

VIETNAM NATIONAL UNIVERSITY HCM
INTERNATIONAL UNIVERSITY

School of Computer Science & Engineering



FLAPPY BIRD PROJECT

Course: OBJECT - ORIENTED PROGRAMMING

Group members:

No	Full name	Student ID
1	Hoàng Việt Quang	ITITDK21049
2	Trần Xuân Mạnh	ITDSIU21100
3	Trần Vi Quý	ITCSIU21214

Contents

1. Abstract.....	2
2. Introduction.....	3
3. Game overview	4
4. UML diagram	7
Overview.....	9
Bird.java.....	10
Cloud.java.....	10
GameBackground.java.....	10
GameElementLayer.java.....	11
GameForeground.java.....	11
GameOverAnimation.java	12
Pipe.java.....	12
MovingPipe.java	12
PipePool.java	13
ScoreCounter.java.....	13
WelcomeAnimation.java	13
5. Game design	14
6. Conclusion	20
7. References.....	20
8. Our Git Project.....	21

1. Abstract

Flappy Bird game, developed and published by dotGEARS, is inspired by classic arcade games such as Super Mario Bros and Helicopter Game. In Flappy Bird, the player's goal is to navigate a bird through a series of pipes without crashing. The game is simple yet challenging. The player taps the screen or just presses only one key on keyboard to make the bird flap its wings and rise, while gravity constantly pulls it down. The objective is to fly as far as possible between sets of pipes. If the bird touches any pipes or the ground, the game is over, and the player must start over.

In this undertaking, the team of “**Po Pox**” created Flappy Bird – Po Pox to bring players a version of a side-scrolling endless runner game with addictive and

straightforward gameplay. The theme of the game is based on the idea of maneuvering the bird through gaps between pipes, requiring precise timing and quick reflexes. Along with the simple concept, the basic rules of Flappy Bird are straightforward, but the game includes innovative features that make it engaging and challenging. The pixel art style and retro sound effects add to the nostalgic feel of the game. The game encourages players to focus and strategize to achieve higher scores. This is the product of an effort to provide players with a fun and addictive game with straightforward yet challenging gameplay.

Keyword: flap, bird, pipe, observation, entertainment, game, object – oriented programming

2. Introduction

In today's rapidly advancing Software Technology industry, a higher level of programming skill is becoming increasingly essential. As a result, the traditional Procedural-Oriented Programming approach is no longer sufficient to meet all requirements. This has led to the development of a new method called "Object-Oriented Programming," following Alan Kay's principles, to address

these challenges.

This project was specifically designed using the Java language, incorporating Object-Oriented Programming. This approach has effectively resolved several issues that commonly arise when using the traditional Procedural-Oriented Method:

- The code becomes more transparent, easily understandable, and concise.
- The project represents a cohesive logical system, achieved by combining numerous related classes.
- Each class contains multiple methods that perform distinct behaviors unique to that class.
- Resources can be effectively reused, enhancing overall efficiency.

The purpose of this project is to design a basic game by using OOP (Object – Oriented Programming) method. Thus, our team decided to create a game that is named “Flappy Bird” followed the four pillars of this measure. This project will go through the game overview design, describing how the game is implemented and the programming functions and libraries used in the design.

Besides the requirements of the course, our team also wants to create this project on account of learning and practicing the techniques throughout the semester. On the other hand, we considered it a good opportunity to have a special mark in the progress of the future career as a programmer. Therefore, the final version is the effort and hard – working of all team members without much interference from other people or available source codes from the Internet.

3.Game overview

Overview

Flappy Bird is a classic arcade-style game that challenges players to navigate a bird through a series of pipes without crashing. The game is simple yet

highly addictive, offering endless gameplay with increasing difficulty. The objective is to achieve the highest possible score by passing through as many pipes as possible.

Game Rules

1. Objective

- The objective of Flappy Bird is to guide a bird through a series of pipes without hitting them.

2. Controls

- Pressing the "Space" key (or tapping on the screen in mobile versions) makes the bird flap its wings, causing it to ascend slightly.

3. Movement

- The bird descends automatically due to gravity.
- Flapping the wings provides upward momentum, allowing the bird to ascend.
- The player must time their flaps carefully to navigate through openings between pipes.

