**PROJECT REPORT**

**on**

**Android-based-2D-gaming**

**(CSE IV Semester Mini project)**

**2020-2021**

****

**Submitted to: Submitted by:**

Ms. Lisa Gopal Mr. Ritesh Kumar

(CC-CSE-F-IV-Sem) Roll No.: 1918609

**Guided by:** CSE-F-IV-Sem

Mr. Avnish Panwar Session: 2020-2021

(Resource Person)

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

**GRAPHIC ERA HILL UNVERSITY, DEHRADUN**

**CERTIFICATE**

### Certified that Mr. Ritesh Kumar (Roll No.- 1918609) has developed mini project on “Android 2D gaming application” for the CS IV Semester Mini Project Lab in Graphic Era Hill University, Dehradun. The project carried out by Students is their own work as best of my knowledge.

Date:

(Ms. Lisa Gopal) (Mr. Avnish Panwar)

**Project Co-ordinator Project Guide**

**CC-CSE-F-IV-Sem** Resource Person

(CSE Department) (CSE Department)

GEHU Dehradun GEHU Dehradun

**ACKNOWLEDGMENT**

We would like to express our gratitude to The Almighty Shiva Baba, the most Beneficent and the most Merciful, for completion of project.

We wish to thank our parents for their continuing support and encouragement. We also wish to thank them for providing us with the opportunity to reach this far in our studies.

We would like to thank particularly our project Co-ordinator Ms. Lisa Gopal and our Project Guide Mr. Avnish Panwar for his patience, support and encouragement throughout the completion of this project and having faith in us.

At last but not the least we greatly indebted to all other persons who directly or indirectly helped us during this work.

**Mr. Ritesh Kumar**

**Roll No.- 1918609**

**CSE-F-IV-Sem**

**Session: 2020-2021**

**GEHU, Dehradun**

**TABLE OF CONTENTS**

1. **INTRODUCTION** 
   1. About Project
   2. C# Programming Language
2. **Project**
   1. Requirement Analysis
   2. Software Specification
   3. Modules
      1. Installation and Setup
      2. Sprites
      3. Importing Art
      4. Animation
      5. Adding Background
      6. Physics
      7. Coding
      8. Launching
      9. Adding Obstacles
      10. Indicator
      11. Camera Controller
      12. Adding Enemies
      13. Prefabs
      14. Adding Levels
   4. Application
3. **SNAPSHOT OF PROJECT**
   1. Scene view
   2. Game view
   3. Level 1 on Android
   4. Bird is about to hit
   5. Bird after crashing(level 1)
   6. Level 2
   7. One enemy left
4. **CONCLUSION**

**APPENDIX:** CODE

**REFERENCE**

1. **Introduction**

**1.1 About Project**

**Game Description**

This is an android based 2D Gaming Application. It is made with the help of platform named **Unity**(Unity Engine), which is usually used to build many type of games like 2D, 3D, 1st person(FPP), 3rd person(TPP) games.

In this I have made a game which is named Mad Birds, in this we have a bird and we have to kill out enemies over there by targeting at them. As the enemies are killed we reach our next level.

* + 1. **Project Description**

While making the game Mad Birds I used different modules to successfully make the game like, what impact the bird will have while striking an object or enemy and stimulate characteristics of a bird falling towards ground due to gravity. I had to add sprites of bird and enemies, background image, obstacles, also how it would look if enemies are dead. I had to code in C# language to make the bird move like how the bird will be dragged down and released so that it may have some force and strike different objects in the game scene. I also added different levels to the game to make it more interesting.

* 1. **C# Programming Language**

C# was developed around 2000 by Microsoft as a part of its .NET initiative.

C# is a modern, object-oriented, and type-safe programming language. C# enables developers to build many types of secure and robust application that run in the .NET ecosystem. C# is an object-oriented, component-oriented programming language.

C# can also access code written in any .NET compliant language and can also inherit the classes written in these languages.

C# is widely used for developing desktop applications, web applications, and web services. C# is also used in game development on platforms like Unity, Unreal.

**2. Project**

**2.1. Requirement Analysis**

**Hardware Requirement**

Processor: 1.2 GHz

Ram: 2Gb

Storage: 200 Mb

**2.2. Software Requirement**

Android Version: 5.0

* 1. **Modules of Project**
     1. **Installation and setup**

Firstly, we have to install Unity. Then comes the project creation means to set up the platform on which we would be working. Like we would be creating a 2D game so we will set up the platform for creation of a 2D game or if we will make a 3D game then we will set the platform according to it.

