FALLING PLANE GAME USING UNITY

PROJECT PHASE I REPORT

Submitted by

RAHULPRASANTH P 22P603

HARSHAVARTHA K 22P604

ARAVIND KUMARAN A 22P605

MOHAMMED ASRAF S 22P606

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

ANNA UNIVERSITY, CHENNAI

DECEMBER 2023

FALLING PLANE GAME USING UNITY PROJECT PHASE I REPORT

Submitted by

RAHUL PRASANTH P 22P603

HARSHAVARTHAN K 22P604

ARAVIND KUMARAN A 22P605

MOHAMMED ASRAF S 22P606

in partial fulfillment for the award of the degreeof

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

ANNA UNIVERSITY, CHENNAI

DECEMBER 2023

KARPAGAM COLLEGE OF ENGINEERING

(Autonomous)

COIMBATORE – 641 032

FALLING PLANE GAME USING UNITY

Bonafide record of work done by

RAHULPRASANTH P 22P603

HARSHAVARTHAN K 22P604

ARAVIND KUMARAN A 22P605

MOHAMMED ASRAF S 22P606

Dissertation submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING

COMPUTER SCIENCE AND ENGINEERING

of Anna University, Chennai

DECEMBER 2023

……...……..…	…….……..…….
Ms. AARTHI.D M.E., (Ph.D)	**Dr. T. RAVICHANDRAN M.E., Ph.D**
Assistant Professor	
Faculty Guide	Head of the Department

Certified that the candidate was examined in the viva-voce examination held on

…….

……..	……..
Examiner 1	Examiner 2

ACKNOWLEDGEMENT

We would like to show our gratitude to the management of Karpagam College of Engineering **Dr. R. Vasanthakumar, B.E., (Hons), D.Sc.,** Chairman and Managing Trustee, Karpagam Educational Institutions for providing us with all sorts of support in completion of this project.

We express our sincere and profound gratitude to our principal **Dr.V.Kumar Chinnaiyan M.E.,Ph.D** for his guidance and sustained encouragement for the successful completion of this project.

We feel immense pleasure in expressing our humble note of gratitude to our Head of the Department **Dr. T. RAVICHANDRAN ME., Ph.D** for his remarkable guidance and besides his positive approach, he has offered incessant help in all possible ways from the beginning.

We are grateful to our Project Coordinator **Dr. R. KALA M.E Ph.D.,** Assistant Professor, Department of Computer Science and Engineering for his valuable suggestions and guidance throughout the course of this project.

We are thankful to our project guide **Ms. AARTHI. D M.E., Ph.D,** Assistant Professor, Department of Computer Science and Engineering for her valuable suggestions and guidance throughout the arise in the course of the project.

We also extend our thanks to other faculty members, parents, and friends for providing their moral support in the successful completion of this project.

ABSTRACT

In this project, we design and implement a 2D video game named as “FALLING PLANE” on the Unity engine using C# . In this game, the player can control the vertical movement of plane (every pressing on the keyboard makes the plane leap upward for a little bit, and the plane will fall freely without control). As soon as the game begins, tubes will keep appearing from the right side of the screen and moving leftwards. (so that it seems like the plane flying forward). The goal of this game is to control the plane, dodging and passing the incoming tubes, as many as possible. The game is endless until the plane eventually hit one of the tubes, ground.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

The game is a side-scroller where the player controls a plane, attempting to fly between rows of green pipes, which are equally sized gaps placed at random heights, without coming into contact them. Each successful pass through a pair of pipes awards the player one point.If the player touch the pair of pipes, it ends the game. The plane briefly flaps upward each time the player taps the key,If the key is not tapped the plane falls due to gravity. The player is awarded with several milestones,such as a gold medal if they reached hundred points

1.1 Problem Statement

Now days many of the developers and programmers are making many games and it may provide a lot of features to but it is not that much easy to develop a full fledge feature game as a new developer or a programmer. I faced these problems, when I was creating this game.

