Table of Contents

Flappy Bird Game Application

UML Diagram

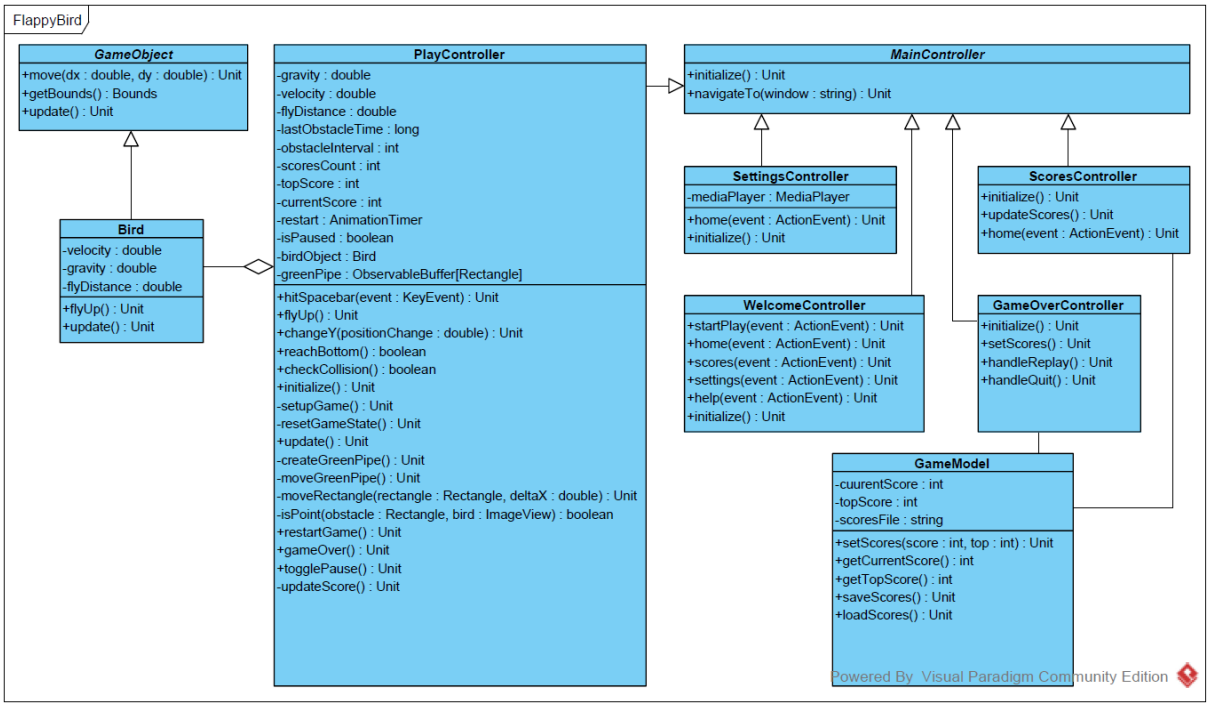

Figure 1: Class diagram of flappy bird game application.

System Functionalities

The application designed is a flappy bird game that allows the player to fly the bird through green pipes and earned scores by avoiding collisions with the green pipes. It consists of a series of functions, in which the four main functions are:

- playing the game (triggered by “Play” button)
- checking the current and top score (triggered by “Scores” button)
- adjusting the background music volume (triggered by “Settings” button)
- viewing user manual of the game (triggered by “Help” button)

Besides, there are also some other functions that enhance the user experience, for instance, the player could navigate to the home window regardless of which window the player is at. This is

made possible by the “Go Back To Home” button in the menu bar (encapsulated within the “Home” in the menu bar), which is illustrated in Figure 2.

