Livrable final

Projet Informatique Individuel

Introduction

Dans le cadre de la deuxième année de l'ENSC nous sommes amené à réaliser un projet informatique en individuel. J'ai choisi pour cette UE, de développer un jeu vidéo 2D à l'aide de la plateforme Unity. La description du jeu est la suivante :

Le démon serpent a envoyé ses créatures maléfiques envahir le royaume. Le roi, furieux, sort de son château pour écraser ces monstres de lui-même. "Moi vivant, pas un serpent ne survivra dans mon royaume ! ", aurait-il crié en franchissant le pont-levis de sa forteresse.

L'application est un jeu de plate-forme dans lequel le joueur pourra contrôler le roi tueur de serpents. Le but est de franchir les obstacles du monde dans lequel il va évoluer tout en écrasant les ennemis sur son passage. Le jeu s'appelle Angry kingz crushes snakes ou simplement "Angry kingz".

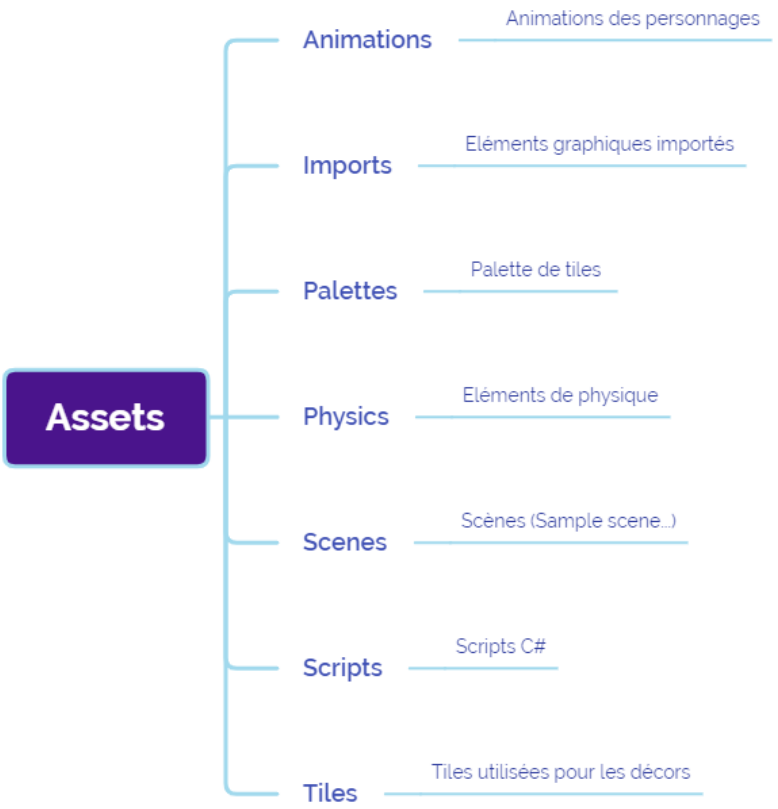
Choix techniques faits

J'ai choisi d'utiliser Unity pour m'aider à développer ce jeu car c'est un outil largement utilisé dans l'industrie du jeu vidéo, même encore actuellement. Le langage utilisé est le C#. Je connais déjà ce langage grâce aux cours de programmation que j'ai pu suivre à l'école. Cependant je n'avais pas encore eu l'occasion de réaliser un jeu vidéo avec une interface graphique vraiment développée grâce à celui-ci, et encore moins avec Unity.

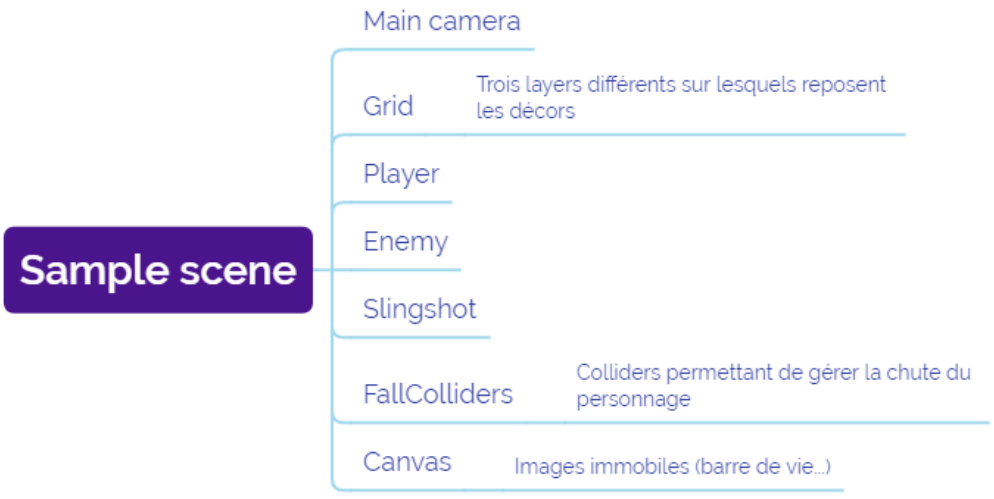
Unity apporte donc une certaine facilité pour mettre en place une interface graphique de qualité, mais aussi certaines mécaniques et fonctionnalités complexes aisément implémentables. On peut par exemple y retrouver la notion de poids d'un objet (gravité) ou encore les colliders et les fonctions pré implémentées liées (*OnTriggerEnter2D*...).