4. Scoring

- Each time the bird successfully passes through a pair of pipes without colliding, the player earns one point.
- The score increases incrementally as the bird passes each set of pipes.



5. Obstacles

- Pipes are positioned at various heights on the screen, creating gaps for the bird to fly through.
- The player must maneuver the bird through these gaps to avoid collision.
- If the bird collides with a pipe or the ground, the game ends.

6. GameOver

- The game ends when the bird collides with a pipe or the ground.
- Upon game over, the player can press the "Space" key again to reset the game and start over.
- The final score achieved in the game is displayed to the player.

7. Difficulty

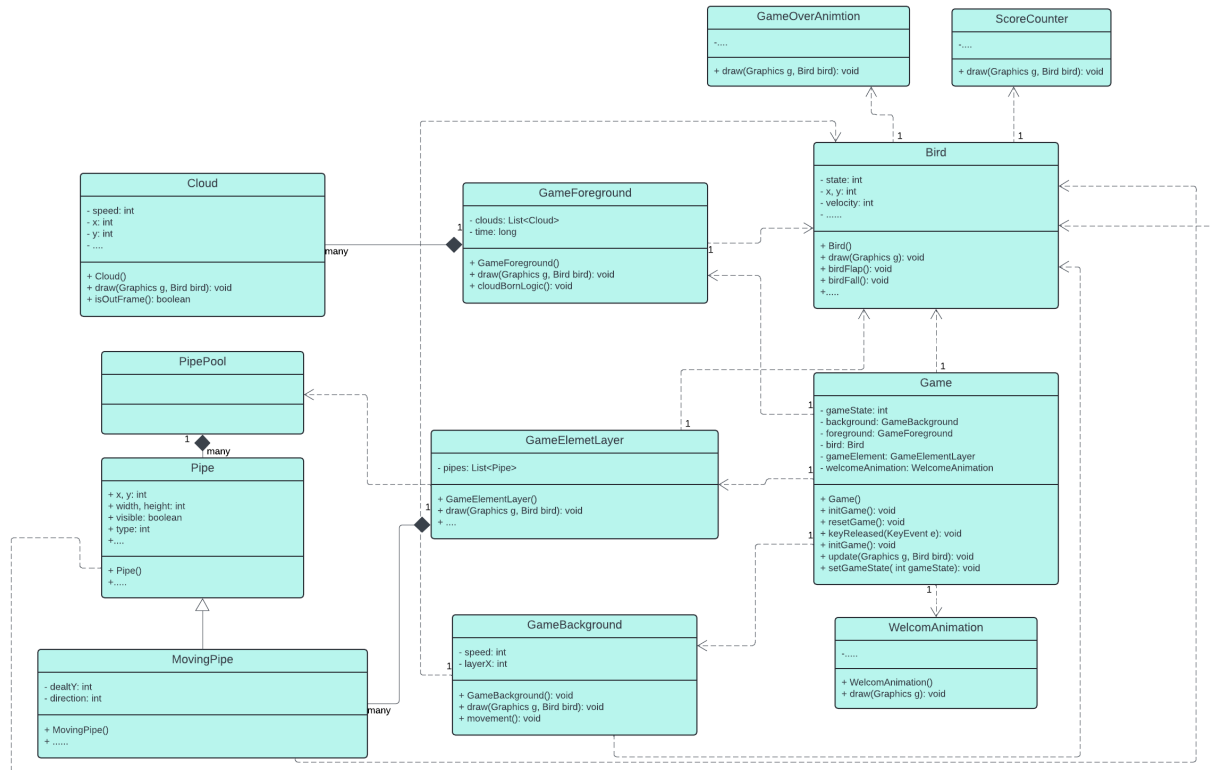
- The game will randomly spawn three different types of pipes: NormalPipe , HoverPipe , and NormalPipe that can move up and down

