

2D Game Using Unity and C#

Submitted in partial fulfillment of the requirements for the award of Bachelor of Computer Science .

By

RAJESH KUMAR.M (Reg.No.39290084)

GOKUL R KRISHNAN (Reg.No.39290025)



Bachelor of computer science

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MARCH - 2022



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**



Accredited with Grade “A” by NAAC
(Established under Section 3 of UGC Act, 1956)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **RAJESH KUMAR.M (Reg. No. 39290084)** and **GOKUL R KRISHNAN (Reg. No. 39290025)** who carried out the project entitled **“2D GAME USING UNITY AND C#”** under my supervision from November 2021 to April 2022.

Internal Guide

Dr. MAHESWARI, M.E., Ph.D.,

Head of the Department

Dr. L. Lakshmanan M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **RAJESH KUMAR (Reg. No. 39290084),Gokul Krishnan(Reg.No.39290025)** hereby declare that the Project Report entitled "**2D GAME USING UNITY C#**" done by me under the guidance of **Dr. MAHESWARI, M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Computer Science.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

Acknowledgement

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA, M.E., Ph.D.**, Dean, School of Computing and **Dr. S. VIGNESHWARI, M.E., Ph.D.**, and **Dr. L. LAKSHMANAN, M.E., Ph.D.**, Head of the Department, Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.MAHESWARI , M.E., Ph.D.**,for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computer Science and Engineering who were helpful in many ways For the Completion of the project.

Abstract

Game development is one of the most interesting field to work on gaming community and gaming developers communities around the world are collaborating to make their imagination and skill come true in the form of a game.

Playing Games is one of the most relaxing things you can do and games also gives you an opportunity to improve your logical skills which is of paramount importance.

Playing games is one of the best way to earn money by participating in various tournaments that went unaffected by the pandemic .some of the famous tournaments conducted around the world are CRL league which has cash prize of \$200000 dollars and most of the games such as CS Go and Fortnite have exicting tournaments that is conducted all year around the world.

Game development is also a great domain to take because no one will stop playing games in the world .

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	
	LIST OF FIGURES	
1	INTRODUCTION	
	1.1 OVERVIEW OF THE PROJECT	1
2	LITERATURE SURVEY	2
3	AIM AND SCOPE OF THE PRESENT INVESTIGATION	
	3.1 AIM OF THE PROJECT	4
	3.2 SOFTWARES USED	4
	3.2.1 UNITY	5
	3.2.2 C# LANGUAGE	7
	3.2.3 VS CODE	10
4	EXPERIMENTAL OR MATERIAL AND METHODS	
	4.1 DESIGN METHODOLOGY	11
	4.1.1 TRADITIONAL METHOD	11
	4.1.2 MODERN METHOD	11
	4.1.3 RULES OF THE GAME	11
	4.2 ARCHITECHURE DIAGRAM	12
	4.3 MODULE EXPLANATION	
	4.3.1 SCRIPTS	12
	4.3.2 PREFABS	16

	4.3.3 SCENES	20
	4.3.4 INPUTS	21
	4.3.5 EXPORT ENGINE	22
	4.4 PREDICTION	22
5	RESULTS AND PERFORMANCE	
	5.1 GAME SELECTION	23
	5.2 GAME EXECUTION	23
	5.3 LOW END LAPTOP TEST	25
6	CONCLUSION AND FUTURE ENHANCEMENT	
	6.1 CONCLUSION	26
	6.2 FUTURE ENHANCEMENT	26
	APPENDIX	27

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
4.1	ARCHITECHURE DIAGRAM	12
4.2	ENEMY	18
4.3	WEAPON	18
4.4	BULLET	19
5.1	GAME FOLDER	23
5.2	MAIN MENU	23
5.3	FIRST SCENE	24
5.4	WIN SCREEN	24

CHAPTER 1

1. INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

Game development is an interesting field which is relevant to this day , it is believed that this domain has the potential to be one of the most fastest growing domain in the world in fact it is estimated that nearly 3 thousand games were released in the month of January and February alone .

It is important to note that game development is also not easy, Developing a game is a intensive and a costly process which requires a lot of Programmers , voice over artists and other parts which a company has to spend in millions to develop a game and even then game is not ready you as a developer has to give the gamers a reason to play your game and games are also prone to bugs which causes the games to be refunded which makes game developing a hard way to make money, coupled with the ability to acquire cracked version of the game easily causes some companies to go bankrupt and loose their business.

