Arnaud Curie | Johann Fouchard | Pascaline Guichard | Juliette Gagnepain | Guillaume Clerc | Noa Barbosa

IUT Dijon | departement informatique

Rapport

Livrable n°1



# Introduction

Le projet consiste en la réalisation d’un clone du jeu « The Binding of Isaac ». C’est un jeu indépendant d’action-aventure en 2D isométrique. Le style du jeu est volontaire gore et effrayant mais est adouci par des designs cartoons. Nous avons décidé pour notre clone de reprendre un style similaire dans l’univers d’Alice au pays des merveilles. L’objectif du projet est de réaliser la version la plus abouti possible du jeu.

Dans ce document, nous nous pencherons sur l’existant, à savoir le jeu que l’on souhaite copier. Puis nous réaliserons une étude de l’art, en recherchant les différentes caractéristiques d’autres jeux similaires, pour ainsi déceler les différents point forts et points faibles du jeu « The Binding of Isaac ». Nous pourrons ainsi ajuster notre copie en conséquence. Bien que pour ce projet le langage de programmation nous soit imposé, nous réaliserons quand même une comparaison des technologies envisageables.

# Etude de l’existant

# Etude de l’art

Voici quelques fonctionnalités qui me sont venues à l’esprit pour améliorer notre clone d’Isaac et ainsi dépasser nos concurrents.

## 1ère fonctionnalité : changement de salle par téléportation

Lors d’un changement de salle, au lieu de faire passer notre personnage par une porte et avoir une animation vers la gauche, la droite, le haut ou le bas, j’ai pensé à un système de téléportation. Ainsi, dans quelques-unes des salles, il y aurait un objet spécial disposé dans un des coins de la salle, et ce dernier ne serait activable que lorsque tous les ennemis présents dans la salle sont morts. Un fondu au noir et l’apparition progressive de la salle suivante remplacerait l’animation de translation initiale. De plus, cette fonctionnalité est adaptée à l’univers que nous souhaitons créer pour notre jeu.

## 2ème fonctionnalité : possibilité d’esquiver

En ce qui concerne de déplacement d’Isaac, j’avais pensé qu’il serait intéressant d’ajouter un système d’esquive (qui consiste à une accélération limitée dans le temps). Une barre de remplissage apparait quelque part sur l’écran et lorsque celle-ci est entièrement remplie (le temps de remplissage varie selon le niveau de jeu), le joueur peut esquiver les ennemis à sa guise (il peut ou non utiliser toute la barre de chargement).

## 3ème fonctionnalité : épreuves de discrétion

Cette fonctionnalité pourrait rendre le jeu plus intéressant et moins répétitif sur la durée : des épreuves de discrétion et non de combat.

## 4ème fonctionnalité : Epreuves bonus

Connaissant le jeu vidéo Five Night At Freddy’s 4 (FNAF 4), j’ai eu l’idée d’adapter un des éléments à notre clone : les épreuves bonus. FNAF 4 est composé de 5 niveaux (5 nuits de 6h chacune), et entre chaque niveau, le joueur a la possibilité de faire disparaître 2h de la nuit suivante s’il réussit un mini jeu. S’il réussit, il n’a que 4h à jouer la nuit suivante au lieu des 6h habituelles. Mais s’il meurt avant d’avoir fini le niveau, alors le bonus ne s’applique et il doit recommencer la nuit avec 6h à jouer. Nous pouvons adapter cette mécanique à Isaac en remplaçant les 2h en moins par un objet bonus.

# Présentation des technologies envisageables

Pour pouvoir réaliser notre projet, le clone du jeu The Binding of Isaac, nous avons le choix à la fois entre de nombreux langages de programmation, de Framework ou même d’IDE.

Traditionnellement pour ce genre de projet les langages de programmation de haut niveau sont utilisés. Parmi tous ceux disponibles nous pouvons citer le Java, le C#, le C++, Python Ruby et encore bien d’autre.

Pour le développement de l’application, notre client souhaite que le tout soit codé en Java. Si cela correspond à notre premier critère, nous pouvons cependant nous demander s’il cela reste cohérent et adéquat à notre projet.

