***Project Report***

***On***

**Write Project Title**

**“Sleep less night”**

***Submitted to***

**Shri Ramdeobaba College of Engineering & Management, Nagpur**

**(**An Autonomous College of Rashtrasant Tukadoji Maharaj Nagpur University**)**

***for partial fulfillment of the degree in***

**Master in Computer Application**

**Second Semester, Shift I**

***Developed by***

**Mr. Vaibhav Shewale**

**Miss. Pournima Bambal**

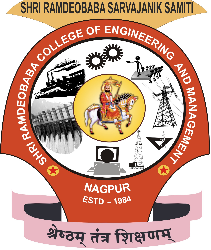
***Under the Guidance of***

**Prof. Kaushik R. Roy**

**Assistant Prof., MCA, RCOEM**

**Prof. Satyajit S. Uparkar**

**Assistant Prof., MCA, RCOEM**

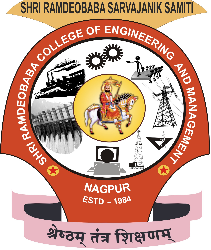


**Department of Computer Application**

**Shri Ramdeobaba College of Engineering & Management**

**Nagpur-13**

**2018-2019**

****

**CERTIFICATE**

*This is to certify that the Project Report on*

**“Sleep less Night”**

*is a bonafide work and it is submitted to*

**Shri Ramdeobaba College of Engineering & Management, Nagpur**

**(**An Autonomous College of Rashtrasant Tukadoji Maharaj Nagpur University**)**

*By*

***Mr. Vaibhav Shewale***

***Miss. Pournima Bambal***

*For partial fulfillment of the degree in*

**Master in Computer Application**,

**Second Semester, Shift I**

*during the academic year 2018-2019*

*under the guidance* *of Prof. Kaushik R. Roy and*

*Prof. Satyajit S. Uparkar.*

Name & Signature of Internal Examiner s Name & Signature of External Examiner

**Prof. Kaushik R. Roy**

**Prof. Satyajit S. Uparkar**

**Dr. P. S. Voditel**

Head, Department of Computer Application

RCOEM, Nagpur

# Acknowledgements

We wish to express our deep gratitude to our guide **Prof. Kaushik R. Roy, and**  **Prof. Satyajit S. Uparkar** Department of Computer Application, for all the advice, encouragement and constant support they has given us throughout our project work. This work would not have been possible without their support and valuable suggestions.

We are grateful to **Dr. P. S. Voditel,** Head of the Department of Computer Application for her cordial support, valuable information and guidance. Which help us in completing this task through various stages.

We are also grateful to **Dr. Rajesh S. Pande, Principal, Shri Ramdeobaba College of Engineering & Management, Nagpur** for giving us the necessary facilities to carry out our project work successfully.

We obliged to faculty members of ShriRamdeobaba College of Engineering & Management, Nagpur for the valuable information provided by them in their respective fields. We are grateful for their cooperation during the period of our project work.

**Name of the Projectees**

**Mr. Vaibhav Shewale**

**Miss. Pournima Bambal**

# Contents

**Page No.**

**Abstract** *iii*

**List of Figures** *iv*

1. **Brief Review of Project 2**
   1. Title
   2. Introduction
   3. Objectives
2. **Preliminary System Analysis 3**
   1. Present Games in the Domain
   2. Feasibility study:

1. Technical feasibility.

2. Economical feasibility.

3. Operational feasibility.

1. **Formal Game Proposal 5**
   1. Considerations for your game idea
   2. Game Prototype
2. **Software/ Hardware Requirements 7**
   1. Tools
   2. Platform
   3. Languages To Be Used
3. **Game Design 8**
   1. Sprites
4. **Game Implementation 10**
   1. Environment variable setting
   2. Sample Code
   3. Output Screens
5. **Maintenance of Game 21**
   1. System testing
   2. Post Implementation and Updation
6. **Conclusion 23**
   1. Future scope
   2. Limitation of the System & Modification

1. **Bibliography/ References** 24

1. Appendices
   1. Appendix-A : Synopsis
   2. Appendix-B: UNITY tool environment 25

# Abstract

The project is about Serving game “Sleep Less Night”. With the help of C# Script this project was created. The project also used assets, sound effects from Unity Assets Store and from Youtube. The project had been developed with the Unity version 2018 3.3f1. The project contains single levels. The character “Little Tommy” in our project has to kill zombie bunny, teddy and elephant by a gun and using obstacles for cover up, until he gets killed.

The zombie may very well be one of the most influential modern monsters. Besides the pop-cultural hype surrounding the undead, they increasingly receive the attention of scientists. The game is in endless cycle until our character finally gives up. The game is easy and very interesting to play.

*iii*

# List of figures

|  |  |  |
| --- | --- | --- |
| **Serial No.** | **Description** | **Page No.** |
| **Figure 3.1** | **Game Prototype** | **5** |
| **Figure 5.1** | **Player** | **8** |
| **Figure 5.2** | **Gun** | **8** |
| **Figure 5.3** | **Obstacle** | **9** |
| **Figure 5.4** | **Texture** | **9** |
| **Figure 6.1** | **Room** | **10** |
| **Figure 6.2** | **Pause Menu** | **19** |
| **Figure 6.3** | **Level** | **19** |
| **Figure 6.4** | **Output screen shots** | **20** |
| **Figure 6.5** | **Output screen shots** | **20** |

*v*

# CHAPTER 1

**Brief Review of Project**

* 1. Title

“Sleep Less Night”

* 1. Introduction

Nowadays, videogames are huge software projects developed by various teams. Therefore, it is no longer possible to have a single programmer coding a game.

The game being developed is a Third person player Zombie adventure Shooter Game. The player has to kill the zombifies toys, because he is been trapped in his dreams with the toys the only way to escape is killing them. The game contains single level.

The game is easy and very interesting to play. There are many zombies which are attacking to player and player has to kill them by a Gun. We hardly managed to reach finish line of the level through C# script.

1.3 Objectives

Following are the objectives of this game-

* Purpose of the game is to survive and killing the zombify toys and surviving.
* “Sleep Less Night” is a simple and addicting 3D shooter game where you play the role of a character “Little Tommy”.
* The player has to survive and kill enemies and create the high score, for the other player to set.It is simple and interesting game.

