Tower Defense

Mise en œuvre d’un graphe interactif

Création d’une variante du célèbre jeu Tower Defense dans le cadre d’un cours d’algorithmique.

ASD2

Lazhar Farjallah / Aurélien Da Campo / Pierre-Dominique Putallaz

Heig-vd

ASD

TD

Table des matières

[1 Analyse préliminaire 4](#_Toc251182353)

[1.1 Introduction [LAZHAR] 4](#_Toc251182354)

[1.2 Organisation [LAZHAR] 4](#_Toc251182355)

[1.3 Objectifs [LAZHAR] 5](#_Toc251182356)

[1.4 Planification initiale [AURELIEN.OK] 7](#_Toc251182357)

[2 Analyse 8](#_Toc251182358)

[2.1 Concepts algorithmique [LAZHAR] 8](#_Toc251182359)

[2.2 Etude concurrentielle et rétrospective [AURELIEN.OK] 8](#_Toc251182360)

[2.2.1 Bref historique 8](#_Toc251182361)

[Détails des plus célèbres jeux « Tower Defense » 9](#_Toc251182362)

[2.2.2 Essor actuel sur l’Internet et avenir 11](#_Toc251182363)

[2.3 Etude de faisabilité [AURELIEN.OK] 12](#_Toc251182364)

[2.3.1 Risques techniques 12](#_Toc251182365)

[2.3.2 Risques concernant le planning & ressources humaines 13](#_Toc251182366)

[2.3.3 Risques concernant le budget. 13](#_Toc251182367)

[3 Conception 14](#_Toc251182368)

[3.1 Dossier de conception 14](#_Toc251182369)

[3.1.1 Systèmes d’exploitation [LAZHAR] 14](#_Toc251182370)

[3.1.2 Outils logiciels [LAZHAR] 14](#_Toc251182371)

[3.1.3 Librairies externes [LAZHAR] 14](#_Toc251182372)

[3.1.4 Interface graphique [AURELIEN] 14](#_Toc251182373)

[3.1.5 Architecture de l’application [AURELIEN.OK] 16](#_Toc251182374)

[3.1.6 Schémas UML [PIERRE-DO] et [AURELIEN] 19](#_Toc251182375)

[3.1.7 Gestion de la concurrence [PIERRE-DO] 19](#_Toc251182376)

[4 Réalisation 20](#_Toc251182377)

[4.1 Dossier de réalisation 20](#_Toc251182378)

[4.1.1 Résultat de l’interface graphique [AURELIEN] 20](#_Toc251182379)

[4.1.2 Résultat de l’implémentation du maillage [PIERRE-DO] 20](#_Toc251182380)

[4.2 Description des tests effectués [LAZHAR] 20](#_Toc251182381)

[4.3 Erreurs restantes [LAZHAR] 20](#_Toc251182382)

[5 Conclusions *[LAZHAR]* 20](#_Toc251182383)

[5.1 Objectifs atteints / non-atteints 20](#_Toc251182384)

[5.2 Points positifs / négatifs 20](#_Toc251182385)

[5.3 Difficultés particulières 20](#_Toc251182386)

[5.4 Avenir du projet 21](#_Toc251182387)

[6 Annexes 22](#_Toc251182388)

[6.1 Sources – Bibliographie [LAZHAR] 22](#_Toc251182389)

[6.2 Journal de bord de chaque participant [LAZHAR] 22](#_Toc251182390)

[6.3 Manuel de TDA maillage [PIERRE-DO] 22](#_Toc251182391)

[6.4 Manuel d'Utilisation [LAZHAR] 22](#_Toc251182392)

[6.5 Archives du projet [LAZHAR] 22](#_Toc251182393)

# Analyse préliminaire

## Introduction [LAZHAR]

Ce projet de fin de semestre consiste à créer une application ludique mettant en œuvre des algorithmes et structures de données étudiées en cours. Nous avons choisi pour cela de créer une variante du célèbre jeu « Tower Defense », dans lequel des personnages se déplacent d’un point A à un point B selon un chemin optimal. En effet, on aura pour cela besoin d’une structure de graphe ainsi que les algorithmes associés, ce qui colle parfaitement avec la contrainte de départ car nous les avons étudiés en cours.

Le but est également ici de créer une application « didacticiel » qui sera présentée dans les futurs cours de cette unité d’enseignement. Ce projet permettra en effet de montrer une application réelle de l’utilisation d’algorithmes associés à des graphes (en particulier celui de recherche du chemin le plus court entre deux nœuds).

Nous commencerons par effectuer une analyse du projet, notamment en ce qui concerne l’organisation, les objectifs ainsi que la planification initiale. Puis nous détaillerons les étapes de conception et de réalisation. Enfin, avant de conclure, nous présenterons les fonctionnalités de l’application finale sous forme de mode d’emploi et nous présenterons des captures d’écran.

