notamment changer la composition des équipes et démarrer le jeu à tous moment. |  |
|  | **3. Rejoindre une partie réseau**  Plaçons nous maintenant du coté du client qui veux se connecter. Si le serveur d’enregistrement est atteignable, le client verra la liste des serveurs disponibles avec leurs caractéristiques.  Le client sélectionne une partie et la rejoint… |
| **3.1 Attente de joueurs**  …Le client arrive alors dans le formulaire d’attente d’autres joueurs où il peut voir les joueurs déjà connectés. Il peut changer d’équipe si de la place est disponible. Il attend ensuite que tout le monde soit connecté ou que le modérateur décide de lancer la partie.  La partie commence… |  |
| **4. Jeu multi-joueurs en mode Versus**    Voici la fenêtre de jeu réseau. Grâce aux divers menus, le joueur pourra entre autre créer et gérer des tours, voir des informations sur les créatures (Box Sélection) et envoyer des vagues de créatures à l’équipe adverse. Un petit chat fera peut-être son apparition si le temps le permet (à priori non planifié dans les itérations). | |

# Rôle des participant au sein du groupe [OK]

Voici un tableau présentant les différents rôles (standard) des membres du groupe.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Aurélien** | **Lazhar** | **Pierre-Do.** | **Romain** |
| **Représentants des utilisateurs** |  | 🞮 |  | 🞮 |
| **Chef de projet** | 🞮 |  |  |  |
| **Analyste** | 🞮 |  |  | 🞮 |
| **Architecte, concepteur** | 🞮 | 🞮 | 🞮 | 🞮 |
| **Programmeur** | 🞮 | 🞮 | 🞮 | 🞮 |
| **Responsable des tests** |  |  | 🞮 |  |
| **Responsable de la configuration** |  | 🞮 |  |  |

# Itérations [OK]

## Itération 1 – Serveur d’enregistrement + Interface graphique

**Durée : 1 semaine – 30 avril 2010 au 7 mai 2010**

**Implémenter complètement la partie Serveur d’enregistrement ainsi que son interface graphique.**

Le serveur d’enregistrement permet d’enregistrer les serveurs de jeu sur un serveur central afin de fournir la liste de ces serveurs aux clients. Ces derniers peuvent ensuite choisir la partie qu’ils veulent rejoindre.

**Pourquoi cette itération prend-t-elle place ici ?**

C’est une petite partie fournissant une bonne introduction à la notion de communication CLT-SRV notamment pour les points suivants :

- Création de notre premier protocole réseau (fixation des standards)

- Première communication CLT-SRV en Java (Création des classes de base)

- Intégration du tout dans une interface graphique cohérente. (Schéma d’interface)

**Résultat attendu :** le client peut enregistrer ses parties sur le serveur d’enregistrement et voir la liste de toutes les parties en attente de joueur(s) depuis une interface graphique.

|  |  |  |
| --- | --- | --- |
| **Fonctionnalités attendues** | | |
| **Ok** | **Responsable(s)** | **Fonctionnalité** |
| 🗆 | Lazhar | Etablir une connexion client / serveur avec échange de message |
| 🗆 | Lazhar | Enregistrer une partie sur le serveur d’enregistrement. |
| 🗆 | Lazhar | Voir les parties inscrites sur le serveur d’enregistrement |
| 🗆 | Lazhar | « Désenregistrer » d’une partie sur le serveur d’enregistrement. |
| 🗆 | Lazhar | Mettre à jour les informations d’une partie. |
| 🗆 | Aurélien | Interface graphique pour l’enregistrement de la partie |
| 🗆 | Aurélien | Interface graphique pour voir les parties inscrites |

## Itération 2 – Serveur de Jeu + Architecture

**Durée :** 1 semaine - 7 mai 2010 au 14 mai 2010

**Pourquoi cette itération prend-t-elle place ici ?**

On prépare tous les éléments pour les fusionner ensuite (avec adaptations quasis certaines)

**Création de l’application client / serveur pour le jeu. En parallèle nous commencerons la restructuration de l’architecture pour correspondre à un jeu multi-joueurs.**

Il s’agit de mettre place (sans interface) une communication entre un joueur et le serveur de Jeu.