Architecture de l'application

L'application est organisée en 2 parties : Assets et Packages. Packages contient tous les packages dont Unity a besoin pour fonctionner. Assets est organisé de cette manière :



Au sein même de l'application, on retrouve la "sample scene" qui est organisée de cette façon :



J'ai aussi pu appliquer certains principes de la méthode SOLID, comme par exemple le principe de responsabilité unique. Chaque script C# a une seule utilité :

- PlayerMovement : déplacements du joueur
- PlayerLanding : "atterrissage" du joueur sur la catapulte
- EnemyPatrol : déplacements des ennemis
- WeakSpot : point faible de l'ennemi
- ...

Graphismes, fonctionnalités et gameplay

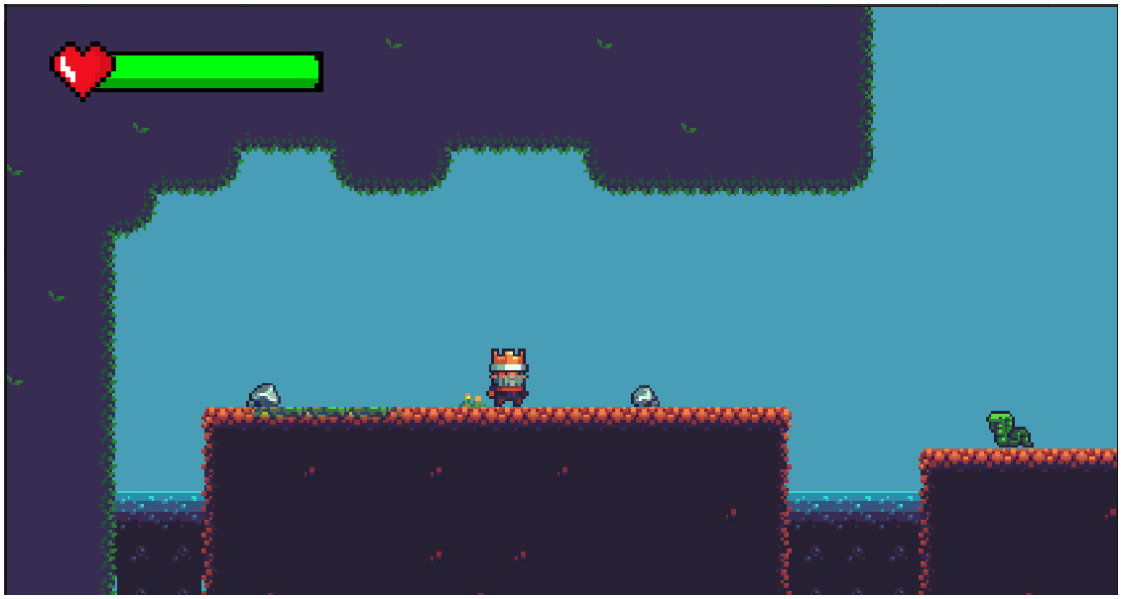


Figure 1 : Début du jeu

Concernant l'aspect graphique, j'ai entièrement reprit les outils (tylesheets...) montrés dans un tutoriel. La plupart des fonctionnalités de base du jeu sont aussi beaucoup inspirés de ce tuto. La grande originalité de ce projet réside dans l'ajout de la mécanique de catapulte inspirée du jeu angry birds.

Figure 2 : atterrissage dans la catapulte

Figure 3 : catapultage

Au delà de ça, le joueur peut sauter sur des monstres pour les éliminer, et quand il tombe dans le vide il perd de la vie et est téléporté au début du niveau.