## Organisation [LAZHAR]

Les membres participant à ce projet sont :

**Etudiant 1** :

Aurélien Da campo, *aurelien.dacampo@heig-vd.ch*

**Etudiant** **2** :

Pierre-Dominique Putallaz, *pierre-dominique.putallaz@heig-vd.ch*

**Etudiant 3 (responsable de projet)** :

Lazhar Farjallah, *lazhar.farjallah@heig-vd.ch*

Nous prévoyons de nous répartir les tâches de manière suivante entre chaque entité du groupe :

1. *Responsable de projet*
   1. Rédactions, administrations
   2. Suivi des rendus (*deadlines*)
   3. Surveillance et coordination
   4. Développement
2. *Etudiant 1*
   1. Création de l’interface graphique
   2. Gestion de l’affichage
   3. Rendu graphique
   4. Interaction avec l’utilisateur
3. *Etudiant 2* 
   1. Algorithmique
   2. Implémentation des algorithmes de graphe
   3. Fournir les briques logicielles pour permettre la construction de la partie fonctionnelle de l’application

## Objectifs [LAZHAR]

Les objectifs de ce projet sont les suivants :

* Illustrer le concept de graphe de manière ludique et interactive.
* Acquérir de l’expérience dans la planification et l’accomplissement d’un projet conséquent.
* Utiliser et découvrir des librairies existantes implémentant le concept de graphe
* Apprendre à mettre en œuvre une interface graphique en Java.
* Séparer le travail en plusieurs niveaux d’abstraction pour faciliter l’élaboration et l’évolutivité de ce projet.
* Comprendre la nécessité d’utiliser des algorithmes complexes dans les applications informatiques.
* Mettre en œuvre un algorithme de recherche de chemin le plus court (ACPC).
* Respecter le design pattern MVC (*Model* – *View* – *Controller*) qui structure un programme en trois couches principales.

## Planification initiale [AURELIEN.OK]

Le projet se déroulera du **18 novembre 2009 au 15 janvier 2010**, ce qui représente un total de 25 périodes en classe (par personne). Nous prévoyons également de passer un total d’environ au moins 25 périodes par personne en dehors des heures encadrées. **Au total, c’est environ 150 périodes de travail** que nous allons planifier comme suit :

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **nov.09** | | **déc.09** | | | | | **janv.10** | | |
| **Taches** | **18** | **25** | **2** | **9** | **16** | **23** | **30** | **6** | **13** | **15** |
| Analyse |  |  |  |  |  |  |  |  |  |  |
| Conception |  |  |  |  |  |  |  |  |  |  |
| Réalisation |  |  |  |  |  |  |  |  |  |  |
| Tests |  |  |  |  |  |  |  |  |  |  |
| Documentation |  |  |  |  |  |  |  |  |  |  |
| Planification |  |  |  |  |  |  |  |  |  |  |

Remarque : Vue l’ampleur du projet et le temps disponible pour celui-ci, nous avons décidé de ne réaliser que la planification initiale pour le projet et de ne pas détaillé plus profondément les durées des sous-tâches. Les sous-tâches en question correspondent au chapitre de ce dossier que vous découvrirez ci-dessous.

# Analyse

## Concepts algorithmique [LAZHAR]

## Etude concurrentielle et rétrospective [AURELIEN.OK]

Dans ce chapitre, nous souhaiterions vous présenter divers jeux qui ont été inspirés par le concept innovant de « Tower defense ». Nous commencerons ce chapitre avec une petite présentation des origines de ce genre de jeux et détaillerons les plus célèbres d’entre eux. Pour finir, nous aborderont encore un sujet d’actualité, il s’agit du grand essor que subissent actuellement ce genre de jeux sur l’Internet et de leur avenir sur la toile.

### Bref historique

De nos jours, Il existe une multitude de jeux vidéo qui exploitent ce concept qui ne date pas d’hier. En effet, les premiers jeux vidéo de stratégie en temps réel comme « Age of Empire » sorti en 1997 sont réellement des précurseurs en la matière.

C’est en juillet 2002 qu’une boite de développement nommée « Blizzard Entertainment » lance son troisième opus de la série **Warcraft** et officialise sous la forme d’un « mod 1 » la première version officielle du concept des jeux applelés « Tower Defense ».

La première version de ce mod fut un succès et était déjà très aboutit. En effet, elle offrait déjà énormément de possibilités et fonctionnalités qui seront en majorité reprises par les jeux dérivés. Notons qu’il existe fondamentalement deux types de Tower Defense et que ces deux ont été imaginés dans la version de Blizzard. La différence entre ces deux types repose sur le positionnement des tours. On les appelle, avec ou sans labyrinthe (with / without mazing). Voici une brève description de chacun d’eux :

- Dans une version à labyrinthe, les tours ne peuvent être placées que **le long de l’itinéraire des ennemis (donc chemin des créatures statique)**. Le but est alors de trouver le placement optimal et la meilleure combinaison de tours.

- Dans une version sans labyrinthe, le joueur peut placer ses **tours sur l’itinéraire des ennemis qui les contournent (implémentation obligatoire d’un algorithme de recherche de chemin)**. La stratégie est alors de créer des chemins qui forcent les vagues d’ennemis à rester le plus longtemps possible sous le feu des tours.