8. High Score

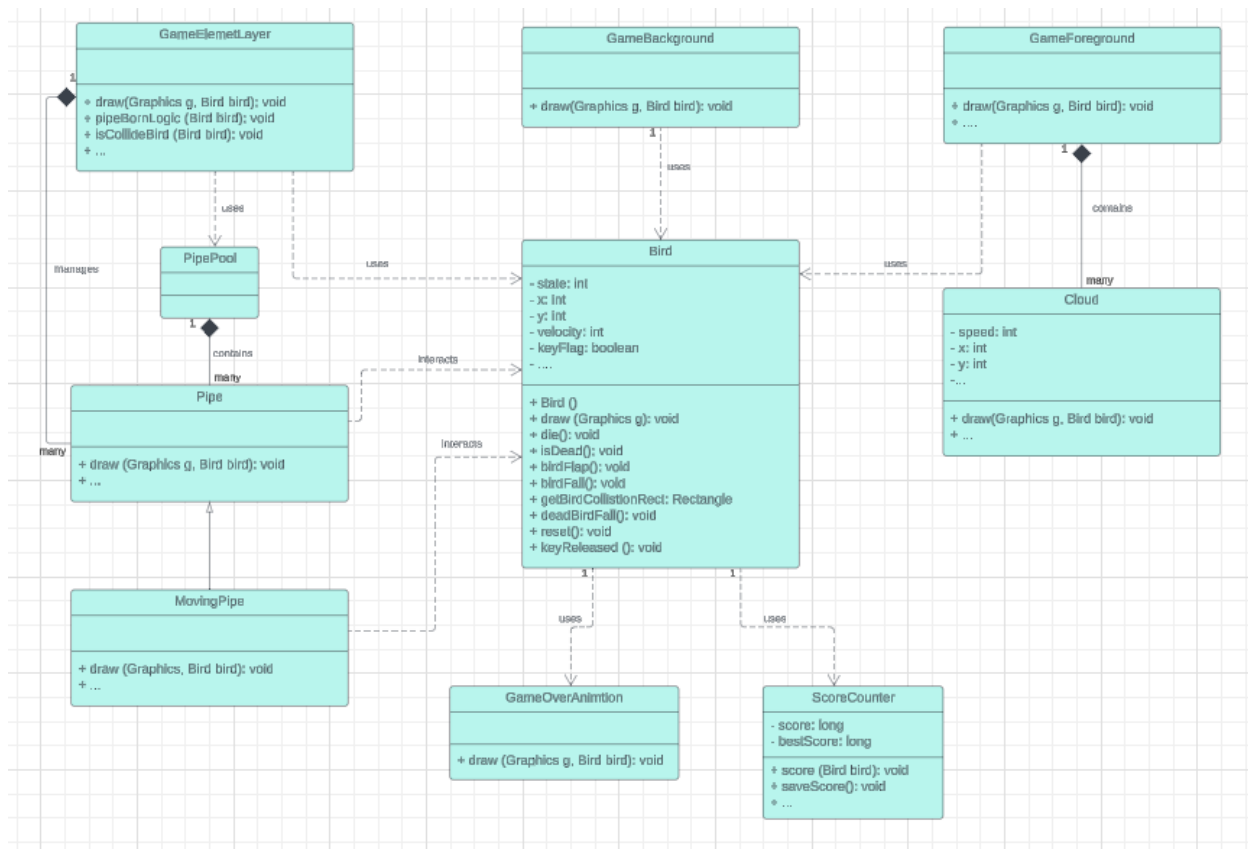
- Players can strive to achieve a high score by successfully navigating the bird through as many pipes as possible without colliding.
- The game typically keeps track of the player's highest score achieved across multiple play sessions.

