CHAPELLE Quentin
CHALIEZ Arnaud

Projet 1ère année

Développement et création d'un jeu vidéo

Année scolaire 2011-2012

ISIMA
INSTITUT SUPÉRIEUR D'INFORMATIQUE DE MODÉLISATION ET DE LEURS APPLICATIONS

Remerciements

Nous tenons à remercier notre tuteur Mr Luc TOURAILLE qui nous a permis de mettre sur pied ce projet et qui nous a accompagnés à chaque étape de son développement.

Table des matières

Table des figures et illustrations

Résumé

Ce projet a été développé dans le cadre de la première année de l'ISIMA (Institut Supérieur d'Informatique, de Modélisation et de leurs Applications).

Il fut proposé dans le but de découvrir et de comprendre le processus de développement d'un jeu vidéo en équipe. Pour se rapprocher au plus près d'un projet industriel, nous avons souhaité nous imposer des difficultés au niveau des plates-formes utilisées (développement sur PC et sur XBOX 360) ainsi qu'au niveau des techniques (design patterns*,...) nous apportant ainsi une flexibilité et une facilité de maintenance accrue.

Après avoir choisi une technologie (le framework* XNA) ainsi qu'un type de jeu à réaliser ("Space Invader"), nous nous sommes engagés dans une phase d'analyse qui nous a permis, à terme, de faire les meilleurs choix technologiques possibles.

Nous avons ensuite commencé le développement à proprement parlé du jeu. Pour être le plus efficace possible et nous rapprocher des conditions en entreprise, notre tuteur, Mr Luc TOURAILLE a mis à notre disposition un serveur Subversion (SVN) ainsi que d'autres outils regroupés dans une forge*.

Malgré les difficultés rencontrées et notamment celle du travail d'équipe, nous avons finalement développé un jeu vidéo disposant de bases solides et pouvant être facilement amélioré. En effet l'analyse effectuée auparavant nous as permis de développer un jeu pour lequel il est aisé d'ajouter des niveaux, des types d'ennemis, des périphériques, … et ce, de manière aisée et rapide.

Mots clés: Jeux vidéo, Analyse, entreprise, design pattern, développement, équipe, multiplateforme.

Introduction

Ce projet a était proposé dans le cadre des projets de deuxième semestre de la première année de l'ISIMA.

Il a pour but principal de recréer, de manière plus restreinte, les conditions de travail dans le domaine des jeux vidéo. Pour se faire, nous avons travaillé sur quasiment toutes les étapes du développement et de la production d'un jeu vidéo. Cependant, le projet visant à développer des compétences dans le monde de l'ingénierie informatique et le temps imparti étant relativement court, nous avons laissé de côté l'aspect créatif (scénario, graphisme, musique, …) au profit de l'aspect technique (analyse, optimisation, …).

Nous avons donc repris un grand classique des jeux vidéo, « The Space Invaders » renommé pour l'occasion « ZZInvaders » (le principe de ce jeu est détaillé plus tard dans le rapport).

Notre objectif principal n'était pas seulement d'obtenir un jeu fonctionnel mais surtout d'obtenir un jeu modulable. Nous nous sommes donc donné différentes contraintes techniques intéressantes, qui pourraient tout à fait être celles d'un studio de développement.

Tout d'abord, nous avons souhaité que le jeu soit multiplateforme, ainsi nous avons opté, pour des raisons matérielles, pour le développement sur PC (Windows) et Xbox 360.

Ensuite, comme énoncé précédemment, nous avons voulu développer un jeu où la maintenance serait très aisée, pour cela nous avons utilisé des techniques de programmations telles que les design patterns.

Enfin nous avons aussi souhaité travailler avec une technologie que nous ne connaissions peu voire pas du tout. Dans ce rapport nous allons vous présenter de manière détaillée ce travail. Pour ce faire nous allons suivre le plan suivant :

1. **Introduction de L'étude**
    - **a.** Rappel du sujet de l'étude
    - **b.** Analyse du problème
    - **c.** Étude du problème
2. **Matériel, méthode**
    - **a.** Conception d'une solution
3. **Résultats et discutions**
    - **a.** Le produit réalisé
    - **b.** Le produit dans son fonctionnement

L’étude

1. Introduction de L’étude

a. Rappel du sujet de l’étude

L’étude du sujet vise donc à créer un jeu vidéo fonctionnel reprenant le principe de “Space Invaders” qui est un jeu de type shoot them up* apparut à la fin des années soixante-dix, sous forme de jeu d’arcade. Il consiste à tuer des aliens se déplaçant en groupe de manière régulière grâce à un vaisseau ne pouvant se déplacer que horizontalement (voir images ci-dessous).

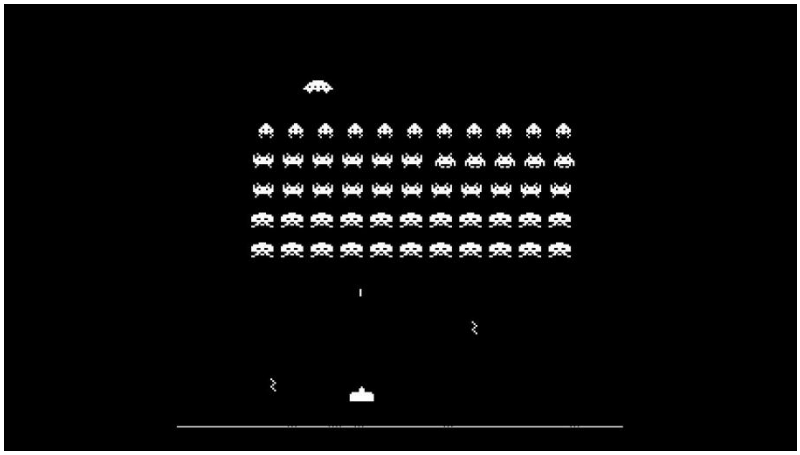


Figure 1 - Space Invaders original

Ce projet fût mené avec les mêmes contraintes techniques que pourraient rencontrer un studio de création de jeux vidéo. Ce fût un moyen ludique de découvrir un autre univers de travail riche en technologies diverses et variées parfois méconnues. Au terme de ce projet, nous devions donc livrer un jeu multiplateforme, fonctionnel et évolutif. L’objectif étant de nous habituer à de nouvelles méthodes de développement (notamment en utilisant de nouveaux outils) et à nous permettre de développer un jeu évolutif. Malheureusement, le temps imparti pour le projet étant court et sans créneaux dédiés, toutes les contraintes concernant l’expérience de jeu, le confort du joueur, … tel que le scénario, les graphismes, … n’ont pas pu être inclus de manière aussi importante que nous l’aurions souhaité.

b. Analyse du problème

Les jeux vidéo actuels sont très souvent adaptés sur de nombreux formats (console de salon, console portable, PC, appareil mobile, …) nous devions donc nous même être dans la capacité de déployer notre code sur des plates-formes multiples et variées, c’est ainsi que notre première contrainte fût fixée.

De plus les studios de jeux disposent de nombreux corps de métiers différents : scénariste, designer, etc. qui travaillent en permanence côte à côte. Le projet est donc assez instable et doit pouvoir être modifié assez aisément. La facilité de maintenance fût donc le deuxième gros objectif (ex: le développement du jeu Game of Thrones basé sur les ouvrages de George R. R. Martin a commencé bien avant qu'une série télé n'ait était enregistrée, les différences entre le jeu et la série aurait pu être catastrophique au point de vue des ventes, cependant grâce à une bonne conception, les changements à effectuer non que très peu impactés les délais de développement du jeu source). Dans cette optique la conception d'une structure solide fût nécessaire pour permettre un maximum d'adaptabilité et de flexibilité.

Enfin le dernier objectif fût de développer le jeu avec une technologie ou du matériel que nous maîtrisions peu, voire pas du tout, visant ainsi à parfaire notre capacité à nous former nous-même sur des technologiques nouvelles, ce qui est le quotidien de tout informaticien.

Tous ces critères devant nous mener à un jeu vidéo fonctionnel en un temps relativement restreint, nous avons dû faire les bons choix technologiques.