2

# CHAPTER 2

**Preliminary System Analysis**

* 1. Present Games in the Domain
* [First-person](https://en.wikipedia.org/wiki/First-person_shooter) [survival horror video game](https://en.wikipedia.org/wiki/Survival_horror_video_game) developed by [*Unity Game Engine*](https://en.wikipedia.org/wiki/Ubisoft_Montpellier) and published by Unity Tutorials*.*
* *Survival Shooter*was developed by [*Unity*](https://en.wikipedia.org/wiki/Ubisoft_Montpellier) *Developers*.
* Zombies are a popular theme for video games, particularly of, but not limited to, the
* [*stealth*](https://en.wikipedia.org/wiki/Stealth_game)
* [*survival horror*](https://en.wikipedia.org/wiki/Survival_horror)
* [*first-person shooter*](https://en.wikipedia.org/wiki/First-person_shooter)
* *Third-person surviving*
* [*role-playing game*](https://en.wikipedia.org/wiki/Role-playing_video_game)

Important horror fiction media franchises in this area include

* *Silent Hill*
* [*Resident Evil*](https://en.wikipedia.org/wiki/Resident_Evil)
* *The Walking Dead*
* [*Dead Rising*](https://en.wikipedia.org/wiki/Dead_Rising_(series))
* [*The House of the Dead*](https://en.wikipedia.org/wiki/The_House_of_the_Dead_(series))
* [*Dead Island*](https://en.wikipedia.org/wiki/Dead_Island)
* [*Left 4 Dead*](https://en.wikipedia.org/wiki/Left_4_Dead)
* [*Dying Light*](https://en.wikipedia.org/wiki/Dying_Light)
* [*State of Decay*](https://en.wikipedia.org/wiki/State_of_Decay_(video_game))
* [*The Last of Us*](https://en.wikipedia.org/wiki/The_Last_of_Us)

and the Zombies game modes from [*Call of Duty*](https://en.wikipedia.org/wiki/Call_of_Duty) title series and so on.

3

* 1. Feasibility study:

A feasibility study is performed when we want to know whether a project is possible under the given certain circumstances :

1. *Technical feasibility.*

Technical feasibility is one of the first studies that must be conducted after a project has been identified. Technical feasibility are as follows:-

* The relevant technology based on unity search engine 2018 3.3f1 is used to develop this game.
* The hardware requirement for developing “Sleep Less Night”

game is 2GB RAM, 1 Tb storage, 2 GB graphic cards.

1. *Economical feasibility.*

Economic feasibility determines whether the required software is capable of generating financial gains for an organization. It involves the cost incurred estimated cost of hardware and software. It can be assume that benefits of project is greater than the cost. So if this project can develop easily than it is used for the evaluation of the proposed game.

We have calculated the cost benefit analysis is by considering man hour spend in developing the game.

1. *Operational feasibility.*

Operational feasibility assesses the extent to which the required software performs a series of steps to solve business problems and user requirements. It determines how user friendly our game is. In operational feasibility we ensure that how easily our game is played by user, we have used high quality graphics & sound effects along with shortcut keys.

4

**CHAPTER 3**

**FORMAL GAME PROPOSAL**

* 1. Considerations for your game idea

The game being developed. The player has to survive the Level and create a high score. The game contains a single level. In that level the player has to kill the terrifying zombie that are in toys form and coming to hunt down the player, the player can complete the game by only way is by dying. Zombies are been spawned in a specific period of time which attack the player and player has to kill them by a Gun.

* 1. Game Prototype

Test the game prototype, using unity. We play the game in unity in each step. When we create a new terrain and other things firstly we check using Third Person control view how my terrain looks. In each scripting we test how the script will work and point out the problem and solves them.

Scene has one character, the Third Person player. The goal for the first level is to move our player forward.

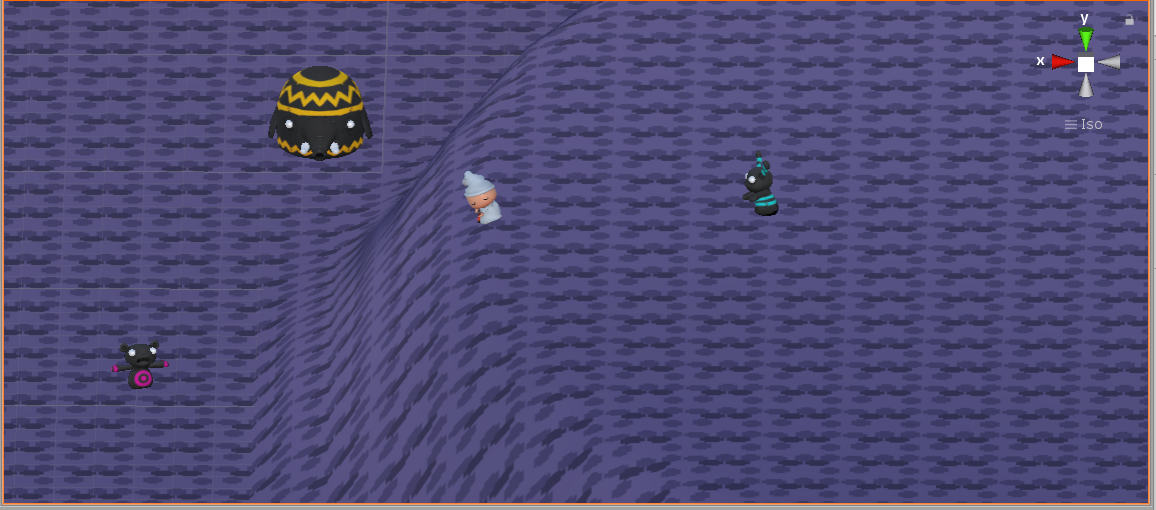
We have used arrow keys for moving and mouse button for shooting. If we release the arrow keys character will simply stand.



**Fig**

**5**

We then added some extra characters that we will use it as a zombie and tried experimenting with different kind of layouts and area.



**Fig 3.2**

6

**CHAPTER 4**

**SOFTWARE/ HARDWARE REQUIREMENTS**

* ***System Requirement :***

Win : 7/8/10 (32/64bit)

Ram : (1GB )minimum

Processor : Intel® Core(TM) i3-4160 @3.60ghz (Quad Core) 3rd gen or equivalent.

