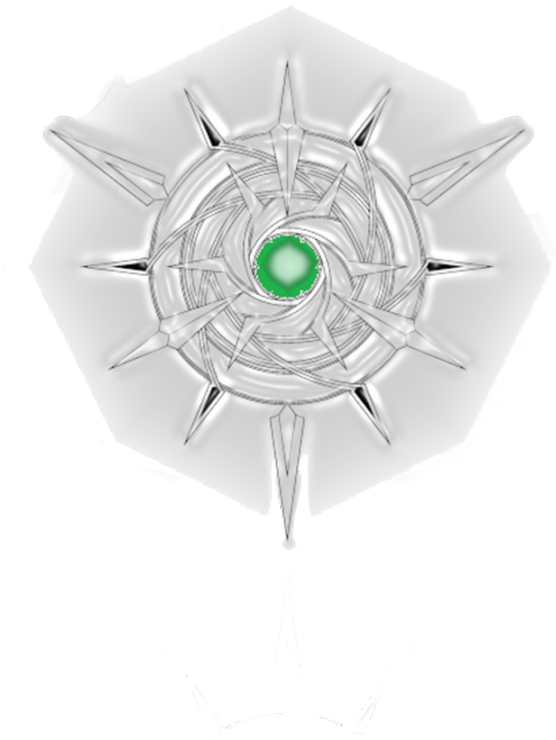
Rapport de projet - Dawn of Gates

0_Dev Team
Lariau Steven (lariau_s)
Brenet Timothée (brenet_t)
Lafargue Victor (lafarg_b)



Epita S2 - XX/06/15

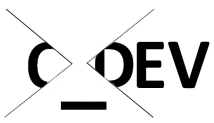
Remerciements

De nombreuses personnes nous ont apportés de l'aide dans la la réalisation de notre projet, sans la communité du forum d'unity pour le jeu et de stack overflow pour le c# en général et le site nous n'aurions pu faire face à tous les problèmes rencontrés lors du développement.
Nous tenons aussi à remercier HTML5Up, le site web sur lequel nous avons été cherché le template css pour notre site web.

Introduction

Nous sommes trois, Steven, Lariau Timothée Brenet, et Victor Lafargue, à avoir travailler sur notre projet, intitulé Dawn of Gates.
C'est un jeu en réalité virtuelle, développé avec et pour l'Oculus Rift.
De type action, le personnage principal met en scène un ninja, dans un environnement fermé, ayant des capacités hors du commun.

Sommaire



1 Présentation du projet

1.1 Un jeu en réalité virtuelle

Dawn of Gates est un jeu en réalité virtuelle, cela signifie que le joueur ne se contente pas seulement de jouer devant son écran, il est réellement émergé dans le jeu. Il met un casque sur sa tête, possédant deux écran, un pour chaque oeil, ayant pour but de faire croire au joueur que celui-ci est réellement dans le jeu.
Il existe plusieurs modèles de casques de réalité virtuelle, mais nous avons choisi Oculus Rift, car celui-ci est l'un des pionniers dans la réalisation d'un casque de jeu en réalité virtuelle pour grand public. Bien qu'il ne soit pas encore sortie en version définitive, la version développement est déjà assez stable, et de plus une assez grande communauté est présente sur Internet pour nous aider en cas de problèmes.
L'Oculus Rift n'est pas une console de jeu, à lui tout seul il ne permet pas de joueur à des jeux en réalité augmenté. C'est un accessoire, il se branche à un ordinateur afin de pouvoir fonctionner. Afin de développer des jeux Oculus, les créateurs ont mis à notre disposition, gratuitement, un SDK, permettant de réaliser des applications pc ou mobiles. Le site web propose en plus des versions de ce sdk s'intégrant aux moteurs de jeu Unreal Engine et Unity. C'est ce dernier que nous avons choisi, afin de pouvoir réaliser notre jeu en réalité virtuelle sur Unity. Nous avons choisi cette plateforme afin de ne pas se contenter d'un jeu 3D banal. Le développement d'un jeu 3D oculus demande certes plus de travail, et se révèle plus difficile, mais au final nous sommes bénéficiaires, car les difficultés rencontrés nous forment et nous rendent meilleurs.

1.2 L'histoire

Le synopsis de notre histoire commence par le personnage principal qui se ballade avec sa petite amie, qui se fait alors volée son collier par un méchant. Notre personnage, qui se révèle être un ninja, poursuit alors le renégat afin de récupérer le collier. Il arrive alors dans un étrange bâtiment, où il découvre alors des capacités spéciales. Il devra alors les utiliser afin de combattre les différents ennemis et passer les obstacles qu'il rencontrera au fil des niveaux, jusqu'à atteindre le dernier niveau, afin de combattre le boss final, le voleur, afin d'espérer pouvoir récupérer le collier.

1.3 Gameplay

1.3.1 Le personnage

Le ninja possède de nombreuses facultés, il peut notamment lancer des shurikens. Il peut ainsi combattre ses ennemis. Mais les shurikens ont aussi une autre utilité. Lorsque ceux-ci touchent un mur, ils ouvrent un "portail", c'est à dire une porte qui mène vers un autre portail ouvert précédemment. Le ninja peut alors se téléporter, et utiliser cette capacité afin de venir à bout des différentes énigmes du jeu.
Il a aussi une capacité de "slow motion", c'est à dire qu'il peut ralentir le temps, pendant un laps de temps limité. Cette faculté, utilisé à bon escient, aide aussi notre héros à venir à bout de tous les dangers présents sur sa route.

