Angry birds

Features and functionalities

We plan to implement all of the basic features and functionalities listed in the A+ assignment. We are also fairly certain that we can implement the high score system lists, star rating algorithm, and sound effects for birds and enemies. We have also put some thought on implementing multiple game modes.

We don't want to limit ourselves to these features, so if we have time, we will try to implement the other additional features listed in the assignment. If we come up with some other good features, we might also add those to the game.

Architecture/Structure

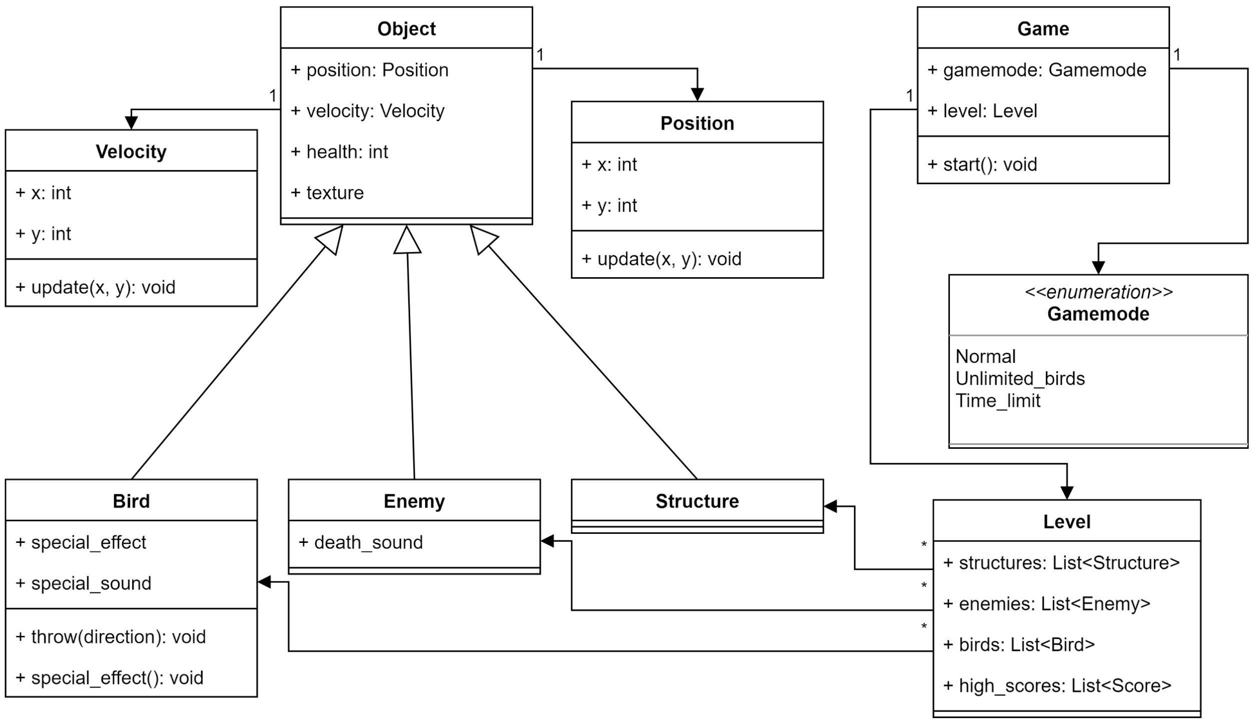
The project consists of a main Game class. The Game classes responsibilities are: level loading and saving, choosing gamemode and handling win conditions. The main class is also responsible for starting the game.

The Game class includes an attribute for a Level object. The Level class contains information on the birds the user can throw, the enemies that are still alive, and structures which are on the Level. The birds, enemies, and structures will most likely be stored as pointers in a list or vector. The Level class also includes the high scores for the level in question.