1) *Un mod est un jeu vidéo créé à partir d'un autre jeu vidéo, ou une modification du jeu original, sous la forme d'une greffe qui se rajoute à l'original, le transformant parfois complètement.*

### Détails des plus célèbres jeux « Tower Defense »

**Warcraft 3 (le mod)**



La plus célèbre version du jeu reste sans nul doute la version originale développée par la célèbre boite de Los Angeles appelée « Blizzard Entertainment ».

Avec son mode multi joueurs permettant de défendre un labyrinthe commun à plus de 8 joueurs, cette version a encore énormément de charme auprès des joueurs sur l’Internet et dans les réseaux locaux de jeux (LAN).

Points forts :

*+ multi joueurs*

*+ énormément de type de tours et créatures*

*+ en 3D*

*+ mode avec et sans labyrinthe*

*+ outils de création de maps*

*+ et j’en passe…*

**Desktop Tower Defense**



Une version très récente du concept et une des plus jouée actuellement sur l’Internet.

Ce qui rend cette version différente des autres jeux de type « Tower Defense » est que cette dernière se déroule sur une carte totalement vierge ! C’est a vous de créer entièrement votre propre labyrinthe.

Lien : <http://www.handdrawngames.com/DesktopTD/game.asp>

Points forts :

*+ flash (pas d’installation)*

*+ mode sans labyrinthe*

*+ très complet*

**Onslaught 2**

Onslaught 2 est un autre classique des jeux de Tower Defense.

Lien : <http://onslaught.playr.co.uk/>

Points forts :

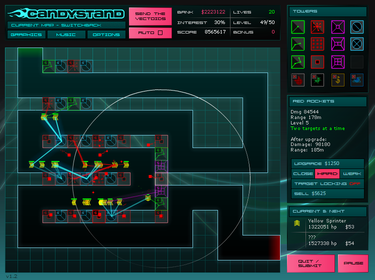
*+ flash (pas d’installation)*

*+ grand choix de tour*

*+ 4 niveaux de dificultés*

*+ possibilité de créer ses propre cartes*

**Vector Tower Defense**



Un tower defense avec un design futuriste est classe.

Lien : <http://www.candystand.com/play/vector-td>

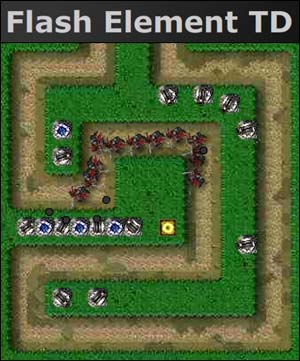
Points forts :

*+ flash (pas d’installation)*

*+ Design original*

*+ Ambiance générale*

*+ Maps sympa*



**Flash Element TD**

Flash Element TD est un jeu de type [Tower Defense](http://www.freegamesnews.com/fr/?cat=44) inspiré par le désormais célèbre jeu de stratégie en temps réel [Warcraft 3](http://fr.wikipedia.org/wiki/Warcraft_3).

Développé et mis à jour actuellement plus ou moins quotidiennement par David Scott, le jeu est très vite devenu un gros succès avec 250 000 visiteurs journaliers, 10 jours à peine après sa sortie (janvier 2007).

Points forts :

*+ flash (pas d’installation)*

*+ Respect du design de Warcraft 3 en 2D*

*+ On retrouve les éléments originaux de Warcraft 3*

*+ Ambiance*

**Epic Tower Defense**

Voici un premier essai avec Unity tournant autour de la licence Warcraft : Epic Tower Defense. Ce n’est pas encore tout a fait au point mais ça risque de donner de très belles choses dans le futur.

Lien : <http://www.shockwave.com/gamelanding/epictowerdefense.jsp>

Points forts :

*+* ***Unity !*** *(pas d’installation)*

*+ en 3D*

*+ licence Warcraft 3*

*+ Projet très ambitieux*

### Essor actuel sur l’Internet et avenir

Les « **Tower Defense »** rencontrent actuellement un grand succès sur l’Internet, notamment grâce à la technologie Flash qui est utilisée dans la majorité des cas.

Mais ce n’est qu’une première étape car avec l’arrivée des environnements riche, de la 3D dans les navigateurs, nous risquons de voir bientôt débarquer de bien belles choses. Notamment en provenance d’un environnement du type d’[Unity](http://unity3d.com/unity/).

**A DEPLACER !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!**

**Sources**

Unity

<http://unity3d.com/unity/>

Listes d’un très grand nombre de « Tower Defense »

<http://www.logiste.be/blog/125-tower-defense-games/>

<http://www.freegamesnews.com/fr/?cat=44>

Wikipedia du célèbre jeux « Warcraft 3 »

<http://fr.wikipedia.org/wiki/Warcraft_III>

Wikipedia décrivant le concept des jeux « Tower Defense »

<http://fr.wikipedia.org/wiki/Tower_defense>

**A DEPLACER !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!**

## Etude de faisabilité [AURELIEN.OK]

Ce chapitre permet de prendre le temps d’analyser et cerner les points délicats qui pourraient rendre le projet impossible. En parallèle de ce dernier il est possible que les membres du projet commencent déjà à mettre en place quelques implémentations afin de répondre concrètement à cette étude. Pour chacun des sujets abordés nous nous posons une question et tentons d’y répondre. Nous décomposons cette étude en trois-sous chapitres présentés ci-dessous.