1.2 Proposed Solution

Falling Plane is a game where the player tries to make the Plane cross the maximum number of pillars while it is fly. If the Plane hit any pillar, or to the ground, it will eliminate and the game will be over (ends). Pillar's height come up randomly and can come in variable velocity. The player tries to navigate the Plane safely through the maximum possible number of hurdles. Each hurdle crossed successfully adds one to the score.

1.3 Customer

As this is a desktop application game for the only use of desktop and operating system devices so all the active users or those who will download it will be the direct customers of this game developers and it is the developer's utmost duty to fulfill the customers with their requirement.

1.4 Affected Groups

Affected people are as follows:-

- Children
- Teenage

These above-mentioned people will be affected by this game, as it is the purpose or main objective of the game to be downloaded and played by these mentioned groups of People

1.5 Goals and Objectives

- This game is a very good game having all the basic features and basic functionalities require to make a game so, these functionalities make it a very good user experience which is the main purpose for any of the game
- Have a good user interface for better user engagement.
- Full fledge functional game that user would like to play.

1.6 Assumptions

Following are the assumptions when the project was being under development:-

- Flappy plane game will be an addictive game
- It will be the way of entertainment for everyone especially for children and teenagers
- It loses stress.

2 UNITY GAME ENGINE

Unity is a cross-platform game engine developed by Unity Technologies, which is primarily used to develop video games and simulations for computers, consoles and mobile devices. First announced only for OS X, at Apple's Worldwide Developers Conference in 2005, it has since been extended to target 27 platforms. Unity is an all purpose game engine that supports 2D and 3D graphics, drag and drop functionality and scripting through C#.

Development Environment:

-Unity provides a comprehensive integrated development environment (IDE) for game development.

- It supports both C# and a JavaScript-like language called UnityScript for scripting.

Cross-Platform Support:

- Unity allows developers to create games for a wide range of platforms, including Windows, macOS, Linux, iOS, Android, Xbox, PlayStation, Nintendo Switch, web, and more.

- Developers can create games for multiple platforms simultaneously with minimal code changes.

Graphics and Rendering:

- Unity offers a robust rendering engine that supports high-quality 2D and 3D graphics.

- It supports advanced rendering features like dynamic lighting, shaders, post-processing effects, and real-time global illumination.

Physics and Simulation:

- Unity includes a physics engine that allows for realistic physics simulations and interactions.

- This engine can handle rigid body dynamics, collision detection, and more.

Asset Management:

- Unity's Asset Store provides a marketplace for developers to purchase or sell 2D and 3D assets, scripts, tools, and plugins.

- The engine also has powerful asset management tools for importing, organizing, and optimizing assets.

Scripting and Customization:

- Developers can use C# or UnityScript to write custom scripts for gameplay, AI, and more.

- Unity's component-based architecture allows for easy customization and extensibility of game objects.

Animation and Audio:

- Unity supports the creation of complex animations, including character animations and particle systems.

- It has a robust audio system for sound effects, music, and spatial audio.

Networking and Multiplayer:

- Unity offers networking solutions for building multiplayer games, including real-time multiplayer, matchmaking, and server hosting options.

Editor and IDE:

- Unity's editor is highly customizable and provides tools for scene design, level editing, asset management, and debugging.
- It supports visual scripting with the Bolt visual scripting tool.

AR and VR Support:

- Unity has built-in support for creating augmented reality (AR) and virtual reality (VR) applications, making it popular for these emerging technologies.

Monetization:

- Unity offers a wide range of monetization options, including in-app purchases, advertising, and analytics tools.

Community and Ecosystem:

- Unity has a vast and active developer community that shares knowledge and resources.
- It also has a rich ecosystem of third-party plugins and extensions.

Licensing:

- Unity offers both free and paid licensing options. Unity Personal is free for small developers, while Unity Pro provides advanced features for larger teams and studios.

Deployment and Distribution:

- Unity makes it easy to package and deploy your games to various platforms, including app stores, console marketplaces, and web portals.

Documentation and Learning Resources:

- Unity provides extensive documentation, tutorials, and online courses to help developers get started and improve their skills.

Unity's versatility, ease of use, and the ability to target multiple platforms have made it a popular choice for both indie and professional game developers. It's also used for a wide range of applications beyond gaming, including architectural visualization, training simulations, and interactive experiences.

