C++ Programming Project Documentation: Kong Kart

1. Overview

The software features a simple 2-player driving game. The players control the car centered on their respective sides of the split screen. Each player is required to complete 3 laps around the map and the one who first completes the task is the winner. Winning the game requires not only skillful driving, but also utilizing the accelerating effect of the coins scattered across the field. The game features 2 tracks.

The software only ended up requiring one external library, SFML. The library was utilized for implementing the graphical elements of the program. While Box2D was originally intended to be used for implementing the game physics, the realistic scope of the later stages of development required it to be omitted. Other, simpler solutions for implementing the most rudimentary of physics were used instead.

The cars are controlled through the WASD keys and the arrow keys. Both cars start off with the same set maximum speed, but this maximum speed can be raised through acquiring coins that are placed in tactical positions on the map.

The game does not feature the originally planned sound effects or items, the reasons for this, as well as reasons for several other omissions from the original plan will be addressed in the 6th section of this document.

The main omissions were due to the change to map creation, as we left out dividing the map into separate tiles and instead used the SFML-provided xy-grid structure. The coins are an effort to fill in for some of the functionality of the omitted Item class, as they also change the state of the cars.

2. Software structure

The code for the project is featured in a singular main file. The start of the file has the code for 2 structures (struct), Car and Coin. Car was originally planned to be a class of its own, but as its structure ended up being far simpler than originally anticipated, it was simplified to a struct to ensure memory handling issues would not be present as the struct only features parameters of type float. Each Car's position on the map is updated through the function move, which takes in two parameters of type float, speed and angle, in order to modify the x and y parameters of the Car.

There are 4 distinct windows that the software opens in succession: The starting screen, the two races and finally the ending screen that shows the race times for each racer.

Running the main.cpp file creates an instance of a game by creating an SFML graphics window, which is sustained through a while loop. Said while loop then calls the necessary functions, namely the move function for each car.

The folder also features the graphics elements to be utilized through SFML. Such elements include the font, the map, the car and the coin sprites.

3. Origins of assets

The game uses several assets derived from the hit video game series Mario Kart. The effect of the coins is the exact same as in the original series. The design of the coins is original, as they were hand-crafted utilizing the free graphical editing software gimp, but other things such as the map and the font are slightly modified versions of assets found online on Nintendo fandom forums.

The car sprites themselves were found online as well, but they appear to be originally unused assets extracted from the game files of the original Super Mario Kart. The riders are the characters Donkey Kong and Luigi, red and green respectively.

The starting screen is also a modified asset of the original Super Mario Kart. The modifications were done with Adobe Photoshop.

All of the text implemented through SFML uses a font that was found on dafont.com. The font emulates the official Mario font and is provided in the git.

4. Instructions for building and using the software

SFML must be configured in the terminal used to run the software. Apart from that, the rest should be handled by an automatic CMake configuration.

The selection of the game mode (1 player, 2 player) can be done using the arrow keys and as previously stated, the cars themselves are controlled through WASD and the arrow keys.

5. Testing

Most of the testing was done using a MacBook Pro M1, which proved to be the most reliable and efficient platform, most likely due to its underlying UNIX structure.

The game was also later tested on a Windows machine running Windows Subsystem Linux with Ubuntu 20.04. Virtual Box was used to allow a client to access the screen of the computer.

Thorough testing was always done after the implementation of any larger features, starting with the simplest of things such as even opening a window to more recent additions such as the coins.

As plenty of the work was done in the same physical location, testing was always dynamic, and issues were able to be fixed immediately.

6. Work log

As is stated in many of the earlier weekly meeting files, the project suffered from a severe lack of suitable hardware for development of software utilizing a graphical user interface. Out of the

3 original members of the group who were planning on using a Windows machine for development, only one managed to find a configuration that allowed for testing of the product. Several issues in both terminal access and the configuration of the main library, SFML, were faced.

As such, the lone MacBook was the only computer capable of proceeding with development for the first 2 weeks. However, due to some apple peculiarities in folder handling, the utilization of git on the computer was restricted. Due to this, alternative methods of exchanging code were utilized. Such methods include but were not limited to telegram messages of singular lines and complete .zip-files of a working build.

Most of the work on the actual, final code was completed in the final 2 weeks of the project timeline as code written before this could not be tested and validated effectively. Moreover, at this point a member of the original group informed the rest of the group that he would not be able to complete the project or the course due to time constraints. In addition, the said member was limited by the GPA requirements set by the school he is leaving on exchange to.

At this point, the project was reconfigured in accordance with the functional and validated rudimentary set-up developed in the Mac environment. As the member who quit the course was in charge of developing several incremental classes, for example Tile and Car, the planned class structure had to be discarded in favor of a validated approach as the structure of a level couldn’t be created without said classes. A lack of class structure made collaborative work way more difficult than what was originally planned.

The coding work ended up being divided between the 2 machines that could test the software. The final machine was used to document and work on the readability of the program and editing of assets.