

# RAPPORT

## C++



*LE DRÉAN Yoann  
MARTEL Benoît*

*16/12/2019*

# Table des matières

- Difficultés rencontrées..... 4
  - Saut .....5
  - Gestion de plusieurs inputs simultanés .....5
  - Suppression d’objets.....5
  - Surcharge d’opérateur .....5
- Fonctionnalités ..... 6
  - Plateformes .....7
  - Bonus .....7
  - Monstres .....7
- Evolutions ..... 8
  - Plateformes mobiles verticales..... 9
  - Nouveaux bonus ..... 9
  - Inertie horizontale ..... 9
  - Monstre mobile..... 9
  - Diagramme de classes .....10

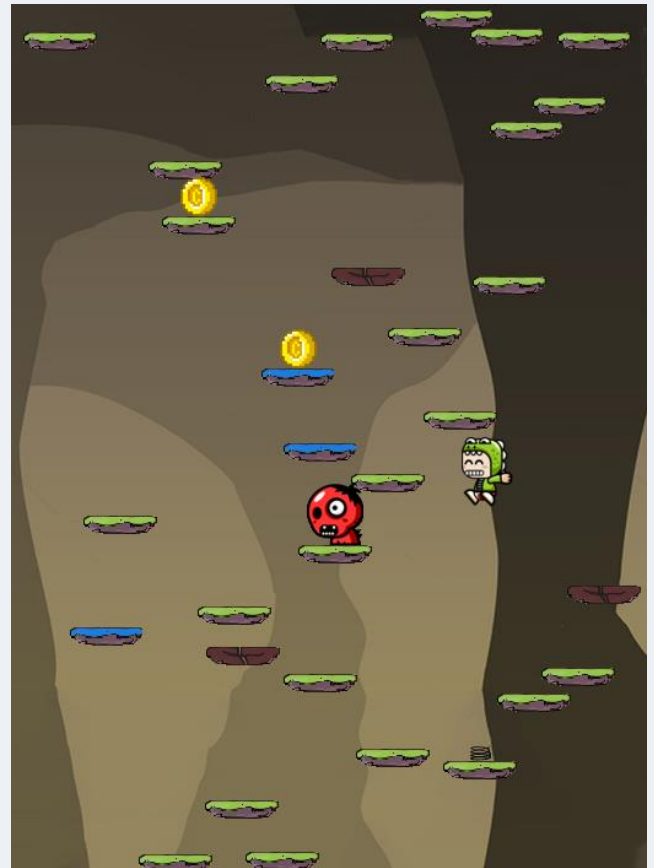
# INTRODUCTION

*Nous avons développé un jeu en C++, en utilisant la librairie Qt. Nous sommes partis du jeu Doodle Jump, dont nous avons repris les mécaniques de base. Nous avons ensuite modifié les textures, afin de créer notre jeu : Crocro Jump. Nous avons aussi ajouté certaines fonctionnalités non présentes dans le jeu de base, comme par exemple les pièces. Toutes ces fonctionnalités seront détaillées dans ce rapport.*

*Le but du jeu est d'aller le plus haut possible. Le score augmente lorsque le joueur monte, ou lorsqu'il ramasse des pièces. Pour monter, il faut sauter de plateforme en plateforme. La partie se termine lorsque le joueur tombe en bas de l'écran, ou se fait toucher par un monstre. Il est possible de sortir de l'écran à gauche ou à droite pour passer de l'autre côté.*

*La difficulté est croissante. Au fur et à mesure de l'ascension, de moins en moins de plateformes sont disponibles et les probabilités d'apparition de certains types de plateforme augmentent.*

*La génération du niveau est aléatoire, mais il y a toujours au moins une plateforme accessible permettant au joueur de monter.*



# **Difficultés rencontrées**

## Saut

La première difficulté rencontrée, pour le développement de ce jeu, a été la mécanique de saut. Le personnage saute automatiquement, il rebondit.

Au tout début, le personnage montait et descendait, avec une vitesse constante, le saut n'était pas réaliste.