**2.3.2. Sprites**

Sprites are simple 2D objects that have graphical images(called textures) on them. Unity uses Sprites by default when the engine is in 2D mode. Whenever Unity makes a new sprite, it uses texture. This texture is then applied on a fresh Game Object, and a Sprite Renderer component is attached to it. This makes our Game Object visible with our textures, as well as gives it properties related to how it looks on-screen.

**2.3.3. Importing Art**

To import art firstly, download flappy bird sprites.

After that select all bird images and drop it in our project(Hierarchy tab), then save it with an extension(.anim) and it creates an animation, and we see the bird flapping the wings.

**2.3.4. Animation**

Animations only change the visual representation of an object while Animators change the physical properties of the objects.

**2.3.5. Adding Background**

Download the background image and drop it in the project(Hierarchy tab).

Select "Order in layer" and set it to a -ve number so, that our bird must not render behind our background image and the bird will be visible.

**2.3.6. Physics**

How the bird will fall???  
We will select "Add Components" after that select Physics 2D(Rigid Body 2D) and we have gravity in our bird, when we click on the play button the bird starts falling down.

Then we would look for collision so that the bird must not continuously fall down, it must collide and stop. For that we will again select "Add Components" after that select Physics 2D(Box Collider 2D).

For collision both our ground and bird must be Rigid Body.

**2.3.7. Coding**

Click on assets tab and create a C# script and here we would be writing a code so that our bird will fly.

using- to use special methods and classes

public- access modifier

MonoBehaviour- it allows to be a component that we can attach to our objects in game

void start()- start is called before the first frame update

void update()- update is called once per frame

OnMouseDown()- it will be called every time we click on our bird

OnMouseUp()-it will be called when we releases our click

To find the directionToInitialPosition(mouse position) we have to subtract (initial position ~~-~~- current position)

After that we will be adding force to directionToInitialPosition.

For Dragging

When we are dragging our mouse around we want the object to move underneath our mouse.

OnMouseDrag()- create a vector3 new position and transform position to new position

As it's a 2D game so we will have only two axis(i.e. x axis and y axis) and z axis will be zero.

For Launching

Awake()- it will reposition the bird to it's initial position

Update()- Firstly check if the bird was launched then, a condition that velocity is less than 0.1 and set the time sitting of the bird on ground.

To load the whole scene to it's original form by an if condition(i.e. if the current position of x axis is greater than 10 || if the current position of x axis is less than 10 || if the current position of y axis is greater than 10 || if the current position of y axis is less than 10 || if time sitting is greater than 3 ).

**2.3.8. Launching**

How bird will move?

It will move when we add some force to it.

We will select bird and then we will select "Add Components" after that select Rigid Body 2D Component and set gravity to zero, and in this way our bird will not fall on it's own.

**2.3.9. Adding Obstacles**

Download a box(crate) and drag it into the project(Hierarchy tab).

We will select "Add Components" after that select Physics 2D(Box Collider 2D), Rigid Body 2D.

Line renderer- It draw laser beams between two points or objects. This will be rendered on run time using a line renderer component.

It will make a line from the bird to the starting point.

On mouseDown line renderer is set to true and on mouseUp it is set to false. Cause when we drag the mouse down we can see the line in which direction it is pointing and as it is released the line must not show up.

**2.3.10. Indicator**

Download monster sprites, after that select all monster images and drop it in your project(Hierarchy tab).

We will select "Add Components" after that select Physics 2D(Box Collider 2D), Rigid Body 2D.

We will be using indicators so that we must get to know is our bird hitting correct or if the monsters are dead we, get to know.

**2.3.11. Camera Controller**

Install it with “Cine Machine” package manager.

It is used to put the bird, monster and crates in a single scene.

**2.3.12. Enemies**

Click on assets tab and create a C# script and here we would be writing a code for enemies.

On collision whenever enemy collides with ground, bird or anything we would get an information about the collision.

If bird hits crates or enemies destroy the game object.

**2.3.13. Prefabs**

Prefabs are a special type of component that allows fully configured Game Objects to be saved in the project for reuse. These assets can then be shared between scenes, or even other projects without having to be configured again.