### Risques techniques

Nous décrivons ici les risques techniques liés principalement au développement de l’application.

* **Graphe & algorithmes**

**Question :** Arriverons-nous à implémenter les algorithmes nécessaires au bon fonctionnement du jeu ?

**Réponse :** Nous avons eu un cours dédié à l’apprentissage des graphes et aux algorithmes qui nous permettent de comprendre leur implémentation.

**Question :** Devrons-nous créer notre propre TDA ou trouverons-nous une librairie ou un TDA déjà réalisée et fonctionnelle afin de nous éviter de devoir tout refaire ?

**Réponse :** Nous avons trouvé une librairie gratuite et libre qui nous offre exactement ce que nous voulions. Celle-ci s’appelle jGraphT (<http://jgrapht.sourceforge.net/>)

* **Interface Java**

**Question :** Arriverons-nous à réaliser l’interface graphique avec Java ?

**Réponse :** Les membres du projet ont déjà de bonnes connaissances en interface graphique avec Java et se sentent capable de réaliser cela. De plus nous suivons en parallèle un cours de Java et nous aborderons les notions d’interface graphique.

* **Architecture de logiciel**

**Question :** Serons-nous capable de faire cohabiter tous les éléments du jeu dans un ensemble d’une façon logique ?

**Réponse :** Plusieurs membres du projet ont déjà réalisé des jeux vidéo durant leur formation et savent plus ou moins comment structurer une telle architecture. De plus avec les notions actuelles de « programmation orienté objet », nous sommes capable de concevoir cela sans trop de problème.

**Question :** Arriverons nous à réalisé l’ « inverse engineering » basé sur les versions actuel du jeu ?

**Réponse :** L’étude concurrentielle et nos expériences nous on permit de bien comprendre les éléments clefs du concept. Après quelques schémas d’essais nous avons réussi à cerner les objets et leur interaction entre eux.

### Risques concernant le planning & ressources humaines

Nous abordons ici les risques liés au travail des membres et de leur communication.

* **Délais**

**Question :** Arriverons-nous à fournir un logiciel fonctionnel dans les temps ?

**Réponse :** Le temps pour le projet est court. Ce pourquoi nous allons dans un premier temps nous concentrer sur l’essentiel et évité les choses superflues comme les animations ou la gestion des scores et choses similaire. De plus, comme le sujet est très intéressant nous allons prendre beaucoup de notre temps libre pour réaliser ce travail.

* **Répartition des tâches et travail d’équipe**

**Question :** Partagerons-nous suffisamment bien les taches de façon équitable entre les membres du projet ?

**Réponse :** Nous avons décomposé le projet en plusieurs couches bien distinctes (chef de projet, gestion de maillage et des algorithmes, interface graphique, conception du jeu) et avons distribué chacune delle aux membres. Nous avons également prévue des tâches optionnelles (musique, design, gestion des scores, etc.). Si un membre termine complètement sa partie il pourra alors se concentré sur les parties annexes**.**

* **Versionnage & mise en commun**

**Question :** Comment allons-nous gérer les versions du jeu et la mise en commun des parties ?

**Réponse :** plusieurs membres du projet connaissent très bien des application de gestion du versionnage et gestion de projet de développement. Les membres qui ne sont pas encore familiarisés avec ces notions apprendront à les utiliser. Ces genres de logiciels sont extrêmement puissants et permettent un gain de temps énorme.

### Risques concernant le budget.

Aucun budget nécessaire

# Conception

## Dossier de conception

### Systèmes d’exploitation [LAZHAR]

Le système d’exploitation sur lequel nous travaillons est Windows (XP et Seven). Cependant, grâce au choix qui a été fait d’utiliser un encodage de type UTF-8 ainsi que celui du langage portable Java, nous pouvons sans soucis travailler sur un environnement Linux ou Mac par exemple.

### Outils logiciels [LAZHAR]

Nous développons avec l’IDE Eclipse (version 3.4 et supérieure) intégrant tous les outils nécessaires au développement d’applications Java. Ce logiciel est très largement répandu dans le monde des développeurs et est très utilisé. Il possède de nombreuses fonctions spécialement conçues pour augmenter la productivité des développeurs et leur simplifier la vie (comme le *refactoring* par exemple). De plus, cette plateforme nous permet d’ajouter toute une série de plugins qui peuvent ajouter des fonctionnalités, telles que SVN (logiciel de gestion des versions du code). En ce qui concerne la génération des diagrammes de classe UML, nous utilisons le plugin ***eUML 2.0*** de chez ***Soyatec***.