* ***Technical description of project :***

Developed with **UNITY 5**

Assests taken from **Unity Assets Store**

* + - * + Character **-** *Third person player*
        + Environment **–** *Basic Home set.*
        + Material **–** *Character skin, material skin, lights, etc.*
        + Packages **–** *sfx sound, third person controller, github.*

Sounds effects inspired from **“I am Going In”** and “**The Frankenstein**”.

**Innovativeness:** Third person view,

New camera angle.

* 1. Tools

Using Unity software tools and Microsoft Visual Studio tools for scripting.

* 1. Platform

Deploy Game for Windows, Mac operating systems.

* 1. Languages To Be Used

In our project we used two types of languages C# and Java script. For creating Pause menu of our game Java script was used. C# script was used for character, obstacle, camera, finish line.

7

**CHAPTER 5**

**GAME DESIGN**

* 1. Sprites

A sprite is a bitmap graphic that is designed to be part of a larger scene. It can either be a static image or an animated graphic. Examples of sprites include objects in 2D video games, [icons](https://techterms.com/definition/icon) that are part of an application [user interface](https://techterms.com/definition/user_interface), and small images published on websites. Graphic artists created small 2D images that were used to represent characters and other objects. Developers referenced these sprites in the [sourcecode](https://techterms.com/definition/sourcecode) and assigned properties such as when the sprites were displayed and how they interacted with other sprites.

We have used 4 sprites in our game:

1. ***Player***



1. ***Gun***



**8**

1. ***Obstacle (Zombies)***



1. Texture



9

**CHAPTER 6**

**GAME IMPLEMENTATION**

* 1. Environment variable setting

**1) Layout:**

For creation of terrain we were first going to made our own layout and design but we faced different kind of difficulties and compiler error, so we have gone with the pre made layout with prefabs of archticture.



**Fig : Room**

**2) Sky**:

To create sky we imported skybox freebie assets from Unity Assets store. At first it gave nice look to the game during creation, but during gameplay character was unable to look up so it was waste because of heighted wall.

10

* 1. Sample Code

**Pause Menu Script :**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PauseMenu : MonoBehaviour

{

public static bool GameIsPaused = false;

public GameObject pauseMenuUI;

// Update is called once per frame

void Update()

{

if (Input.GetKeyDown(KeyCode.Escape))

{

if(GameIsPaused)

{

Resume();

}else

{

Pause();

}

}

}

public void Resume()

{

pauseMenuUI.SetActive(false);

Time.timeScale = 1f;

GameIsPaused = false;

}

void Pause ()

{

pauseMenuUI.SetActive(true);

Time.timeScale = 0f;

GameIsPaused = true;

}

public void QuitGame()

{

Debug.Log("Quiting the game...");

Application.Quit();

}

}

11

**Enemy Attack Script :**

using UnityEngine;

using System.Collections;

public class EnemyAttack : MonoBehaviour

{

public float timeBetweenAttacks = 0.5f;

public int attackDamage = 10;

Animator anim;

GameObject player;

PlayerHealth playerHealth;

EnemyHealth enemyHealth;

bool playerInRange;

float timer;

void Awake ()

{

player = GameObject.FindGameObjectWithTag ("Player");

playerHealth = player.GetComponent <PlayerHealth> ();

enemyHealth = GetComponent<EnemyHealth>();

anim = GetComponent <Animator> ();

}

void OnTriggerEnter (Collider other)

{

if(other.gameObject == player)

{

playerInRange = true;

}

}

void OnTriggerExit (Collider other)

{

if(other.gameObject == player)

{

playerInRange = false;

}

}

void Update ()

{

timer += Time.deltaTime;

if(timer >= timeBetweenAttacks && playerInRange && enemyHealth.currentHealth > 0)

{

Attack ();

}

if(playerHealth.currentHealth <= 0)

{

anim.SetTrigger ("PlayerDead");

}

}

void Attack ()

{

12

timer = 0f;

if(playerHealth.currentHealth > 0)

{

playerHealth.TakeDamage (attackDamage);

}

}

}

**For Enemy Movement :**

using UnityEngine;

using System.Collections;

public class EnemyMovement : MonoBehaviour

{

// This element is not "public" because the enemies are instantiated during gameplay, so the reference to the player needs to be established at runtime.

Transform player;

PlayerHealth playerHealth;

EnemyHealth enemyHealth;

UnityEngine.AI.NavMeshAgent nav;

void Awake ()

{

player = GameObject.FindGameObjectWithTag("Player").transform;

playerHealth = player.GetComponent <PlayerHealth> ();

enemyHealth = GetComponent <EnemyHealth> ();

nav = GetComponent <UnityEngine.AI.NavMeshAgent>();

}

void Update ()

{

// Update() is being used here because the enemies are being controlled using NavMesh and not physics.

if(enemyHealth.currentHealth > 0 && playerHealth.currentHealth > 0)

{

nav.SetDestination (player.position);

}

else

{

nav.enabled = false;

}

}

}

**For Player Movement:**

using UnityEngine;

public class PlayerMovement : MonoBehaviour

{

public float speed = 6.0f;

int floorMask;

float cameraRayLength = 100.0f;

Vector3 movement;

13

Animator anim;

Rigidbody playerRigidBody;

void FixedUpdate()

{

// GetAxisRaw returns a value of -1, 0 or 1. There is no acceleration or "smoothing" when using this; the action is immediate.

float h = Input.GetAxisRaw("Horizontal");

float v = Input.GetAxisRaw("Vertical");

Move(h, v);

Turning();

Animating(h, v);

}

void Animating(float h, float v)

{

// "walking" is true if either h or v do not equal 0. "v" and "h" are set based on the Input.GetAxis value.

bool walking = h != 0.0f || v != 0.0f;

anim.SetBool("IsWalking", walking);

}

}

**For Player Shoot :**

using UnityEngine;

public class PlayerShooting : MonoBehaviour

{

public int damagePerShot = 20;

public float timeBetweenBullets = 0.15f;

public float range = 100f;

float timer;

Ray shootRay;

RaycastHit shootHit;

int shootableMask;

ParticleSystem gunParticles;

LineRenderer gunLine;

AudioSource gunAudio;

Light gunLight;

float effectsDisplayTime = 0.2f;

void Awake ()