2.1 UNITY PLATFORMS

Unity is particularly popular for mobile game development and much of their focus is on mobile platforms and desktop. Unity3D's 2D pipeline is a more recent addition to the engine, and is less mature than the 3D pipeline. Despite this Unity is an adequate platform for developing 2D games even when compared to other dedicated 2D engines, particularly if you plan to release the game across multiple mobile devicesUnity is also a good choice for VR development, although VR is a very smallmarket at the moment. The mobile and PSVR markets are the largest in VR, and Unity is already well positioned to port games to many platforms such as PS4 and PC, or many different mobile markets.The engine targets the following graphics APIs: Direct3D on Windows and Xbox One; OpenGL on Linux, macOS, and Windows; OpenGL ES on Android and iOS; WebGL on the web; and proprietary APIs on the video game consoles.

Unity is notable for its ability to target games for multiple platforms. The currently supported platforms are Android, Android TV, Facebook Gameroom, Fire OS, Gear VR, Google Cardboard, Google Daydream, HTC Vive, iOS, Linux, macOS, Microsoft HoloLens, Nintendo 3DS family, PlayStation 4, Tizen, tvOS, WebGL, Wii U, Windows, Windows Phone, Windows Store, and Xbox One

3 .SYSTEM REQUIERMENTS

- The basic concept of the game that should be the look proper and feel of the game.

- Visual studio Engine should have the appropriate knowledge of this engine how it works and how things are related or interconnected with each other means prefers, script, game management, scenes UI elements, assets and the integration of elements.

- C# is the main language used in scripting in Windows Application Form (WAF) so a comprehensive knowledge of this language is mandatory that is how classes and objects are made in C# with using Windows Application Form (WAF).

- Good knowledge of user interface is preferred as the main concern of every game is to grab and stay the user to play game maximum time so we can generate lea from these potential Users.

4 .LIST OF USE CASE

Menu bar

In the Menu bar the player will get the options through which the player can start a new game, exit the game, check out the rules, regulation of it and lastly about this game.

New game

By clicking the New game user will get the user interface and run it as a .exe file. This how the user will get interaction to the game and play it.

About game

By clicking this option user will be informed by information about the game and its

versions

Rules

On clicking the option of Rules it will show the rules of the game that how user can play it.

Jump

On pressing space bar the plane will jump across the barriers which are trees and hurdles.

Barrier cross

There will be two barriers i.e. trees and hurdles the User have to jump across these barriers.

Score

User will display about the score and increment of it as user will pass the barriers.

Exit

The exit button will exit the application.

