Project Documentation

1. **Overview**: what the software does, what it doesn't do? (this can be taken/updated from the project plan)

Game logic

A classic Angry Birds game is developed. The game's mechanism is to throw bird objects and get points based on performance, where a higher score is given to:

- Destroy more blocks and pigs, or cause a chain reaction with a single shot
- Use fewer birds to clear the level
- Utilize birds' special abilities effectively

After choosing the game level from level 1 to level 3, the data is loaded from the level file. The higher the level, the more difficult the game becomes. Higher difficulty in game level refers to:

- More pigs to kill with fewer birds available
- More complex construction structure

Players can see their scores in real-time. Once the game is finished, the player will get a star rating (maximum 3 starts) to show how well the player played.

Apart from the basic type, there are two special birds implemented: speed bird and explode bird. Speed bird will speed up towards the target if the player presses mouse after the bird is released. The exploding bird will explode after pressing the mouse and blast away anything around it.

Game features

Our game has the following features:

- Stars to rate how well the player played and some logic to calculate them.
- Level editor to create levels and to save them in a file
- Sounds effect
- Excellent graphics(textures, animations)

2. **Software structure**: overall architecture, class relationships diagrams, interfaces to external libraries

https://app.diagrams.net/#G1oC8OIDdc-8rIwZuSGTP_Yggn0FMqK1sa#%7B%22pageId%22%3A%22C5RBs43oDa-KdzZeNtuy%22%7D

3. **Instructions for building and using the software**
   a. How to compile the program ('make' should be sufficient), as taken from the git repository. If external libraries are needed, describe the requirements here
   b. How to use the software: a *basic user guide*

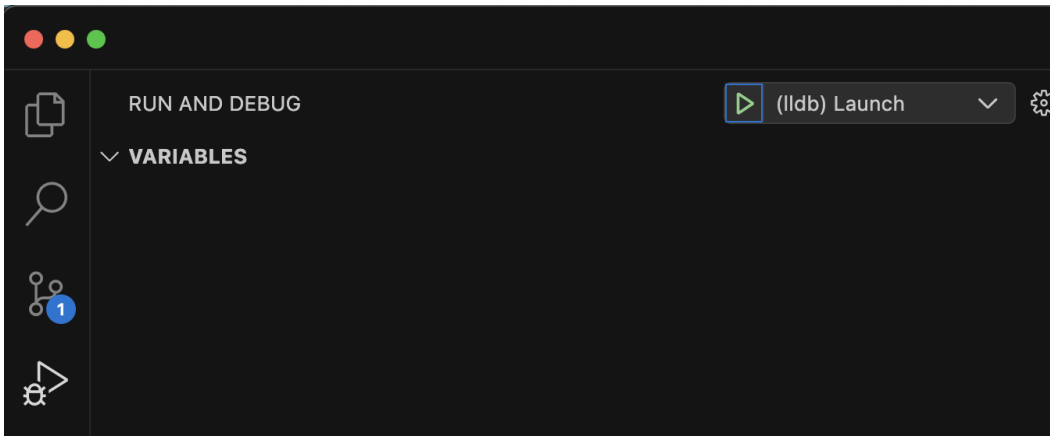
Install Required libraries

Box2D (version 2.4.2 or higher): Install via a package manager, such as:

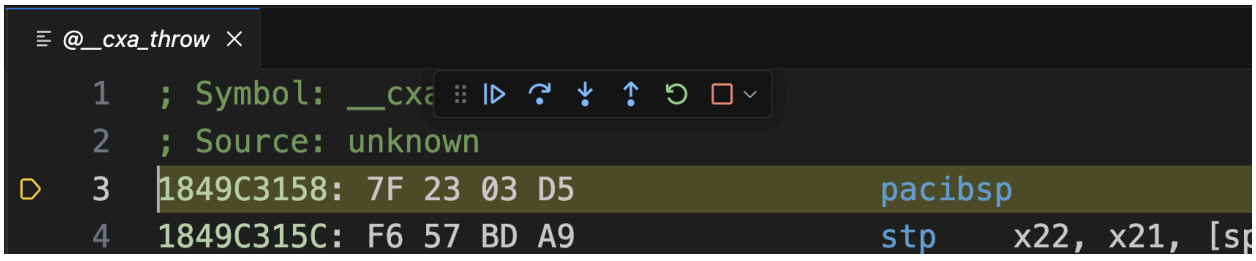
- `Brew install box2d (for mac)`
- `Sudo apt install libbox2d-dev (for linux)`

SFML (version 2.5 or higher): Install via a package manager, such as:

- `brew install sfml (for mac)`
- `sudo apt install libsfml-dev (for linux)`



`Build by RunAndDebug button in VScode and then click(lldb) Launch. Afterwards, you will see @_cxa_throw. Click Step Out to start the Game.`



4. **Testing**: how the different modules in software were tested, description of the methods and outcomes.

Unitest by GoogleTest is used and the program passes all tests.

- Test for Block module:

  Block module represents destructible game objects: ice, wood and stone. Tests validate its construction, interactions(damage), and destruction.

  Create method validates that a `Block` is initialized correctly in the game world.

  Take Damage method ensures that taking damage reduces the block's health points (HP).

  Destroy method validates that a block is added to the `Object::destroyList` when destroyed.

- Test for Bird Module:

  The Bird module encapsulates the behavior of birds used to destroy objects in the game. Tests verify initialization and interaction functionality.

  Create method ensures that a bird is initialized correctly in the game world.

  Attack method ensures the attack mechanic works and the bird cannot attack again immediately.

- Test for Level Module:

  The Level module handles the game levels, including the arrangement of objects and their state. Tests ensure proper loading and initial conditions.

  The load method ensures that the level loading system correctly initializes all objects and state.

  Assertions: Validate expected conditions using Google Test assertions such as:

  - `EXPECT_EQ` for exact matches.
  - `EXPECT_NE` to ensure non-null values or differing results.
  - `ASSERT_NE` for critical checks that halt testing upon failure.

5. **Work log**: a detailed description of the division of work and everyone's responsibilities. For each sprint, describe the tasks that were planned, and what was completed, along with a summary which tasks did each team member contribute during the sprint and how much time was used.

Initially the group had 5 people. The sprint plan was re-made after One person quitted in the first week. Due to inactivity of two team members, the group was split and the sprint plan was not suitable again. After splitting, the group had only 2 people, which made communication easy. Since we found it difficult to estimate how much time is needed, we adopted a flexible schedule: we will just do whatever we feel like doing and notify the other team members. This approach has worked very well. The team also worked very often together on campus which made communication very efficient.

Xin Lin : Took care of textures, GUI, button, game texts, animations mainly.

Jaakko Rautapää: Took care of Game objects rendering, Object implementation, bird launching