4. UML diagram

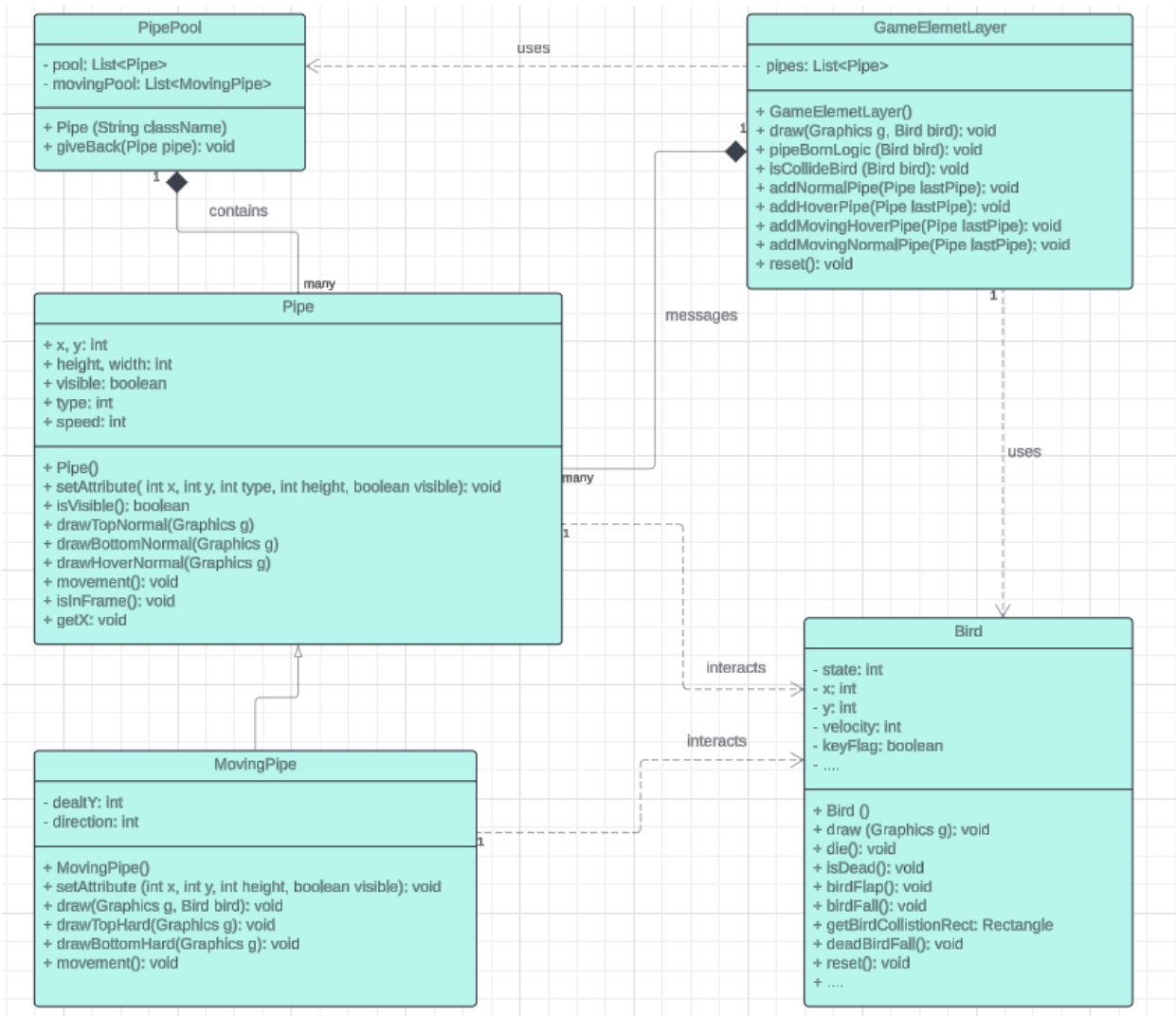
4.1 Overview



4.2 Bird



4.3 Pipe



4.4 Funtion of class:

Overview

The Flappy Bird game implemented in Java is composed of several components that work together to create the gameplay experience. The main components are the **Bird**, **Cloud**, **GameBackground**, **GameElementLayer**, **GameForeground**, **GameOverAnimation**, **Pipe**, **MovingPipe**, **PipePool**, **ScoreCounter**, and **WelcomeAnimation** classes. Each component is responsible for a specific aspect of the game, such as rendering graphics, handling user input, and managing game state.

Bird.java

The `Bird` class handles the bird's state, movement, and rendering.

Key Features:

1. **State Management:** The bird can be in one of several states, including normal, flapping up, falling, dead falling, and dead.
2. **Movement:** The bird's movement is controlled by gravity and user input. The bird falls due to gravity and flaps upwards when the player presses a key.
3. **Collision Detection:** The bird's collision rectangle is used to detect collisions with pipes.
4. **Rendering:** The bird's image is updated and drawn on the screen based on its state and position.

Implementation Details:

- The bird's images are loaded from resources and stored in a 2D array.
- The bird's position and velocity are updated based on its state.
- The bird's collision rectangle is used to check for collisions with pipes and the ground.
- The `draw` method renders the bird on the screen based on its current state and position.

Cloud.java

The `Cloud` class represents the clouds in the background that move horizontally across the screen.

