Computer Graphics And Animation Mini Project- CIE 03

Introduction :

Flappy Bird is a popular mobile game that gained widespread attention in 2014 for its simplistic gameplay and addictive nature.In order to guide a bird through a series of pipes without touching them, players must tap the screen to cause the bird to flap its wings. The goal is to successfully go through as many pipes as you can in order to earn the greatest score.In this project, we'll use the Java programming language to create a Flappy Bird game. Due to its simplicity of use and platform independence, Java is a popular language for game creation. Through this project, we will investigate fundamental game creation ideas like graphics, animation, and handling user input. The graphical user interface will be made using the Java Swing toolkit, and game logic will be included to make the game experience interactive and interesting. So get ready to explore the fascinating world of game development and build your own Java-based Flappy Bird game!

Software Requirements :

1. Java Development Kit (JDK): To compile and execute Java code, JDK is necessary. The Java Compiler and Java Runtime Environment (JRE) are included.
2. Integrated Development Environment (IDE): Eclipse or NetBeans are two examples of suggested IDEs for Java application development. IDEs offer a practical setting for editing, debugging, and testing code.
3. Java Swing : A GUI toolkit called Java Swing enables programmers to design graphical user interfaces for Java applications. It offers a collection of game interface-creating tools, including buttons, labels, text fields, and panels.

Design/Game Logic:

The game was developed in different files in the following order :

1) Util

a. The code defines a Util class that provides a loadImage() method to load an image from a given file path. The method uses a cache of previously loaded images to avoid reloading them every time they are needed.
b. The loadImage() method first checks if the requested image is already in the cache, and returns it if it is.
c. If the image is not in the cache, the method reads the image from the file using the ImageIO.read() method, and adds it to the cache.
d. If the file cannot be read, the method prints an error message to the console and returns null.
e. The Util class is mainly used by the Render class to load images for rendering.

2)Render

a. The Render class has four fields: x and y for the position of the image, image for the image to be rendered, and transform for any transformation to be applied to the image.
b. The class has two constructors: a default constructor that doesn't initialize any fields, and a parameterized constructor that initializes the x, y, and image fields with the provided values. The Util.loadImage() method is used to load the image from the provided image path.
c. The Toolkit.getDefaultToolkit().sync() method is called to synchronize the graphics state to avoid flickering or other visual artifacts when rendering the image.
d. The Render class does not have any methods to update or manipulate its fields. Instead, other classes, such as the Bird and Pipe classes, use Render objects to render images on the screen by setting its fields appropriately.
e. The Render class is not responsible for rendering the image on the screen; this is handled by the App class, which uses the Graphics.drawImage() method to draw the image at the specified position and with the specified transformation.

3)Pipe

a. The Pipe class represents the pipes in the game that the bird must fly through.
b. It has fields for the pipe's position, width, height, speed, and orientation (north or south).
c. The reset() method sets the pipe's width, height, and x position to their initial values, and sets the y position randomly if the pipe is oriented south.
d. The update() method updates the pipe's position based on its speed.
e. The collides() method checks if a given rectangle (specified by its position and dimensions) collides with the pipe.
f. The getRender() method returns a Render object that contains information about how to draw the pipe on the screen, including its image.

4)Bird

a. The Bird class represents the player character in the game.
b. It has fields for the bird's position, width, height, velocity, and gravity.
c. The update() method updates the bird's position and velocity based on gravity and user input.
d. The getRender() method returns a Render object that contains information about how to draw the bird on the screen, including its image and rotation.

5) Keyboard

a. The Keyboard class contains a private constructor, and the instance of the class is created using a static method called getInstance.
b. The getInstance method returns an instance of the class if it is null, otherwise, it returns the existing instance.
c. The class also has an array of boolean values called keys that corresponds to the 256 possible keys on a keyboard.
d. The class implements the three methods required by the KeyListener interface: keyPressed, keyReleased, and keyTyped.
e. The class provides a method called isDown that takes an integer key code as an argument and returns a boolean value indicating whether the corresponding key is pressed.

6)Game

a. The code consists of four classes: App, Game, Bird, and Pipe, as well as two utility classes: Keyboard and Util. Here's a summary of each class and its purpose:
b. App - This class creates a window and sets up the game loop.
c. Game - This class contains the main game logic, including updating the bird, moving pipes, and checking for collisions. It also handles rendering of game objects.
d. Bird - This class represents the bird object in the game. It contains its position, velocity, and sprite.
e. Pipe - This class represents the pipes in the game. It contains its position, size, and orientation.
f. Keyboard - This class handles input from the keyboard.
g. Util - This class contains utility methods, including loading images from disk.

h. The Game class has a paused Boolean variable to keep track of whether the game is paused or not.
i. The game also has a pauseDelay, restartDelay, and pipeDelay integer variables to keep track of delays between certain events.
j. The Game class has a bird and an ArrayList of pipes, as well as a keyboard instance.
k. The Game class has a score, gameover, and started Boolean variable to keep track of game status.
l. The update method in the Game class updates the game state, including moving the bird, moving the pipes, and checking for collisions.
m. The getRenders method in the Game class returns an ArrayList of Render objects, which represent the game objects that need to be rendered.
n. The Game class has several private helper methods that handle input, move pipes, and check for collisions.
o. The Game class has a restart method that resets the game state to its initial values.