### Librairies externes [LAZHAR]

Nous utilisons la librairie externe ***JGraphT*** codée en Java. En effet, cette librairie possède toutes les briques logicielles nécessaires à la création de graphes. Nous l’utilisons comme une boîte noire sans se soucier de son implémentation.

### Interface graphique [AURELIEN]

Ce chapitre nous permettra d’imaginer et de penser le résultat final de l’interface graphique. Grace à cette étude, nous pourrons plus facilement élaborer et trouver les classes qui en découleront.

Commençons par lister les actions possibles et informations visibles dans la fenêtre de jeu :

* Voir les créatures se déplacer sur le terrain et les tours tirer sur les créatures
* Choisir et poser une tour sur le terrain
* Améliorer ou vendre une tour sélectionnée
* Voir les informations d’une tour (ces caractéristiques)
* Voir les informations d’une créature sélectionnée
* Voir les informations du jeu (score, argent, vies restantes)
* Voir les informations de la vague de créatures suivante et courante
* Lancer la vague de créature suivante
* Quittez le jeu
* Ouvrir la fenêtre « A propos »

Ensuite nous élaborons un schéma d’interface graphique permettant d’implémenter toutes les actions et informations citées ci-dessus. En voici le résultat :

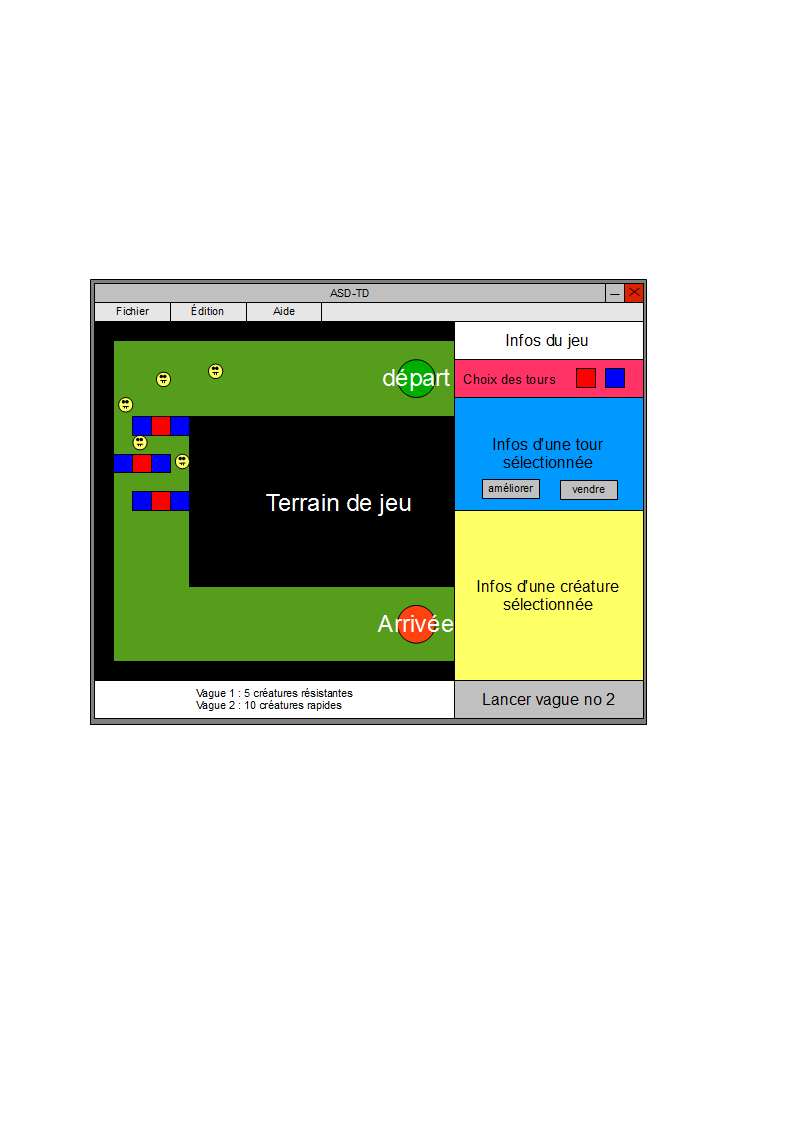


Figure Schéma de la fenêtre principale

La fenêtre est décomposée en 4 parties :

La première partie, celle au nord, contient le menu permettant de quittez l’application (fichier -> quitter) et d’ouvrir la fenêtre « à propos » (aide -> à propos). Le menu Edition permettra d’offrir d’autres fonctionnalités comme l’affichage du maillage (facultatif).

La deuxième concerne l’affichage du terrain et de l’animation du jeu. C’est ici que l’on verra bouger les créatures et tirer les tours. C’est aussi dans cette partie que le joueur posera ces tours et sélectionnera les tours et les créatures.