Key Features:

1. **Movement:** Clouds move from right to left across the screen.
2. **Scaling:** Clouds are randomly scaled to add variety to the background.
3. **Rendering:** Clouds are drawn on the screen and are removed when they move out of the frame.

Implementation Details:

- Clouds are created with random positions and scales.
- The `draw` method updates the cloud's position and renders it on the screen.
- The `isOutFrame` method checks if a cloud has moved out of the screen.

GameBackground.java

The `GameBackground` class handles the rendering and movement of the game's background.

Key Features:

1. **Scrolling Background:** The background image scrolls horizontally to simulate movement.

2. **Layer Management:** Multiple layers of the background are drawn to create a continuous scrolling effect.

Implementation Details:

- The background image is loaded from resources.
- The `draw` method renders the background image and updates its position to create a scrolling effect.
- The `movement` method updates the background layer's position.

GameElementLayer.java

The `GameElementLayer` class manages the pipes that the bird must navigate through.

Key Features:

1. **Pipe Management:** Pipes are generated, moved, and removed as needed.
2. **Collision Detection:** Checks if the bird collides with any pipes.
3. **Score Counting:** Increases the player's score when the bird successfully passes through a pair of pipes.

Implementation Details:

- Pipes are stored in a list and updated each frame.
- The `pipeBornLogic` method controls when new pipes are generated.
- The `isCollideBird` method checks for collisions between the bird and the pipes.
- The `draw` method renders all pipes on the screen.

GameForeground.java

The `GameForeground` class manages the foreground elements, such as clouds, that add depth to the game's visuals.

Key Features:

1. **Cloud Management:** Handles the creation, movement, and removal of clouds.
2. **Rendering:** Draws all foreground elements on the screen.

Implementation Details:

- Clouds are stored in a list and updated each frame.
- The `cloudBornLogic` method controls when new clouds are generated.
- The `draw` method renders all clouds on the screen.

GameOverAnimation.java

The `GameOverAnimation` class handles the animation and display of the game over screen.

Key Features:

1. **Game Over Screen:** Displays a game over message and the player's score.
2. **Score Display:** Shows the current and best scores.

Implementation Details:

- The game over images are loaded from resources.
- The `draw` method renders the game over message and scores on the screen.

Pipe.java

The `Pipe` class represents a single pipe obstacle in the game.

Key Features:

1. **Pipe Types:** Supports different types of pipes, such as top, bottom, and hover pipes.
2. **Collision Detection:** Each pipe has a collision rectangle for detecting collisions with the bird.
3. **Movement:** Pipes move horizontally across the screen.

Implementation Details:

- Pipe images are loaded from resources and stored in a static array.
- The `setAttribute` method initializes a pipe's attributes.
- The `draw` method renders the pipe on the screen.
- The `movement` method updates the pipe's position.

MovingPipe.java

The `MovingPipe` class extends the `Pipe` class to add vertical movement to the pipes.

Key Features:

1. **Vertical Movement:** Pipes can move up and down to increase difficulty.
2. **Direction Management:** Handles the direction of the pipe's movement.

Implementation Details:

- Inherits from the `Pipe` class and adds vertical movement logic.
- The `movement` method updates the pipe's vertical position based on its direction.

PipePool.java

The `PipePool` class manages a pool of pipe objects to optimize performance.

Key Features:

1. **Object Pooling:** Reuses pipe objects to avoid creating new objects frequently.
2. **Pipe Management:** Provides methods to get and return pipes to the pool.

Implementation Details:

- Pipes are stored in lists and reused to improve performance.
- The `get` method retrieves a pipe from the pool.
- The `giveBack` method returns a pipe to the pool.

ScoreCounter.java

The `ScoreCounter` class handles the player's score and best score.

Key Features:

1. **Score Management:** Tracks the current and best scores.
2. **Persistence:** Saves and loads the best score to/from a file.

Implementation Details:

- The score is updated when the bird successfully passes through pipes.
- The best score is saved to a file and loaded when the game starts.
- The `score` method increases the current score.
- The `saveScore` method saves the best score to a file.

WelcomeAnimation.java

The `WelcomeAnimation` class handles the animation and display of the welcome screen.

Key Features:

1. **Welcome Screen:** Displays a welcome message and instructions.
2. **Flashing Notice:** Flashes a notice to attract the player's attention.