Pour ne pas être surpris par le temps, nous avons séparé notre projet en plusieurs "Thèmes" nous permettant ainsi d'avoir à l'issu de chacune de ces sous-parties un jeu parfaitement fonctionnel avec plus ou moins de fonctions. Voici la liste des quatre principaux thèmes:

Thème 1 :

- Analyse poussée pour établir la meilleure technique de développement possible.
- Création d'un mode de jeu traditionnel (reprenant le Space Invaders normal déplacement selon l'axe des abscisses uniquement).
- Création de menus permettant de changer les options, la gestion du son, …

*Ce **thème 1** est l'objectif principal, le but étant de concevoir une base solide permettant une implémentation rapide et efficace de toutes fonctionnalités supplémentaires (dans notre cas les thèmes 2, 3 et 4). Pour ce faire, après une analyse suffisante du projet, nous étudierons les solutions, telles que les design patterns, à mettre en place.*

Thème 2 : Création d'un deuxième mode de jeu où les déplacements sont possibles selon l'axe des abscisses mais aussi celui des ordonnées.

Thème 3 : Création d'un mode multijoueur, ou deux adversaires pourront s'affronter en ligne, vaisseau contre vaisseau.

Thème 4 : Portage du jeu sur XBOX et/ou Windows phone.

Autre piste : création d'un mode 3D, gestion du XBOX Live, Kinect ...

Ces thèmes furent eux-mêmes fractionnés en objectifs. Certains furent définis comme obligatoires pour arriver à un jeu un minimum fonctionnel et d'autres plus optionnels, permettant d'améliorer l'expérience utilisateur dans le cas où le temps imparti ne serait pas dépassé.

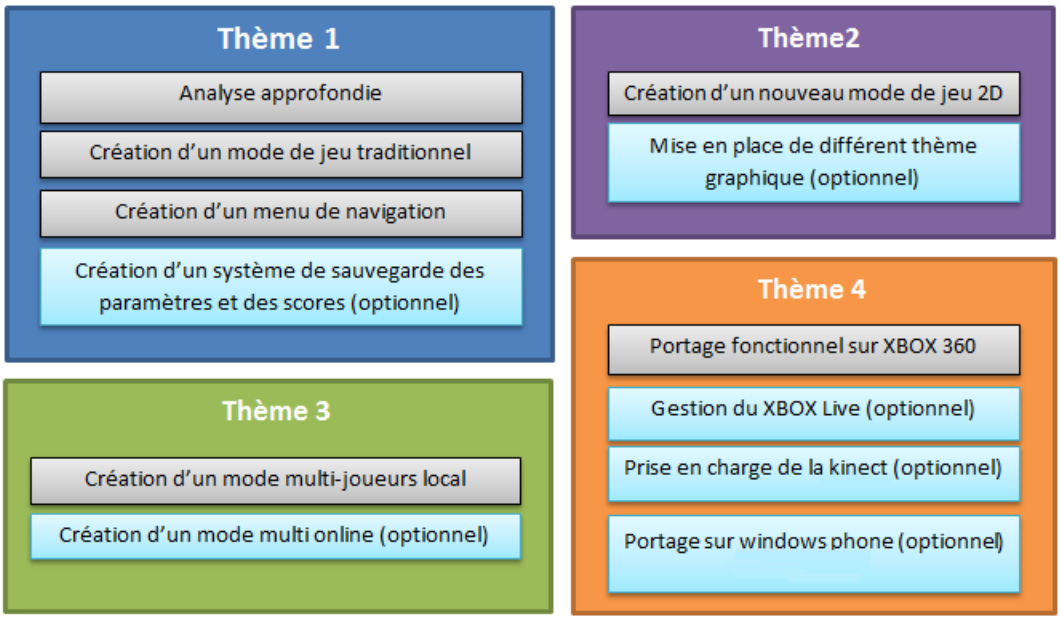


Figure 2 - Découpage du projet

A l'issue de tous ces thèmes, nous aurions donc dû obtenir un jeu solo et/ou multijoueur, paramétrable, jouable en ligne, fonctionnant sur Xbox 360, Windows phone, PC (Windows) et disposant de plusieurs modes de jeu. Cette liste de fonctionnalité n'était pas exhaustive, comme précédemment écrit, d'autres pistes pouvaient être rajoutées selon l'imagination de l'équipe de développement.

c. Étude du problème

De nombreux moteurs graphiques, physiques, … permettent de créer des jeux vidéo de façon plus ou moins rapide, pour des plates-formes diverses et variées en utilisant des langages tout aussi variés. Les plus célèbres d'entre eux (Unity, Urealengine, Ogre …) ont servis à de nombreux éditeurs de jeux pour concevoir les plus grands box-offices. Cependant, ces solutions demandent généralement un temps de développement assez important ou ne consiste qu'à utiliser des logiciels ayant une abstraction quasi totale avec le code source. Nous avons donc opté pour le framework XNA, pour plusieurs raisons:

- Il permet un développement rapide, tout en utilisant uniquement des techniques de programmation

- Le framework est efficace et assez similaire à ce que nous connaissons déjà, nous avons donc pu nous concentrer sur l'analyse et la conception à proprement parler et non sur la prise en main d'un tout nouvel environnement.

- Beaucoup de documentations et d'exemples sont disponibles sur internet.

- Il est possible de développer des applications compatibles Kinect

- Raisons économiques puisque la licence MSDN fournit par l'ISIMA nous donne accès à tous les outils nécessaires gratuitement.

Une fois la technologie principale choisie, nous avons pu entrer dans le vif du sujet et commencer l'analyse.

2. Matériel, méthode

a. Conception d'une solution

La conception de notre jeu a nécessité l'utilisation de nombreux outils et méthodes. Certains connus, d'autres non, avec ou sans documentation.

Dans cette partie nous allons vous présenter les différentes ressources et méthodes ainsi que nos réussites, nos problèmes et difficultés issus de leurs utilisations. Tout d'abord nous allons présenter les différents outils que nous avons utilisés.

- ***Les outils***

Le Framework

XNA (officiellement XNA's Not Acronymed) est un framework constitué de plusieurs librairies .NET permettant de développer des jeux vidéo qui pourront s'exécuter sur différentes plateformes telles que les multiples systèmes Windows, mais aussi sur la console de Microsoft à savoir la XBOX 360 et plus récemment les Smartphones sous Windows phone.

XNA est donc un ensemble de classes et de méthodes communes aux différentes plateformes énoncées ci-dessus. Il est ainsi possible de porter un jeu développé pour Windows via XNA, directement sur XBOX 360 sans changer le code ou très peu et même de partager le code d'une plateforme à l'autre.

L'attrait de ce Framework est de faciliter grandement le développement de jeux vidéo en apportant une couche d'abstraction importante. Il est donc plus aisé et plus rapide de développer un jeu en XNA qu'en C++, langage majoritairement utilisé dans le développement de ceux-ci. Cependant XNA est moins performant et donc peu utilisé dans le cadre de projets professionnels ou seulement pour tester des concepts qui seront ensuite retraduits en C++.

XNA utilise le langage C# (présenté ci-dessous) du framework .NET. Le Visual Basic est aussi supporté depuis peu (mai 2011) mais de manière limité.

Le framework peut être découpé en plusieurs couches, certaines sont gérées de façon native par XNA tandis que d'autres doivent être gérées par le développeur. Voici un schéma illustrant cette structure.