{

shootableMask = LayerMask.GetMask ("Shootable");

gunParticles = GetComponent<ParticleSystem> ();

gunLine = GetComponent <LineRenderer> ();

gunAudio = GetComponent<AudioSource> ();

gunLight = GetComponent<Light> ();

}

void Update ()

{

timer += Time.deltaTime;

if(Input.GetButton ("Fire1") && timer >= timeBetweenBullets && Time.timeScale != 0)

14

{

Shoot ();

}

if(timer >= timeBetweenBullets \* effectsDisplayTime)

{

DisableEffects ();

}

}

public void DisableEffects ()

{

gunLine.enabled = false;

gunLight.enabled = false;

}

void Shoot ()

{

timer = 0f;

gunAudio.Play ();

gunLight.enabled = true;

gunParticles.Stop ();

gunParticles.Play ();

gunLine.enabled = true;

gunLine.SetPosition (0, transform.position);

shootRay.origin = transform.position;

shootRay.direction = transform.forward;

if(Physics.Raycast (shootRay, out shootHit, range, shootableMask))

{

EnemyHealth enemyHealth = shootHit.collider.GetComponent <EnemyHealth> ();

if(enemyHealth != null)

{

enemyHealth.TakeDamage (damagePerShot, shootHit.point);

}

gunLine.SetPosition (1, shootHit.point);

}

else

{

gunLine.SetPosition (1, shootRay.origin + shootRay.direction \* range);

}

}

}

15

**For player Health:**

using UnityEngine;

using UnityEngine.UI;

using System.Collections;

using UnityEngine.SceneManagement;

public class PlayerHealth : MonoBehaviour

{

public int startingHealth = 100;

public int currentHealth;

public Slider healthSlider;

public Image damageImage;

public AudioClip deathClip;

public float flashSpeed = 5f;

public Color flashColour = new Color(1f, 0f, 0f, 0.1f);

Animator anim;

AudioSource playerAudio;

PlayerMovement playerMovement;

PlayerShooting playerShooting;

bool isDead;

bool damaged;

void Awake ()

{

anim = GetComponent <Animator> ();

playerAudio = GetComponent <AudioSource> ();

playerMovement = GetComponent <PlayerMovement> ();

playerShooting = GetComponentInChildren <PlayerShooting> ();

currentHealth = startingHealth;

}

void Update ()

{

if(damaged)

{

damageImage.color = flashColour;

}

else

{

damageImage.color = Color.Lerp (damageImage.color, Color.clear, flashSpeed \* Time.deltaTime);

}

damaged = false;

}

public void TakeDamage (int amount)

{

damaged = true;

currentHealth -= amount;

healthSlider.value = currentHealth;

playerAudio.Play ();

16

if(currentHealth <= 0 && !isDead)

{

Death ();

}

}

void Death ()

{

isDead = true;

playerShooting.DisableEffects ();

anim.SetTrigger ("Die");

playerAudio.clip = deathClip;

playerAudio.Play ();

playerMovement.enabled = false;

playerShooting.enabled = false;

}

public void RestartLevel ()

{

SceneManager.LoadScene (0);

}

}

**For enemy Management:**

using UnityEngine;

public class EnemyManager : MonoBehaviour

{

public PlayerHealth playerHealth;

public GameObject enemy;

public float spawnTime = 3f;

public Transform[] spawnPoints;

void Start ()

{

InvokeRepeating ("Spawn", spawnTime, spawnTime);

}

void Spawn ()

{

if(playerHealth.currentHealth <= 0f)

{

return;

}

int spawnPointIndex = Random.Range (0, spawnPoints.Length);

Instantiate (enemy, spawnPoints[spawnPointIndex].position, spawnPoints[spawnPointIndex].rotation);

}

}

17

**For Score maintenance:**

using UnityEngine;

using UnityEngine.UI;

using System.Collections;

public class ScoreManager : MonoBehaviour

{

public static int score;

Text text;

void Awake ()

{

text = GetComponent <Text> ();

score = 0;

}

void Update ()

{

text.text = "Score: " + score;

}

}

**For Game Over:**

using UnityEngine;

public class GameOverManager : MonoBehaviour

{

public PlayerHealth playerHealth;

Animator anim;

void Awake()

{

anim = GetComponent<Animator>();

}

void Update()

{

if (playerHealth.currentHealth <= 0)

{

anim.SetTrigger("GameOver");