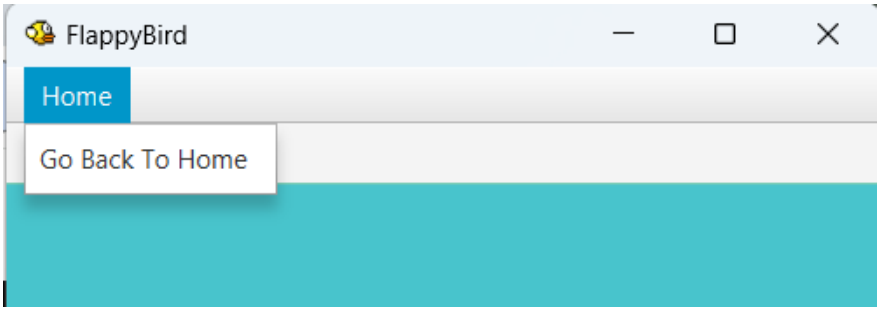

Figure 2: “Go Back To Home” function in the menu bar

Figure 3 shows the user interface of the game application when it is first launched, it acts as the home window where player could navigate to other windows as desired. The four main functions are also visible in the home window.

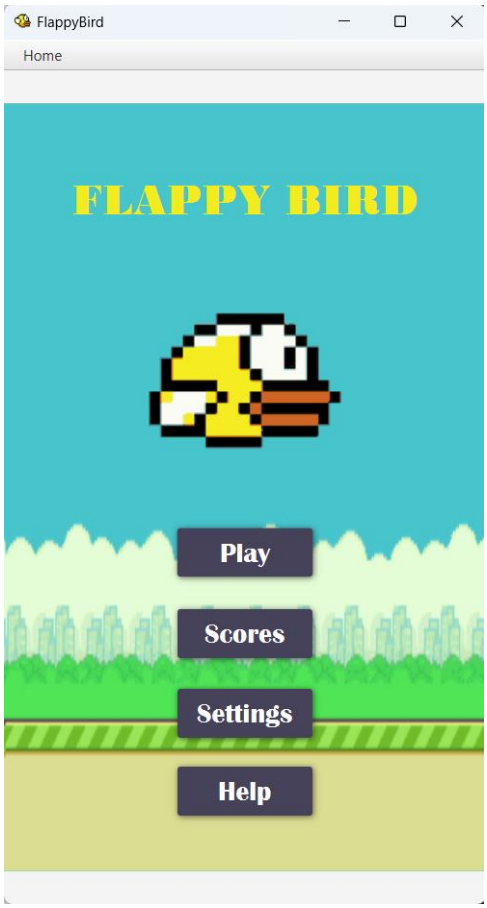

Figure 3: Home window of the flappy bird game application

The detailed descriptions of the four main functions can be seen in the following paragraphs.

Play Game

The “Play” button in the home window will direct the player to a play window (shown in Figure 4) that allows the player to start playing the game. At the top of the play window, there is a number which represents the score collected by the player. If the player successfully pass through the gaps between the green pipes without collision, the player will be awarded one point. The score number at the top of the window will be updated automatically while the player is playing. To play the game, player should hit the “spacebar” on the keyboard to make the bird fly up, otherwise the bird will fall by itself. In order to pause the game while playing, the player could press the “Escape” key on the keyboard to pause and also to resume the game.

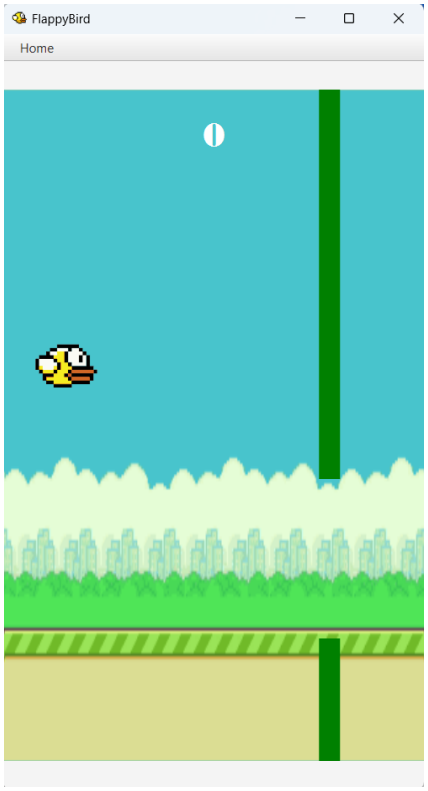

Figure 4: Play window of flappy bird game application

When the bird collides with the green pipes, or falls off the screen, a game over window will be shown, indicating the game has ended. In the game over window (shown in Figure 5), the current score and the top score are displayed. The “Replay” button will clear the current score

of the player and direct the player back to the play window to replay the game. The “Quit” button will direct the player to the home window.