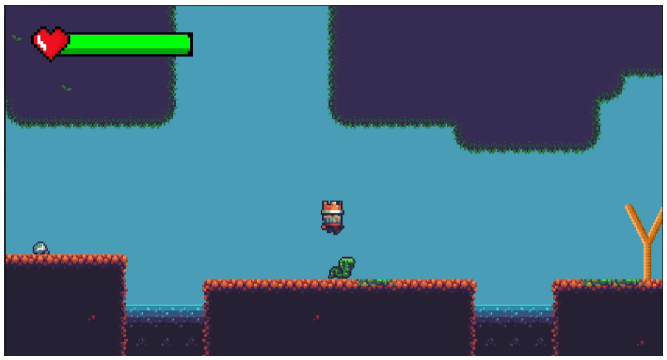


Figure 4 : saut sur un serpent

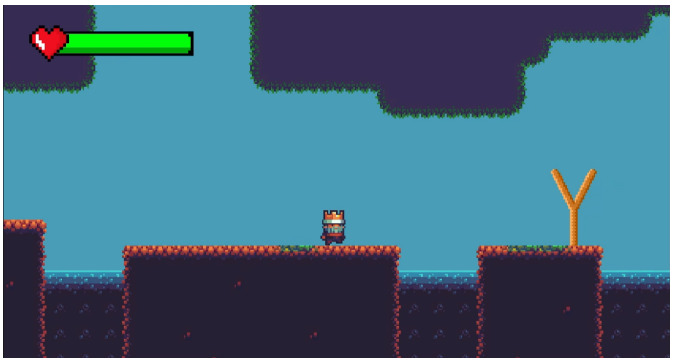


Figure 5 : plus de serpent

Figure 6 : chute dans le vide

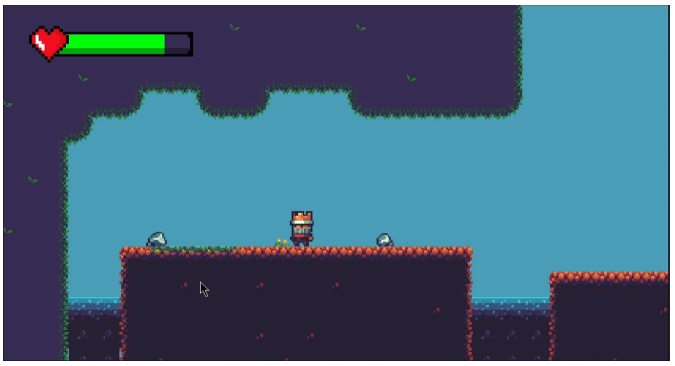


Figure 7 : téléportation au début du niveau et perte de vie

Comment jouer ?

- **se déplacer :** flèches directionnelles droite / gauche
- **sauter :** barre espace
- **se catapulter :** tirer le personnage dans une direction en cliquant dessus (drag) et lâcher la souris

Gestion de projet

Organisation avec le tuteur

Tous les mardis, un point d'avancement était prévu avec le tuteur. Une semaine sur deux, c'était un simple compte rendu par mail sur ce qui a été fait pendant la semaine, l'autre semaine, un rendez-vous en présentiel, à l'école.
Afin de pouvoir partager mon projet plus facilement, je l'ai mis en ligne sur Github : [Dépôt github]