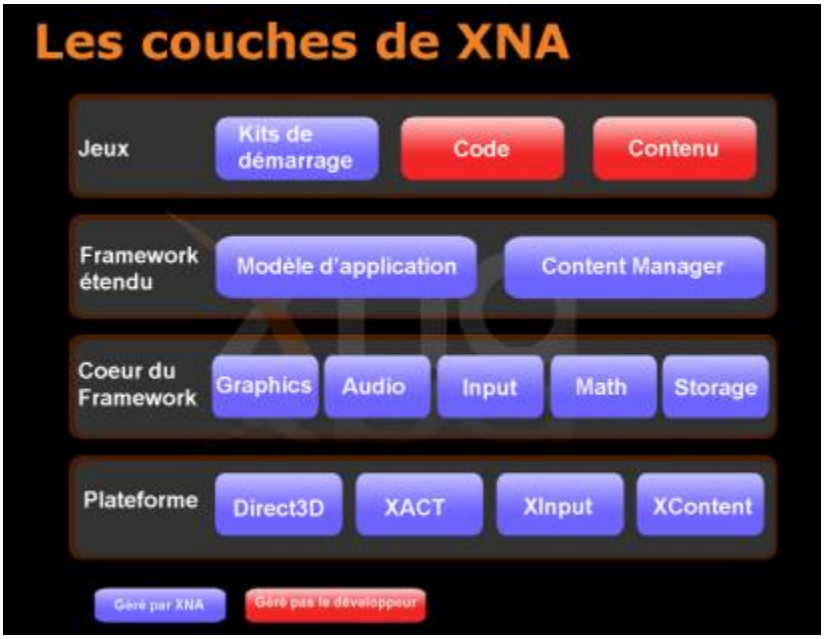


Figure 3 - Les couches de XNA

XNA étant l'un des outils majeur de Microsoft pour inciter les développeurs indépendants à s'intéresser à la XBOX 360, il est accompagné d'une documentation et d'exemples en quantité qui permettent de comprendre très rapidement le fonctionnement de ce framework. De plus, il introduit une manière particulière assez intuitive pour programmer. Le schéma ci-dessous illustre bien la logique XNA.

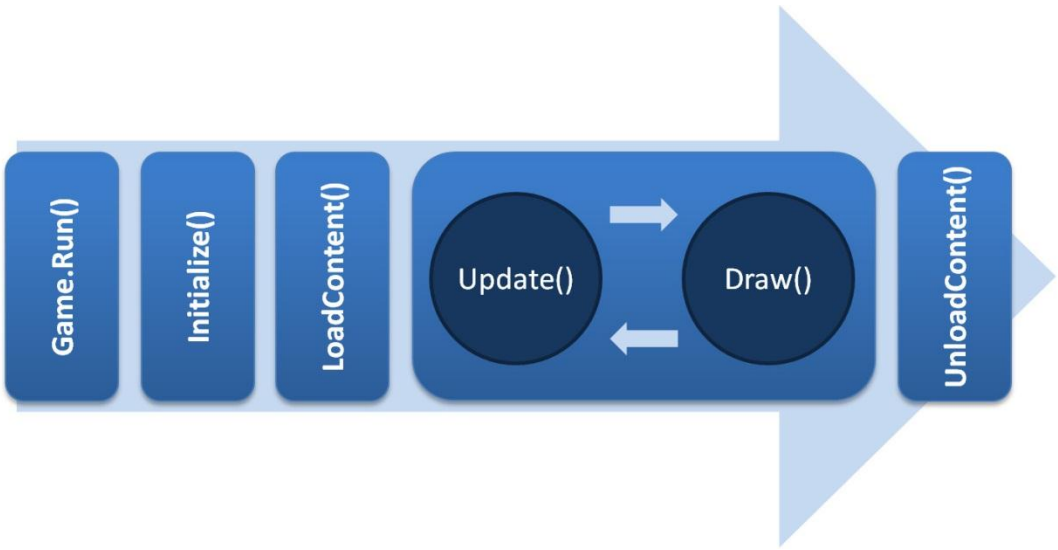


Figure 4 - Fonctionnement de XNA

Dans un premier temps nous avons une classe Game qui est le début de toute la chaîne. Ensuite, chaque élément, faisant parti du processus, se doit de contenir un certain nombre de méthodes standards. Tous d'abord Initialize(), qui, comme le nom l'indique permet d'initialiser l'objet concerné. Ensuite LoadContent() qui permet le chargement de certaines ressources, comme des sons, des images, … Les méthodes Update() et Draw() qui fonctionnent de pair. La première traite les données en les mettant à jour selon les actions du ou des joueurs, tandis que la seconde se charge de l'affichage. Enfin la fonction UnloadContent() permet de décharger les éléments préalablement chargés.

Le Langage

Le C# est un langage développé par Microsoft pour exploiter facilement toutes les capacités de sa plateforme .Net. Spécifiquement conçu pour exploiter les spécifications CLI, C# est donc le langage principal du framework Microsoft .Net.

Le C# s'inspire de la syntaxe générale de C/C++ pour les instructions, et de certaines caractéristiques de Java comme la gestion automatique de la mémoire (Garbage Collector).

Le C# est un langage orienté objet qui bénéficie d'innovations censées le rendre plus sûr, pour créer des programmes à la fois simples et robustes.

Le code C# est, comme les autres langages .NET, converti en CLR* (Common Language Runtime). Ce code précompilé s'exécute sur une machine virtuelle plutôt volumineuse, et donc très pénalisante pour un système embarqué. Les communautés .Net ou C# sont le plus souvent

orientées vers Microsoft et le développement Windows cependant la rapidité et les performances restent suffisantes pour un déploiement sur Smartphones (Windows phone).

XACT **La Gestion du son**

Cross-platform Audio Creation Tool (XACT) est une API de haut niveau publiée par Microsoft qui permet l'utilisation d'une partie de DirectX. Cette API est utilisée pour la création audio / lecture audio dans XNA. Elle permet notamment d'utiliser Xaudio pour la Xbox, DirectSound pour Windows XP, et la nouvelle pile audio sur Windows Vista et Windows 7. Ce choix technologique c'est imposé de par sa présence native dans le framework XNA ainsi que sa compatibilité native avec la Xbox 360.

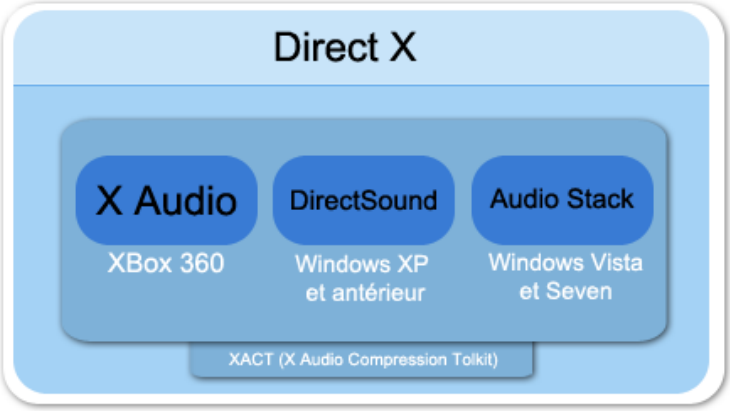


Figure 5 - XACT, partie de Direct X

Le système de gestion de versions

Subversion (SVN) est un système de gestion de versions, distribué sous licence Apache et BSD. Il a été conçu pour remplacer Concurrent Versions System (CVS).

Un logiciel de gestion de versions est un logiciel qui permet de stocker un ensemble de fichiers en conservant la chronologie de toutes les modifications qui ont été effectuées dessus.

Les logiciels de gestion de versions sont utilisés notamment en ingénierie du logiciel pour conserver le code source relatif aux différentes versions d'un logiciel.

Intégré à la forge de l'université Blaise Pascal, l'utilisation de SVN nous as permis de centraliser le code source et donc de pouvoir travailler plus efficacement de façon collaborative.

C'était aussi une sécurité pour pouvoir retourner sur d'anciennes versions du projet si d'éventuels problèmes survenaient.