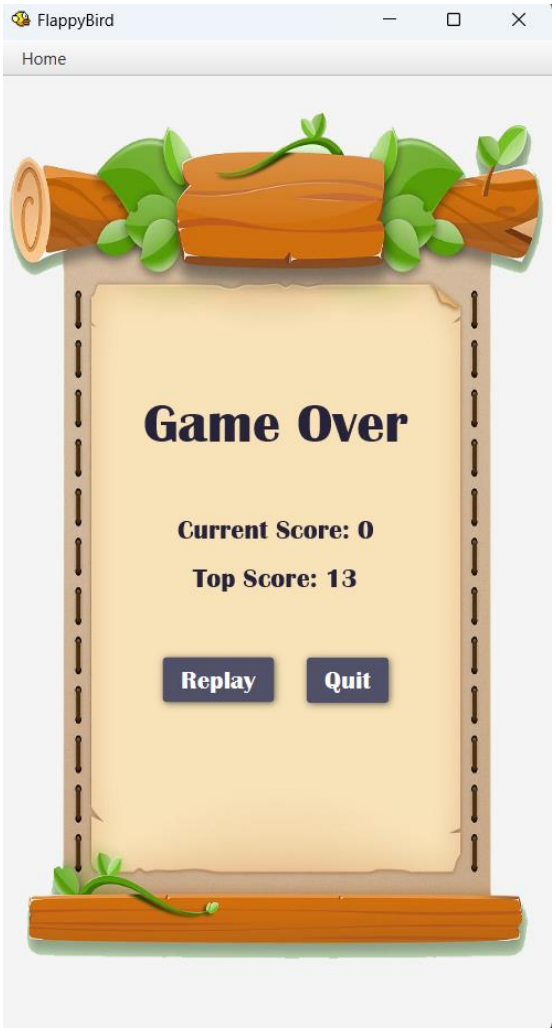

Figure 5: Game Over window of the flappy bird game application

Check Scores

If the player wishes to check the current and top score, the player could click on the “Scores” button in the home window, and a scores window (shown in Figure 6) will be shown. The current score and the top score of the player are updated and stored in a text file (“scores.txt”) and displayed in this window. The current score shows the points collected by the player since the last game, and the top score shows the highest score achieved by the player. The “Back” button in the scores window will direct the player back to the home window.

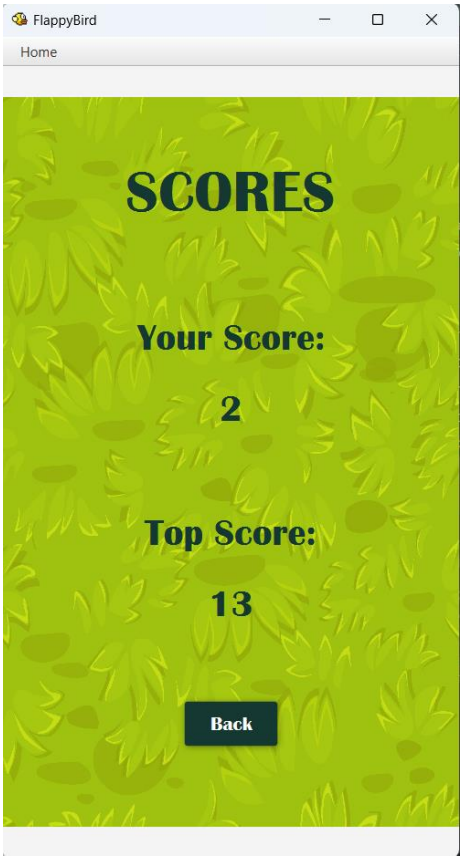

Figure 6: Game Over window of the flappy bird game application.