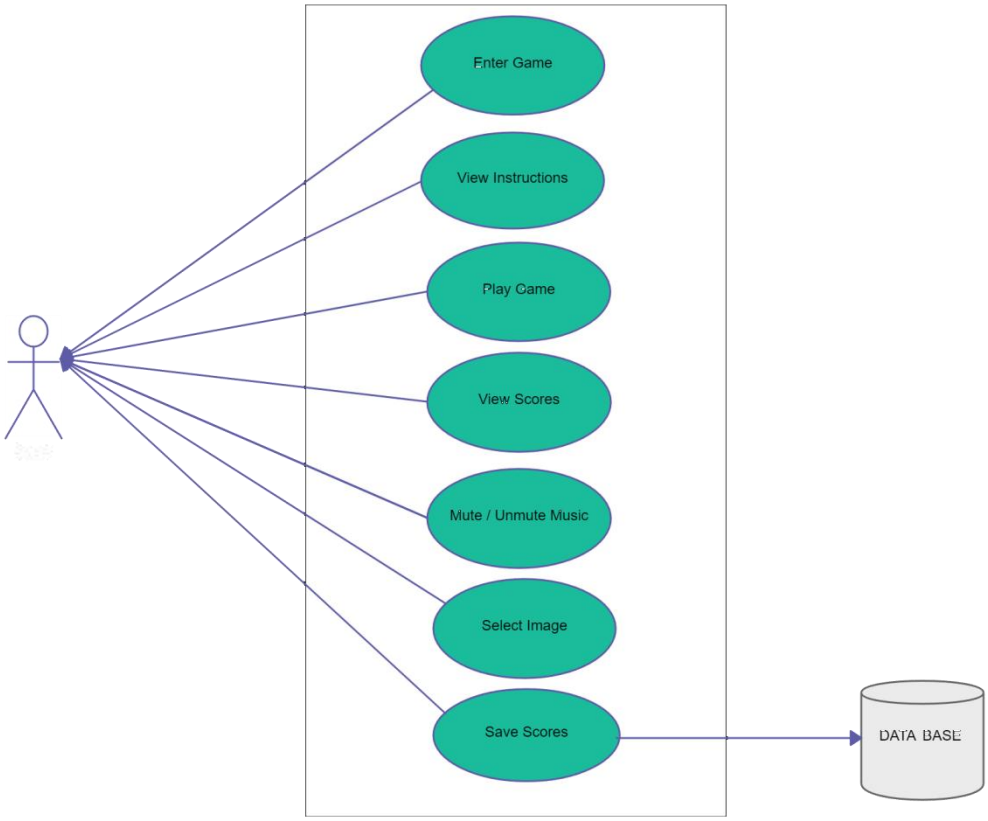

Figure 1: Use Case

5. LOGIC CONTROLLER

We implement the game logic by using C programming language. The game logic controller should realize the functions which are indicated below: updating the location of the plane from the keyboard, implementing the game rule (whether the game is over or not, computing how many pillars the plane has passed), generating the appropriate audio in terms of the game rule, and controlling the generation of sprites.