We have to assign cloud particle prefab to enemies so that on collision enemy should die and it can be reused.

**2.3.14. Adding Levels**

Click on assets tab and create a C# script and here we would be writing a code for levels.

To check if all of the enemies are dead there's a condition(if enemy is not dead return the enemy) and the same level will continue.

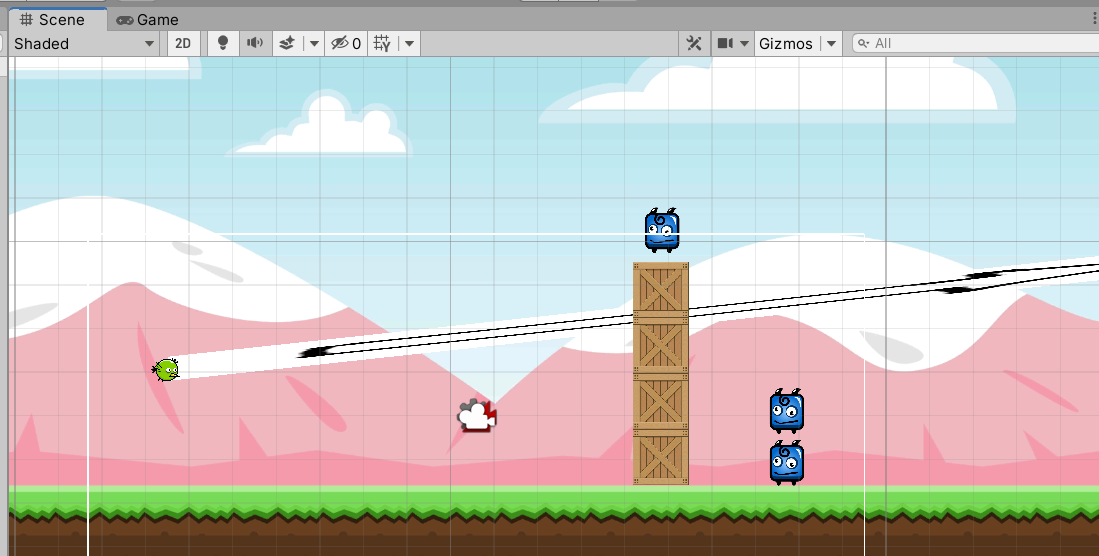
To load another level we will go to file(build settings) and we will drag level2 from our scenes to build settings and save it and we can play level2 as level1 is completed.

**2.4. Application**

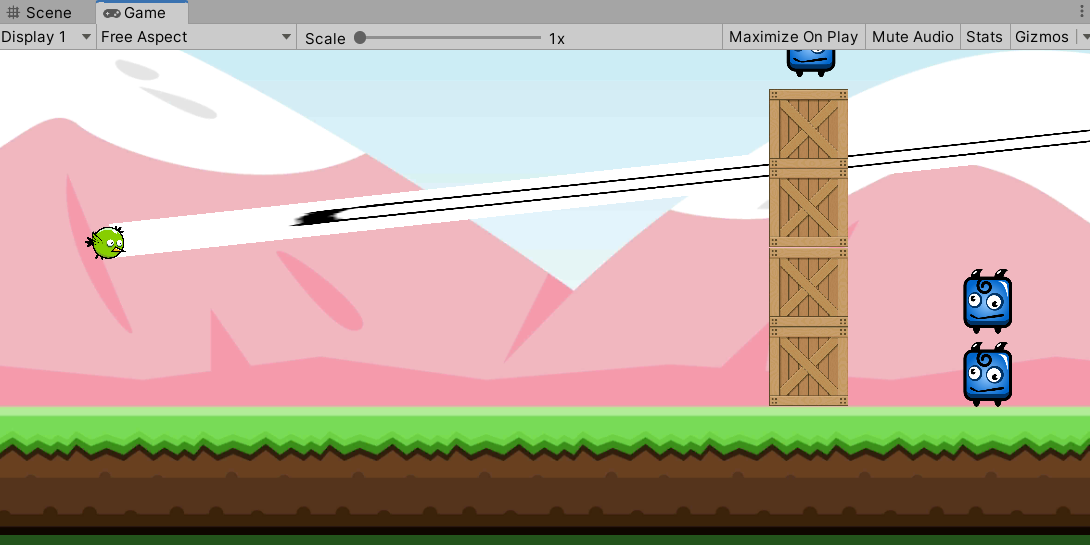
The application is based on C# programming and a platform named Unity.