1.3.2 Environnement

Le héros se déplace ainsi de niveaux à niveaux. Le jeu se situe en lieu clos, le personnage reste dans le bâtiment et parcours les salles une à une. Ces salles ont un style graphique mélangeant à la fois un style asiatique et un style toon. Une musique asiatique accompagne aussi le personnage tout au long de ses péripéties, afin d'améliorer l'ambiance de jeu.

1.3.3 Les ennemis

Notre ninja rencontre plusieurs types d'ennemis. Tous d'abord il y a des tourelles, elles sont fixes, et peuvent uniquement tirer des flèches blessant le joueur. Certaines visent le joueur, d'autres peuvent uniquement tirer en ligne droite.
En plus, il existe aussi des ennemis mobiles, ils effectuent des patrouilles dans les différents niveaux, et lorsqu'ils détectent le joueur, ils lui courent après et l'attaquent avec une épée.

1.3.4 Le boss final

Le boss final est différent des autres ennemis. Il est bien plus difficile à tuer, et possède une intelligence artificielle dédiée. Il est impossible à vaincre sans une technique spéciale.

2 La team

2.1 Modifications du groupe

Au départ il y avait un quatrième membre dans notre équipe, Idris Sequeira, mais il a décidé d'arrêter Epita une fois la première année terminée, et ne s'impliquait plus du tout dans le projet. Nous avons donc décidé de l'exclure du groupe.

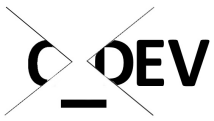
2.2 Présentation des membres

2.2.1 Steven Lariau

Ce projet sera sans aucun doute une grande aventure, vouloir réaliser un jeu en réalité virtuelle, c'est un choix ambitieux, mais qui me permettra de découvrir de nouveaux domaines de la programmation. Bien que je connaisse déjà un peu le développement web, notamment javascript avec node, mais aussi la programmation en Java, C#, C++, tout ce qui relève de la 3D et de l'IA par exemple sont des notions que j'ai hate de découvrir afin de réaliser ce projet. Etant un expert dans l'art de "commencer des projets colossaux, d'y passer des journées entières dessus en partant dans tous les sens puis d'arrêter complètement", j'espère ainsi apprendre à mener un véritable projet à son terme, à travailler en équipe, et à faire face aux différents problèmes sans se décourager.

2.2.2 Timothée Brenet

Passionné d'informatique depuis la quatrième, j'ai appris la programmation grâce au site du zero. J'ai d'abord suivi les cours sur le C puis je me suis tourné vers les langages web comme l'html, php, javascript et drupal. Je me suis finalement majoritairement concentré sur le développement en Java.J'ai participé à quelques projets en groupe de programmation en Java, la plupart centrés sur la creation d'un jeu en 2D grâce a la librairie lwjgl. Ce projet de sup, m'intéresse donc énormément et je pense qu'il va me permettre de me faire progresser en C# notamment et de créer mon premier jeux 3D avec Unity. Je suis donc très motivée pour ce projet, je travaillerais surtout sur la programmation en C# et le portage sur l'Oculus Rift.



2.2.3 Victor Lafargue

Je m'appelle Victor Lafargue, je suis originaire de Seine-Et-Marne près de Fontainebleau. J'ai toujours été fasciné par la technologie, l'électronique et l'informatique. Ce sont les domaines d'expertises d'Epita et c'est pourquoi j'ai choisi cette école. J'ai déjà un peu d'expérience en informatique surtout en C# et en python que je connaissais avant de venir ici. J'ai aussi déjà essayé de créer le début d'un jeu-vidéo avec OpenGL ce qui pourra peut-être aider. Ce projet informatique en équipe et sur le long terme est une superbe occasion d'avoir un avant-gout de ce qu'est le développement informatique. Il me permettra, j'en suis sûr d'énormément progresser de démontrer ma détermination à réussir et à terminer un grand projet.

2.3 Organisation

Pour travailler en groupe, nous utilisions tout d'abord Github. Cette plateforme de versionning nous permettait ainsi de pouvoir partager et travailler à plusieurs sur le projet sans problème.
Nous utilisons aussi Skype afin de communiquer, de prendre des décisions sur le projet, de demander son avis à d'autres personnes du groupe, etc. Pour les trois soutenances nous nous sommes plus ou moins organisées de la même façon. La première étape était de se répartir les tâches, de décider qui allait faire quoi. Ensuite, nous réalisions chacun nos tâches respectives, avant de se réunir et de mettre notre en commun pour régler les incohérences et les différents bugs.

