Angry Birds - Documentation
Group 2

Kankaanpää Kalle 793456 Juusti Mikko 793469

Hintsala Jaakko 886790 Porio Sami 793540

December 2021

1 Overview

We sought out to do an Angry Birds like game in C++. The game plays like the hit game from Rovio. It has an appealing main menu from which the player is able access a level selector. After selecting the level the game starts. The goal of the game is to shoot different kinds of birds from a slingshot across a field and hit evil pigs. After hitting every pig the level is cleared. The catch is that the player has a limited amount of birds to achieve this and clearing the level without using all of the available birds grants extra points.
The game has three levels from which to choose. Also a moving camera helps recon the evil pig fortress. The game does not support for custom levels.

2 Software structure

The structure of the game has changed a bit from the original plan, but not too much.

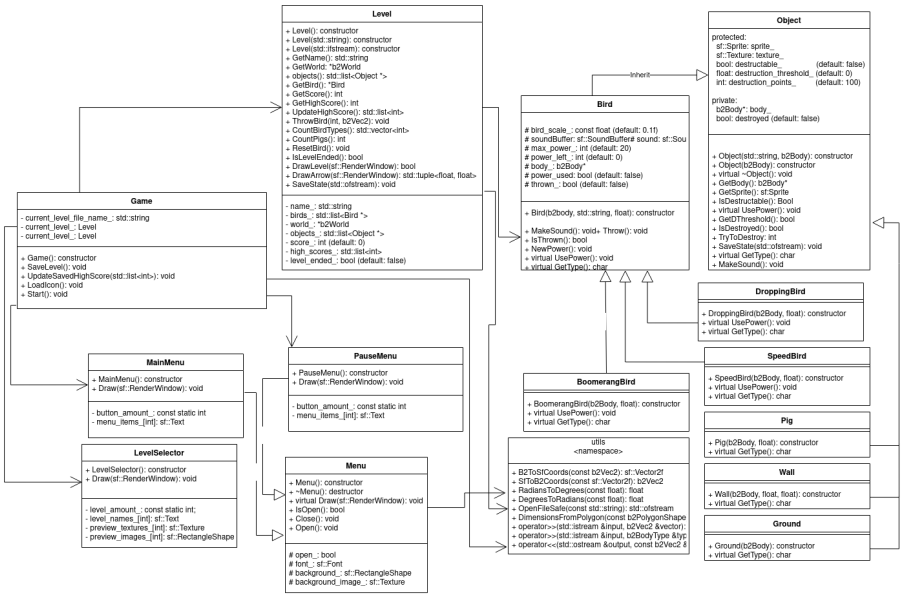

Figure 1: UML-diagram of the software structure. Presented as an object oriented UML model.

2.1 Game class

We have ha all governing Game-class which controls the menus and switching between them and levels. It also handles the calls for level saving and loading from files.

2.2 Level class

The Game-class has an instance of the Level-class. This class has the information about the SFML graphics world and also the Box2d world. It is also responsible for drawing the game and running the physics engine.

2.3 Object class

The object class is a super class for all the objects in the game world. Bird, wall, ground, pig all inherit the object class. All of the objects have a physics body in Box2d and a sprite image for SFML. All objects keep track of their

own position and orientation. One notable exception is the bird class which has multiple children, these are all the different kind — and also special — birds.

3 Instructions

The program has been designed from the start to be easy to use across different desktop platforms. After cloning the git repository, ...

After all that, if the cmake compilation ran without problems the executable named 'angry_birds' has been created in the build/ folder. Executing this will open the program and set the user to the main menu. In the main menu, there are a few options to choose from. Play button will open the level selector from which the game itself can be accessed. The high scores button will show high scores of different players in different levels. The exit button will exit the game, throw the user back to desktop and close the program execution.

Playing the game itself is designed to be very intuitive. After choosing a level the player is inserted into the selected level. On the top of the screen is shown a pause button, amount and types of still available birds, the remaining number of evil pigs, and current score with the high score of the level.

Figure 2: Example status of the current level hud.

The player can shoot a bird by using their mouse on the left hand side of the bird sitting in the slingshot.

Figure 3: The arrow shows the launch angle and the power. The longer the arrow is the more power the bird will have.

By clicking the left mouse button while the arrow is visible the bird is hurled towards the evil pigs. When the bird is in flight, pressing the left mouse button again will activate the bird power which is different for every kind of bird. All birds have some power, testing them out will teach the player what each bird does.

To complete the level the player has to clear it of pigs. Points are awarded for each pig destroyed and also for each bird left in the arsenal.

4 How to compile the program

Compiling is made easy by utilizing CMake, its lists, git submodules and all-ready build scripts. By running the platform specific build.sh or build.ps1 script on Linux or Windows respectively will make the program and build all necessary file structures for the build of the program.

5 Testing

6 Work log

We had a rough start as we started building the program on top of the Qt graphics library. This idea was totally abandoned after three weeks of building and debugging. After disbanding Qt library and qmake totally we started the game development with SFML and Box2d only. We had relatively regularly a meeting to which all of us attended. The average time of these meetings were 3-6 hours. Each week we checked how are the parts designated to each member of the group progressing. We merged the complete parts to master from their own feature branches. We also helped each other out if someone had gotten really stuck or had something common to ask. Also after all this we decided what to do next week allocating some new job or feature implementation to each member somewhat according to the schedule we had made in the project plan.