Rapport de Projet : Sim Race

Austin LAROQUE, Joseph REGNAULT

May 4, 2025

Contents

1 Introduction

Ce rapport présente le travail effectué sur le projet Sim Race, réalisé dans le cadre du cours de L3 de POO. Il abordera les points importants de la réalisation du projet, des informations sur les choix effectués pendant la réalisation du jeu, ainsi que l'architecture du logiciel.

2 Structure et Fonctionnement du Jeu

2.1 Modes de jeu

Le jeu comporte deux modes différents :

- **Mode 1 joueur** : Dans ce mode, le joueur peut naviguer librement sur la carte, et doit récupérer un maximum de poissons et les ramener au port dans le temps imparti.
- **Mode 2 joueurs** : Dans ce mode, deux joueurs s'opposent : ils doivent ramener plus de poissons que l'adversaire dans leur camp, mais leurs cargaisons peuvent leur être subtilisées.

2.2 Fonctionnalités implémentées

Les principales fonctionnalités implémentées sont :

- Un menu principal permettant de sélectionner le mode de jeu.
- L'instanciation du bateau, ou des bateaux en mode deux joueurs (prise en compte des différences entre les joueurs).
- L'implémentation de mouvements réalistes pour les bateaux ainsi que d'un sillage les suivant lors des déplacements.
- La création des zones de port, où les joueurs pourront respectivement aller déposer leurs poissons.
- La création et la gestion des obstacles et des collisions avec les bateaux.
- La création et la disposition de poissons collectables sur la carte.

3 Architecture du code

3.1 Organisation des fichiers

Le dossier `/appTest/src/` contient les fichiers source C++ du projet, organisés de façon modulaire. Chaque classe est définie dans son fichier d'en-tête `.h` et implémentée dans son fichier source `.cpp`. Le fichier `main.cpp` est le point d'entrée de l'application, et permet l'initialisation du jeu.

3.2 Description des classes

L'implémentation du jeu est réalisée grâce aux entités suivantes :

- **Game** représente le coeur du programme, initialisant les variables nécessaires au bon fonctionnement du jeu et gérant l'initialisation et la coordination des différentes fonctionnalités des autres entités. Il utilise leurs différentes propriétés et implémente la majorité de la logique du jeu.
- **Menu** permet d'afficher le menu principal dans la fenêtre du jeu.
- **Boat** contient tout le code lié aux bateaux, dont celui des déplacements, des transferts de poissons et des sillages.
- **Port** gère l'implémentation des ports, notamment de la détection des livraisons de poissons.
- **Obstacle** s'occupe de la création des obstacles, de leur positionnement aléatoire et permet d'implémenter les collisions avec les bateaux.
- **Fish** est chargé de créer les poissons, de vérifier la validité de leur placement et permet leur collecte.

4 Difficultés et évolution du projet

Nous avons commencé le développement du jeu en suivant les consignes données dans le document de présentation du projet. Les premiers problèmes que nous avons rencontrés concernaient la compilation, avec `cmake`, que nous avons dû étudier plus en profondeur afin de comprendre son fonctionnement et régler les erreurs. Une fois ces difficultés surmontées, nous avons pu débuter l'implémentation du jeu, en C++, en commençant par le code du bateau. Ici, nous avons eu du mal à obtenir des déplacements réalistes, et

avons dû, pour remédier à ce problème, analyser des programmes que nous avions réalisés dans d'autres langages, et essayer de les convertir en C++. Ensuite, nous avons travaillé sur les autres fonctionnalités du jeu, et, une fois terminées, nous avons cherché à le rendre plus original ; c'est là que nous avons eu l'idée de la course aux poissons.

5 Travail en groupe

5.1 Organisation du travail

Nous avons pu collaborer tout au long du développement en utilisant Git pour partager le code et suivre notre avancement respectif. Cette organisation nous a permis de travailler de manière pratique et efficace. Austin a pris en charge l'intégration générale du projet, ainsi que les ajustements finaux, notamment le déroulement des parties, et Joseph a travaillé sur les déplacements du bateau et des fonctionnalités plus mineures. Cette répartition nous a permis d'avancer de notre côté, de manière complémentaire.

6 Conclusion

Ce projet nous a permis d'apprendre à développer un jeu en C++, à travailler en POO, par exemple en mettant en place l'approche modulaire, ainsi qu'à utiliser des outils comme `cmake`. De plus, l'aspect collaboratif du travail nous a permis d'avoir un aperçu de la façon dont les équipes de développement peuvent coopérer pour obtenir des solutions et produire des logiciels de façon efficace.