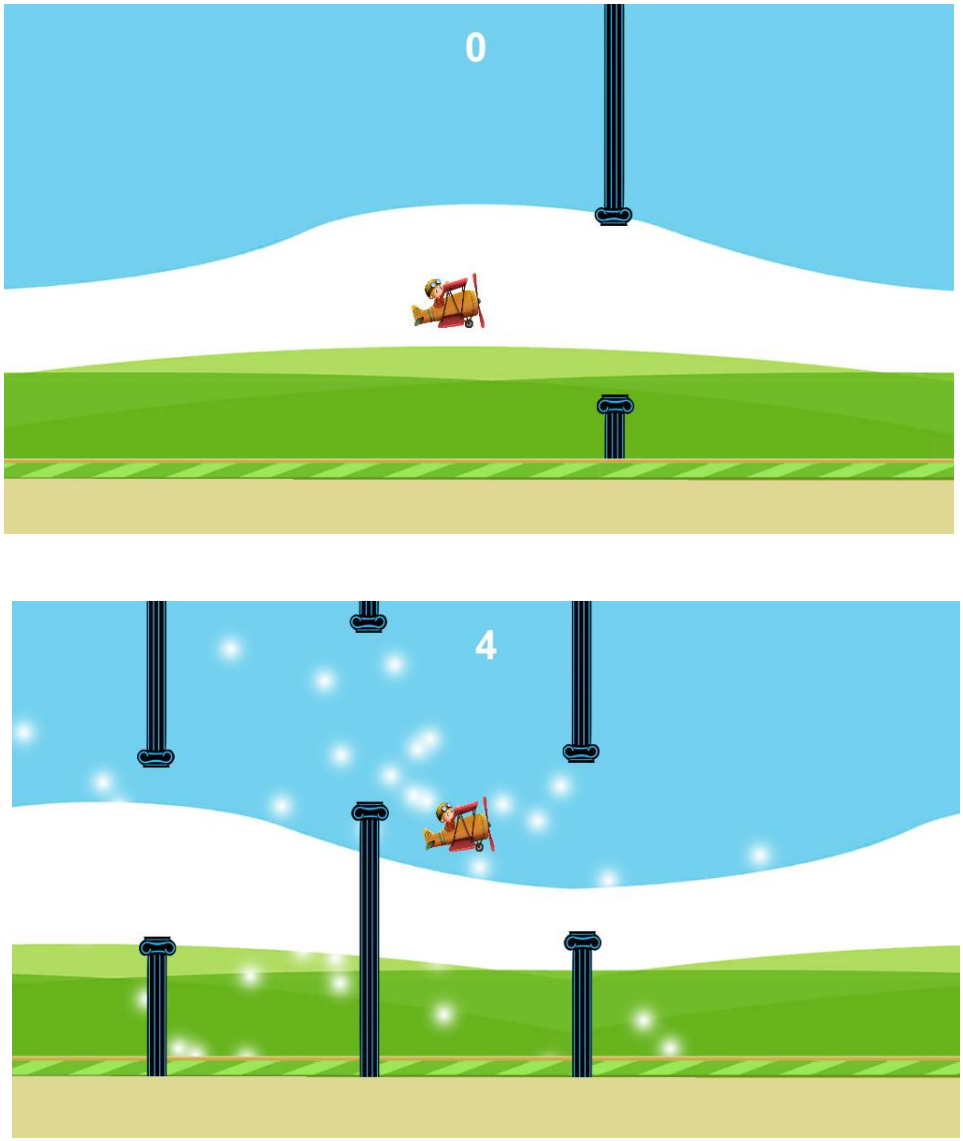

Figure 2: Game View

1. **Character Control:**

- The player controls a plane character.
- The character is in a constant state of falling due to gravity.
- Player input (typically a tap on the screen) causes the plane to flap its wings, providing a vertical lift and countering gravity.

2. **Obstacle Generation:**

- The game generates a series of vertically arranged pipes, which act as obstacles.
- These pipes have gaps between them, and the plane must pass through these gaps to progress.

3. **Collision Detection:**

- Continuous collision detection is used to check if the plane collides with the pipes or the ground.
- A collision with any pipe or the ground results in the end of the game.

4. **Scoring:**

- The player earns points for successfully passing through the gaps between pipes.
- The score increases with each successful passage.
- The player's current score is displayed on the screen during gameplay.

5. **Gravity and Physics:**

- The plane is subject to gravity, causing it to fall when not flapping.
- The game employs physics to determine the plane's velocity and acceleration based on user input and gravity.

6. **Game Over:**

- The game ends when the plane collides with a pipe or the ground.
- A game over screen is displayed, showing the player's score and providing the option to restart the game.

7. **Level Progression:**

- The game may progressively increase the difficulty by making the gaps between pipes narrower or increasing the scrolling speed.
- This encourages players to continually improve their skills.

9. **User Interface (UI)**:

- The game includes UI elements like the main menu, score display, and game over screen.
- The UI may also have buttons for starting a new game or accessing options.

10. **Persistence:**

- The game may store high scores and progress data to provide a sense of achievement and competition among players.

The simplicity of the game logic, combined with the challenging gameplay, is a key factor in falling plane addictive nature. Players are drawn to the game by its easy-to-understand mechanics, and they are motivated to keep playing to beat their previous high scores. This straightforward yet compelling game logic contributed to the game's viral success during its brief time in the mobile gaming spotlight.

6.1 PLANE AND GRAVITY

The graphics we follow is the standard system where the screen moves in the X-Direction and the user will have to trigger the up- down movement by the press of the key.The plane's Y position moves down by a certain speed each frame. Each frame this speed is increased to simulate gravity.When a key is pressed, the plane's speed is set to a high number in the upward direction.The pipes share the same width, and the same space height between the segments. Each pipe has its own X position, and Y position the space.

Figure 3: Plane(Main Character)

Since there are only two pipes in the playing area at one time, the information for only two pipes needs to stored. Once a pipe goes out of the playing area, it gets a new space position and its X position is reset to the right edge of the playing area. Digital trophies are given to the player. Also the background is a png file that is attached and it keeps on re- looping as the plane progresses throughout the game. Various other facades are there in the game that give a complete look of the dynamism of the game.

