Angry Birds: Group 3

Object oriented programming with C++

ELEC-A7151

Project Plan

Authors:

Marc Aguado 886732

Roope Kontiainen 707691

Laura Heikkilä 778727

Venla Valve 783974

November 5, 2021

Scope of the work

The goal of our project is to create an actually playable angry birds like game that is close to the original game in terms of the playing experience. The number of content in the game will be fewer: different levels, birds, Easter eggs, etc., because we have considerably less resources than the original creators of Angry birds. Instead we will prioritize the things that are essential for the gaming experience itself: some basic funny graphics, realistic 2D physics, and of course a gameplay system that works.

The main strategy of the project is the following. First, we will implement a very basic version of the game: a bird that flies and some obstacles that can break. When that is working, we will gradually add more complexity to the game. We advance with just about equal speed on all the different aspects of the game, e.g. graphics, physics and content.

This type of strategy is safe. Even if we run out of time, we still have something to show for. Moreover, even if we make it to the advanced phase, it is safer to implement small steps on different directions, rather than go for large one dimensional improvements. This is because each successful step guarantees the end product to be a little better. Also, it is easier to think how the next step will be implemented, when the previous is already known. This strategy will also allow us to divide roles in the group based on different game aspects, because all aspects are being worked on simultaneously.

Minimum features

The game should have a simple user interface with basic graphics, along with information on the number of birds and enemies left.

The throwable objects are birds, and at least one of them comes with a speed boost, explosion or other special effect. The speed boost will likely be the easiest special effect to implement.

There are three different levels that can be loaded from external files. Their difficulty should increase from level to level.

The bird follows simple physics: it can fly, hit objects and fall.
Obstacles and pigs can break, loose hp and fall when the bird hits them.

When thrown, the bird angle and speed can be adjusted with the mouse.

A level is passed if the player destroys all pigs and the level is failed if the player runs out of birds.

More ambitious features

If time permits, we will sound effects to the game, for example when the bird is thrown or an enemy is hit.
We will also implement more special birds that look different and have a special effect.

Adding more than three levels would also make the game experience more interesting.

The objects, including both birds and obstacles should act according to more advanced physics, that is, they can fly, rotate, fall, collide, reflect, lose hp and break.

The player can choose, in which order the birds are thrown.
Additional features also include adding a level menu.

A level is passed with a score that is computed based on the amount of birds used and obstacles broken.

The scores are saved with a player nickname that the player chooses.

Implementation

The Level Loader Module is the main module of the project, it's goal is to read, understand and load the level files of the project (Title Screen, Level Selector, Level 1, Level 2, Level 3). All the other modules "connect" to this module so that it can interpret and add the proper objects into the screen so that the game works properly.

The Controls Module is going to handle all control related aspects of the game, like for instance launching the bird, selecting the level from the selector. Basically it's going to handle the input it receives from the keyboard and mouse accordingly.

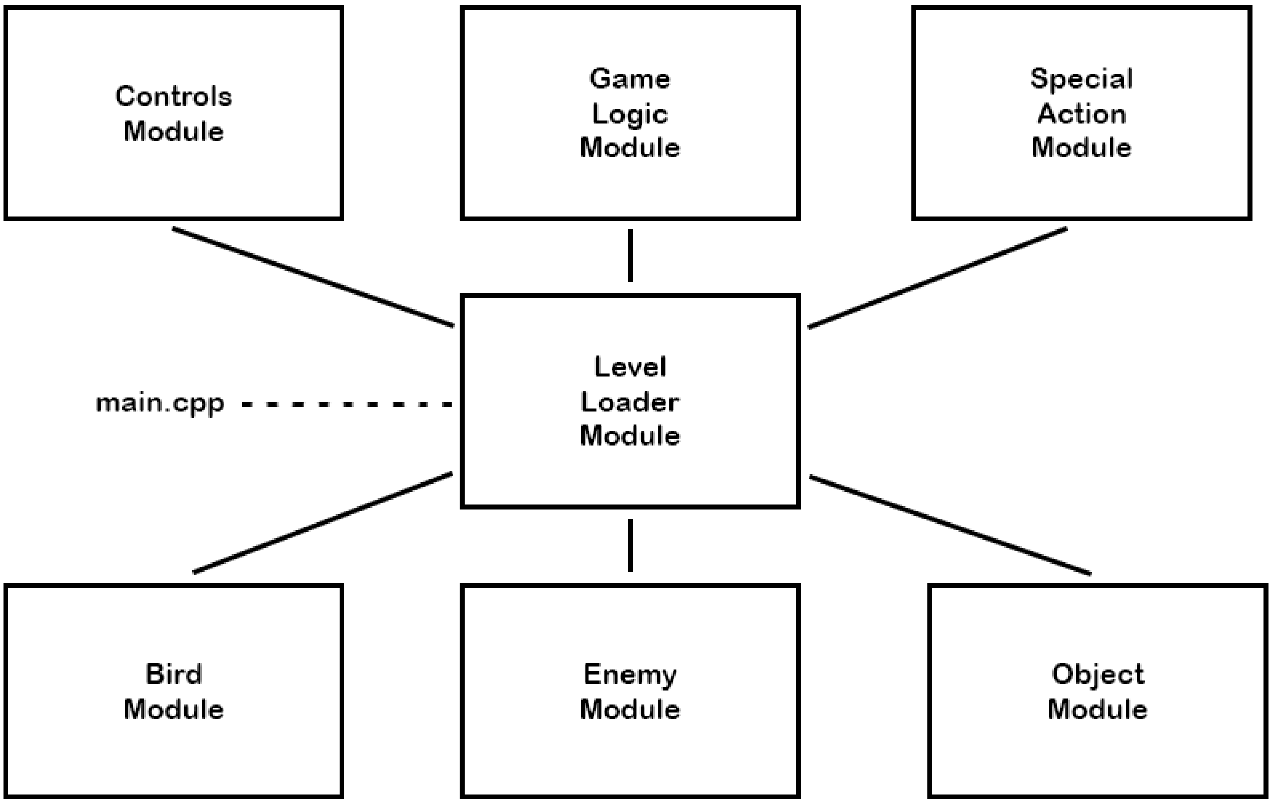
The Bird Module is supposed to allow the LLM to understand what a bird is (abstract class) and what type of birds are there (Bird Type A) so that it can load instances of them into the level. Each Bird Type class will have specific attributes such as actions, graphics, and physical characteristics.