1. **Snapshots of Project**

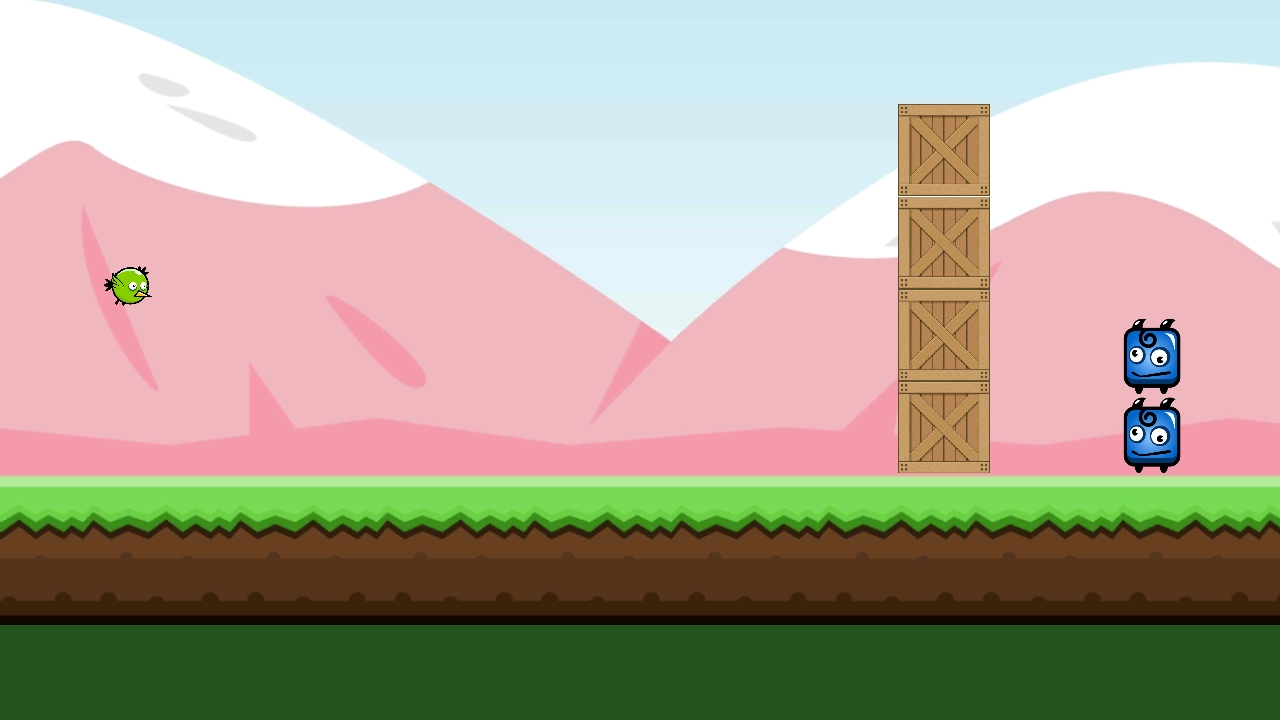
**3.1. Scene View**

****

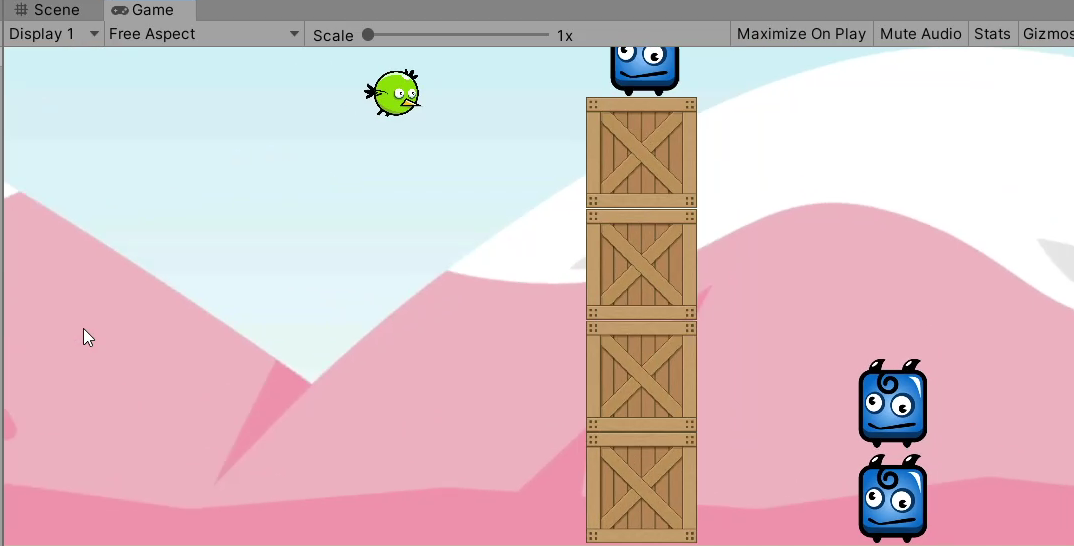
**3.2. Game View**

****

**3.3. Level 1 on Android**

****

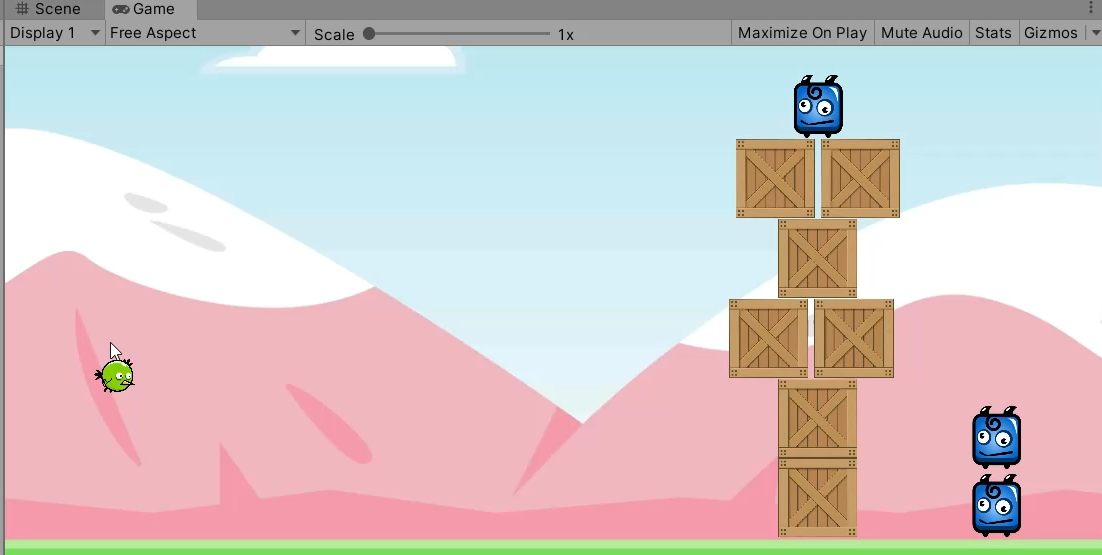
**3.4. Bird is about to hit**

****

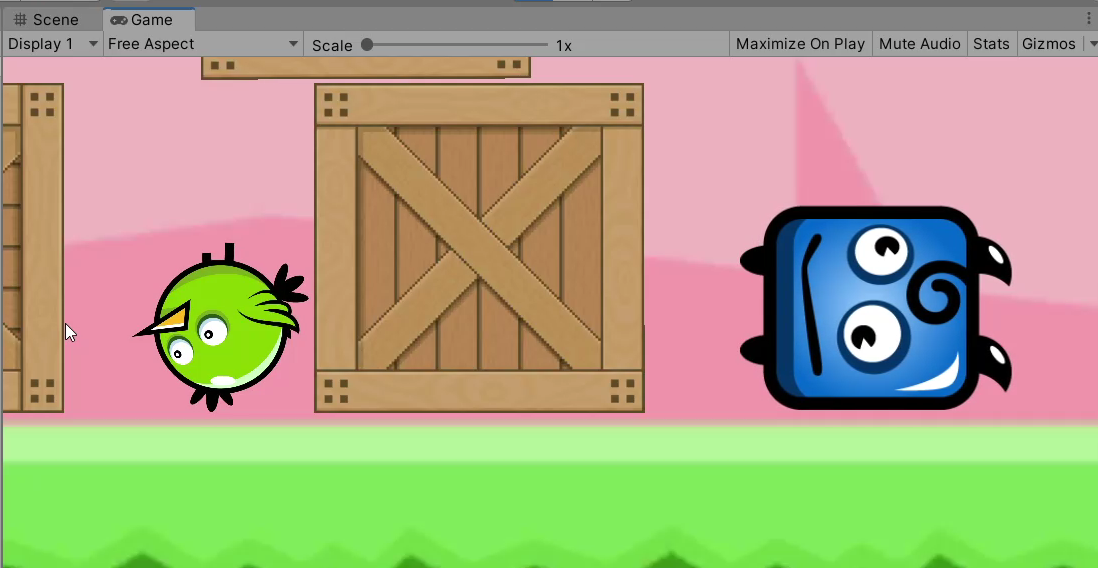
**3.5. Bird after crashing(level 1)**

****

**3.6. Level 2**

****

**3.7. One enemy left**

****

1. **Conclusion**

In the conclusion of this project, I would like to say that coding with C# language is fun and while creating a project like this, it has not just been a good experience but it also helped in the development of my creativity and logical thinking. I would be more than happy to work on other projects like this in future because it’s just amazing to work with C# language. The program is working and I hope, it’s also bug-free.

**Appendix**

**Code:-**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Bird\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Bird : MonoBehaviour

{

Vector3 \_initialPosition;

private bool \_birdWasLaunched;

private float \_timeSittingAround;

[SerializeField] private float \_launchPower=500;

private void Awake()

{

\_initialPosition = transform.position;

}

private void Update()

{

GetComponent<LineRenderer>().SetPosition(0, transform.position);