}

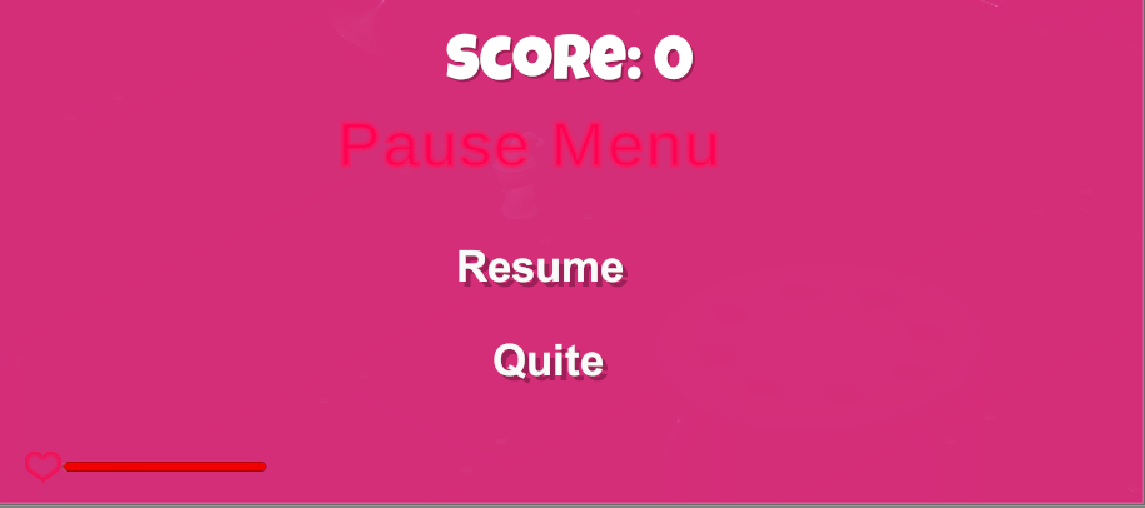
}

}

18

* 1. Output Screens

**Pause Menu**



**Fig**

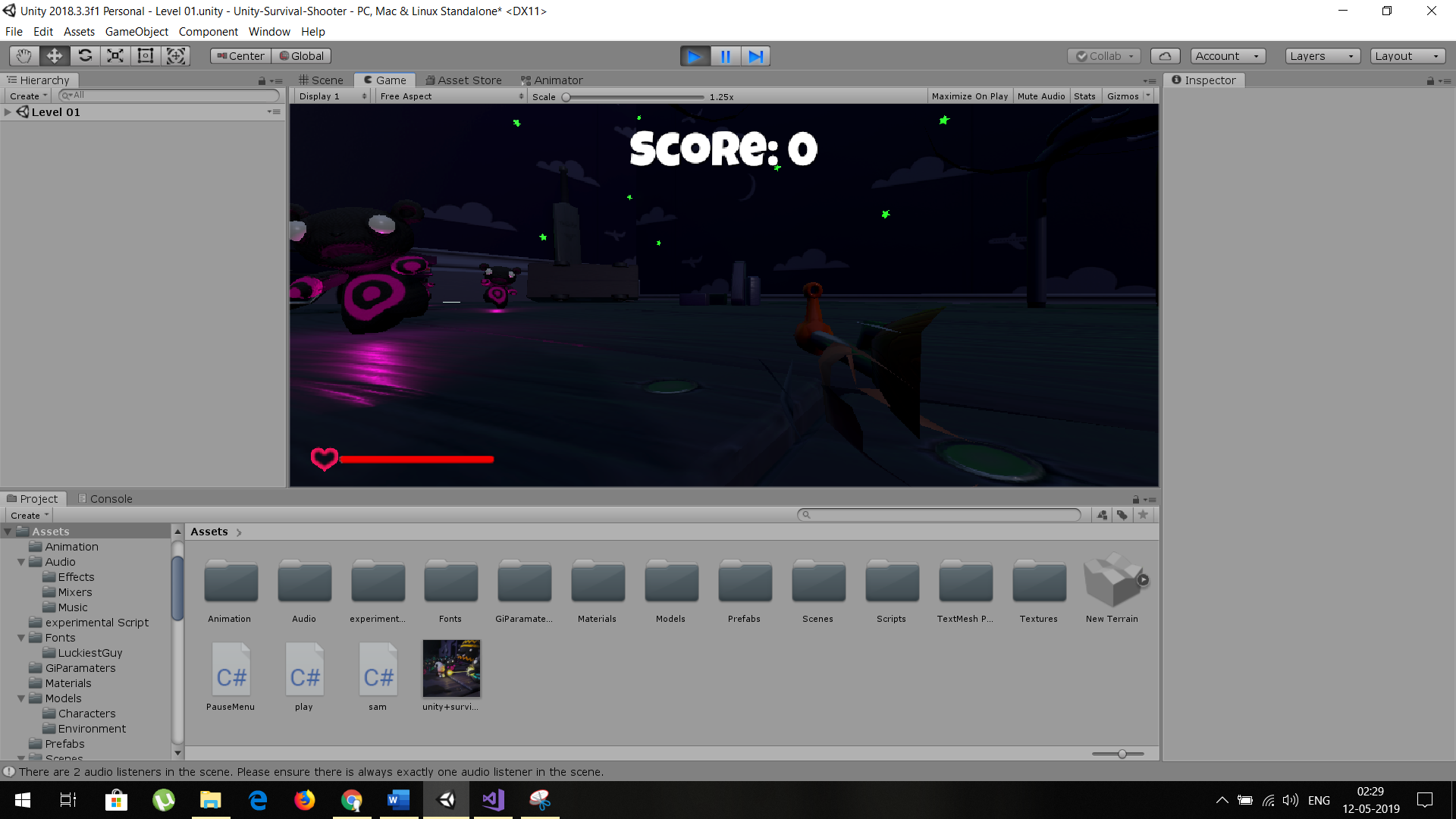
**Level**



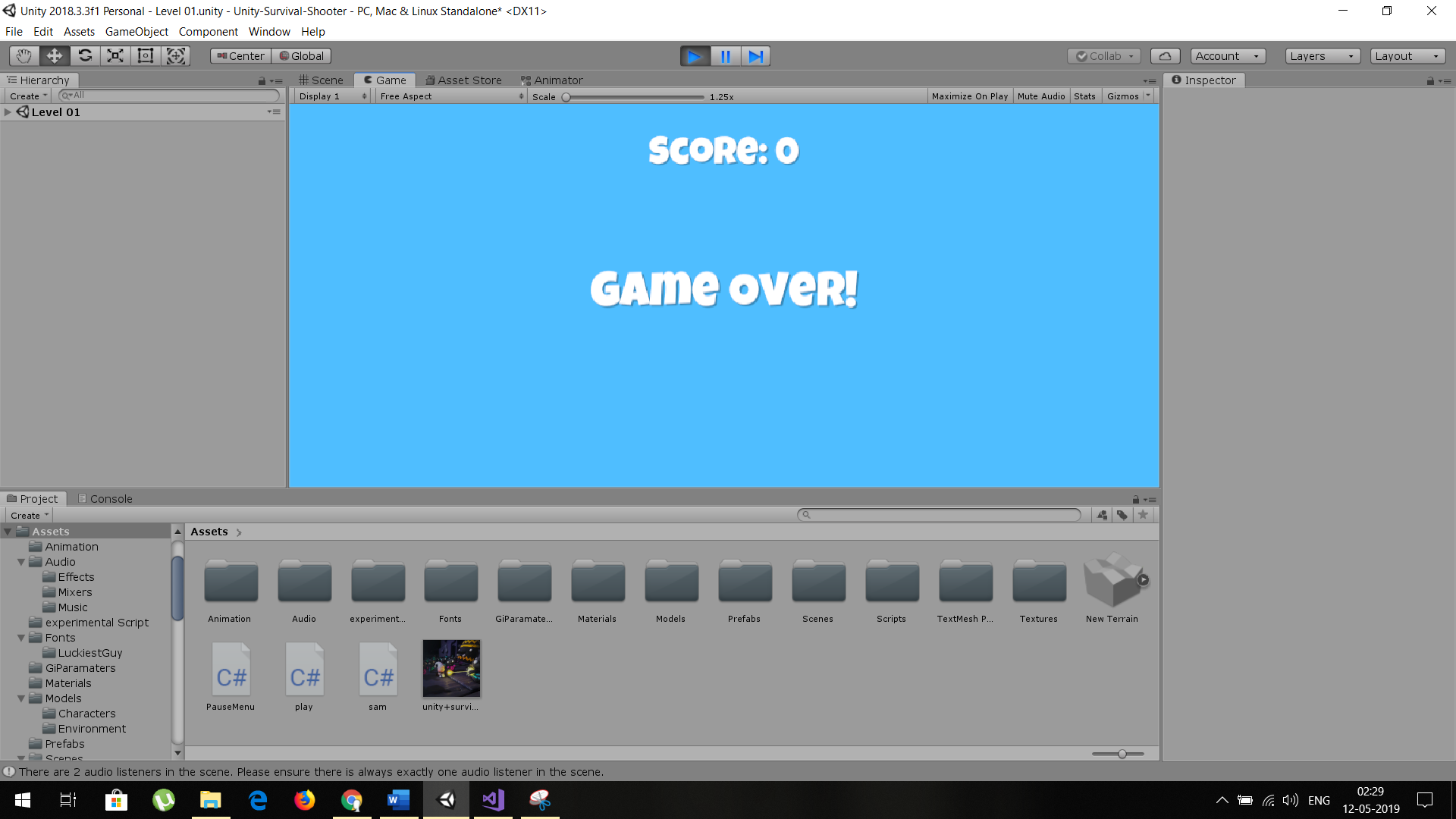
**Fig**

19

**Output Screen :**



**Fig**



**Fig**

20

**CHAPTER 7**

**MAINTENANCE OF GAME**

In this game we can maintain the overall data of regarded to the games and performing the tests find out the difficulties and bugs create in our game. And finds out the actual problem regarded to the scripts, asset, prefab or any other things and find out the solution on them.

* 1. System testing

In this game we can perform certain test and check the overall performance of the game. We can also test the scripts with the help of debugger, we also check the all animations and all assets tools of our game and tests the assets and animations are properly working or not, if not the finds out the problem and solve them.

The audio source files are also checked and test all sources and working or not. We can also test the unused assets, scripts; temporary and unimportant files are available in project and remove them to reduce the size of game and increasing the performance of the game.

Due to the increasing complexity of modern engineering systems, availability is more and more emphasized when systems are used. In order to develop availability sound products, this paper presents a supporting method to evaluate the reliability and maintainability of the feasible solution in the early design phase.

* 1. Post Implementation and Updation

This game, we implement in windows and Linux operating system. In future, we are planning to upgrade the game more, further and want to implement it on Android devices as well.

After Deployment, we are still doing Upgradation and changes to game

like

Changing the user control

Adding a new Level

Keeping Track of the High Score

Particle Effects to the character or enemy

Adding a side story for the character

Introducing new Enemies

And many more…

**21**

The evaluation is implemented based on some scaling factors that are determined according to the differences of the involved engineering activities between the feasible solution and an existing system.

22

**CHAPTER 8**

**CONCLUSION**

In this game, we create a “Sleep less night” game and implement the various types of functionalities in our game.

* 1. Future scope

The future scope of this game is very high because the everyone’s loves for playing the games.

And our game is related to the Shooting and hence the new generation people especially boys are loves shooting or to killing the obstacles game. In future we can include the various types of functionality in our game increase the visible quality of our game. And we give the update of our games also. So the future of our game is good.