La troisième zone est la partie de droite qui est composé de 4 boites :

* *« Infos du jeu »* permettant au joueur de voir l’état de la partie caractérisé par le score, l’argent du joueur et le nombre de vies restantes.
* *« Choix des tours »* offrant la possibilité au joueur de sélectionner une tour à acheter (pour acheter la tour sélectionnée, le joueur doit alors cliquer sur une zone du terrain pour choisir où mettre la tour)
* *« Info d’une tour sélectionnée »* permettant de montrer les caractéristiques d’une tour en particulier (prix, dégâts, cadence de tir, bonus, etc.) et c’est ici que le joueur peut améliorer ou vendre une tour.
* *« Info d’une créature sélectionnée » qui permet de voir les* caractéristiques d’une créature (nom, santé, vitesse, etc.)

Le quatrième cadran permet au joueur de voir des informations sur les vagues suivante ainsi que de lancer la vague suivante d’ennemis. Nous pourrons également diffuser des messages d’erreur ou autre dans la zone d’affichage des infos vagues.

Les autres fenêtres (fenêtre à propos, et menu principal d’accueil) de l’application ne nécessitent pas une étude particulière. En effet, elles seront relativement simples et nous ont demandé uniquement un petit croquis sur papier.

### Architecture de l’application [AURELIEN.OK]

Nous avons choisi d’implémenter une architecture très répandu dans la conception de logiciels, il s’agit du célèbre design pattern **MVC** (Modèle – Vue – Contrôleur). Les étudiants responsables de cette partie s’étaient déjà familiarisés avec cette notion durant d’autres projets d’études. Malheureusement nous n’allons pas détailler et décrire le fonctionnement du pattern mais présenter comment nous pensons l’implémenter dans ce jeux vidéo.

**MVC dans un jeu vidéo**

Dans un jeu vidéo, il est difficile de séparer distinctement les vues des modèles car les éléments des gestions sont parfois intrinsèquement liés à leur affichage. En effet, quasiment tous les éléments comme les tours, les créatures, le terrain, la gestion des sons et les animations (attaques ou autre) sont bien des objets faisant partie du modèle mais d’un autre coté, il est difficile de ne pas les intégrer dans les vues car les interactions entre eux peuvent parfois passer par des affichages et traitement particuliers.

Imaginons l’affichage des animations, lors de la construction de l’image représentant les animations sur le terrain. Est-il mieux d’établir un standard d’affichage pour toutes les animations séparant ainsi les vues des animations ou de dire à l’animation dessine toi et lui passer le pointeur vers les outils de dessin ? A priori, la première solution est la meilleure mais dans ce cas, cela nous force à avoir un standard de pour dessiner l’objet se qui peut limiter considérablement la diversité et rentre très difficile certains résultats souhaités. Par exemple, certaines animations pourraient être représentées par une image et d’autre peuvent être construites avec des traits ou forme géométrique de base. Pour cette raison nous essayerons de séparer un maximum les modèles des vues dans une certaine mesure de portabilité dans le cas ou un jour, nous déciderons par exemple de passer à la 3D ou autre changement d’interface.

**Contrôleur principale**

Après plusieurs schémas et tests, nous avons décidé de simplifier l’architecture en créant une classe qui s’occupera de gérer toutes les interactions entre les vues et les modèles. La classe qui jouera ce rôle sera la fenêtre principale du jeu. Celle-ci sera donc informée de tous les événements du joueur et des modèles (sous la forme d’écouteurs) et son travail sera d’en informer les autres modèles et toutes les parties de l’interface. Ci-dessous nous vous présentons un schéma simplifié mettant en œuvre les différentes parties citées ci-dessus :