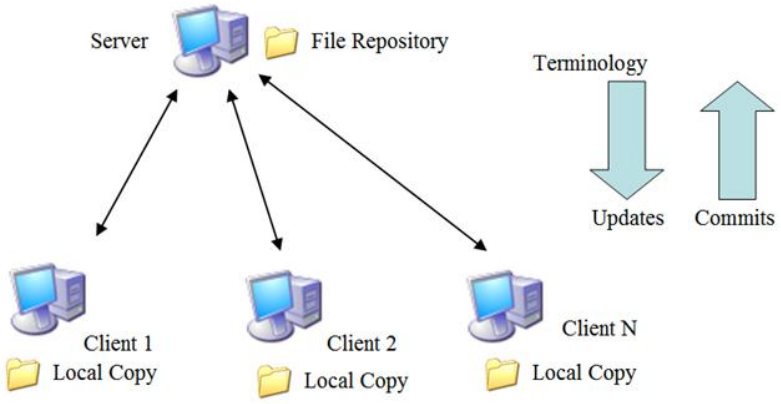


Figure 6 - Fonctionnement de SVN

L'IDE*

Pour développer en C# nous avons utilisé Visual Studio 2010 qui est un ensemble complet d'outils de développement développé par Microsoft et disponible uniquement sur Windows. Il permet le développement d'applications en ASP.net, en Visual Basic, en Visual C++, C# et J#. Cet IDE (Integrated Development Environment) est aujourd'hui le seul outil convaincant pour développer des applications en technologie .NET

Le Formalisme d'analyse

L'UML (Unified Modeling Language) est issu de la fusion du booch, de l'OMT et de l'OOSE, il met à disposition du développeur un panel d'outils majoritairement graphiques permettant de mener de manière rapide et efficace l'analyse d'un projet.

Le logiciel de création de diagramme UML

Argo UML est un logiciel de création et d'édition de schéma UML libre et relativement intuitif par rapport à la plupart de ces concurrents libres.

Le Profiler

The Indiefreaks Game Profiler for XNA est un outil de profiling adapté à XNA. Un profiler est une application permettant l'analyse de la consommation de mémoire et du temps CPU d'une application.

Maintenant que nous vous avons présenté de manière assez exhaustive les différents outils et méthodes que nous avons utilisés, nous allons vous parler de notre organisation.

- ***L'organisation***

Un bon projet commence par une bonne organisation. En effet comment tirer parti de toutes les ressources humaines sur un projet si celles-ci ne sont pas ordonnées ? Nous avons appris à nos dépends au cours de ce projet que cette organisation est plus importante qu'il n'y parait.

Dans le cadre de ce travail, nous avons eu la chance de pouvoir travailler à deux binômes. De plus, quasiment chaque personne de de ces binômes avait des parcours différents, les points faibles et forts de chacun étaient donc généralement différents et complémentaires.

Cette différence peut être une grande force si l'organisation de l'équipe est correctement pensée mais si ce n'est pas le cas on peut arriver très rapidement à un projet statique, qui n'avance pas ou de mauvaise qualité.

Notre premier travail dans ce projet fût de mettre un place une analyse cohérente de notre futur projet toute en commençant à prendre en main les outils qui nous ont servis pour la suite. Une foi cette phase initiale terminée nous avons mis en place un guide de style permettant d'avoir un code propre clair et compréhensible rapidement pour tous les membres de l'équipe.

A cet instant, tout était prêt pour démarrer la phase de développement. Notre première approche fût de laisser libre tout à chacun de faire ce qu'il voulait quand il voulait, … mauvaise idée … nous sommes vite arrivés à une situation où le projet n'avançait peu voire pas et où notre analyse préalable perdait de son efficacité. Tout le monde se gênait dans son travail (travail conflictuel sur une même classe, développement lent d'une classe peu utile à un binôme bloquant l'autre binôme, ...). Nous avons donc adopté une autre stratégie.

En effet nous avons organisé de manière hebdomadaire des réunions entre les deux binômes pour fixer des objectifs personnels. Ainsi chaque semaine chacun de nous avait un objectif précis à remplir pour la semaine d'après. Nous avons pu ainsi gagner en rapidité et en qualité au niveau du projet ainsi qu'en capacité d'organisation en équipe.

- ***Présentation de la solution***

L'un de nos objectifs principaux étant de développer un jeu facilement adaptable et évolutif, nous avons dû mettre en place des solutions et des techniques particulières. Dans cette partie nous allons vous en présenter certaines. La liste n'est pas exhaustive car le jeu se voulant très modulable il serait trop long et peu intéressant de détailler toutes les classes et les techniques employées du projet surtout que beaucoup d'entre elles se ressemblent et partent du même principe.

Commençons par le modèle principal, qui a dirigé la majeure partie de l'analyse du projet.

Modèle global

Le projet s'articule autour du modèle de conception "Modèle-Vue-Contrôleur (MVC)"

Cette architecture permet de réduire les efforts du programmeur lorsqu'il souhaite élaborer un système fondé sur de multiples représentations (les vues) de données identiques (le modèle).

Le lien entre les deux est assuré par un contrôleur qui assure la distribution des informations de façon à ce que les modifications intervenant dans le modèle se reflètent automatiquement au niveau des vues.

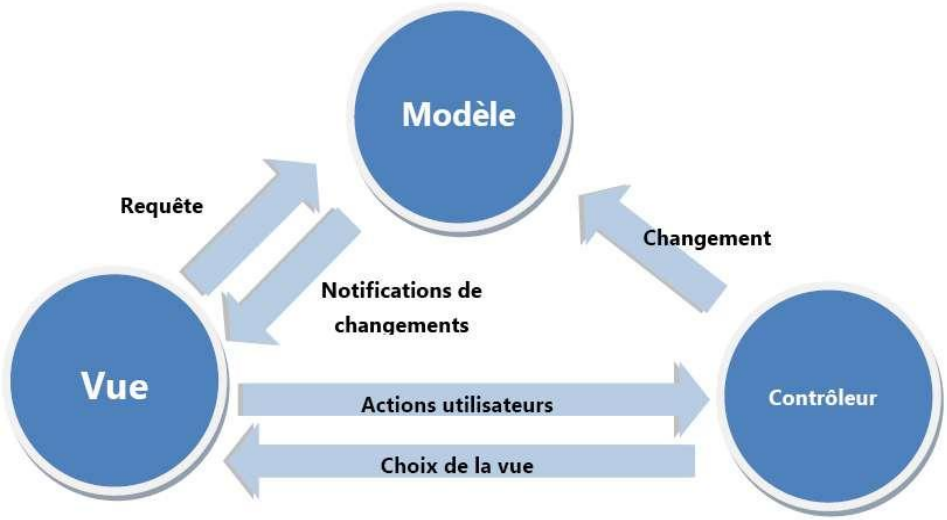


Figure 7 - Modèle MVC

Dans le cas de notre jeu nous avons ainsi découpés nos différentes classes/entités :

Le modèle :

Le modèle contient les différents éléments du jeu. On retrouve alors les différents acteurs d'une partie (joueur, ennemis, blocs, lasers …) mais aussi des classes servant de passerelle entre des données externes et le jeu tels que la gestion des sauvegardes et du son.

Différentes classes du modèle utilisent d'autres concepts d'architecture tels que la fabrique. La fabrique de création (ou factory) est une classe qui n'a pour rôle que de construire des objets. Cette classe utilise des interfaces ou des classes abstraites pour masquer l'origine des objets.

Ce concept nous a permis d'aborder la création des différents éléments intervenant dans une partie de manière simplifiée.

La vue :

La vue concerne tous les éléments visuels du jeu. C'est ici qu'est géré l'affichage des éléments à l'écran. On retrouve ainsi deux principaux types d'éléments que sont :

- les entités d'une partie :

Les différentes entités d'une partie (vaisseau, ennemis, blocs …) se gèrent de manière autonome. Toutes ces entités (du modèle) héritent d'un élément plus abstrait (Element) qui contient un Sprite. Le Sprite est la classe majeure de cette partie de la vue, elle représente un objet graphique affiché à l'écran.

- les écrans :

A chaque écran du jeu (écran de début, menu, game over …) correspond une vue. Celle-ci permet l'affichage à l'écran des différents éléments qui composent l'écran auquel elle est associée. Ainsi l'écran de début par exemple se chargera d'afficher le fond d'écran du jeu ainsi que le texte qui doit apparaitre ("appuyer sur start ou entrer" dans cet exemple-là).

Le contrôleur :

Le contrôleur s'occupe de gérer toutes les interactions entre la vue et le modèle. On a cependant séparé deux types d'interactions.

- les interactions liées aux périphériques de commande :

Afin de permettre l'implémentation de nouveaux types de contrôles dans le futur, et trouvant le contrôle des différents périphériques nativement dans XNA assez laborieux nous avons mis en place un système de contrôle de périphérique. Ce système différencie deux types de périphériques utilisant un rapport avec les positions différent.