2.4 Plannings de répartition des tâches

2.4.1 Première période

	Timothée	Steven	Victor	Total
Code (Base)	20	20	20	60
Graphismes	0	0	40	40
Animations	10	0	30	40
Niveaux	6	6	8	20
IA	15	20	15	40
Audio	0	0	0	0
Site web	0	50	0	50
Réseau	0	0	0	0

2.4.2 Deuxième période

	Timothée	Steven	Victor	Total
Code (Base)	30	25	25	80
Graphismes	0	0	60	60
Animations	20	0	40	60
Niveaux	25	25	5	55
IA	25	30	15	70
Audio	25	0	0	25
Site web	0	60	0	60
Réseau	0	10	10	20

2.4.3 Troisième période

	Timothée	Steven	Victor	Total
Code (Base)	40	30	30	100
Graphismes	0	0	100	100
Animations	20	0	80	100
Niveaux	35	35	30	100
IA	45	35	20	100
Audio	33	34	33	100
Site web	10	90	0	100
Réseau	0	50	50	100

3 Capacités du personnage principal

3.1 Déplacements

3.2 Les caméras

3.3 Le lancer de shurikens

3.4 Les portails

L'utilisation de portails est l'une des fonctionnalités les plus importantes du jeu, mais aussi l'une des plus complexes à réaliser. J'ai fait de nombreuses tentatives et de nombreux échecs avant d'arriver à une solution fonctionnelle et viable. Cette partie du développement, fut, pour ma part, l'une des plus difficiles à réaliser, j'y ai passé beaucoup de temps, mais ie ne me suis iamais

découragé, et j'ai finalement eu, un résultat, qui n'était certes pas parfait, mais du moins à la hauteur de nos espérances.
Les portails permettent à notre personnage de se déplacer plus rapidement, et de franchir des obstacles qui ne pourraient pas l'être en se déplaçant normalement. Le ninja possède deux portails, qu'il peut ouvrir sur n'importe qu'elle surface. Lorsque deux portails sont placés, le personnage, peut alors se déplacer du premier portail au second, et inversement, en rentrant dans l'un des portails.

3.4.1 Ouverture et placement

Un portail est sensé s'ouvrir lorsqu'un shuriken lancé par le ninja touche un mur. De plus, si il n'y a qu'un seul portail placé, celui-ci reste fermé. Ce n'est que lorsque les deux portails sont placés qu'ils s'ouvrent. Le clic de la souris détermine quel portail est lancé. Le clic gauche correspond à un portail gauche, et le clic droit à un portail droit.

Pour cela, lorsque le shuriken est lancé, un raycast est réalisé afin de savoir quel objet le shuriken va rencontrer. Le script de placement ne commencera que si la cible est un mur. Pour détecter si c'est un mur ou non, j'ai mis un Tag nommé "room" sur toute la pièce. Je n'ai plus qu'à vérifier si le tag de l'object trouvé par le raycast est bien "room". Si c'est le cas, je calcule la distance entre le shuriken et le mur, et lorsque le shuriken a parcouru cette distance, le portail est placé à l'endroit où se trouve le shuriken, et le shuriken est détruit.
Le raycast me donne aussi le vecteur de la face touchée. J'utilise ce vecteur afin que le portail soit bien posée à plat, quelle que soit la face du mur, du sol ou du plafond.

Ensuite viens le problème d'ouverture des portails. Les portails ne sont ouverts uniquement que si les deux portails sont placés.
Par défaut, lorsqu'un portail est placé, celui-ci est fermé. Si l'autre portail est ouvert, alors cela signifie que les deux portails sont placés, et l'animation du portail est lancée. Si le portail est placé mais fermé, cela signifie que c'est la première que les deux portails sont placés, et dans ce cas l'animation d'ouverture est activée sur les deux portails. Enfin, si l'autre portail n'a pas encore été lancé, alors celui-ci reste fermée.

3.4.2 L'effet portal

L'effet portal, c'est que lorsque le joueur regarde un portail, il voit à travers celui-ci, ce qu'il y a en face de l'autre portail. Cela donne un effet de téléportation bien plus réaliste et impressionnant. Pour réussir cela, j'ai beaucoup réfléchi, fait de nombreuses tentatives, regardés sur les forums, des vidéos.
Ma première idée était tout d'abord de réussir à faire un simple miroir. J'espérais ensuite pouvoir l'adapter aux portails, le premier portail serait un miroir du second, et inversement. J'ai trouvé sur internet un tutoriel expliquant comment faire un miroir avec un script et un shader, mais celui-ci était très complexe, et je n'ai pas réussi à bien le comprendre pour pouvoir l'adapter à mon problème.

Finalement, en cherchant sur les forums, j'ai trouvé une solution beaucoup plus simple : les texture renderer. Ils permettent de sauvegarder tout ce qu'enregistre une caméra. Je n'ai alors plus qu'à placer une caméra sur chaque portail. Le portail de gauche à un matériel possédant la texture renderer sauvegardé par la caméra du portail de droite, et inversement. Ainsi j'obtiens l'effet désiré.
Les textures renderer ont une certaine résolution, plus celle-ci est élevée, plus l'image serra de meilleure qualiité, mais le jeu demandera alors plus de ressources. J'ai donc du faire un compromis entre qualité et performance, mais les portais restent tout de même réalistes.

Bien que le résultat final est très simple et court à implémenter, il m'a fallu beaucoup de temps de recherches et de réflexions, ainsi que de faux pas, avant d'arriver à cette solution. Dans cette situation, mes connaissances en programmation ne m'ont pas réellement servi, c'est surtout la recherche et la connaissance des fonctionnalités d'Unity qui a fait la différence.