Planning et choix effectués

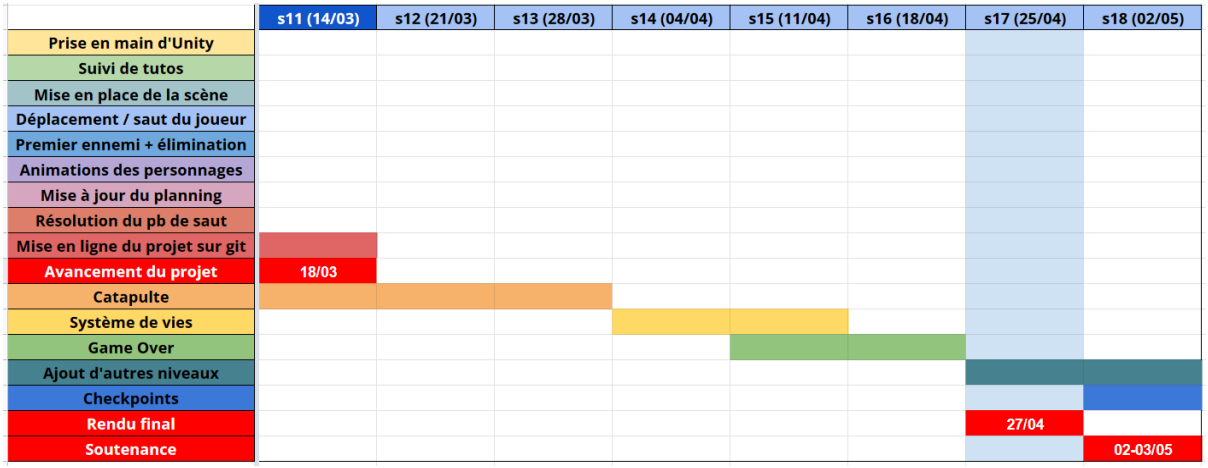


Figure 8 : Planning prévisionnel depuis le point d'avancement

Les 2 dernières fonctionnalités (ajout d'autres niveaux et checkpoints) étaient des fonctionnalités bonus, à réaliser en cas d'avancement plus rapide que prévu.
La fonctionnalité principale, à mettre en place à tout prix était le système de catapulte.

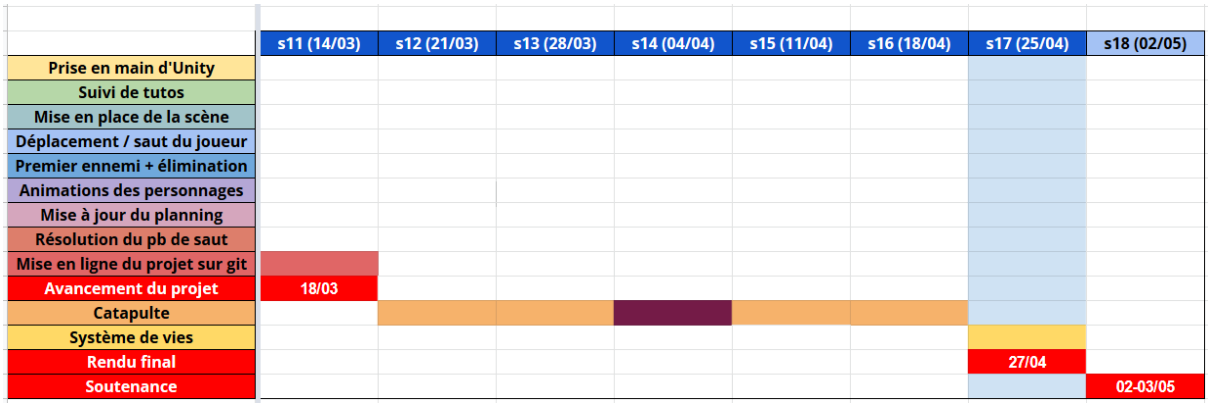


Figure 9 : Planning effectif depuis le point d'avancement

Légende	
	Période de congés
Date	Jalon
	Incapacité de travail

La fonctionnalité de catapulte a pris plus de temps que prévu, elle s'est étendue sur 4 semaines au lieu de 3 (+ 1 semaine d'incapacité pour cause de maladie). Ce retard de 2 semaines m'a poussé à prioriser certaines fonctionnalités. J'ai décidé de mettre en place le système de vie et de gérer les chutes du personnage (pas dans le planning car rapide à faire), puis de lier les deux pour faire perdre de la vie au personnage quand il tombe.

Bilan personnel

Origine du projet

Au tout départ du projet, l'objectif que je m'étais fixé était de réaliser un jeu type Angry birds. Cependant, en me renseignant sur Unity et en apprenant à le prendre en main j'ai trouvé plus intéressant de faire un jeu de plate-forme 2D. C'est alors qu'est né Angry Kingz, un mélange de platformer et de jeu style angry birds.

Difficultés rencontrées

La mise en place de la fonctionnalité de catapulte a été la plus grande source de difficultés, puisqu'il n'existe pas d'équivalent sur internet (ou alors je ne l'ai pas trouvé). L'objectif ici étant d'arriver à faire grimper un personnage pouvant se mouvoir dans la catapulte, bloquer ses mouvement quand il est dans la catapulte et enfin, arriver à l'envoyer dans les airs.

Au début j'ai commencé par incorporer la fonctionnalité de catapulte avec spawn d'oiseau automatique. L'intérêt était de pouvoir se reposer sur du code déjà existant. Cependant, tout ne s'est pas déroulé immédiatement comme prévu.