- Le positionnement relatif : Certains périphériques utilisent un positionnement relatif, par exemple le clavier. Lors de l'utilisation de la touche droite par exemple on va demander à l'objet de se décaler de x à droite par rapport à sa position courante.
- Le positionnement absolu : Certains périphériques renvoient un positionnement absolu, c'est le cas de la souris par exemple dont on récupérer ses coordonnées à l'écran sans s'occuper de son déplacement vis-à-vis de sa position précédente.

La gestion des périphériques est présentée plus exhaustivement après.

- les interactions liées aux actions du joueur au sein du jeu:

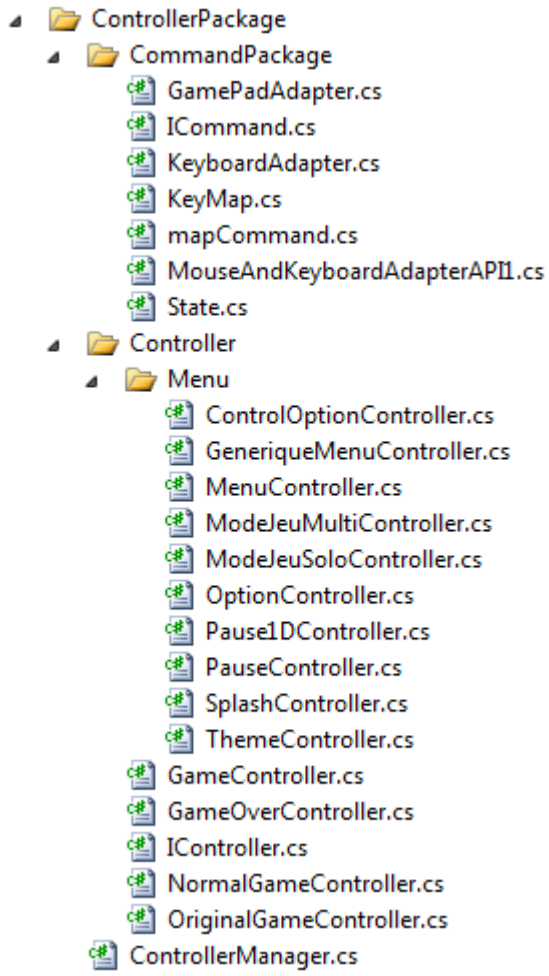
Certaines actions du joueur (la grande majorité) amènent des changements à considérer par rapport à l'état précédent du jeu. Lorsque le joueur décide d'appuyer sur une touche associé au tir par exemple, le contrôleur devra s'occuper de repérer cette action et de notifier à la vue et au modèle la création d'un laser. C'est aussi le cas lorsque le joueur se déplace dans les menus et appuies sur la touche associée à la commande de validation, le contrôleur s'occupera alors du changement d'écran.

Certaines actions ne découlent pas directement d'une action liée au périphérique, c'est le cas notamment des collisions, le contrôleur doit repérer la collision et agir en conséquence (baisse des points de vie, destruction(s) nécessaire(s) ...).

Ainsi pour gérer tous les différents aspects du jeu et pour garder toujours ce niveau d'abstraction nécessaire pour accomplir nos objectifs, nous n'avons pas un unique contrôleur mais une multitude.

Voici la liste de ceux-ci:

Figure 8 - Liste des contrôleurs



Nous voyons bien sur cette capture d'écran les différentes "familles" de contrôleurs et nous retrouvons un mécanisme similaire pour chacune d'entre elles. Ainsi le schéma type utilisé dans la structure du contrôleur du jeu est la suivante:

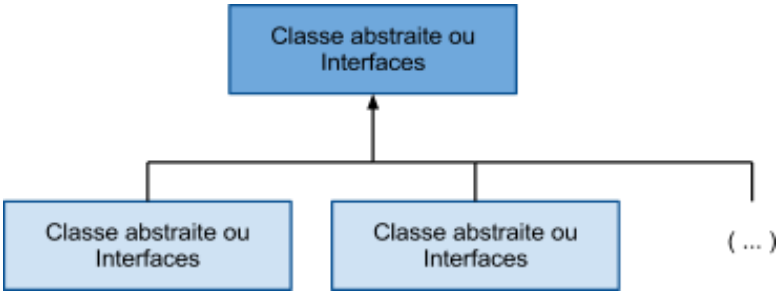


Figure 9 - Structure des contrôleurs

Ensuite chacune de ces "familles" est gérée par un contrôleur principal qui est un peu le chef d'orchestre entre tous les contrôleurs.

Solutions particulières

Le modèle MVC est l'élément clé de notre analyse, mais il est loin d'être la seule solution que nous ayons mise en place pour atteindre nos objectifs. Pour réussir à développer un jeu le plus adaptable possible nous avons dû réfléchir à des solutions d'abstraction le plus haut possible.

Notre première réflexion concerna la gestion des périphériques.

Gestion des périphériques

Actuellement de nombreuses nouvelles interfaces de contrôle viennent bouleverser notre manière d'utiliser nos programmes habituels. Pour rendre ce jeu attractif et suivre cette évolution matérielle, l'abstraction du mode de contrôle fût donc une chose très importante. Nous avons donc conçu le système suivant qui nous permet d'ajouter très facilement la gestion d'un nouveau périphérique.

Ce niveau d'abstraction s'est vite heurté à un problème, comment gérer, avec la même fonction, une souris et un clavier ?

En effet le clavier indiquera une direction de l'objet contrôlé par rapport à sa position à un instant t, tandis que la souris, elle, va donner directement la position finale de l'élément.

Nous avons décidé de regrouper les périphériques en deux groupes. Comme l'illustre le schéma ci-dessous, nous avons une gestion de type directionnel qui, comme expliqué précédemment, incrémente dans une direction ou une autre la position actuelle ainsi que la gestion de type positionnel qui donne directement la position finale du déplacement.

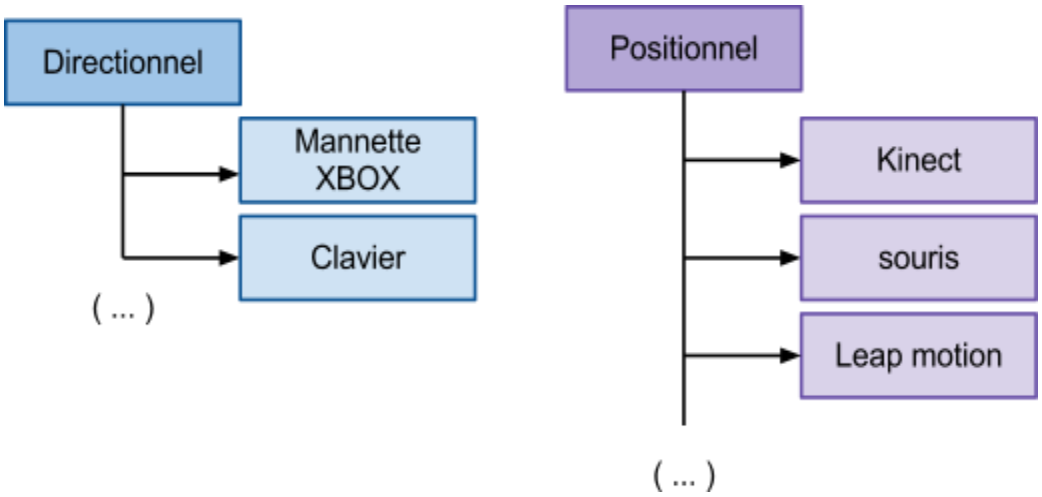


Figure 10 - Types de périphériques

Toutes les interfaces présentes sur le schéma ci-dessous ne sont pas actuellement gérées mais il représente bien le système que nous avons voulu mettre en place. Ainsi nous savons que, peu importe la technologie utilisée, par exemple le futur système leap motion, nous pourrons facilement adapter notre jeu en concevant un nouvelle classe du type positionnel.