Adjust Background Music Volume

The game application has a music playing in the background once the application runs. The initial music volume is set to 40% and player could adjust it based on their preferences. To adjust the volume, player could navigate to the settings window by clicking the “Settings” button in the home window. In the settings window (shown in Figure 7), there is a slider to receive input from player to adjust the music volume. The value of the slider ranges from 0 to 100, this indicates that the player could set the music volume from 0% to 100% by sliding the slider. There is also a “Back” button to allow players to return to the home window.

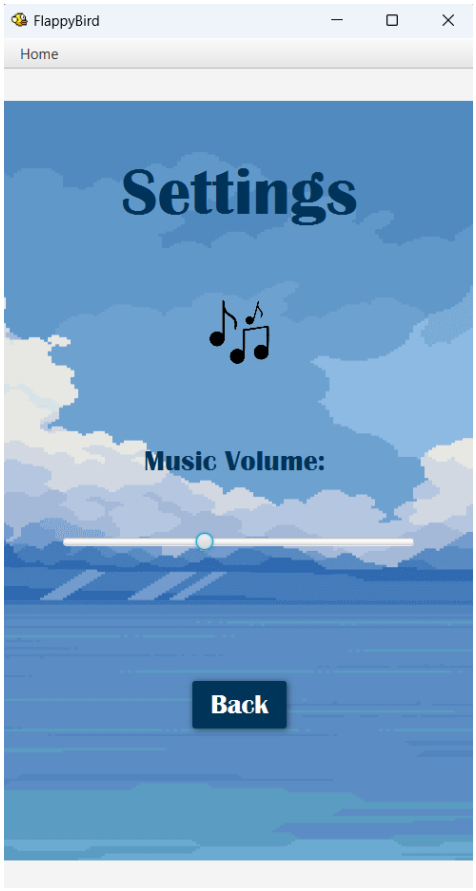

Figure 7: Settings window of the flappy bird game application

View User Manual Guide

This game application provides a user manual guide for players who are not familiar with the flappy bird game. It is located in the help window (shown in Figure 8) and it guides players on how to play the game by introducing the rules and controls of the game. The help window can be accessed by clicking the “Help” button in the home window. After reading the user manual guide, the user could click on the “Back” button to return to the home window.

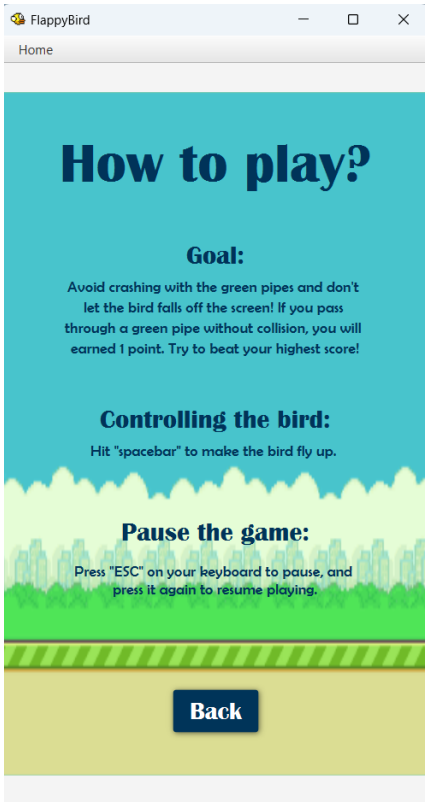

Figure 8: Help window of the flappy bird game application

Personal Reflections

Object-oriented programming (OOP) concepts

Overall I have implemented OOP concepts such as inheritance, polymorphism and abstract class in my game application.

Inheritance:

An abstract class – *GameObject* is created and it acts as a superclass for *Bird* class in this project, this abstract class provides all the basic functionalities for its subclasses. In other words, the *Bird* class inherits from the *GameObject* class. In the *GameObject* class, there are methods like *move(), getBounds*(), and *update()* which the subclass (*Bird*) can reuse and inheritance allows the subclass (*Bird*) to add more specific functionalities while ensuring it can access the common functionalities declared in the superclass after extending from its superclass (*GameObject*). For instance, a more specific method, *flyUp()* is being introduced in the subclass (*Bird*) which depicts a more specific behaviour.