3.4.3 La téléportation

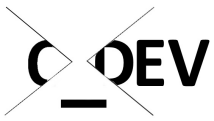
La téléportation décuple les possibilités de déplacement du personnage. Elle permet d'ajouter un gameplay intéressant pour les différents niveaux du jeu. Le principe de la téléportation est que le joueur rentre dans un portail, il ressort par un autre portail. La aussi j'ai pas mal réfléchi avant d'arriver à l'implémentation finale. Tout d'abord, j'ai pensé à dédoubler le personnage lorsqu'il commençait à rentrer dans un portail. Son double étant en train

de sortir de l'autre portail. Mais cette solution était bien trop compliquée à implémenter et aurait complexifié tous les autres scripts du personnage.
Finalement j'ai opté pour une solution plus simple, lorsque le ninja rendre dans le portail, il ressort immédiatement par l'autre portail, avec l'orientation adéquate.
Pour cela, chaque portail possède un box collider. Lorsque le ninja rentre en collision avec le box collider d'un portail, il est téléporté à la même position et la même rotation que l'autre portail. Malheureusement, cela ne fonctionnait pas, car une fois le ninja téléporté, il déclenchait alors le box collider de l'autre portail, et se retéléportait, et ceci indéfiniment. Pour régler le problème, le personnage avance automatiquement tout droit afin de sortir du collider.

L'implémentation de la téléportation fut assez laborieuse, de nombreux essais furent réalisés avant d'arriver à ce résultat, notamment au niveau de l'orientation et de la position du personnage, afin d'éviter que le personnage sorte dans le sens opposé par exemple, et d'éviter les cas de téléportation infini. Néanmoins certains cas subsistent, comme le placement d'un portail au sol, ainsi qu'un autre au plafond, juste au-dessus du premier, mais le joueur peut toujours s'en sortir en se déplaceant pendant la chute.

3.5 Le slow motion

Le slow motion, c'est la capacité du ninja à ralentir le temps. Lorsque cela le produit, la vitesse de tous les objects et de tous les ennemis est considérablement ralenti. Seule la vitesse du joueur reste la même.
Pour cela j'utilise une variable de "temps", un float, qui par défaut est à 1. Tous les objets influencés par le slow motion ont leur vitesse multiplié par ce nombre. Lorsque le slow motion est activé, la valeur de cette variable change. Pour ralentir le temps par n, il suffit de mettre la variable à 1/n. Ainsi, la vitesse de tous les objects affectés par le slow motion est divisée par n. Une fois le slow motion désactivée, la variable revient à 1.
L'implémentation du slow motion en elle-même a été assez courte, mais avant cela, il a fallu se concerter, afin de savoir s'il fallait limiter l'utilisation du slow motion ou pas. De plus, le choix du taux de ralentisement a aussi un impact important sur notre jeu. Un taux trop faible l'aurait rendu peu utile, alors qu'un taux trop élevé aurait rendu le jeu ennuyeux. J'ai donc essayé de nombreuses fois le slow motion, avec des taux différents, afin de choisir celui qui me paraissait le plus approprié.



Pour activer et désactiver le slow motion, il suffit d'appuyer sur la touche espace. Afin d'ajouter du gameplay, nous avons préféré limiter l'utilisation du slow motion. Le ninja a une certaine quantités d'énergie, qui diminue lorsque le slow motion est activé, et qui augmente lorsque le slow motion est désactivé. L'activation du slow motion si le personnage n'a plus d'énergie. Cela permet au joueur d'éviter certains ennemis ou projectiles en activant le slow motion, et grâce à la limite d'utilisation, il ne peut pas l'utiliser en permanence, et doit donc s'en servir à bon escient.

3.6 La vie

Le joueur possède une certaine quantités de vie, qui diminue lorsqu'un ennemi l'attaque. Il meurt lorsqu'il n'a plus de vie.

4 Fonctionnement des niveaux

4.1 L'audio

La difficulté de la partie audio ne réside pas dans l'intégration des sons aux jeux, mais dans les choix de ceux-ci. Pour la musique de fond de notre jeu, nous devons trouver une musique de style asiatique, correspondent bien à l'ambiance du jeu. Après plusieurs recherches, j'en ai trouvé une faisant l'affaire.
Pour l'intégrer à Unity, je me suis servi tout d'abord de la camera comme audio listener, il permet d'entendre les sons. De plus, plus un objet produisant du son est éloigné de l'audio listener, plus le son résultant sera faible, et inversement. Ainsi, plus un objet est près du ninja, plus il produira un bruit fort, ce qui est l'effet voulu.
La caméra a en plus une audio source, qui est le fichier son de la musique de fond, ainsi, elle a en permanence la même intensité sonore.

La musique de fond n'est qu'une petite partie de l'audio de notre jeu. Bien qu'elle contribue fortement à l'ambiance, cela ne suffit pas. Il faut rajouter en plus des bruitages, lorsque le joueur lance un shuriken, que l'ennemi utilise son épée, qu'un portail s'ouvre, qu'un personnage ou un ennemi est touché. Plus ou moins toutes les interactions avec le jeu nécessiterait des bruitages,

afin de permettre une meilleure immersion dans le jeu. L'implémentation des sons n'est pas très difficile à réaliser, il suffit de mettre une source audio sur les objets concernés, puis il est ensuite possible de joueur, mettre en pause, ou arrêter la source audio depuis le script.
La difficulté réside dans le choix même des bruitages. Il faut aller sur un site web proposant des bruitages libres et gratuits, et en trouver qui correspondent bien à ce que nous voulons, mais aussi à notre style de jeu, c'est à dire un style asiatique, afin de plonger le joueur dans l'ambiance des différents niveaux.