* 1. Limitation of the System & Modification

**The limitation of our game is**

1. It is simply a endless loop game.
2. We can do only kill the obstacles (like zombies) in this game.
3. In present state of our game, the modification of Stage and other things this kind of functionality is not be included.

**Modification in our game is**

Change the animation of obstacles in level because they can follow the player at finish the game. Hence the utilization of CPU unusually increases.

* Modify the way zombie chase the player.
* Modified sound effects.
* Added extra Walls to other side.
* Changed the layout of the environment.
* Modification in times script also.

23

**CHAPTER 9**

**BIBLIOGRAPHY/ REFERENCES**

**References**

**Book:** Unity in action – Joseph Hocking

**Links:**

<https://www.youtube.com/watch?v=JivuXdrIHK0>

<https://youtu.be/DOFUueI_d_I>

<https://youtu.be/A42jEolCGdg>

<https://youtu.be/yqfrREmWvC0>

<https://youtu.be/rhRuaFT55wk>

<https://www.assetstore.unity3d.com>

<https://unity3d.com/>

<https://unity3d.com/learn/tutorials>

24

**CHAPTER 10**

Appendices

* 1. Appendix-A : Synopsis

**SHRI RAMDEOBABA COLLEGE OF ENGINEERING AND MANAGEMENT**

**DEPARTMENT OF COMPUTER APPLICATION**

Session 2017-18

**Course Code :** MCP539-1

**Course Name** : Game Programming Lab

**Team Name : *Pirates***

**Student name : Roll No :**

Vaibhav Shewale  48

Pournima Bambal 14

**Year/Semester/Shift :** 1st/Sem-II/Shift-I

**PROJECT SYNOPSIS**

**Title of Project :** Sleep Less Night

**Goal :** The game being developed is a Third person player Zombie adventure Shooter Game. The player has to kill as many zombie as they can in this game and create a High score, it is an casual zombie shooter game.

Toys become Zombies and trying to kill the player and end trying to end the Night

**Technical Details :**

* **System Requirement :**

Win : 7/8/10 (32/64bit)

Ram : (1GB )minimum

Processor : Intel® Core(TM) i3-4160 @3.60ghz (Quad Core) 3rd gen or equivalent.

* **Technical description of project :**

Developed with **UNITY 5**

Assests taken from **Unity Assets Store**

* + - * + Character **-** *Third person player*
        + Environment **–** *Basic Home set.*
        + Material **–** *Character skin, material skin, lights, etc.*
        + Packages **–** *sfx sound, third person controller, github.*

Sounds effects inspired from **“I am Going In”** and “**The Frankenstein**”.

**Innovativeness:** Third person view,

New camera angle.