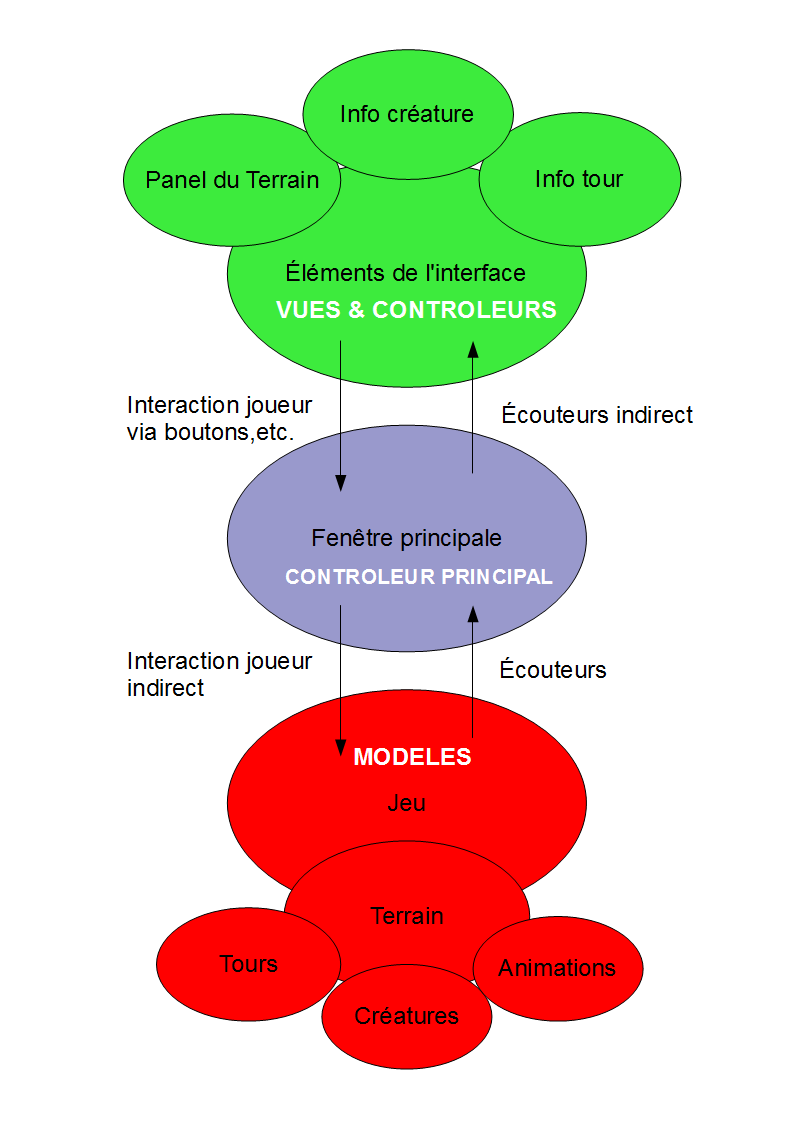


Figure Contrôleur principal

**Notre organisation**

Voici un résumé des différentes classes et leur(s) emplacement(s) dans le pattern :

**Vues**

Toutes les fenêtres

Les panels contenus dans les fenêtres

Les animations, sons et autres classes modèles (le moins possible)

**Modèles**

Les tours

Les créatures

Le terrain

Les données du jeu

Les animations

Les sons (lancés par des modèles et pour les vues)

**Contrôleurs**

Boutons et objets événementiels contenu dans les fenêtres

Fenêtre principale de diffusion (explication à la page précédente)

### Schémas UML [PIERRE-DO] et [AURELIEN]

#### Maillage **[PIERRE-DO]**

#### Interface graphique (vue et controleur) **[AURELIEN]**

#### Gestionnaire du jeu (model) **[AURELIEN.OK]**

Nous avons décidé de fournir un schéma UML simplifié (sans getter/setter et attributs de gestion spécifique) afin de rendre la compréhension plus aisé. Voici comment nous pensions organiser nos classes :

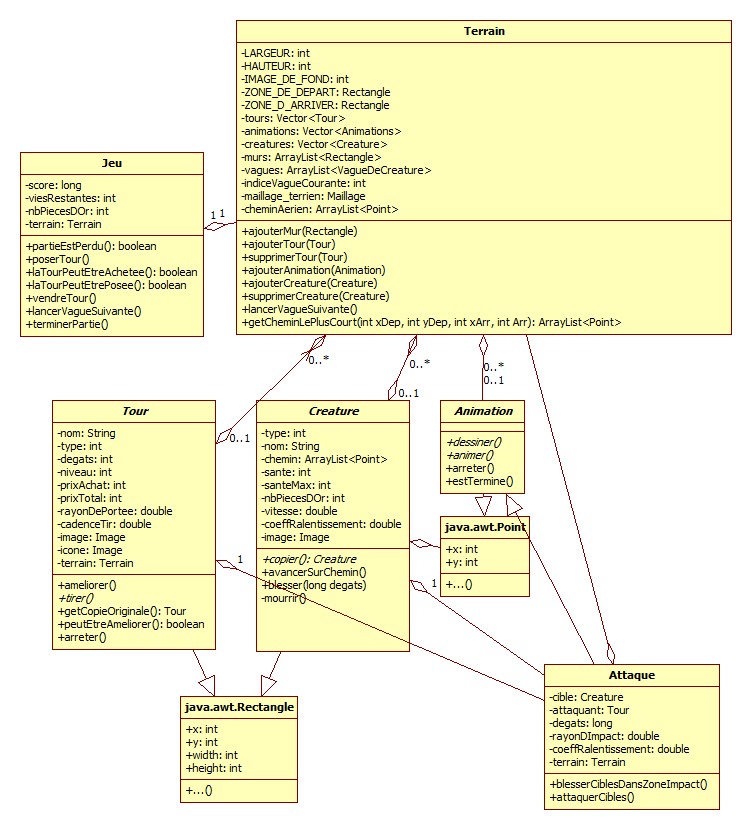


Figure Schéma de classe du jeu

Partons de la classe principale que nous nommons **Jeu**. Celle-ci contiendra les données du joueur et de la partie de jeu. Elle permettra également d’encapsuler le terrain. La classe **Terrain**, elle, contiendra toutes les collections d’éléments comme les tours, les créatures et les animations. C’est également elle qui encapsulera le maillage et définira les zones de départ et d’arrivée des créatures. Les murs (simple rectangle) du terrain permettront de rendre des zones non accessibles pour les créatures et les tours (pose impossible). Passons maintenant à la classe **Tour**, cette entité est caractérisée par un prix d’achat, des dégâts, une cadence de tir, un rayon de portée et j’en passe. Celle-ci peut tirer sur des créatures et pour cela elle a besoins du terrain pour savoir ou sont les créatures. Une **Créature** est caractérisée principalement par sa santé, sa vitesse de déplacement et son nombre de pièces d’or gagné par le joueur lors de son décès. Les tours et les créatures héritent de la classe Rectangle fourni par Java qui nous permet de gérer les collisions. Ensuite, le dernier groupe éléments principales est les **animations**. Une animation sera surtout utilisée d’un point de vue esthétique mais les attaques sont des animations qui joueront un rôle sur les modèles.