4.2 Transition entre les différents niveaux

Nous avons longuement réfléchi à comment passer d'un niveau à un autre. La première idée était de mettre une porte menant à des escaliers, qui permettrait d'accéder au niveau suivant. Finalement, nous avons décidé d'utiliser une toute autre approche.
Afin de passer entre les niveaux différents, qui correspondent à différents étages du bâtiment, le personnage utilise des ascenseurs. Il en existent de deux types : les ascenseurs de début de niveau, et les ascenseurs de fin de niveau. Chaque niveau possède les deux types d'ascenseurs, excepté le premier et le dernier niveau.
Ces ascenseurs sont placés dans des endroits stratégiques. En début de niveau, ils sont toujours assez éloigné des premiers ennemis, afin d'éviter que le joueur se fasse attaqué à peine sorti de l'ascenseur. Quand à l'autre ascenseur, il est placé à l'opposé du niveau, dans un endroit aussi dénudé d'ennemis, montrant bien que le niveau est terminé.
Le joueur commence dans l'ascenseur de début de niveau. Celui est fermé, puis s'ouvre automatiquement après quelques secondes, permettant au joueur de découvrir le niveau.
L'ascenseur de fin de niveau est ouvert et possède un box collider. Lorsque le joueur le déclenche, les portes se referment après quelques secondes. Le niveau suivant est alors chargé.

4.3 Sauvegarde des scores

A chaque fois que le joueur termine un niveau, une requête HTTP est envoyée au serveur afin de sauvegarder le temps réalisé par le joueur. Pour cela, le timestamp est sauvegardé lorsque le niveau commence, puis le timestamp

est de nouveau récupéré lorsque le niveau se termine, afin de savoir la durée en secondes. Cette durée est envoyée au serveur par une requête HTTP, avec en plus le numéro du niveau correspondant. Si le joueur est hors ligne, ou que le serveur ne peut être atteint pour tout autre raison, le temps n'est tout simplement pas sauvegardé.

Depuis l'utilisation de MySQL, le pseudo du joueur est aussi sauvegardé. Cela signifie que maintenant seul les joueurs connectés peuvent envoyer leur score au serveur, afin d'apparaître sur les classements avec son pseudo.
La requête effectuée n'est donc uniquement effectuée que si le joueur est connecté, et une nouvelle information est envoyée afin d'identifier le joueur, l'index de son pseudo. Cela permet au serveur de connaître les meilleurs temps ainsi que les pseudos des joueurs qui les ont réalisés.

5 Les différents niveaux

5.1 Le premier niveau

5.2 Le second niveau

5.3 Le niveau final

5.3.1 Scripts dédiés au niveau

Je me suis occupé des scripts permettant d'implémenter les spécificités du dernier niveau.
Pour les dalles, le but était qu'une fois que le personnage aura activé les deux dalles, alors la porte de la salle du boss final s'ouvrira.
Pour ce faire, j'ai mis un box collider sur chacune des dalles, lorsque celui est déclenché, l'état de la dalle, inactif par défaut, devient actif, et l'état de l'autre dalle est vérifiée. Si elle est, elle aussi, active, l'animation d'ouverture des portes de la salle du boss se déclenche.

J'ai aussi géré les scripts pour la salle de gauche, avec les arbalètes et le slow motion. Tout d'abord, les arbalètes se comportent différemment des premières rencontrées, celles-ci se contentent de tirer tout droit et ne visent pas le joueur. De plus elles tirent uniquement lorsque le joueur est à l'intérieur de la salle et lorsque la dalle n'est pas encore activée.

Pour mettre en place ces arbalètes, j'ai refait au nouveau script, différent du script de base pour les arbalètes. Celui-ci est bien moins complexe, et se contente d'utiliser un timer afin de créer des flèches et leur donner la même orientation que l'arbalète, afin que celles-ci aient une trajectoire rectiligne.
De plus, je mets un box collider sur la salle, ainsi qu'un script, qui active le script des arbalètes lorsque le joueur déclenche le collider, si la dalle n'a pas encore été déclenchée, et qui le désactive lorsque lorsque le joueur quitte le collider ou lorsque la dalle est déclenchée.

6 L'interface

6.1 Le Menu principal

6.2 Le Menu multijoueur

6.2.1 Connexion

Le menu de connexion est réalisé à l'aide de la nouvelle fonctionnalités d'Unity 4.6 : les canvas 2D. De nouveaux gameobjects ont été implantés, permettant de faire bien plus facilement une interface 2D, sans avoir à utiliser des plans 3D et à placer la caméra au bon endroit pour avoir une impression de menu 2D.
Le menu contient deux champs de texte, un pour le nom de compte, et un autre pour le mot de passe. De plus, il contient aussi un bouton connexion, une fois que le joueur a rentré ses identifiants, il clique sur ce bouton. Une requête HTTP est alors envoyée au serveur, contenant les identifiants de connexion. Si les identifiants sont exacts, le serveur renverra l'index du joueur, qui est utilité pour de futures requêtes avec le serveur, et le joueur passe ensuite au menu suivant. Si les identifiants sont incorrects, le serveur renvoie -1, et un message d'erreur s'affiche alors, demandant au joueur de réessayer.

6.2.2 Choix de la partie

7 Le Site Web

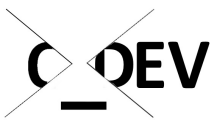
7.1 Buts

La Site web présente notre projet. Il se doit d'être beau esthétiquement, simple et épuré, afin de donner envie aux joueurs de jouer à notre jeu. Il sert aussi à présenter notre équipe de développement, et permet de télécharger le cahier des charges, le rapport, ainsi que notre jeu bien entendu.
Il possède aussi des fonctionnalités dédiés aux joueurs, il est possible de s'inscrire pour se connecter sur le site web et en jeu. Il est aussi possible de voir les performances des autres joueurs.
Il permet aussi en grande partie au jeu de fonctionner, notamment avec le multijoueur qui nécessite de communiquer avec le serveur, ainsi qu'avec le système de connexion dans le jeu afin de pouvoir sauvegarder les classements sur le serveur.