Grâce à notre cursus, nous connaissons et avons l’habitude de manipuler du Java, du C# et du C++. Nous allons donc comparer ces trois langages entre eux et laisser de côté les autres. En effet, apprendre un nouveau langage pourrait nous prendre trop de temps et être contre-productif dans la réalisation du projet alors que ceux que nous connaissons pourrait être suffisant.

Une première chose qui les différentie est leur date de création : C++ a été créé dans les 1983, 1999 pour Java et 2001 pour C#. Les trois sont encore mis-à-jour C++ est donc le plus ancien et pourrait bénéficier d’une documentation plus complète au vu de son âge, mais pourrait souffrir d’une syntaxe vieillissante. Concernant Java et C#, ils sont plus récents mais ils sont cependant plus utilisé et possède une communauté plus présente et active.

Une autre différence entre ces trois langages est que les projets en Java et C# sont indépendants de la plateforme sur laquelle ils se trouvent. Cela veut dire que nous pouvons coder une seule application et elle pourra s’exécuter sur n’importe quelle machine qu’importe son système d’exploitation. En revanche, il faut que l’ordinateur possède un interpréteur de ce langage qui va s’occupe de le traduire en langage que la machine pourra directement comprendre.

Prenons l’exemple de Java. Un fois le code écrit, il est compilé pour pouvoir être exécuté. Le compilateur converti le Java en bytecode. Ce byte code peut alors être exécuté sur n’importe quel système exploitation si ce dernier possède un Java Runtime Environment (JRE) compatible. Il est a noté, malgré tout, que Java s’installe plus facilement sur différent support puisqu’il est implémenté de base dans les systèmes sous Android.

Une autre différence importante est sur le temps d’exécution. Pour les projets les plus conséquents, utiliser C++ présente un grand avantage puisqu’il s’exécute plus vite que C# et Java su fait qu’il ne passe pas par une machine virtuelle. Cependant pour la taille de notre jeu, même si nous utilisons Java qui est plus lent, cela ne nous posera pas de soucis, ne nous impactant pas.

Finalement, nous pouvons voir une tendance se dessiner avec nos choix. Java et C# semble assez semblable, comparé à C++. Cela vient de leur date de création, très proche, mais aussi de la raison de la création de C# qui a été lancé pour concurrencer Java en proposant une version améliorée qui lui-même devait être une amélioration de C++.

Nous nous orientons plus favorablement vers Java ou C#. Si C# semble être plus privilégié pour réaliser des projets vidéo ludiques, Java peut être intéressant à choisir puisque, dans une situation réelle, il pourrait être plus facile de recruter des personnes pour travailler sur notre projet. En effet, Java est gratuit et plus facilement accessible que C# qui est détenu par Microsoft.

Maintenant que nous savons que Java est adaptée à ce que nous souhaitons faire, nous pouvons nous demander si nous allons utiliser un framework, une bibliothèque regroupant de nombreuses fonctions pré-codées, très utile pour éviter de recoder des méthodes et pour potentiellement développer plus vite notre projet.

Pour Java, de nombreux framework sont adaptés pour la création d’un jeu comme LibGDX ou Slick2d. Cependant notre client ne souhaite pas avoir recours à ces framework, cependant est-ce dérangeant ?

Pas vraiment, puisque si un framework peut faire gagner du temps, apprendre à le maîtriser et à savoir l’utiliser peut couter un temps précieux. Dans notre cas, puisque nous n’avons jamais utilisé ces Framework auparavant nous risquons de ne pas savoir les utiliser correctement et de ne pas être aussi efficace que si nous codions toutes nos fonctions. Nous n’en n’utiliserons donc pas.

Finalement, en ce qui concerne les IDE, nous avons le choix entre plusieurs IDE, Eclipse, IntelliJ IDEA Netbean. Si IntelliJ est l’IDE le plus prisé par les développeurs Java, le fait qu’il soit payant est un frein pour que nous puissions l’utiliser. Nous avons donc NetBean, que nous connaissons bien puisque nous l’avons toujours utilisé, nous savons donc comment l’utiliser. Tout comme Eclipse, il possède des outils de de débogage et de test. Toutefois, Netbeans est plus moderne dans son apparence et plus intuitif. Nous utiliserons donc NetBeans pour coder notre projet en Java, sans Framework.