Faute de temps nous n’allons pas détailler plus cette partie. Pour plus de question nous vous invitons a consulté la  « java doc » fourni en annexe qui détail bien toutes les classes et les membres qui les composes.

### Gestion de la concurrence [PIERRE-DO]

*Fournir tous les document de conception:*

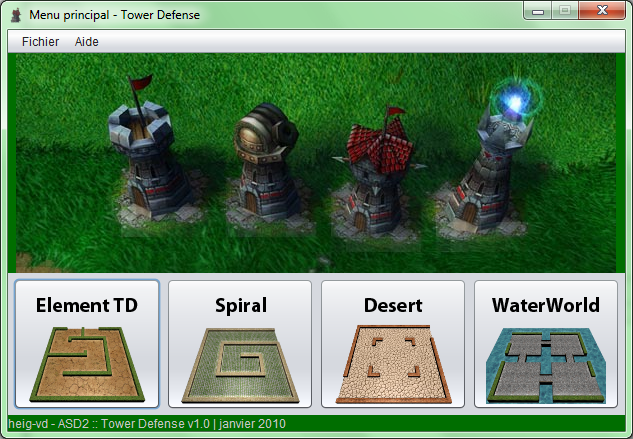
* *le choix du matériel HW*
* *le choix des systèmes d'exploitation pour la réalisation et l'utilisation*
* *le choix des outils logiciels pour la réalisation et l'utilisation*
* *site web: réaliser les maquettes avec un logiciel, décrire toutes les animations sur papier, définir les mots-clés, choisir une formule d'hébergement, définir la méthode de mise à jour, …*
* *bases de données: décrire le modèle relationnel, le contenu détaillé des tables (caractéristiques de chaque champs) et les requêtes.*
* *programmation et scripts: organigramme, architecture du programme, découpage modulaire, entrées-sorties des modules, pseudo-code / structogramme…*

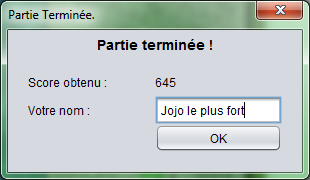
***Le dossier de conception devrait permettre de sous-traiter la réalisation du projet !***

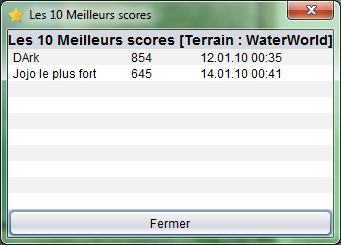
# Réalisation

## Dossier de réalisation

### Résultat de l’interface graphique [AURELIEN]

**

**

**

**

**

### Résultat de l’implémentation du maillage [PIERRE-DO]

*Décrire la réalisation "physique" de votre projet*

* *les répertoires où le logiciel est installé*
* *la liste de tous les fichiers et une rapide description de leur contenu (des noms qui parlent !)*
* *les versions des systèmes d'exploitation et des outils logiciels*
* *la description exacte du matériel*
* *le numéro de version de votre produit !*
* *programmation et scripts: librairies externes, dictionnaire des données, reconstruction du logiciel - cible à partir des sources.*

*NOTE : Evitez d’inclure les listings des sources, à moins que vous ne désiriez en expliquer une partie vous paraissant importante. Dans ce cas n’incluez que cette partie…*

## Description des tests effectués [LAZHAR]

*Pour chaque partie testée de votre projet, il faut décrire:*

* *les conditions exactes de chaque test*
* *les preuves de test (papier ou fichier)*
* *tests sans preuve: fournir au moins une description*

## Erreurs restantes [LAZHAR]

*S'il reste encore des erreurs:*

* *Description détaillée*
* *Conséquences sur l'utilisation du produit*
* *Actions envisagées ou possibles*

# Conclusions *[LAZHAR]*

## Objectifs atteints / non-atteints

## Points positifs / négatifs

## Difficultés particulières

## Avenir du projet

# Annexes

## Sources – Bibliographie [LAZHAR]

*Liste des livres utilisé (Titre, auteur, date), des sites Internet (URL) consultés, des articles (Revue, date, titre, auteur)… Et de toutes les aides externes (noms)*

## Journal de bord de chaque participant [LAZHAR]

## Manuel de TDA maillage [PIERRE-DO]

## Manuel d'Utilisation [LAZHAR]

## Archives du projet [LAZHAR]

*CD, … dans une fourre en plastique*