Ces deux groupes héritent eux même d'une interface périphérique qui permet de mettre en forme un objet état qui sera utilisé dans le reste du jeu. Ainsi au lieu d'exploiter directement

les informations issues du périphérique, on abstrait complètement celui-ci et on utilise les informations de cette classe état.

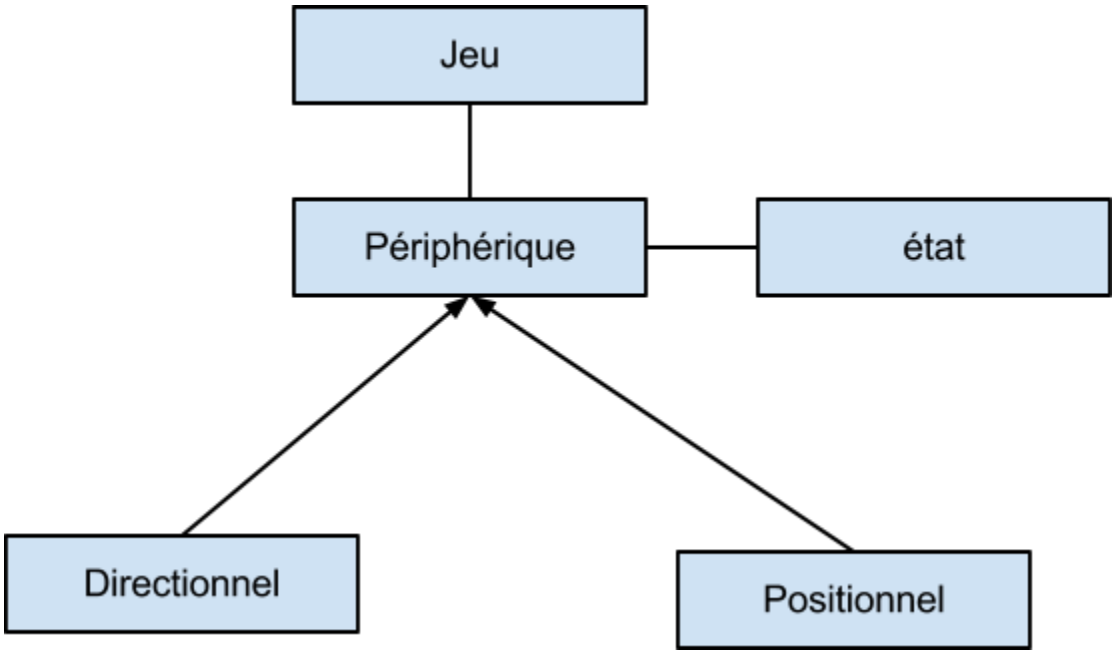


Figure 11 - Structure des périphériques

Cette technique est une version adaptée du design pattern Adapter.

Gestion des éléments du jeu

Dans le *Space Invaders* traditionnel nous n'avons qu'un nombre limité "d'entités". Des Invaders d'aspects variés mais de comportements relativement semblables, un vaisseau (ou laser inter-galactique selon les interprétations), des obstacles et des lasers.

La solution la plus simple aurait été d'implémenter de manière fixe ces éléments. Cependant, nous aurions perdu toute flexibilité dans le développement futur du jeu. Nous avons donc, ici encore, décidé d'apporter un niveau d'abstraction assez important. Ainsi tous les éléments du jeu sont définis de la même manière par une classe "Elément" qui contient toute les informations nécessaires à leur affichage ainsi qu'un attribut de type comportement qui va définir le comportement de l'objet.

En créant de nouveaux "héritiés" de la classe Comportement nous avons la possibilité de modifier le comportement de tous les éléments du jeu très rapidement sans avoir à modifier trop profondément la structure du jeu. Voici un schéma simplifié de la structure que nous avons utilisée.

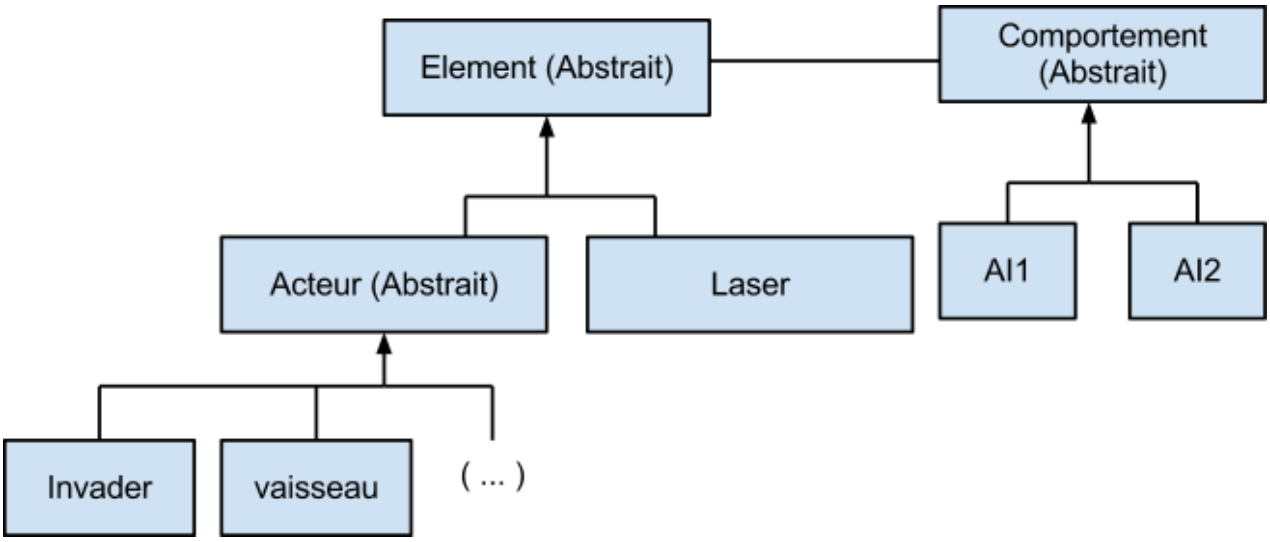


Figure 12 - Structure de la gestion des éléments

Gestion du son

La gestion du son fait appel à un outil externe présenté précédemment qu'est XACT. Cette API est en fait une surcouche qui permet d'utiliser de manière transparente aussi bien XAudio (dédié au son sur Xbox) que DirectSound (dédié au son sur plateforme Windows XP et plus anciens) ainsi que la nouvelle pile audio de Windows Vista et Seven.

Le son devient un projet à part entière dans lequel il est possible de gérer plusieurs banques et catégories de son au format WAVE ou AIF(F). Un fichier projet d'extension .xap est ainsi généré ainsi que deux dossiers contenant respectivement les données de sons à utiliser pour Windows et/ou pour Xbox. Ce fichier .xap, qui reprend une structure de type json, est naturellement interprété par XNA. Ainsi selon si le code est compilé pour Xbox ou pour Windows, il se chargera de récupérer les données de sons dans le dossier approprié.

Cet outil, bien qu'ayant une utilisation assez contraignante, permet une compatibilité totale des différents sons avec Xbox et des différentes versions de Windows ainsi qu'une abstraction avec le projet.

De plus, au sein même du projet, la gestion du son a été vue de manière externe. Ainsi une classe de gestion de son a été créée. Celle-ci permet de jouer certains types de sons par les différents éléments du projet sans réellement savoir ce qu'il se passe derrière. Par exemple lors d'un tir du joueur, indépendamment de la création du laser, le gestionnaire de son est appelé en lui demandant de jouer un son correspondant à un tir de laser.