**25**

**References :**

<https://www.youtube.com/user/Unity3D>

<https://www.youtube.com/watch?v=p8MzsDBI5EI>

<https://www.youtube.com/watch?v=THnivyG0Mvo>

<https://www.youtube.com/watch?v=MFQhpwc6cKE&t=73s>

<https://www.youtube.com/watch?v=gfkTfcpWqAY>

<https://www.youtube.com/watch?v=Vrld13ypX_I>

**Date of Submission :** *11/05/2019*

Vaibhav Shewake

Pournima Bambal Prof. Kaushik R. Roy

Prof. Satyajit S Uparkar

**Name & Signature of Student**  **Name & Signature of Guide**

26

* 1. Appendix-B: UNITY tool environment

Based on Practical list points

This is the phase where you have actually studied, the game programming , i.e. training sessions and implementation of study project.

1. **Introduction to game development : Objectives: , need , scope of game programming.**

**Objectives:**

* Computer Animation and Game Development graduates will have an understanding of critical and aesthetic issues in computer graphics and mixed-media.
* They will know basic aesthetic principles and concepts, and the production process.
* They will be able to effectively use technical, conceptual and critical abilities, and appropriate technology tools.
* They will be effective written and oral communicators with the ability to function as effective members of collaborative multi-disciplinary teams in the production process.
* They will be able to critically evaluate computer graphics and the mixed media.

**Need:**

Game programming requires substantial skill in [software engineering](https://en.wikipedia.org/wiki/Software_engineering) as well as specialization in one or more of the following areas, which overlap heavily to create a game: [simulation](https://en.wikipedia.org/wiki/Simulation), [computer graphics](https://en.wikipedia.org/wiki/Computer_graphics), [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence_(video_games)), [physics](https://en.wikipedia.org/wiki/Physics),

[audio programming](https://en.wikipedia.org/wiki/Audio_programming), and [input](https://en.wikipedia.org/wiki/Input/output). For massively multiplayer online games, additional areas, such as [network programming](https://en.wikipedia.org/wiki/Computer_network_programming) and [database programming](https://en.wikipedia.org/wiki/Database_programming) are often included. Though often engaged in by professional [game programmers](https://en.wikipedia.org/wiki/Game_programmer), many novices may program games as a [hobby](https://en.wikipedia.org/wiki/Hobby).

**Scope:**

We believe the scope of game development will be more pervasive, in terms of the game itself as well as the development of it. We imagine a near future where any individuals, companies or startups could or will develop their own games to embrace the gamification realm and are able to do it the DIY way, without require high cost such as those of game studios**.**

**27**

1. **Game development framework and architect :**

A game engine is a software framework designed for the creation and development of video games. Developers use them to create games for consoles, mobile devices and personal computers. Frostbite 3D Source Game Engine Rage Engine Flash Action Script OGRE Unity 3D Engine Unreal game engine Shiva Game Engine Havoc Vision CryEngine Game Maker Cocos2D

C++ Game Engine Framework & OpenGL for Graphics Objective C Apple IOS game development C#, Java Game Scripting Python, JavaScript, Lua Scripting & supporting behaviour language Html5, Php, Web based browser games Action Script Flash games 4.

1. **Unity engine : should cover 5 major parts**

* [**The Project Window**](https://docs.unity3d.com/Manual/ProjectView.html)**:**

In this view, you can access and manage the assets that belong to your project.The left panel of the browser shows the folder structure of the project as a hierarchical list. When a folder is selected from the list by clicking, its contents will be shown in the panel to the right. You can click the small triangle to expand or collapse the folder, displaying any nested folders it contains. Hold down Alt while you click to expand or collapse any nested folders recursively.

* [**The Scene View**](https://docs.unity3d.com/Manual/UsingTheSceneView.html)**:**

The Scene View is your interactive view into the world you are creating. You will use the Scene View to select and position scenery, characters, cameras, lights, and all other types of Game Object. Being able to Select, manipulate and modify objects in the Scene View are some of the first skills you must learn to begin working in Unity.

* [**The Hierarchy Window**](https://docs.unity3d.com/Manual/Hierarchy.html)**:**

The default Hierarchy window view when you open a new Unity project

The **Hierarchy**window contains a list of every GameObject (referred to in this guide as an “object”) in the current Scene. Some of these are direct instances of Asset files (like 3D models), and others are instances of Prefabs, which are custom objects that make up most of your game. As objects are added and removed in the Scene, they will appear and disappear from the Hierarchy as well.

28

* [**The Inspector Window**](https://docs.unity3d.com/Manual/UsingTheInspector.html)**:**

Projects in the Unity Editor are made up of multiple [GameObjects](https://docs.unity3d.com/Manual/GameObjects.html) that contain scripts, sounds, Meshes, and other graphical elements such as Lights. The Inspector window (sometimes referred to as “the Inspector”) displays detailed information about the currently selected GameObject, including all attached [components](https://docs.unity3d.com/Manual/Components.html) and their properties, and allows you to modify the functionality of GameObjects in your Scene. The Inspector in its default position in Unity.

* [**The Toolbar**](https://docs.unity3d.com/Manual/Toolbar.html)**:**

The Toolbar consists of seven basic controls. Each relate to different parts of the Editor.

 Transform Tools – used with the [Scene View](https://docs.unity3d.com/Manual/CustomizingYourWorkspace.html)

 Transform Gizmo Toggles – affect the [Scene View](https://docs.unity3d.com/Manual/CustomizingYourWorkspace.html) display

 Play/Pause/Step Buttons – used with the [Game View](https://docs.unity3d.com/Manual/CustomizingYourWorkspace.html)

 Cloud Button - opens the [Unity Services](https://docs.unity3d.com/Manual/UnityServices.html) Window.

 Account Drop-down - used to access your [Unity Account](https://docs.unity3d.com/Manual/UnityIAPSettingUp.html).

[Layers](https://docs.unity3d.com/Manual/Layers.html) Drop-down – controls which objects are displayed in Scene View

[Layout](https://docs.unity3d.com/Manual/CustomizingYourWorkspace.html) Drop-down – controls arrangement of all Views

* [**The Game View**](https://docs.unity3d.com/Manual/GameView.html)**:**

The Game View is rendered from the Camera(s) in your game. It is representative of your final, published game. You will need to use one or more Cameras to control what the player actually sees when they are playing your game. For more information about Cameras, please view the [Camera Component page](https://docs.unity3d.com/Manual/class-Camera.html).