Additionally, inheritance is implemented in the controller classes. The subclasses (*PlayController, ScoresController, SettingsController, WelcomeController, and GameOverController*) inherits from the superclass (*MainController*). The MainController is also an abstract class that provides a blueprint for its subclasses, allowing the subclasses to utilize the common methods declared in the abstract class and implement additional features. All the subclasses are able to reuse the methods declared in their superclass (*MainController*). For example, the method, *navigateTo()* that is declared in the superclass (*MainController*), is being used in all of its subclasses to change to other windows consistently within the game application.

Polymorphism:

As mentioned above, there are two groups of classes that implement inheritance. The *GameObject* class and *MainController* class, both act as the superclass for their respective subclasses. These two groups also exhibits the polymorphism behaviour by allowing the subclasses to modify the methods declared in the superclass. For instance, the *GameObject* class has an *update()* method and its subclass (*Bird*) overrides that method and adds additional game logics within that method. On the other hand, the *MainController* class has a method called *initialize(),* and it is being overridden in several subclasses such as *GameOverController, PlayController,* and *ScoresController* to perform different implementations when needed.

Problems encountered

Game Over Window Display:

I have encountered a problem where the game over window keeps appearing instantly after launching the play window, before I could even play the game. The expected outcome should be launching the game over window after the game has ended in the play window. During the debug process, I made use of the *println()* methods to print out the bird's Y position and the plane height to identify the possible reasons behind the problem. I discovered that the problem could be because the plane height was set to 0 in the beginning. To further elaborate this, the logic to launch the game over window is when the bird's Y position is greater than the plane height, indicating the bird already falls off the screen. However, since the plane height is 0 after the play window is first initialized, which is lesser than the initial bird's Y position (0.2), the game over window is launched every time the play window is first initialized. To solve this, I explicitly initialize the plane height to be 800 when the play window is first launched. In this way, the game over window will only be shown when the bird falls off the screen (bird's Y position is greater than the plane height) or when it collides with the green pipes.

Manage Scores Across Two Controllers:

Another problem I faced in the making of the flappy bird game application is how to manage the scores accessed by two controllers (*GameOverController* and *ScoresController*). When the game ended, the current score and top score in the game over window and scores window should be updated and show the same results. However, only the scores in the game over window is reflected but the scores in the scores window remain zero. To solve this, I created a central repository, *GameModel* class, to store the current and top scores after every game. The GameModel class is responsible for writing and storing the scores into a "scores.txt" file, and every time when both of the controllers are trying to access the current score and top score, they would need to go through the *GameModel* class. By doing so, the scores are guaranteed to be consistent and maintainable.

Strengths and weaknesses

Strengths:

In this project, OOP concepts such as inheritance, polymorphism, and abstract class were implemented, which would make the game application more flexible and scalable, causing the process of adding new features easier in the future. Furthermore, the game application has a simple and user friendly user interface, allowing players to navigate to the desired window and carry out designated functions smoothly and effortlessly. The game application also comes with a feature that allows players to adjust the music volume based on player's preference, thus enhancing the user experience.

Weaknesses:

This flappy bird game application is considered fairly simple and player could get bored after playing it for some time. It lacks more advanced features such as levels of difficulty, storing results of different players, and power-ups which would make the game more customized and challenging. In addition, the transition from the home window to the play window is not as smooth when player clicked on the "Play" button in the home window, causing a slight delay when launching the play window and could disrupt the user experience.

References

Random Code. (2022, April 22). *JavaFX and Scene Builder - Flappy Bird: MVP* [Video]. YouTube. https://www.youtube.com/watch?v=qjMN7FqikxI

Da9el. (2022). FlappyBirdJavaFX. GitHub https://github.com/Da9el00/FlappyBirdJavaFX/tree/main/src/main/java/com/example

OpenAI. (2024) ChatGPT (GPT-4o) [Large language model]. https://chat.openai.com/chat

Chin, T. M. (2024). AddressApp source code [Source code].