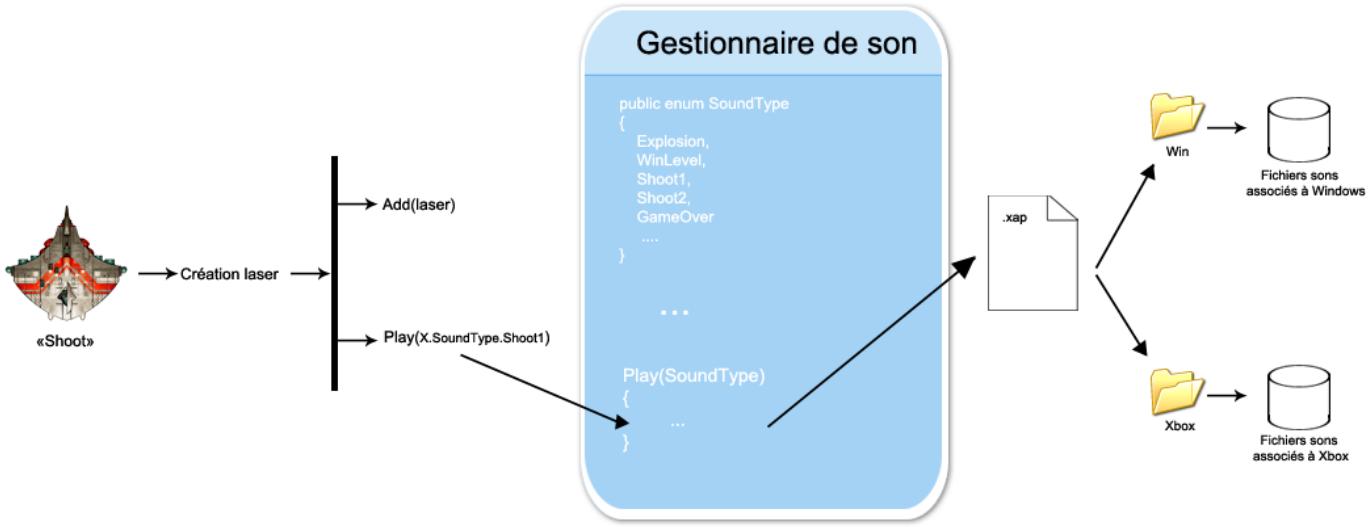


Figure 13 - Fonctionnement de la gestion du son

3. Résultats et discussion

a. Le produit réalisé

Le jeu est à ce jour fonctionnel, la dernière release est une version bêta du produit final. Nous avons mis en place, conformément à nos attentes, deux modes de jeu différents ainsi que la possibilité de jouer en multijoueur (sur une même machine).

Comme énoncé précédemment le niveau d'abstraction est assez élevé, il est ainsi très aisé de rajouter des types de monstres, comportements spécifiques, niveaux... voire même de nouvelles fonctionnalités à part entière comme par exemple la récolte d'objets au cours du jeu qui permettront l'utilisation de facultés spéciales.

Le jeu dispose d'un menu de navigation complet, fonctionnel et intuitif. Certain écrans d'option ne sont pas terminés mais la structure du menu est entièrement implémenté. Voici le schéma fonctionnel représentant ce menu.

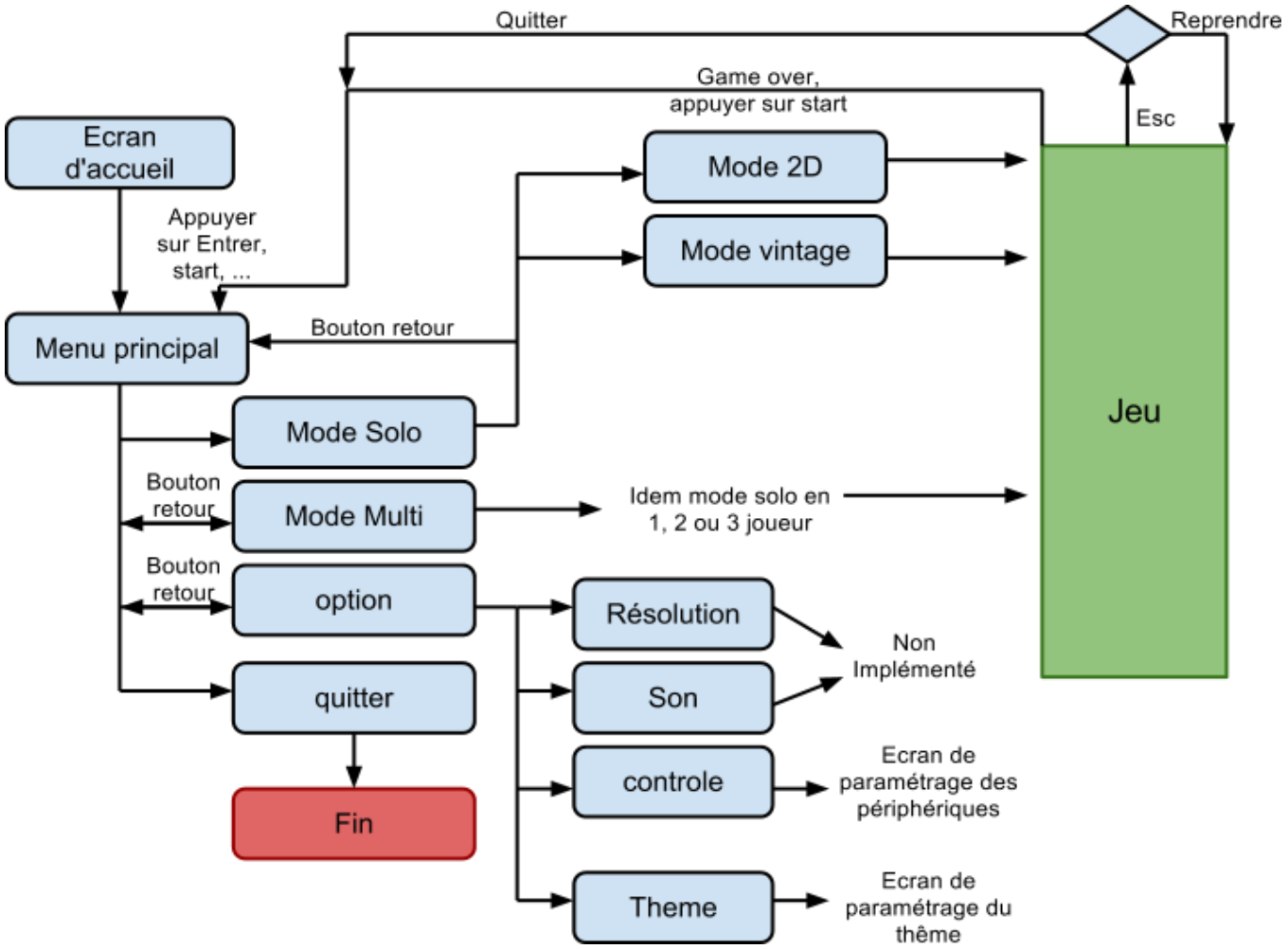


Figure 14 - Diagramme d'état du jeu

Cependant nous n'avons pas eu le temps de réaliser tous les objectifs optionnels. Ainsi les sauvegardes, bien que partiellement implémentées ne sont pas disponibles à l'heure actuelle. Comme nous l'avions prévu, nous n'avons pas non plus eu le temps de développer l'aspect visuel ainsi que le scénario pour améliorer l'expérience utilisateur, cependant nous sommes parvenus à concevoir un système très efficace permettant facilement de les mettre en place après coup.

b. Le produit dans son fonctionnement

Le principe du jeu original, à savoir le "Space Invaders" a bien été reproduit. Cependant nous avons implanté deux différents modes de jeu, correspondant à des univers et des gameplay différents :

Le mode vintage

Ce mode de jeu correspond intégralement au jeu “Space Invaders” d'origine. Le déplacement du vaisseau du joueur ne s’effectue que de manière horizontale, les ennemis se rapprochent de manière linéaire et à vitesse constante.

Figure 15 - Capture du mode vintage

Le mode 2D

Afin de moderniser le jeu initial, nous avons mis en place un autre mode de jeu permettant un déplacement à la fois horizontal et vertical ainsi qu’une orientation pour le vaisseau. Ce mode de jeu fait appel à des Intelligences Artificielles (IA) plus complexes ainsi qu’à des Sprites disposant d’animations.

Figure 16 - Capture du mode 2D en multijoueur

Ces deux modes de jeu contiennent néanmoins tous deux la possibilité de jouer à plusieurs ainsi que d'utiliser différents périphériques (clavier, souris et manette XBOX).