1. **objects and tools: 2d,3d bodies, camera ,lights etc**

We use 2d Flappy bird game as study project. In this game various 2d bodies

used are Camera, Floor i.ePipeDestroyer and Ceiling, IntroGUI, Score, DeathGUI

and Flappy. Main camera includes Background and SpawnerObject.The SpawnerObjectGameObject has the SpawnerScript which in turn has the duty of spawning the pipe prefabs. We provide the two prefabs as parameters plus a minimum and maximum

29

time for the spawn. The Background is a Sprite GameObject which has a RandomBackgroundScript attached. We provide the two backgrounds from our spritesheet as parameters.

1. **Asset creation : Sprits and prefab**

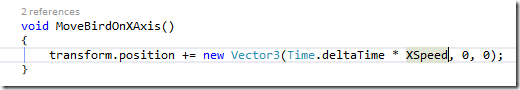
In this game Unity’s spritesheet editor to “slice” the spritesheet and getreferences to specific sprites is used. To create assets sprits or prefab of all game object spritesheet is used.

1. **Adding animation: adding physics , colliders, simulation, C# coding etc**

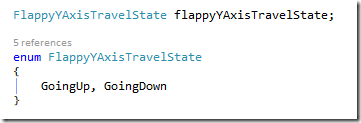
The Flappy Bird is a Rigidbody2D, has a pretty basic Circle Collider. The Ceiling (not visible during the game!) is a simple quad equipped with a BoxCollider2D component that basically prevents our bird from leaving our gameplay screen. The PipeColunnPrefab, which is identical to the PipeColumnPrefab2 with some sprite differences. The prefab is made by two sprites that are kinematic rigidbodies and a triggered BoxCollider2D. Both are tagged as “Pipe”. Plus, between the two pipes there is an empyGameObject with another BoxCollider2D, which is tagged as “pipeblank”. One can easily make out that when the bird hits the pipes, the player will lose whereas when it hits the “pipeblank” the score will increase by 1. The Floor GameObject holds 2 sprites of the floor, has a BoxCollider 2D and a FloorMove script and is tagged as “Floor”. The PipeDestroyerGameObject does the job of destroying (and thus, saving performance and memory) the pipes that our bird surpasses. The script code destroys the parent of the Pipe or Pipeblank, i.e. the entire prefab.

1. **Basic movements and player control : use of C# coding , functions, keys for movements etc.**

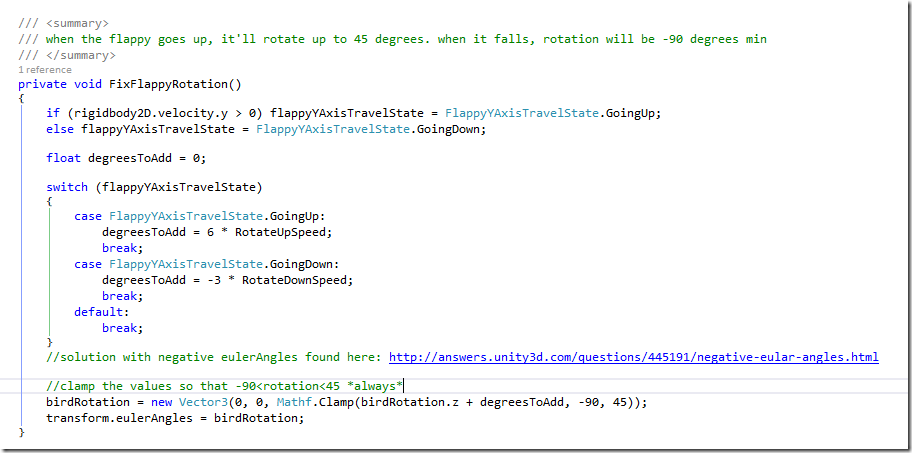
In order to move our Bird to the right, we use this method.

The bird begins to look downwards as it falls and looks up when jumping. Consequently, the bird has two states, one going u[](http://studentguru.gr/cfs-file/__key/communityserver-blogs-components-weblogfiles/00-00-00-00-89-metablogapi/image_5F00_4AFC0CD3.png)p and one going down.

30

[](http://studentguru.gr/cfs-file/__key/communityserver-blogs-components-weblogfiles/00-00-00-00-89-metablogapi/image_5F00_46195917.png)

We use the below method to make sure that the Flappy bird always has the proper rotation on the z axis.

[](http://studentguru.gr/cfs-file/__key/communityserver-blogs-components-weblogfiles/00-00-00-00-89-metablogapi/image_5F00_6151B218.png)

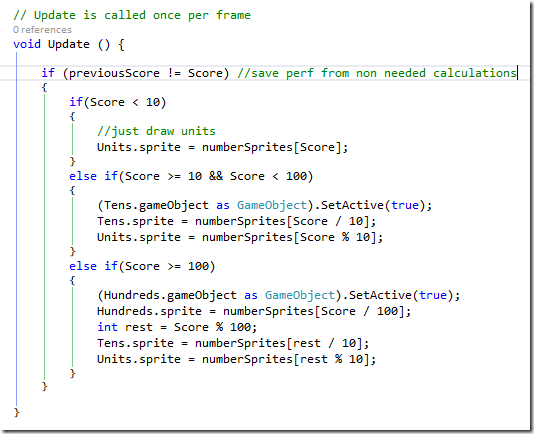
1. **Menu and interface elements : start screen , scoring , dialog boxes etc**

The IntroGUIGameObject shows the sprites of the start screen whereas the DeathGUI (which is disabled at the start of the game) shows the sprites of the death screen.

The Score is a GameObject that holds a reference to the ScoreManagerScript which contains score-handling code. The Script is provided the sprites for our numbers. It starts with a single sprite enabled (Units) and two other Sprites disabled, specifically Tens and Hundreds, which are activated when the player’s score is larger than 10 and 100, respectively. It also stores reference to our sprite digits.

31

In order to save performance, the sprite/score calculation is achieved if and only if the score has changed from the previous update call. Then, the respective sprites are enabled and the relevant score is shown.



1. **Running a project : overall integrity, executable files etc.**

The executable file of flappy bird game is made by using build and settings options in file menu of unity. For deploying the game on different OS i.e Windows, linux, MacOS, webgl, android particular option is used. The icon and reolution setting is also done.

32