7.2 Base du site

Tout d'abord, je suis partit sur un site web statique, c'est à dire ne comportant que des fichiers HTML, CSS et javascript. Afin d'avoir un beau design, j'ai décidé d'utilisé un modèle CSS. Un autre avantage des modèles est qu'ils s'adaptent automatiquement à toutes les résolutions, tous les navigateurs et tous les appareils. J'en ai trouvé un qui me convenait sur le site HTML5 Up. Toutes les pages web du site ont le même menu, la même en-tête et le même bas de page, seul le corps du document varie.

Ensuite, j'ai décidé de partir sur un site dynamique, indispensable pour l'interaction avec les visiteurs du site. Pour cela, j'ai choisi d'utiliser PHP, et j'ai développé le site web avec XAMP, un suite logistiques regroupant tous les programmes nécessaires pour faire tourner un serveur local.
Ce changement m'a permis tout d'abord de modifier la structure de mon site, au lieu recopier le même code de base dans chaque page HTML, j'ai fait une seule page PHP regroupant tout ce code. Ensuite toutes les autres pages HTML sont remplacés par des pages PHP ne contenant que le corps du document. Grâce à l'instruction PHP include, ces pages peuvent ainsi être incluses dans la page de base, selon l'url.



Finalement, j'ai aussi fait le choix d'utiliser MySQL, une base de donnée, qui me permet de stocker les informations des différents membres, et de les récupérer, tout cela avec du code PHP.

7.3 Les différentes pages

Note site web contient différentes pages web statiques, accessible sans restriction. Ce sont les toutes premières pages réalisées, avant l'utilisation du PHP :

- Accueil : Présente rapidement qui nous sommes et notre jeu
- Téléchargements : Permet de télécharger le cahier des charges, les rapports de soutenance, le rapport de projet, les sources ainsi que l'exécutable de notre jeu, pour windows et os x
- Présentation : Présente notre projet, ainsi que les différents membres de l'équipe
- Liens : Une liste de liens vers les différents site web, logiciels et ressources externes utilités pour notre projet

A cela s'ajoute les pages web dynamiques, certaines nécessitant d'être connecté afin de pouvoir les afficher :

- Ccores : affiche les performances des joueurs.
- Inscription : permet de s'inscrire sur le site et le jeu.
- Connexion : permet de se connecter avec ses identifiants personnels.
- Déconnexion : permet aux visiteurs connectés de se déconnecter.

Sur chaque page web, se situe un menu, juste en-dessous du titre de notre projet. Il contient des liens permettant d'accéder à ces différents pages.

7.4 Les scores

Lors du passage à PHP, j'ai rajouté une nouvelle page, la page des scores. Cette page affiche les 5 meilleurs temps réalisés pour chaque niveau.
Ces scores sont en fait stockés dans un fichier sur le serveur, qui est lu par PHP. Ce fichier contient une version textuel d'un tableau PHP. Je peux ainsi "unserialize" ce texte afin de récupérer un tableau PHP contenant les temps pour chaque niveau, que je peux ainsi afficher sur la page.

Comme expliqué précédemment, le serveur reçoit une requête HTTP, envoyée par le jeu, contenant le numéro du niveau terminé, ainsi que le temps réalisé. Un script php va alors "unserialize" le tableau des scores comme pour

l'affichage, mais cette fois il va le modifier, en ajoutant le temps au niveau si celui-ci fait partie des 5 meilleurs, puis va mettre à jour le classement. Le tableau est alors "serialize" puis sauvegardé de nouveau dans le fichier texte.

Lorsque j'ai décidé d'utiliser MySQL, j'ai refait le fonctionnement des scores. Maintenant, la page des scores affiche aussi en plus le pseudo des joueurs. Cette fois les scores ne sont plus stockés dans un fichier texte, mais dans la base de donnée. La page PHP fait plusieurs requêtes SQL, une pour chaque niveau, permettant à chaque fois de récupérer les 5 meilleurs temps. La script de sauvegarde a aussi changé, la requête HTTP contient en plus l'identifiant du membre, c'est à dire que maintenant seul les scores des joueurs connectés sont enregistrés. Le script fait une requête sql pour enregistrer le score dans la base de donnée.

7.5 La partie membre

Je me sers aussi de MySQL pour stocker les informations des différents membres. De nombreuses pages ont été ajoutés :

- Inscription : Une simple page web contenant un formulaire d'inscription : Une fois validé, un script vérifie si les informations sont correctes, et renvoie en conséquence sur la page d'inscription avec les erreurs affichées, ou sur une page temporaire confirmant l'inscription, et qui redirige le visiteur sur la page d'accueil après quelques secondes, à l'aide d'un script javascript. Si l'inscription est validée, les informations sont enregistrés dans la base de donnée.
- Connection : Il fonctionne globalement comme l'inscription, sauf que le script PHP va faire une requête sql avec les identifiants entrés. Si aucune entrée n'est trouvée, alors la connection échoue et le visiteur est renvoyé sur la page de connection. Sinon le visiteur voit une page de confirmation avant d'être redirigé.
- Déconnection : Cette page se contente uniquement de déconnecter le visiteur. Un message de confirmation est affichée avant que l'utilisateur soit redirigé vers la page d'accueil.