Object class is a common parent/superclass for the Bird, Enemy, and Structure classes. The Object class includes position and velocity attributes, these might be stored in their own classes or directly in the Object class as integers. Object also has a health attribute which represents how many hits it takes to destroy. The class also contains a texture attribute which specifies the texture of the object. We are not yet sure what is the type of the texture attribute since it depends on how SFML handles textures.

The structure class may not have any new attributes or methods but it overwrites some virtual methods of the Object class. The Bird class is a child of an Object but it's also a superclass for special bird types. These special birds have different textures and special effects with custom sounds for the special effects. The Enemy class is just a simple child of Object with custom death_sound attribute which will be played when the enemy dies.

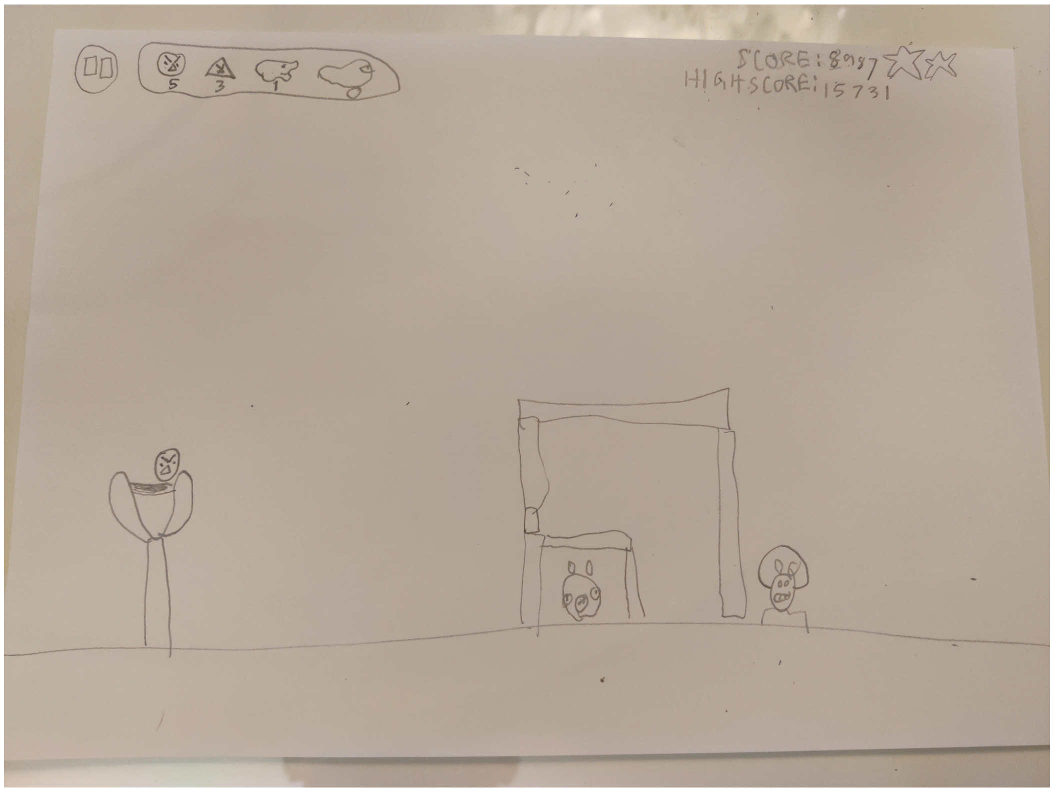
We haven't described any UI classes in the architecture description since we really have no idea what kind of UI classes we need. It also depends on whether we use Qt as an application framework or write the whole user interface with SFML. The diagram is also missing types for some attributes since we don't know yet what types the sounds and textures will be.