7) GamePanel

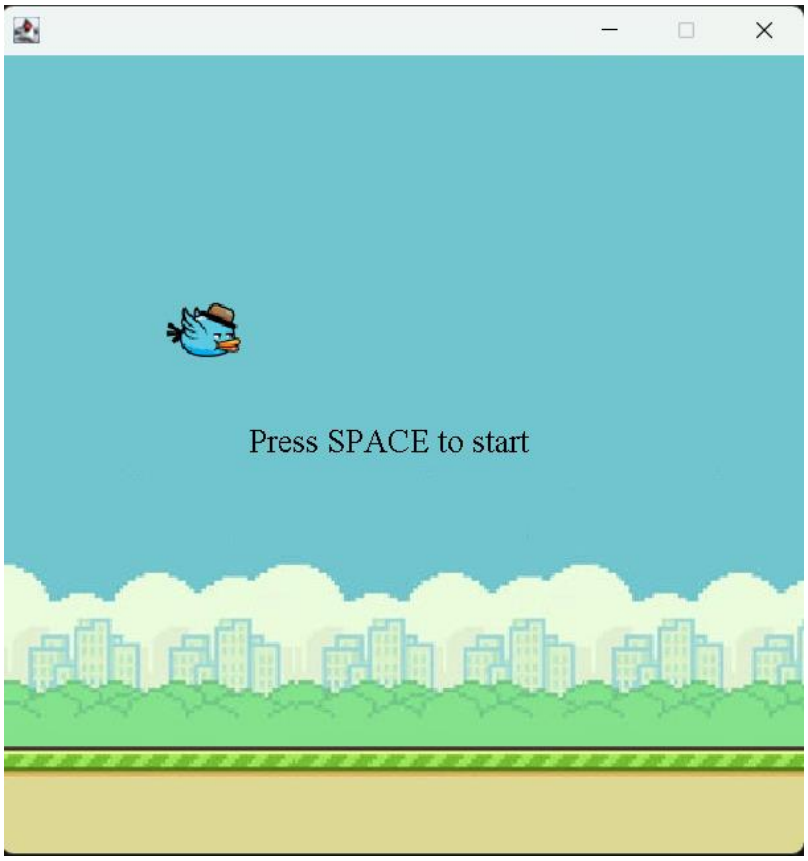
a. The GamePanel class extends JPanel and implements the Runnable interface for multi-threading.
b. It has a private instance of the Game class.
c. The constructor initializes a new instance of the Game class and starts a new thread.
d. The update method updates the game state and repaints the panel.
e. The paintComponent method is called whenever the panel is drawn on the screen. It clears the panel, draws the background and foreground images, draws the bird and pipes, and displays the score or messages if the game is not started or is over.
f. The run method is called in a new thread, and it continuously updates the game state and repaints the panel every 25 milliseconds until the thread is interrupted or an exception is thrown.