6.2 PIPES

Pipes are the most important part of the game because they have to be made sensitive to collision with the plane.Their appearance in the game is totally randomized thereby making it exceedingly difficult for the player to cope as the game proceeds

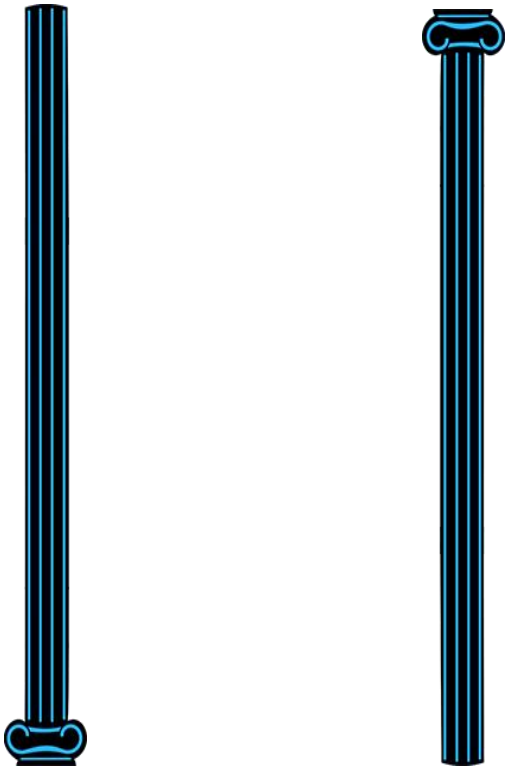

Figure 4: Pipes

The pipes in Falling Plane are a fundamental and iconic gameplay element, contributing significantly to the game's challenge and addictive nature. Here are some details about the pipes in Falling Plane

Design: The pipes consist of two vertically aligned, rectangular columns. They are typically colored green, and their design is intentionally simplistic and retro in appearance. This minimalist design adds to the game's overall aesthetic.

Obstacle Placement: Pipes are placed in the game world in a vertical arrangement, creating a series of ascending and descending challenges for the player-controlled plane character. The objective is to navigate the plane through the gaps between the pipes.

Collision Detection: Contact with any part of the pipes or the ground results in an immediate game over condition. The collision detection is unforgiving, making precise navigation essential for success.

Scrolling: The pipes scroll from right to left across the screen, continuously creating new obstacles. The scrolling speed and frequency of pipe appearances may vary, progressively increasing the game's difficulty.

Gaps: The most critical aspect of the pipes is the gaps they create. These gaps are the spaces between the two columns of pipes, and the player must guide the plane through these gaps without making contact with the pipes.

Scoring Mechanism: Successfully passing through a gap between the pipes earns the player points. The score increases with each successful passage, motivating players to aim for higher scores and compete with themselves or others.

Level Progression: In some versions of Falling Plane ," the game may increase the difficulty over time by making the gaps between the pipes narrower or by adjusting the scrolling speed.

Visual Minimalism: The design of the pipes, along with their stark, uncluttered background, ensures that players can concentrate on the central gameplay without distractions. The game's visual minimalism is a key part of its design.

Symbolism: The pipes have become a symbol of Falling Plane itself. They represent the game's addictive and frustrating nature, and many players instantly recognize them as an emblem of the game.

The pipes in Falling Plane encapsulate the game's essence, embodying the balance between simplicity and challenge that contributed to the game's rapid rise to popularity. These abstract obstacles became an iconic feature of the mobile gaming world, symbolizing the addictive and maddeningly difficult nature of the game.

6.3 COLLISION WITH PIPES

- The left edge of the plane is to the left of the right edge of the pipe The
- right edge of the plane is to the right of the left edge of the pipe
- The top edge of the plane is above the bottom edge of the pipe segment.

6.4 BACKGROUND

The background provides a simple yet essential visual setting for the gameplay. The background is characterized by its minimalistic design, featuring a light blue sky with a few scattered white clouds. This straightforward background design is intentionally unobtrusive, allowing players to focus on the core gameplay elements—the flapping plane and the pipes.The light blue sky creates a serene and uncluttered environment that contrasts with the challenging nature of the game. The absence of complex background scenery or distractions contributes to the game's overall simplicity and helps maintain the player's concentration on navigating the plane through the narrow gaps between the green pipes.

The unassuming background of FALLING PLANE underscores the game's core design philosophy, which emphasizes accessibility, simplicity, and a focus on gameplay mechanics. While the background lacks elaborate details, it complements the game's addictive and challenging nature, making it a perfect canvas for players to test their timing and reflexes as they guide the falling plane through the obstacle course of pipes. the background serves as a crucial yet understated element of the game's overall aesthetic and gameplay experience.