GetComponent<LineRenderer>().SetPosition(1, \_initialPosition);

if (\_birdWasLaunched &&

GetComponent<Rigidbody2D>().velocity.magnitude <= 0.1)

{

\_timeSittingAround += Time.deltaTime;

}

if (transform.position.y > 10 ||

transform.position.y < -10 ||

transform.position.x > 10 ||

transform.position.x < -10 ||

\_timeSittingAround > 3)

{

string currentSceneName = SceneManager.GetActiveScene().name;

SceneManager.LoadScene(currentSceneName);

}

}

private void OnMouseDown()

{

GetComponent<SpriteRenderer>().color = Color.red;

GetComponent<LineRenderer>().enabled = true;

}

private void OnMouseUp()

{

GetComponent<SpriteRenderer>().color = Color.white;

Vector2 directionToInitialPosition = \_initialPosition - transform.position;

GetComponent<Rigidbody2D>().AddForce(directionToInitialPosition \* \_launchPower);

GetComponent<Rigidbody2D>().gravityScale = 1;

\_birdWasLaunched = true;

GetComponent<LineRenderer>().enabled = false;

}

private void OnMouseDrag()

{

Vector3 newPosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);

transform.position = new Vector3(newPosition.x , newPosition.y);

}

}

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Enemy\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Enemy : MonoBehaviour

{