Ils utilisent aussi les mêmes représentations graphiques des différentes entités (vaisseau joueur, ennemis …) qui sont cependant changeables via un thème à choisir dans le menu.

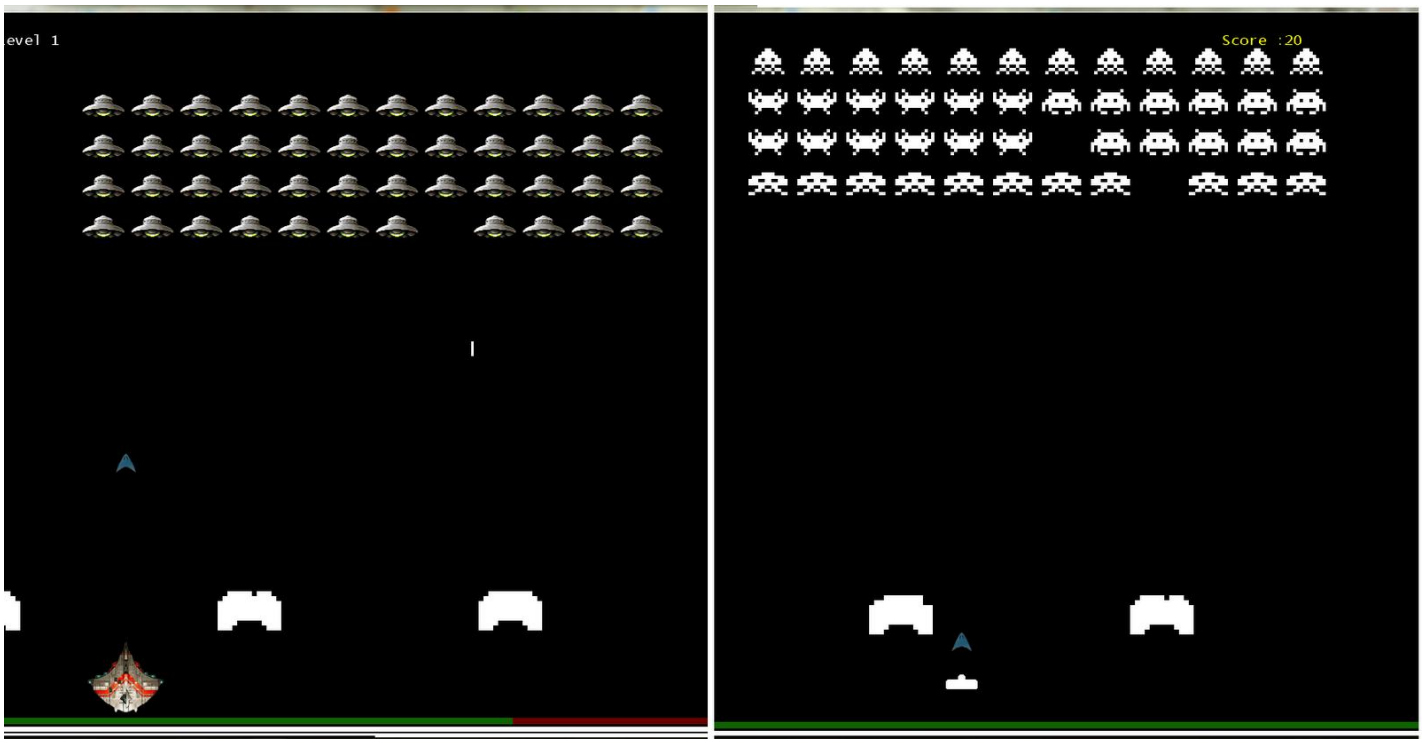


Figure 17 - Capture illustrant la différence entre deux thèmes

Le joueur dispose d'un score qu'il se doit d'incrémenter le plus possible en tuant un maximum d'ennemis et en survivant le plus longtemps possible. Indépendamment de celui-ci il dispose d'un objectif qui consiste à tuer un certain nombre d'ennemis et ainsi pouvoir passer au niveau suivant. Les niveaux suivant contiendront plus d'ennemis voire de nouveaux types d'ennemis plus puissants. La partie se termine alors lorsque la vie du joueur se retrouve à zéro.

Conclusion

Ce projet s'avère être, d'un point de vue personnel, un grand succès. Il nous a en effet permis de revoir nos méthodes d'organisation en groupe, d'apprendre à maîtriser une technologie que nous ne connaissions que très peu et de découvrir très rapidement le monde du jeu vidéo.

L'étude réalisée en début de projet semble viable et a pu faire ses preuves tout au long du développement du jeu. Ainsi, grâce à celle-ci, nous n'avons perdu que peu de temps lorsque certaines erreurs furent commises ou qu'une nouvelle idée s'est présentée à nous.

Au niveau de la réalisation, elle n'est à ce jour pas assez avancée pour être distribuée pour le grand public. L'application fonctionne sur Xbox 360 et PC, les modes de jeux traditionnel et 2D marchent parfaitement, la base de l'application est solide et facile à maintenir, les menus de navigation sont opérationnels, les périphériques pour la plupart bien gérés mais tout l'aspect visuel, sonore, le scénario et tout ce qui fait l'attrait d'un jeu est manquant.

De plus, même si les fonctions sont pour la plupart déjà implémentées, le jeu manque réellement de flexibilité du point de vue du joueur (peu de réglages dans le menu option sont disponibles). Les prochains objectifs qu'ils pourraient y avoir dans le développement de ce jeu concernent donc plus l'interface et le scénario. Cependant, étant donné que presque toutes les fonctions nécessaires au développement de ces deux aspects ont déjà était implémentées, le travail à fournir reste donc assez limité pour atteindre une version grand public.

Nous pouvons donc affirmer que les objectifs principaux du projet ont été remplis. Cette application, bien que inachevée, a donc des perspectives d'avenir intéressantes. En effet, le mode multijoueur (coopération) apporte un réel intérêt vis à vis du jeu de base. Ainsi avec une phase de test pour éliminer les quelques bugs restants, un travail sur l'interface utilisateur ainsi que sur les dernières fonctionnalités inachevées, ce jeu pourrait parfaitement être distribué et devenir une application compétitive dans la catégorie où elle se situe, notamment grâce à la facilité de la mettre à jour en ajoutant des niveaux, des boss, etc... De plus la portabilité relativement aisée sur Windows phone pourrait donner une nouvelle dimension à ce projet.

Lexique

Common Language Runtime (CLR) : Nom choisi par Microsoft pour le composant de machine virtuelle du framework .NET. Il s'agit de l'implémentation par Microsoft du standard Common Language Infrastructure (CLI) qui définit l'environnement d'exécution des codes de programmes. Le CLR fait tourner une sorte de bytecode nommé Common Intermediate Language (CIL). Le compilateur à la volée transforme le code CIL en code natif spécifique au système d'exploitation. Le CLR fonctionne sur des systèmes d'exploitation Microsoft Windows. Le CLR est à .NET ce que la JVM est à Java.

Forge : Système de gestion de développement collaboratif de logiciel. L'objectif d'une forge est de permettre à plusieurs développeurs de participer ensemble au développement d'un ou plusieurs logiciels à travers le réseau Internet.

Framework : En programmation informatique, un framework est un kit de composants logiciels structurels, qui servent à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel (architecture). En programmation orientée objet un framework est typiquement composé de classes mères qui seront dérivées et étendues par héritage en fonction des besoins spécifiques à chaque logiciel qui utilise le framework.

IDE : Environnement de développement intégré, c'est un programme regroupant un ensemble d'outils pour le développement de logiciels.

Shoot them up : Type de jeu originaire des salles d'arcade dérivé du jeu d'action dans lequel le joueur dirige un véhicule ou un personnage devant détruire un grand nombre d'ennemis à l'aide d'armes de plus en plus puissantes, au fur et à mesure des niveaux, tout en esquivant leurs projectiles pour rester en vie.

Design pattern : Ou patron de conception est un arrangement caractéristique de modules, reconnu comme bonne pratique en réponse à un problème de conception d'un logiciel. Il décrit une solution standard, utilisable dans la conception de différents logiciels.