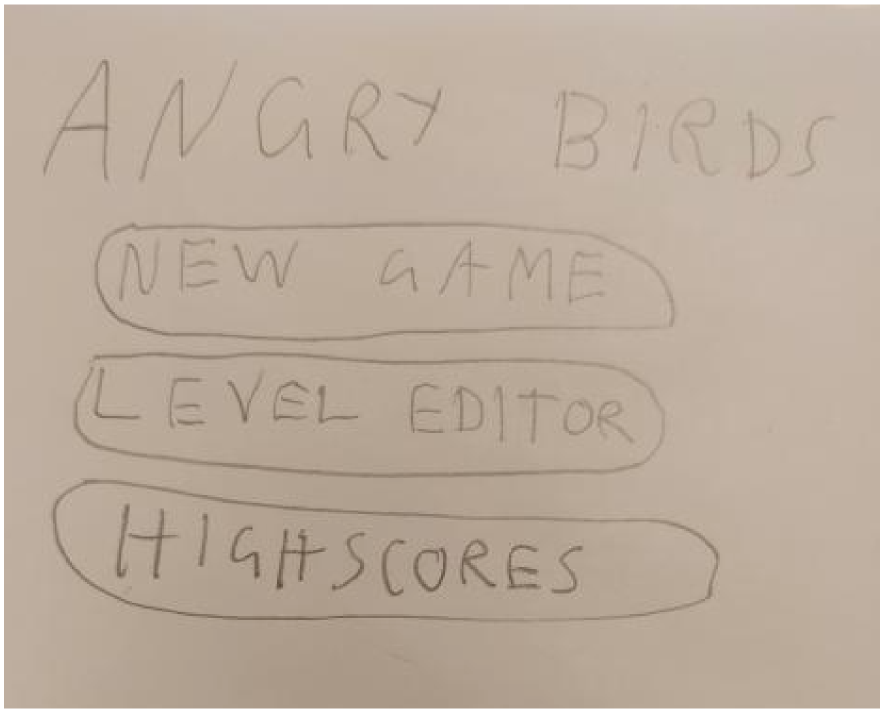


Graphical user interface

Game’s graphical user interface consists of the main menu and the game view. In the main menu, there are possibilities to start a new game, make a new level or check level-based highscores. When the user starts a new game, there is a possibility to choose game mode and level. Also, the user can set their name that will be used in the highscores.

The game view is simple but user-friendly. The game itself will take most of the space and is located in the middle. On the upper left there is a pause button to modify settings or exit the game. On the right of that there are amounts of birds and enemies left shown. There is a high score and current score of the level with stars shown on the upper right corner. When the level is over the game will show a menu with possibilities to restart the level or change to the next game.





Testing

We will create unit tests for essential functionalities in important classes. The GUI will be used extensivelý for testing the game logic and graphics. Printing to standard output will be used at times to solve some individual bugs.

External libraries

Qt

Qt manages our games GUI. Such things as controlling what is currently shown on screen and managing the menu layout and logic will be implemented with it.

SFML

Rendering the actual gameplay and level editing will be done with SFML. It will be very closely integrated with the Qt library. This article shows one way to integrate the two libraries.

Box2D

This library abstracts the 2D physics simulation for us. It works hand in hand with the SFML library.

Bitsery

A serialization library. Used to save, game states, high scores etc. to a file. Serialization will also be used to send game state across a network if we implement multiplayer to the game.

Responsibilities

We decided to spread the responsibilities by the libraries we need to be using.

Kalle is in charge of the basic class structure and game logic. This also includes usage of the Bitsery library to serialize the Level objects.

Sami and Mikko will start to work on the graphical side of the application, which includes Qt and SFML.

Jaakko will be working with Box2D to calculate the flight paths of the birds and collisions with other Objects on the level.

Schedule

On week 44 we were planning and writing this plan. We also tested some of the libraries out.

On weeks 45 and 46 we are going to program the basic class structure and figure out the best way to build the user interface. This also includes learning the libraries.

On weeks 47 and 48 we are going to code the game logic and graphics and physics. We will be writing unit tests along the way as we implement the features.

We will leave week 49 for manual testing and bug fixes. This is just a coarse timeline and it might change if we get stuck on something.