[SerializeField] private GameObject \_cloudParticlePrefab;

private void OnCollisionEnter2D(Collision2D collision)

{

Bird bird = collision.collider.GetComponent<Bird>();

if (bird!=null)

{

Instantiate(\_cloudParticlePrefab, transform.position, Quaternion.identity);

Destroy(gameObject);

return;

}

Enemy enemy = collision.collider.GetComponent<Enemy>();

if(enemy!=null)

{

return;

}

if (collision.contacts[0].normal.y< -0.5)

{

Instantiate(\_cloudParticlePrefab, transform.position, Quaternion.identity);

Destroy(gameObject);

}

}

}

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Levels\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class LevelController : MonoBehaviour

{

private static int \_nextLevelIndex = 1;

private Enemy[] \_enemies;

private void OnEnable()

{

\_enemies = FindObjectsOfType<Enemy>();

}

void Update()

{

foreach(Enemy enemy in \_enemies)

{

if (enemy != null)

return;

}

Debug.Log("You killed all enemies");

\_nextLevelIndex++;

string nextLevelName = "Level" + \_nextLevelIndex;

SceneManager.LoadScene(nextLevelName);

}

}

**Reference**

1. Google(https://www.google.com/)
2. YouTube(https://[www.youtube.com](http://www.youtube.com)/)
3. Tutor(Jason Weimann)
4. Unity(<https://www.unity.com/>)
5. Opengameart(https://opengameart.org/)
6. Javatpoint(https://www.javatpoint.com/)