**Résultat :** Un protocole de communication mis en place pour l’échange de message ente le client et le serveur de jeu. Concernant la restructuration, on attend un mode 1 joueur avec exactement les mêmes fonctionnalités mais avec une architecture beaucoup plus propre.

|  |  |  |
| --- | --- | --- |
| **Fonctionnalités Attendues** | | |
| **Ok** | **Responsable(s)** | **Fonctionnalité** |
| 🗆 | Pierre-Dominique & Romain | Implémentation de tous les messages fournis par le protocole. (liste trop exhaustive pour les citer tous, *se référer au protocole en annexe*) |
| 🗆 | Aurélien | Le mode 1 joueur fonctionne correctement et comme avant. Présentation du schéma de classe ou de domaine. |

## Itération 3 –Intégration du serveur de jeu + Interface du Jeu en réseau

**Durée :** 2 semaines - 14 mai 2010 au 28 mai 2010

**Intégration du serveur dans l’architecture et le jeu fonctionne.**

Le serveur de jeu devra être intégré à l’architecture de l’application (au noyau du jeu). Le client et le serveur pourront alors interagir avec le modèle (point de vue MVC) du jeu. L’interface du jeu permettra d’illustrer ces changements.

**Pourquoi cette itération prend-t-elle place ici ?**

Il est temps de faire fusionner tous les éléments et en faire un programme plus cohérent.

**Résultat :** A la fin de cette itération, le jeu doit fonctionner et tous les messages transitant entre le client et le serveur doivent être correctement traités pas l’entité réceptrice.

|  |  |  |
| --- | --- | --- |
| **Fonctionnalités Attendues** | | |
| **Ok** | **Responsable(s)** | **Fonctionnalité** |
| 🗆 | Aurélien | L’interface permet de solliciter des actions du jeu. |
| 🗆 | Aurélien | Le client peut se connecter à une partie de jeu. |
| 🗆 | Pierre-Dominique & Romain | Le serveur et le client interprètent les messages réseau et modifie correctement le model. Ceci est visible grâce au changement du terrain de jeu. |

## Itération 4 – Lifting de la GUI + Game Design + Amélioration Mode Solo

**Durée :** 1 semaine - 28 mai 2010 au 4 juin 2010

**Revoir le design et faire de notre logiciel un « vrai » jeu vidéo**

Actuellement, au niveau de l’interface graphique, notre jeu ressemble plus un à logiciel applicatif qu’a un jeu vidéo. Nous aimerions dans cette itération rendre notre jeu plus attractif en créant une interface plus agréable. Il serait aussi intéressant de concevoir nos propre ressources (images / sons / etc.) car actuellement, une bonne partie de nos images sont reprises d’autres jeux.

Une bonne chose serait de revoir également toutes les valeurs liées au jeu pour le rendre plus « jouable » (*level design*). Cette partie peut paraître bénigne mais elle est cruciale et très complexe pour ce genre de jeu car il y a énormément d’éléments qui influencent la durée de vie du joueur.

Nous voulons également implémenter un système de progression dans le mode solo pour que le joueur ressente l’envie de finir complètement le jeu.

**Résultat :** Un programme plus esthétique, plus jouable et avec un système de progression.

|  |  |  |
| --- | --- | --- |
| **Fonctionnalités Attendues** | | |
| **Ok** | **Responsable(s)** | **Fonctionnalité** |
| 🗆 | Lazhar | Lifting de l’interface, celle-ci ressemble plus à un jeu. |
| 🗆 | Pierre-Dominique & Romain | Adaptation des valeurs, jeu plus agréable. |
| 🗆 | Aurélien | Système de progression mise en place. |

## Itération 5 – Serveur Web (facultatif)

**Durée :** moins d’une semaine - 4 juin 2010 au 9 juin 2010

**Serveur web de stockage des meilleurs scores**

Il s’agit de mettre en place un serveur de web fournissant un service web de sauvegarde et récupération des meilleurs scores pour les différents terrains de jeu. Le but étant de motiver le joueur à s’améliorer (il doit avoir envie de rejouer le plus souvent possible).

**Résultat :** Un système permettant de sauver et voir les meilleurs scores de tous les joueurs (du monde).

|  |  |  |
| --- | --- | --- |
| **Fonctionnalités Attendues** | | |
| **Ok** | **Responsable(s)** | **Fonctionnalité** |
| 🗆 | Romain | Mise en place du serveur web |
| 🗆 | Aurélien | Intégration du service dans l’application |

# Conclusion [OK]

Le projet avance bien, nous avons maintenant une bonne idée des choses restantes à faire et nous sommes impatients de pouvoir mettre en œuvre tous les artefacts créés dans ce rapport intermédiaire.