Implementation Details:

- The welcome images are loaded from resources.
 - The `draw` method renders the welcome message and flashing notice on the screen.
-

This implementation of the Flappy Bird game in Java is designed to be modular and efficient, using object-oriented principles and techniques like object pooling to optimize performance. Each class has a specific responsibility, making the code easier to understand, maintain, and extend.

5. Game design

5.1. Objects

Objects play the most important role, bringing life and soul to this game. There are various types of objects in this game, and each type of object play a specific role. Various objects in this game are used to create UI, thus making it interact-able by players. Therefore, players can use this opportunity to make many special formations, strategies or even performing experiments between objects.

In addition, many objects have the ability to interact with other objects as well, making the game more consistent in terms of game logic. Bird, Pipe, Cloud, Score. Each of them will have different interactions with each other or with players in the game:

- **Bird:** The player-controlled character that navigates through obstacles.
- **Pipes:** Obstacles that the bird must navigate through.
- **Score:** Tracks the player's score.
- **Clouds:** Background elements that add visual depth.

5.2 Game Scene

Scenes are the collection of atmospheres in the game.

Similar to other games, scenes in this game are also very important, as they

can navigate players to make correct decisions based on each scene. Not only that, they can also make the environment of the game become more colorful and lively.

Scenes also provide User Interface, allowing players to interact with them. Many UI are useful since they help players have a better experience of the game. There are total 3 scenes in this game