Nous avons eu une difficulté pour créer une gravité, ou du moins une simulation de gravité. Pour ce faire, le personnage a un attribut représentant sa vitesse sur l'axe Y. A chaque rebond, cette vitesse est réinitialisée à -5. Ensuite, à chaque déplacement, la vitesse est augmentée d'une constante *gravité*. Ainsi, le joueur a une vitesse décroissante en montant. Lorsque la vitesse atteint 0, il commence à retomber, à une vitesse croissante.

## Gestion de plusieurs inputs simultanés

Le joueur peut effectuer 3 actions :

- Se déplacer vers la gauche
- Se déplacer vers la droite
- Tirer au-dessus de lui

Pour que le joueur puisse tirer tout en continuant de se déplacer dans une direction, nous avons créé un vecteur, une liste d'actions. Lors de l'appui sur une touche, cette touche est ajoutée à la liste. Lorsque le joueur relâche la touche, celle-ci est retirée de la liste.

A chaque fois que nous effectuons les actions correspondant aux touches, nous parcourons toute la liste et effectuons chaque action correspondant à chaque touche.

## Suppression d'objets

Nous avons également eu des problèmes liés à l'oubli de suppression de certains objets, comme par exemples les QTimer propres à certains objets comme les projectiles.

Cet oubli entraînait un ralentissement global du jeu, le rendant injouable.

Ce problème était lié au fait que nous découvrions le langage C++, et donc la suppression des objets. Nous avons donc corrigé ce problème en implémentant les destructeurs des objets concernés.

## Surcharge d'opérateur






L'une des contraintes demandées était la surdéfinition d'un opérateur. Nous avons choisi de redéfinir l'opérateur < (inférieur) de GameObject, qui est la classe dont toutes les autres héritent.

Cette surdéfinition nous permet de savoir si un objet est plus haut qu'un autre dans la scène. Nous avons eu des difficultés pour réaliser cette surdéfinition, et pour l'utiliser.

# Fonctionnalités




# Plateformes

Il existe plusieurs types de plateformes :

-  - Les plateformes **basiques** : Elles n'ont rien de spécial, ce sont les plateformes par défaut.
-  - Les plateformes **cassables** : Le joueur ne peut pas rebondir dessus, elles se cassent lorsque le joueur tente de rebondir dessus.
-  - Les plateformes **mobiles** : Elles se déplacent de gauche à droite de l'écran.
-  - Les plateformes **qui disparaissent** : Elles disparaissent une fois que le joueur a sauté dessus, elles sont donc à usage unique.
-  - Les plateformes **explosives** : Lorsque le joueur atteint la hauteur de ces plateformes, elles explosent au bout d'un certain temps.

## Bonus

Plusieurs bonus sont disponibles, et apparaissent aléatoirement sur les plateformes :

-  - Le **ressort** : Lorsque le joueur saute dessus, il rebondit beaucoup plus haut qu'un saut normal
-  - Le **jetpack** : Lorsque le joueur le ramasse, il est propulsé en hauteur pendant 3 secondes.
-  - La **pièce** : Elle augmente le score de 100 points.

## Monstres

Les monstres ont une faible chance d'apparaître sur les plateformes.  
Pour l'instant, nous n'avons qu'un type de monstre :

-  - Monstre **immobile**

Le joueur perd la partie s'il touche le monstre. Il peut cependant tuer le monstre en rebondissant dessus (Ce qui fait par ailleurs rebondir le joueur plus haut qu'un saut classique)  
Il est également aussi possible de tirer sur les monstres pour les tuer.

# **Evolutions**



## Plateformes mobiles verticales

Il est possible de rajouter des types de plateforme, comme par exemple une plateforme se déplaçant verticalement. La plateforme mobile actuelle serait donc divisée en deux sous-classes : Horizontale et Verticale.

## Nouveaux bonus

D'autres bonus pourrait être ajoutés au jeu : Des pièces rapportant encore plus de points, de nouveaux accessoires faisant monter le joueur ou encore des malus diminuant le nombre de points.

## Inertie horizontale

Nous avons aussi envisagé de créer une inertie au mouvement horizontal, pour que le joueur ne s'arrête pas d'un coup lorsqu'il arrête de se déplacer, pour rendre le jeu un peu réaliste.

## Monstre mobile

Enfin, nous pourrions ajouter d'autres types de monstres, comme par exemple un monstre volant, se déplaçant de gauche à droite de l'écran.

# Diagramme de classes