The minimalist background serves several key purposes within the game:

Visual Focus: By keeping the background unobtrusive and devoid of distractions, the player's attention is directed entirely towards the central elements of the game: the plane and the pipes. This design choice reinforces the game's core mechanics, making it clear that the primary objective is to navigate the plane through the challenging obstacles presented by the pipes.

Enhanced Challenge: The simplicity of the background heightens the perceived difficulty of the game. With nothing to provide visual context or cues, players are left with a stark, unrelenting experience in which the pipes and the ground become the sole, unforgiving adversaries. This design choice amplifies the sense of challenge and the addictive nature of the gameplay.

Accessibility: The minimalist background contributes to the game's accessibility. Players of all ages and backgrounds can easily understand and engage with the game, as there are no complex visual elements to interpret or navigate. This simplicity helps Falling Plane transcend language and cultural barriers.

Distinctive Style: The game's distinctive visual style, characterized by its clean lines, stark contrast, and minimal use of color, became an iconic feature of Falling Plane This visual simplicity contributed to the game's recognizability and its cultural impact.

Gameplay Clarity: Backgrounds should not overpower the foreground elements (such as characters and interactive objects). They need to strike a balance between visual appeal and gameplay clarity. This means making sure that background elements don't distract from the core gameplay.

In summary, the background in Falling Plane plays a pivotal role in shaping the game's overall experience. Its minimalism and lack of visual distractions focus the player's attention on the core gameplay elements, intensifying the game's challenge and addictive quality. Falling Plane is a prime example of how a well-considered background can significantly contribute to a game's identity and impact, making it an iconic title in the world of mobile gaming.

Figure 5: Background

6.5 GROUND

The ground is a critical component of the game environment, contributing significantly to the gameplay and challenge. The ground in falling plane represents the horizontal boundary at the lowermost part of the game screen. It is a constant presence throughout the game and is a key obstacle that players must navigate to keep their avian protagonist afloat. The central objective of the game is to guide the plane through a series of vertically aligned, green pipes that extend from the top of the screen to the ground.

The ground's significance in falling plane lies in the consequences of making contact with it. If the player allows the plane to touch the ground, the game immediately ends, resulting in a "Game Over" scenario. This is akin to the primary risk of colliding with the game's primary obstacle, the pipes. As such, the ground serves as a critical boundary that the player must deftly avoid, often demanding precision timing and skillful taps on the screen to keep the plane in the air.The presence of the ground adds an additional layer of complexity to the game's already challenging mechanics. Players must not only maneuver the plane through the narrow gaps between the vertically positioned pipes but also maintain the plane's altitude to prevent contact with the ground.

The ground, along with the pipes, encapsulates the essence offalling plane. Their synergy creates a sense of relentless, yet straightforward, challenge that captivated players worldwide. The game's minimalist design, combined with its punishing difficulty, led to a widespread phenomenon in the mobile gaming world, driving players to repeatedly attempt to surpass their high scores. the ground infalling plane plays a pivotal role in defining the game's core mechanics and difficulty. It represents the fine balance between success and failure, driving players to test their reflexes and patience as they navigate the plane through its perilous journey. The ground, in its simplicity, became a symbol of the addictive and frustrating nature of falling plane and contributed significantly to its rise to

fame and subsequent impact on the world of mobile gaming.In 2D games, the ground serves as a fundamental element of the game environment and plays several key roles in shaping gameplay and overall player experience. Here's a summary of the significance and functions of the ground in 2D games:

It act as a physical barrier the ground defines the lower boundary of the game world, acting as a physical barrier that characters, objects, and obstacles cannot pass through. It provides structure to the game environment, preventing entities from falling infinitely or moving beyond the established play area.

Figure 6: Ground

7.USER CONTROL

KEY	ACTION
X(in-game)	Jump
Space bar(on pause screen)	Restart

8.TECHNICAL GAME DETAILS

PARAMETER	VALUE
Frames per second	60 fps
Resolution	1920 × 1080
Pipe gap	200 px
Gravity	30px per second

9.FUTURISTICS OF FALLING PLANE

Multiplayer and Social Features: Future iterations of Falling Plane could introduce multiplayer modes, allowing players to compete in real-time or collaborate. Social features, such as sharing scores and challenges on social media, could also be enhanced.