Fig 5.1 WelcomeScene

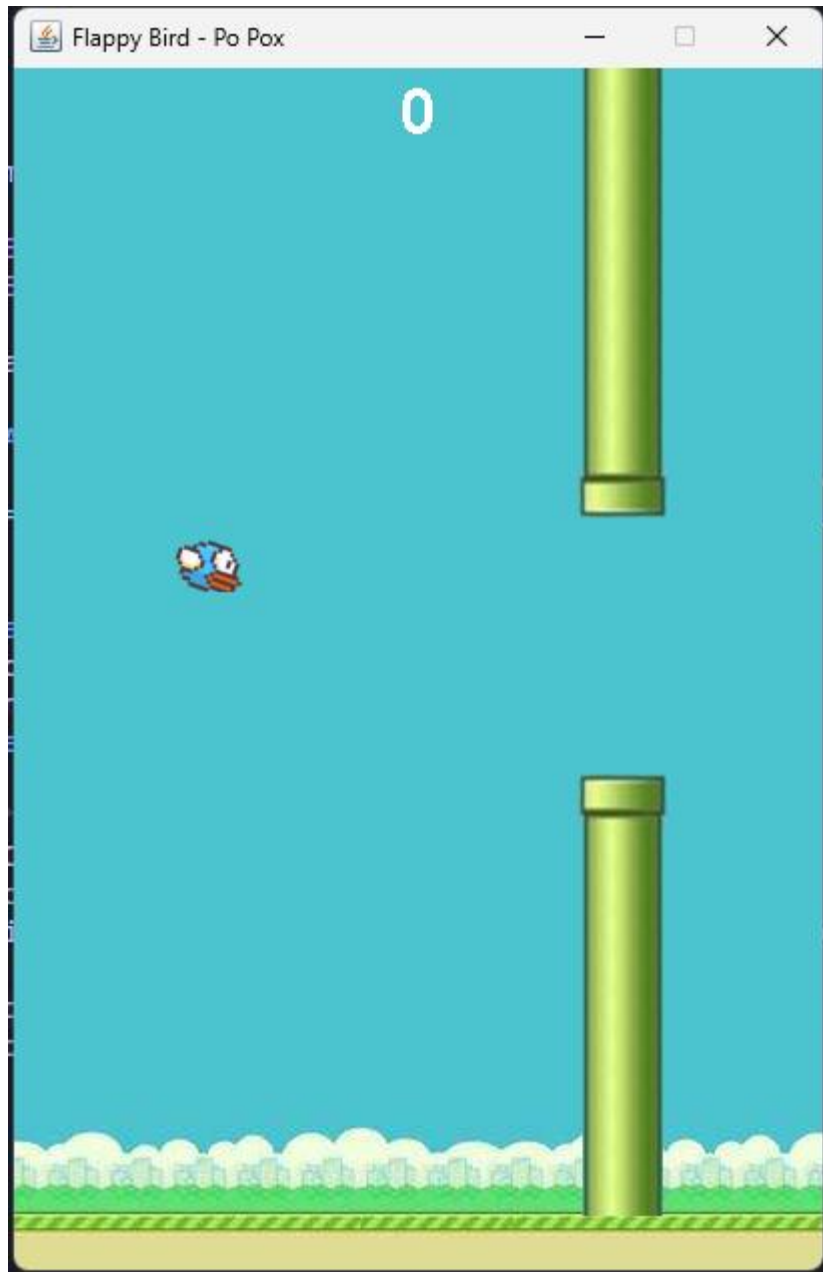


Fig 5.2 Playing Scene

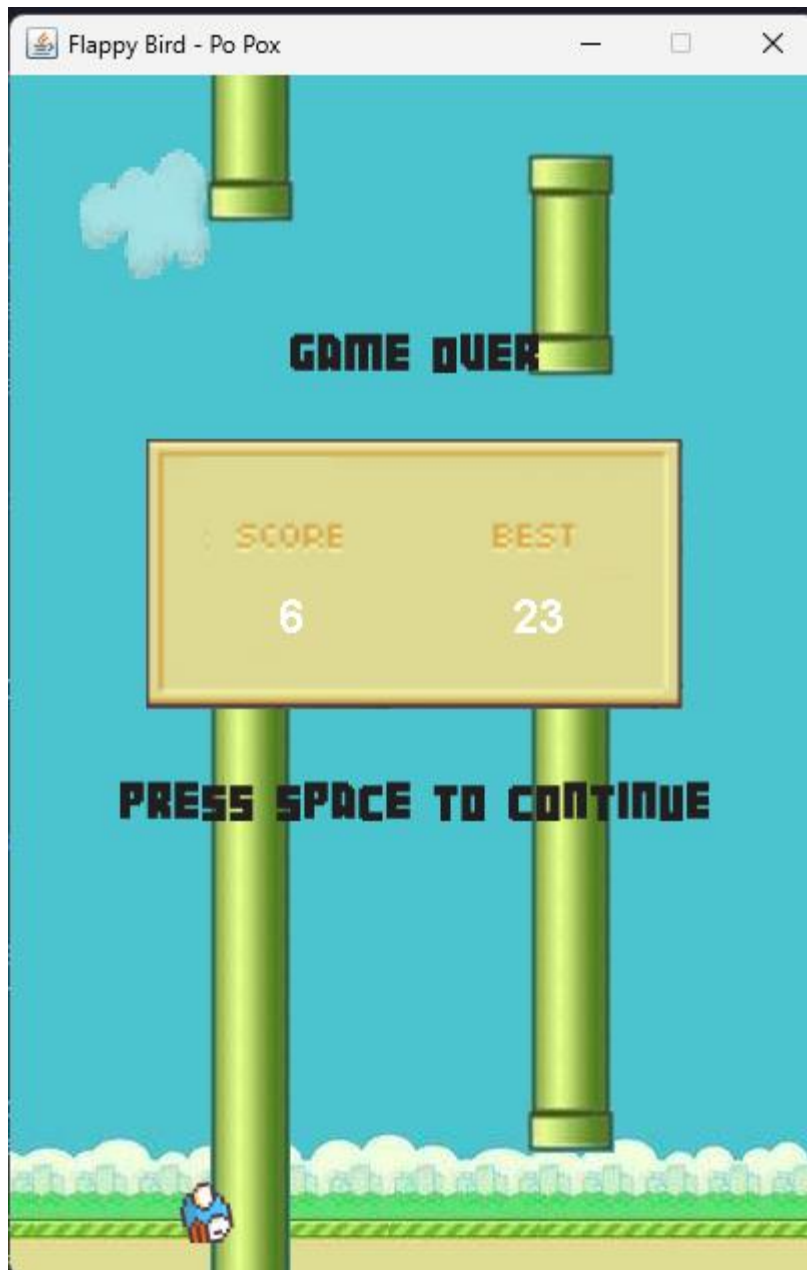


Fig 5.3 GameOver Scene

5.3 Animation

At first, when making the game, the objects themselves did not have any animation, which makes the game look boring and unattractive. Therefore, in order to fix it, animation has come to live.