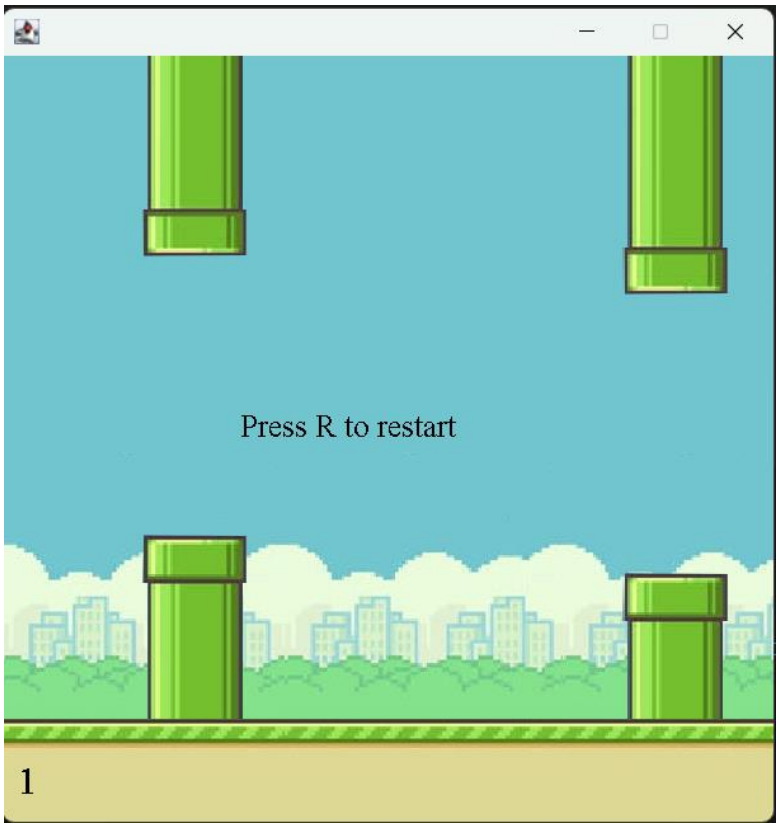
8) App

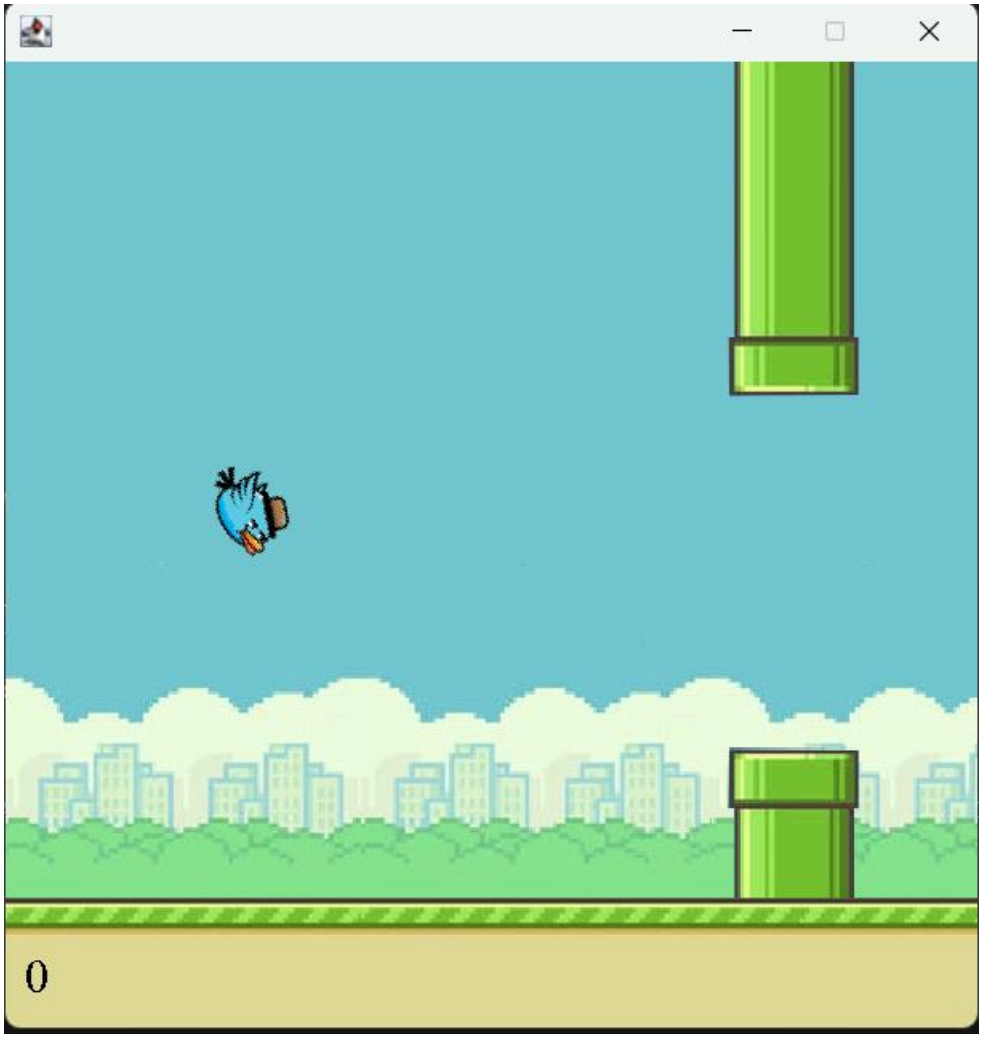
a. This is a Java program that creates a game using Swing.
b. The program contains a class named "App" which has a main method.
c. The "App" class creates a JFrame object, sets its visibility to true, sets the default close operation to exit on close, and centers it on the screen.
d. The "App" class creates a Keyboard object using a Singleton design pattern, and adds it as a key listener to the JFrame object.
e. The "App" class creates a GamePanel object, adds it to the JFrame object, and sets the size of the JFrame object to 500 x 520 pixels. The JFrame object is also set to be non-resizable.
f. The "GamePanel" class extends the JPanel class and implements the Runnable interface. It has a "game" object of the "Game" class.
g. The "GamePanel" constructor creates a new "Game" object and starts a new thread with the "GamePanel" object as the target.
h. The "GamePanel" class has an "update" method which updates the game state and repaints the panel.

i. The "GamePanel" class has a "paintComponent" method which is called when the panel is painted. It paints the game objects using the Graphics2D object. It also displays messages on the screen depending on whether the game has started or is over.
j. The "GamePanel" class has a "run" method which continuously updates the game state and repaints the panel with a delay of 25 milliseconds between each update.

Screenshots:







Resourse:

- Images Used: Bird ,Background, Foreground ,North Pipe, South Pipe (Taken from various Websites)
- Code Learning Basis: W3school
- Java Swing : javapoint.com

Contributers:

1. Yashashree Bedmutha – 02
2. Yojana Naik- 05
3. Jayraj Khivensara - 21
4. Nayan Chaudhari - 26