L'état du visiteur (connecté ou pas), est affiché en permanence dans le menu. Pour cela, j'utilise des sessions, ce sont des variables PHP qui restent après avoir changé de page. Elles peuvent être effacées manuellement, c'est ce qui arrive lorsque le visiteur va sur la page de déconnection. Ou sinon elles s'effacement automatiquement après une certaine durée d'inactivité.

Lorsque le joueur se connecte depuis le jeu, il effectue des requêtes HTTP sur les mêmes pages, mais le serveur détecte que ce n'est pas un navigateur mais le jeu, grâce à une donnée envoyée en plus par le jeu : une variable game. Le serveur, au lieu de renvoyer une page HTML entière, renvoie soit l'id du compte si la connexion à réussie, sinon -1.

7.6 Implémentation du multijoueur côté serveur

Pour permettre l'utilisation du multijoueur, d'autres pages PHP ont aussi été ajoutées, mais celle-ci ne sont jamais utilisées par un visiteur du site web, elles sont réservées pour le jeu.
Afin que plusieurs joueurs puissent jouer, ils doivent tout d'abord être tous les deux connectés bien entendu, mais aussi connectés entre eux. Pour cela, j'ai rajouté une table mysql retenant les joueurs connectés au jeu ainsi que leur adresse ip. Ainsi, lorsqu'un joueur se connecte depuis le jeu, une entrée est rajoutée à cette table, avec son pseudo et son adresse ip.
Ensuite, le jeu peut ainsi demander la liste des joueurs connectés ainsi que leur adresse ip.
Le problème de cette implémentation est que, à partir du moment qu'un joueur se connecte, celui-ci est ajouté à la table, mais il n'est jamais supprimé, donc il reste ainsi déconnecté indéfiniment. Pour palier à ce problème, j'ai décidé de fermer la connection de jeu après 30 minutes d'inactivité.
Pour cela, la table contient un autre champ : le timestamp en secondes de la date de connection. Ainsi, lorsque le jeu demande les personnes connectés, avant de lui donner une réponse, je vais d'abord une requête SQL ayant pour but de supprimer toutes les entrées ayant un timestamp inférieur au timestamp actuel - 1800 (30 * 60). Ainsi, tous les comptes connectés depuis plus de 30 minutes sont supprimés de la table, et n'apparaissent donc pas dans la liste des joueurs connectés.

7.7 Rendu final

Lors du commencement du projet, le site web n'était alors qu'une simple vitrine, présentant le jeu, il était en quelque sorte à part du jeu. Mais au final des soutenances, de nombreuses fonctionnalités ont été ajoutés, permettant au jouer d'interagir avec le jeu, mais aussi avec d'autres joueurs. En fait le terme site web est maintenant un peu réducteur, ce dont il faut parler c'est

du serveur. Notre serveur XAMPP a deux objectifs. Premièrement, il distribue le site web aux visiteurs, leur renvoyant la page web demandées, avec les informations voulues.
Ensuite, notre serveur communique aussi avec le jeu, et sans cette communication, il n'y aurait pas de multijoueur, ni de classements.
Ce serveur n'est pas à part du jeu, c'est une partie de notre jeu, aussi bien que la partie client, réalisée avec Unity.

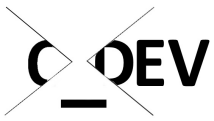
Le serveur permet aussi d'enrichir le site web, en permettant la possibilité de s'inscrire, de se connecter, de se déconnecter, mais aussi de consulter les différents scores des joueurs. Au final notre site web est assez riche et complet, et ressemble à un véritable site web d'un jeu vidéo.

La réalisation de ce site web m'a aussi demandé pas mal d'efforts, les différences rencontrées étaient certes moins importantes que pour le jeu, mais la moindre fonctionnalité à implémenter demande beaucoup beaucoup de lignes de codes en PHP. De plus, il faut aussi paramétrer XAMP, afin que le serveur se lance correctement. Pour la base de données SQL, il faut la rendre accessible depuis PHP, ainsi que créer des tables, ce sont en quelque sorte l'équivalent de nos fichiers sur un disque dur. Les données sont stockées sous formes de tableaux. Chaque table possède une structure propre, avec différents champs, ayant un nom, un type, des fonctionnalités spéciales, chacun de ces champs correspondant à une colonne du tableau.
A cela s'ajoute d'autres problèmes pour le développement des scripts PHP dédiés au jeu. Lorsque ceux-ci ne fonctionnent pas comme attendu, trouver l'erreur est bien plus compliquée, étant donné que le problème pourrait tout aussi bien venir du jeu que du serveur. Cela demande de développer à la fois en se servant d'unity, du navigateur, et d'un éditeur PHP.

8 Conclusions Personelles

8.1 Steven Lariau

Ce projet arrive à son terme, cela fait 6 mois de travail, et je réalise que beaucoup de choses sont arrivées, bien entendues dans le cadre du projet, mais aussi dans un cadre extérieur, et qui ont aussi eu une influence sur ce projet.



