Angry Birds

Group 5

Overview

Our software is a rough version of the famous game Angry Birds, where the game's idea is to shoot enemies (pigs) with birds that are launched from a slingshot. Some of the birds have special actions that boost their performance. Our version of Angry Birds has three levels with increasing difficulty and three different birds, one bird with no special action, two with speed boost, where one of the speed boosts will make the birds shoot straight downwards and the other will increase its speed in the direction that it is already moving. We have also implemented two kinds of obstacles, wooden boxes that can be destroyed and moved according to box2d physics and rocks that are solid and are used to create harder levels.

The game will keep track of the player's score, such that, every destroyed obstacle will increase the score by 50 and every pig by a factor of 100. If the player does not use all the birds that are given in the level, the score will increase by a 100 for every unused bird. The player will receive one to three stars depending on how well he/she has passed the level. After every passed level, the player has the opportunity to save his/her highscore and add a nickname. The highscores can be accessed from the highscores list for example in the main menu.

We have also added some sound effects to make the game more user friendly. For example, the menu and the levels have different theme songs and destroying pigs and shooting birds have their own sound effects. We have also worked on better graphics. We have used Paint to make the birds, pigs, obstacles and level and menu background. The slingshot was implemented using SFML library.

The game does not have a level editor to create levels and it does not have multiplayer nor different game modes.

Software structure

The game has three main classes:

- Graphics that handles everything related to SFML and drawing objects on the screen.
- Map that loads information about the level from a file, stores the Box2D objects and handles the physics simulation using Box2D.
- DynamicObject, a simple class for storing additional information about the objects in the game.

The structure is simple, if a game level is not currently being played, everything is done using the methods of the Graphics-class. When a level is started, the runGame-method of the Graphics class creates a new Map-object and loads the obstacles etc. from a file.

Every Box2D object created by Map has some additional information about it stored as a pointer to a DynamicObject that is stored inside the Box2D object's UserData. This way the Graphics-class can find out more information about the objects.

How to compile the program

The program is compiled using Visual studio build tools, which can be downloaded from https://aka.ms/vs/15/release/vs_buildtools.exe

When compiling the program, be sure to be located inside the angry-birds-group-5/bin directory so that the content files will be properly loaded.

There are two required extensions for Visual Studio Code required, CMake and CMake tools. After installing everything, you can press CTRL + SHIFT + P and choose "CMake: scan for kits". Then choose The Visual Studio Build Tools 2017 Release-amd64 option and press the play-button in the blue bottom bar inside Visual Studio Code.

Note: sometimes the options for scanning for kits may not be visible, then the CMake and CMake tools extensions should be disabled and enabled again.

Instructions for building and using the software

We managed to get our project only working on Windows machines so that is needed for using our game. When you have Windows, testing the game is simple.

Here is instructions for that:

1. Clone the git repository to your computer.
2. Go to the project on your file explorer.
3. Open the folder ‘precompiled’
4. Open the folder ‘game’
5. Launch the application‘DispleasedBirdies’
6. Enjoy the ride

Testing

We used mainly two kinds of testing: printing stuff and testing the game manually.

Printing stuff was an efficient way to detect where the problem was and what did not work in the way we wanted. We used this method a lot in all stages of the project but more in the start of the project.

Testing the game manually was the best way to test the user experience. This way we learned if something was hard or bad to use or if it even worked. The problem with this testing method was that this was not possible before we had some application to launch with user interaction. After we had the application this was the best method to use in testing. Best way to test a game is to play it.

Work log

In our group we did not have a clear division of work, because we had similar schedules and we were able to meet 3-6 times a week. In our meetings everybody worked on some part of the code, depending on what we had planned to accomplish that week, and if one of us got stuck, the others helped. Below is listed our workload per week. By the end of the project, we started to divide the workload more.

Week 44:

- We made the project plan together and submitted it. This took us about 6 hours.

Week 45:

- We downloaded the SFML library and tried to make it work. At first, we could not figure out how to make the cmake work properly, but with the help of our assistant, we figured it out. We met on Monday, Tuesday, Thursday and Sunday. Each day we spent about 3-4 hours figuring out how to import the library.

Week 46:

- We downloaded the box2d library and stumbled across the same problem as in the previous week. The cmake could not find the box2d library and thus, we could not include it in any of the files. We spent the week reading documation about the cmake, each of us focusing on different parts, and by the end of week, we made it work. This time we met on Monday, Thursday and Sunday, each day spending about 4-7 hours.

Week 47:

- We were finally ready to start programming the game. This week our target was to start making some basic features of the game. We implemented the window, bird, obstacles and the physics regarding them. By this point the obstacles didn't get destroyed. The hardest part was to figure out how to get the bird flying and to get familiar with the box2d library. We implemented the movement of the bird in the Graphics class. Lassi and Janne worked on the movement of the bird and setting the window, while Sami and Rami worked on the menu display. We worked on Tuesday, Wednesday, Thursday and Saturday, each day spending about 5 hours.

Week 48:

- On this week we wanted to make a fully functional game. We added the enemies, allowed obstacles to be destroyed, added static obstacles, implemented the slingshot and made it possible to load levels from a file. Lassi focused on game physics (allowing obstacles to be destroyed etc.), Rami on implementing static obstacles, Sami on loading levels from a file and Janne on slingshot. We worked on Wednesday, Thursday, Friday and Sunday, each day working about 4 hours. Hardest part was to figure out how to implement destroying obstacles. We decided that we will destroy obstacles depending on the momentum of the object hitting them. Since momentum is a function of velocity, we implemented the functions using the velocity. So if an obstacle is hit at a certain velocity, it will get destroyed.

Week 49:

- In the last week we focused on graphics, creating sounds, levels and fixing bugs. Lassi and Rami focused on graphics, Sami and Lassi on sounds, Rami and Sami on different levels and Janne on fixing bugs. This week we worked

on Tuesday, Wednesday, Thursday, Friday, Saturday and Sunday, each day working about 3-6 hours. There wasn't anything hard left to do so we did not get stuck on anything but we always seemed to find some bugs left to fix.