Using the definition of animation, the animations in this game are created in one of the most traditional yet effective ways: by fast-forwarding many picture frames of an animation, the objects are given a view that looks extremely like an animation.

Animations in this game originally come from many sources, but most of it comes from wiki-fandom or from the original game.



Fig 5.4 Bird state default when in the menu game



Fig 5.5 When bird hit something and GameOver

```
public static final int IMG_COUNT = 8; // Image count
public static final int STATE_COUNT = 4; // State count
private final BufferedImage[][] birdImages; // Bird image
private final int x;
private int y; // Bird coordinate
private int wingState; // Wing state

// Bird image
private BufferedImage image; // Bird image

// Bird state
private int state;
public static final int BIRD_NORMAL = 0;
public static final int BIRD_UP = 1;
public static final int BIRD_FALL = 2;
public static final int BIRD_DEAD_FALL = 3;
public static final int BIRD_DEAD = 4;

public Bird() {
    counter = ScoreCounter.getInstance(); // Get the score counter
    gameOverAnimation = new GameOverAnimation();

    // Load the bird image
    birdImages = new BufferedImage[STATE_COUNT][IMG_COUNT];
    for (int j = 0; j < STATE_COUNT; j++) {
        for (int i = 0; i < IMG_COUNT; i++) {
            birdImages[j][i] = GameUtil.loadBufferedImage(Constant.BIRDS_IMG_PATH[j][i]);
        }
    }
}
```

Fig 5.6 Methods to import every frame pictures to an array and draw bird base on bird's current state

5.4 Audio

Even if audio is not a must in this game, this feature still has a very significant role in order to bring to players a better experience. Without audio or sound, the game will become boring and players no longer have the interest to play the game.

Many objects of this game have their own audio, thus each audio will have its own unique properties. Each audio also brings to players different emotions and feelings, such as: calm, happy or intense feeling

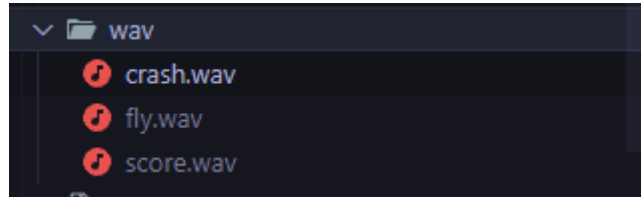


Fig 5.5 Just a simple sound effect used in this game

5.5 Notifications

Notifications in the game serve as important communication tools to inform players about various events, updates, and game-related information. The generation and display of notifications primarily rely on the functionality provided by the Time Logic system and its subsequent descendants. By leveraging this system. Here are some key aspects of notifications in a game like Flappy Bird.

Start game by pressing Space: Notifications can alert players when a game ready to start. It will spam flashing for player know that

Game over and pressing Space to play again: Notifications appear when game is over, player can see the score table and the flashing “Press Space to Countinue”

PRESS SPACE TO FLY

Fig 5.6 Press Space to start the game notification

GAME OVER
PRESS SPACE TO CONTINUE

Fig 5.7 Game Over and Press Space to countinue notification

6. Conclusion

The game is still being developed. After completing a game with some novel features compared to the original version, the team has a more thorough understanding of the four features of OOP, the SOLID principle, and the Design pattern principle, which helps the team be fluent in OOP of game development also in the programming process. Thus, FlappyBird is developed rigorously using the basic idea of OOP, and the game code contains some of four major OOP features (encapsulation, inheritance, abstraction, and polymorphism), the SOLID principle and a design pattern learned from the classroom

7. References

About Github:

<https://www.youtube.com/watch?v=3uq3VCT5V2g&t=334s&pp=ygUPZ2l0aHViIGludGVsbGlq>

https://www.youtube.com/watch?v=mM_drNdss4c&t=1s

About Code:

[Game Loop]

<https://www.youtube.com/watch?v=lW6ZtvQVzyg&t=191s>

[Game GUI by AWT java]

https://www.youtube.com/watch?v=Y5ybC8KS724&list=PLCRogJ_v4BQUuRtS3t_s3TpGFHk8Rsraz&index=7

8. Our Git Project

<https://github.com/ParabolHoang/Flappy-Bird-Po-Pox>