Customization and Personalization: Future versions of Falling Plane could allow players to customize their plane character, pipes, or background. Personalization features might include skins, themes, and in-game purchases.

Cross-Platform Play: Enabling players to enjoy Falling Plane on multiple platforms and devices while maintaining a consistent gaming experience could be a futuristic feature.

SOURCE CODE:

BACKGROUND

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
      public class Background : MonoBehaviour
{
    public GameObject quad;
    private Renderer rend;
    public float speed;
        void Start(){
        rend = quad.GetComponent<Renderer>();
    }
          void Update(){
    Vector2 offset = new Vector2(Time.time*speed,0);
    rend.material.mainTextureOffset=offset;
    }
}
```

PLANE

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
```

```
using UnityEngine.UI;
      public class plane : MonoBehaviour{
public Text scoreText;
public int score;
private Rigidbody2D rb;
public float jumpforce;
    void Start(){
        score=0;
        rb= GetComponent<Rigidbody2D>();
    }
void Update(){
        if(Input.GetKeyDown(KeyCode.X))
        {
            rb.velocity = new    Vector2(rb.velocity.x,jumpforce);
        }
    }
    void OnCollisionEnter2D(Collision2D other){
    if(other.gameObject.tag=="pipe")
     {
    SceneManager.LoadScene(SceneManager.GetActiveScene().name);
     }
    }
 public void UpdateScore(){
```

```
            score++;
        scoreText.text=score.ToString();
           }
        }
```

PIPES

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Pipe : MonoBehaviour{
void Start(){

    }
    void Update(){
       transform.Translate(Vector2.left *1*      Time.deltaTime);
    }
}
```

PIPES GENERATOR

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PipeGenerator : MonoBehaviour{
```

```
        public GameObject pipe;
    public Vector3 pos;
    void Start(){
        StartCoroutine(GeneratePipes());
    }
    void Update(){


    }
IEnumerator GeneratePipes(){
while(true){
yield return new WaitForSeconds(4f);
   float randy = Random.Range(0.15f,3.5f);
pos = new Vector3(1f,randy,0f);
Instantiate(pipe,pos,Quaternion.identity);
      }
    }
}
```

SCORE DETECTOR

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;


public class ScoreDetect : MonoBehaviour
```

```
{
    private plane plane;
        void Start()
    {
        plane = GameObject.Find("plane").GetComponent<plane>();
    }

        void Update(){

    }

    void OnTriggerEnter2D(Collider2D other){
        if(other.gameObject.tag=="Player")
        {
            plane.UpdateScore();
        }
    }
}
```

CONCLUSION

The game logic was deployed successfully and the gameplay was also smooth. The game is suitable for all age groups and its availability on Desktop , there is still some scope to add more features like creating multiple logins or making multi-player compatible game but that is beyond the scope of the script that we have been using.

Using C#, we learned a lot about concept of Graphics as used in Computers. Our course curriculum helped us explore more about the project that we were dwelling into and hence give us a wider perspective of how graphics and simple coding gives us a user experience so good.This report is based on the knowledge, which I acquired during my FALLING PLANE project.

REFRENCES

- https://www.freepik.com/
- https://softuni.org/project-tutorials
- https://www.geeksforgeeks.org/courses
- https://www.w3schools.com/cs/cs_type_casting.php
- https://www.udemy.com/course/how-to-make-a-2d-mobile-game-in-unity-c-flappy-plane-game/
- https://www.coursera.org/learn/introduction-programming-unity?
- https://gamemaker.io/en/blog/how-to-make-a-2d-game
- https://docs.unity3d.com/Manual/Unity2D.html
- https://assetstore.unity.com/
- https://youtu.be/on9nwbZngyw?si=sK0v9-b6pLMzSHkk
- https://youtu.be/j48LtUkZRjU?si=-2qh0CPWEN54qCWX
- https://youtu.be/lgUIx75fJ_E?si=mEauBdyJZ9zORqFa