NOUGIER Guillaume, OHANA Kimberley

2016/2017

Cahier des charges : Runner en console

PrograMMATION ORIENTée objet

Table des matières

[Membres du groupe 2](#_Toc448963921)

[Intitulé 2](#_Toc448963922)

[Définition du projet 2](#_Toc448963923)

[Intérêts et contraintes du projet 2](#_Toc448963924)

[Contraintes techniques 2](#_Toc448963925)

[L’architecture MVC 2](#_Toc448963926)

[Description du projet : 2](#_Toc448963927)

[Technologies utilisées 3](#_Toc448963928)

[Le C++ 3](#_Toc448963929)

[Pourquoi le C++ ? 3](#_Toc448963930)

[Difficultés rencontrées 3](#_Toc448963931)

[Annexes 4](#_Toc448963932)

[MVC : *Model View Controler* 4](#_Toc448963933)

[UML : Unified Modeling Language 5](#_Toc448963934)

[6](#_Toc448963935)

Jeu de plateforme en console Windows

# Membres du groupe

* NOUGIER Guillaume
* OHANA Kimberley

# Intitulé

## Définition du projet

Le but de ce projet est d’intégrer des compétences Programmation Orienté Objet à travers la réalisation d’un projet de programmation.

# Intérêts et contraintes du projet

## Contraintes techniques

L’intérêt de notre projet était de découvrir l’intégration d’une architecture MVC (Model View Controler) à travers la programmation d’un jeu en mode console.

Dans le cadre de notre projet, nous avons dû prendre compte les contraintes suivantes :

* Concevoir et intégrer une architecture MVC
* Utiliser un langage informatique tel que le C++, Java, C#.

### L’architecture MVC

L’architecture **MVC** (modèle, vue et contrôleur) est un concept très puissant qui intervient dans la réalisation d’une application. Son principal intérêt est la **séparation des données** (modèle), de l’**affichage** (vue) et des **actions** (contrôleur).

Intégrer une architecture MVC présente les intérêts suivants :

* Prendre connaissance des interactions entre chaque partie de l’application ainsi que l'intégralité des services et fonctionnalités à coder.
* Concevoir de manière claire et efficace la séparation des données de la vue et du contrôleur.
* Une grande souplesse d’organisation du développement et de la répartition des tâches entre chaque développeur grâce à l’indépendance des données, de l’affichage et des actions.

# Description du projet :

Notre projet consistait à concevoir un jeu de plateforme communément appelé « endless running » (*course infinie).*

Les endless running sont des jeux de plateformes 2D dans lesquels le joueur incarne un personnage qui avance en continu au travers d’un monde théoriquement sans fin (à travers une génération procédurale).

Le but de ce type de jeu est d’aller le plus longtemps et le plus loin possible ainsi que de générer un score le plus haut possible dans le ou les univers avant que le personnage meurt. Un score est affiché, le plus souvent généré par la distance parcourue et/ou par le nombre d’objets ramassés/ d’ennemis tués. Les commandes de jeu sont de simples pressions sur l’écran (sur plateforme mobile) ou bien l’appuie sur des touches ou de simples clic de souris. Ces commandes permettent d’attaquer les ennemis qui arrivent à l’écran ou de faire sauter le personnage afin d’éviter les obstacles.

Dans le cadre de ce projet, nous présentons une version console du jeu jouable sur ordinateur, en utilisant les touches de clavier.

# Technologies utilisées

Afin de réaliser notre mini-jeu, nous avons fait le choix de programmer avec le langage C++

# Le C++

## Pourquoi le C++ ?

Le C++ est un langage de programmation très utilisé pour le développement de mini-jeux. C’est langage très populaire et il régit parmi les langages informatiques les plus utilisés. La sélection d’un langage de programmation dont la communauté est active est fortement utile dans le cadre de son apprentissage.

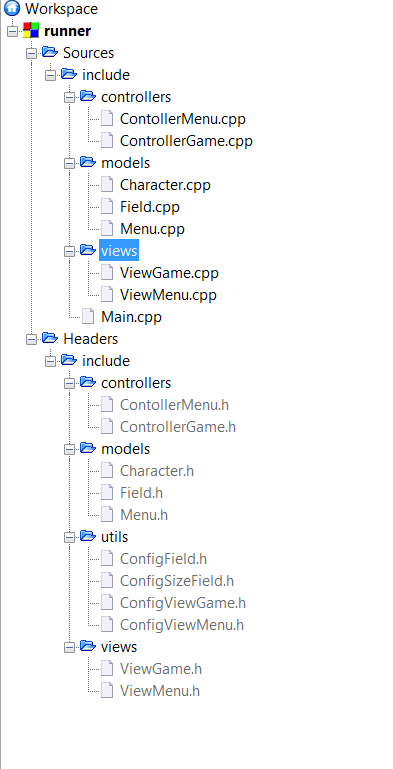
Le C++ est un langage de bas niveau compilé et permet l’utilisation de styles de programmations différents : générique, orienté objet, procédurale, etc.

# Difficultés rencontrées

A travers ce projet, nous avons rencontrés certaines difficultés. L’absence d’utilisation de librairies graphiques a rendu le travail d’intégration de graphismes enrichies, du clavier plus fastidieux. L’interaction entre le programme et le joueur est assez limité ainsi que l’expérience du joueur en elle-même. L’algorithmie de notre projet a donc par ailleurs été moins intuitive et a nécessité une recherche plus élaborée.

# Annexes

# MVC : *Model View Controler*

Ci-dessous, une arborescence de l’architecture Modèle, Vue, Contrôleurs.

# UML : Unified Modeling Language

(cf. page 6)

# 