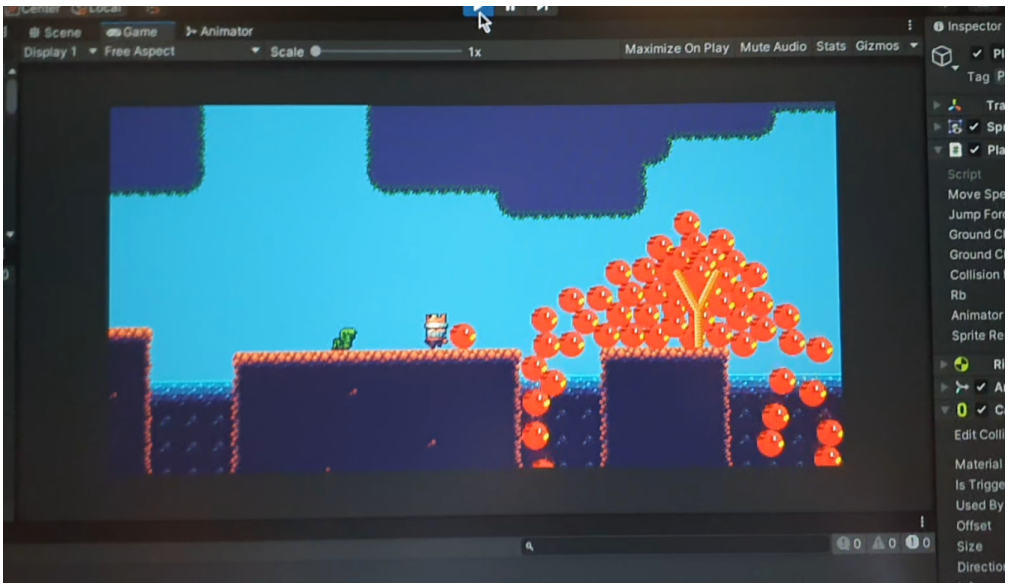


Figure 10 : Une fontaine d'oiseaux

Une fois ce problème résolu, il fallait remplacer ces oiseaux par le personnage du joueur, sans le faire spawn directement dans la catapulte, étant donné que celui-ci existe déjà au début de la partie. Une fois le personnage bien en place dans la catapulte, mon problème a été le suivant :

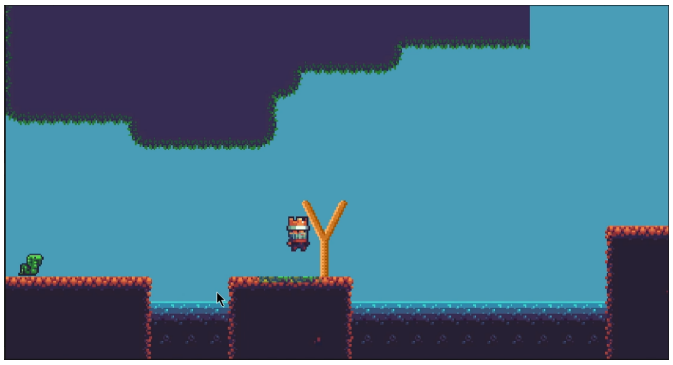


Figure 11 : préparation à l'éjection

Figure 12 : phénomène inattendu

Le personnage se heurtait comme à un mur invisible lorsqu'on lâchait la souris. Cependant ce n'était pas un problème de collision. Le personnage était en fait soumis à deux forces en même temps : celle appliqué par la catapulte et celle de déplacement horizontal, mise à jour à chaque frame. Pour régler le problème il a donc fallu désactiver totalement la fonction de déplacement horizontal pendant que le joueur était dans la catapulte ainsi que pendant qu'il était dans les airs.

Montée en compétence

Grâce à ce projet j'ai pu apprendre à utiliser Unity, une plateforme qui m'intéresse depuis longtemps. Maintenant je suis capable de créer un jeu en 2D et d'y incorporer des fonctionnalités complexes, voir insolites.

Pistes d'évolution

La première piste d'évolution serait d'achever toutes les étapes prévues dans le planning. Pour la suite, il peut être intéressant de penser à créer un écran d'accueil, ajouter des musiques et divers menus (pause, sauvegarde...) afin de rendre le jeu plus attractif. Mettre en ligne le jeu sur le google play store pourrait être l'étape finale du développement d'Angry Kingz.