Dans un premier temps, je réalise que j'ai appris et gagné en expérience sur le travail en équipe, après cette première expérience, je réalise déjà un peu plus ce que travailler à plusieurs sur un même projet signifie réellement. Il ne suffit pas que chacun travaille dans son coin, faisant uniquement ce qu'il veut. Il faut aussi s'organiser, se coordonner. Nous faisans tous plus au moins des parties différentes du projet, mais il ne faut pas par exemple qu'un des membre de l'équipe se retrouve bloqué par ce qu'il a besoin de la partie d'un autre membre pour pouvoir travailler. Il faut aussi éviter de faire plusieurs fois la même chose. De plus, éviter de passer trop de temps sur des fonctionnalités peu utiles et compliquées permet d'avancer plus rapidement.

Ensuite, le but de ce projet était aussi d'améliorer ces compétences en programmation, et j'espérais notamment en apprendre bien plus sur le domaine des jeux vidéos en 3D, qui était pour moi une partie quasiment inconnue de la programmation. A ce niveau là, j'ai été assez déçu. Premièrement à cause d'Unity, certes ce moteur de jeu 3D permet de simplifier énormément la réalisation d'un jeu, en implémentant par défaut bon nombre de fonctionnalités, mais il va beaucoup trop loin, l'abstraction réalisée par Unity est tellement importante que je ne plus vraiment l'impression de programmer. Je pense notamment à la fonction Main, qui existe plus ou moins, dans une forme ou dans une autre dans presque tous les langages de programmation, mais qui n'est pas modifiable sur Unity. Je pense aussi à l'ajout de scripts à des GameObject, qui m'a toujours semblé être une fonction très étrange de procéder, surtout qu'un même objet peut avoir plusieurs scripts. C'est comme ci pour modifier le fonctionnement d'une classe, au lieu de créer une classe fille héritant de la classe voulue, en changeant certaines méthodes pour modifier le comportement, on créerait d'autres classes, qui violerait le principe d'encapsulation et modifierait le gameobject dont on veut changer le comportement.
En fait, Je n'ai pas l'impression qu'Unity est outil pour les programmeurs, on peut même créer son jeu-vidéo entièrement sans taper une seule ligne de code. Le code semble être une fonctionnalité otionnelle, un gadget du framework, qui peut parfois servir. Je n'ai jamais utilité de moteur de jeu avant, alors peut-être que tous les moteurs fonctionnement de la même façon, mais en tout cas j'ai déjà utilisé plusieurs SDK avec des "frameworks colossaux", permettant un peu comme Unity de faire tout très facilement, avec un niveau d'abstraction élevé, comme Visual Studio avec .Net pour faire des programmes windows, Xcode avec Cocoa pour faire des applications OS

X / iOs, Qt pour faire des programmes multiplateformes. Tous permettent de réaliser certaines parties de notre programme sans coder, avec un WYSIWYG, un peu comme Unity, mais le code reste tout de même la majeure partie du travail, et une partie indispensable. De plus, utiliser ces outils pour créer des interfaces est un choix, il est tout à fait possible de faire la même chose rien qu'avec du code, permettant un contrôle total, alors qu'avec Unity nous n'avons pas le choix, nous sommes obligés d'utiliser l'interface.
J'ai donc été déçu par Unity, et j'estime ne pas vraiment avoir appris grand chose dans la réalisation d'un jeu vidéo 3d. Faire ce projet m'a donné envie d'aller voir d'autres moteurs de jeu, notamment Unreal Engine, où l'on code en C++, cela me semble indispensable d'utiliser un langage compilé, générant un code machine optimisé pour faire des jeux-vidéos en 3D.

Néanmoins, j'ai tout de même appris pas mal de choses en programmation, notamment en C#. Unity nous donne accès au framework .NET, je n'ai vu certes qu'une toute partie de celui-ci, il semble énorme et couvre vraiment une large étendue de domaines différents. J'ai trouvé par exemples de nombreuses fonctions pour manipuler les dates, mais aussi pour communiquer avec le serveur en HTTP, ainsi que bien d'autres fonctionnalités pour le réseau. Je pense que de ce côté là .Net est bien réalisé, il possède à la fois une interface haut niveau, permettant facilement de communiquer avec un serveur, d'envoyer et de recevoir des données, mais aussi des interfaces plus bas niveaux, qui demandent plus d'efforts de notre part, mais qui permettent au final d'avoir bien plus de contrôle sur ce que nous faisons, et grâce auxquelles nous pouvons plus ou moins tout réaliser. Nous pouvons ainsi contrôler quelle protocole utiliser, avec quelle machine communiquer, quelles données exactes nous envoyons, comment nous recevons les données, etc.

Pour la réalisation du site web, je connaissais déjà HTML, CSS, Javascript, PHP et MySQL, mais cela ne m'as pas empêché d'acquérir de nouvelles connaissances, notamment pour la communication avec d'autres machines qu'un navigateur web. J'ai appris à ne pas seulement renvoyer des pages HTML, mais aussi du texte brut, traité par notre jeu.
De plus, c'est la première fois que je fais un site web réellement achevé, complet, et avec un véritable design. En effet, par le passé, j'avais utilisé ces langages afin de réaliser de simples pages, ne possédant aucune présentation, seulement du texte.
La réalisation du site web m'a semblé bien différente de celle du jeu, dans le

sens où j'avais vraiment l'impression de programmer. Pour PHP et MySQL je n'ai utilisé aucun framework, je me suis contenté des fonctionnalités de bases du langage, ce qui me donne un assez gros contrôle sur ce que je fais.

Conclusion