The Enemy Module is also supposed to allow the LLM to understand what an enemy is (abstract class) and what type of enemies are there (Enemy1) so that it can load instances of them into the level. Unlike birds, the enemies won't have any special actions.

The Object Module will let the LLM know what an object is (abstract class), what type of objects are there (ObjectA1) and their attributes, so that it can load instances of them into the level. Some objects will have the same attributes except for their size.

The Special Action Item Module will let the LLM know what special objects are there (slingshot, cannon) and load them into the level.

The Game Logic Module will handle all the game logic like for instance, what enemies are dead, has the goal been reached, etc. This module could be merged with the LLM.



Libraries

For implementing the GUI of the game, we intend to use the widget toolkit Qt. There is an extensive YouTube playlist on implementing C++-based games using Qt that we can turn to, if we face difficulties using the library. For the physics of the game, we plan to use Box2D. The library has been used in the development of the actual Angry Birds game, which indicates that it's features serve our goals. For example, collision detection and applying gravity to the trajectory of the bird will be easier to implement with Box2D. We have divided our group to graphics and physics teams, but our goal is that everybody learns to use both packages.

Division of work and responsibilities

The group agreed that it would be beneficial to pick a leader for the project, even though the workload is distributed evenly. The group leader of our group is Marc Aguado. His responsibilities include ensuring that all the members do their part and the project is finished according to schedule. The group members have different levels of experience in game developing, but we aim to divide the workload so that everyone gets to participate in every phase of the project. We hope that this helps us to divide the workload fairly, when no strict roles are assigned. Moreover, we hope that this approach broadens our individual skill sets the most. However, in our first meeting, we did assign some tentative roles, which are presented in the table below.

Group leader	Marc
Graphics	Marc and Venla
Physics	Roope and Laura
Meeting coordinator	Venla

The graphics team, consisting of Marc and Venla, is in charge of the graphics of the game. This includes designing the game world, birds, pigs and objects. Again, the whole team will have a say in the design process but Marc and Venla will make sure that the graphics get done. Roope and Laura are our physics team, and they will familiarize themselves with the physics library we are using. Also, Venla agreed to be the meeting coordinator who arranges the weekly Zoom meetings and reminds the group members to attend them. The group agreed that after each meeting, one member writes meeting notes and updates them to GitLab. This role will rotate through the group, starting from the second meeting.

Planned schedule and meetings

First draft of the schedule is presented in the table below. At this point, we are not yet sure how much each part of the project will take time, so it is likely that the schedule will change during the rest of the course. We have agreed on a meeting time on Mondays, which will be a good time to check the status of the goal of the previous week, and plan the upcoming week. One person writes notes from each meeting, and we will switch turns each week. If necessary, our group can also flexibly arrange extra meetings during the week. There is some flexibility in the weekly schedule, too, and we can add more functionalities if we have time or leave some of them out if we are running out of time.

Week	Goal
43	Project Plan
44	Getting to know the tools and libraries
45	Creating a game world and the first modules
46	Birds, obstacles and pigs are implemented
47	Bird flies and obstacles break
48	More functionalities and levels are added, if in schedule
49	Project is ready
50	Presentation week