2D games are tend to be misunderstood as an outdated technology and no one plays 2D Games anymore but it is not true , 2D Game is still relevant to this day best example is Among us game this game is based on a 2D architecture but it uses 3D sprites another example is Clash Royale and clash of clans which makes millions in dollars in revenue and it is still played by millions of people around the world , compared to creating a 3D game , 2D game requires only less amount of programmers and it can also be played on a computer with less specification which makes it a cheap option for the buyer who is not very rich .

CHAPTER 2

2. LITERATURE SURVEY:

1. Pac-Man Conquers Academia: Two Decades of Research Using a Classic Arcade Game Authors: Phillipp Rohlfshagen ; Jialin Liu ,14th December 2017

It was first named as pucman and for the American market sake it was renamed to Pacman, Their impact goes well Beyond the commercial world of video games and both games have featured in numerous academic research projects over the last three decades.Pac-Man caused coin shortage in japan in its first inception. They also focus on Computer intelligence.The game is visually appealing and 2-D layout allows one to easily visualise the computational logic.Creating millions of mazes to make the more challenge to player is difficult.

2.Profit Optimizing Churn Prediction for Long-Term Loyal Customers in Online Games ; Authors - Eunjo Lee ; Boram Kim , 08th October 2018

Acquiring new customers is hard and therefore this paper argues that companies must make sure that customer base stays constant and number of customers leaving the subscription plan must be less to make more profit, there are many game passes which focusses on the very idea of customer returning to their service ,to prevent churn , two method is proposed, first is setting a target and second is tuning threshold , Real world company called Aion used this proposed model and has success with it.

3. 2D PLATFORMER GAME IN UNITY; Authors - Tejas bosle, kulkarini, patanakar

In this paper, author is telling about the platformer game which they have created, it is a typical platform game which is similar to other platform games and it focusses on user's satisfaction thus this game uses a lot of animations and other thing that are required to

make the game come true and achieve its objectives, and this game also has the default features that will be expected of the platformer game.

4. “Model Driven Game Development: 2D Platform Game Prototyping”

Authors: Montero Reyno and José Á. Carsí Cubel.

This paper proposes a model-driven approach to develop a game as games are getting complex every year using various stereotypes we can create UML elements which makes the game interesting, by dividing the game parts and focussing on each part we can generate great and interesting game which makes it interesting to play

5. “The art and science of level design”, Cliff Blezinsky

This article explains the role of the “Level Designer” and how his role is crucial behind the success of the development process as well as the commercial success of the game. This person plays a crucial part of game development and it also increases the appeal of the game, design in a game is important, it deals with the imagination of a person and what he wants in game thus level designer is important and an interesting part to work on a project

CHAPTER 3

3. AIM AND SCOPE OF THE PRESENT INVESTIGATION

3.1 AIM OF THE PROJECT:

To build a 2D game that takes only less space and less graphics and processing to run and to reach wider range of audience, The created game should be fun and challenging to play and it should also have levels that players has to pass through and if defeated they are given a choice to restart the level. It should be also non age restrictive content so that players can play this game without any age restriction and have fun while playing them.

3.2 SCOPE AND OBJECTIVE:

- Game development is a huge domain with huge potential , developing a game is not an easy task and companies all around the world spend years and millions to create and make the game playable.
- It is also a developing field which grows exponentially more and more than millions of games are released each year
- But in order to run them in their pc they require high end configuration to run such games and with ongoing chip shortages and uncertainty in the pricing of the graphics cards that are required to run those games are increasingly getting in demand, but 2D game requires less memory to run and execute which makes it perfect project idea.
- It is also wrongly said that 2D games are dead and no one plays 2D games anymore but games such as Clash of Clans, Clash Royale, Among us are 2D game with amazing concept , they use 3d sprites but their architecture is 2D Model so 2D game development and games are still alive and will carter the customers who wants to play games at affordable cost.

3.3 SOFTWARE USED:

3.3.1 Unity:

Unity is a game engine that is designed to make game development easy, unity has various useful methods and in built modules that can be used to make games and it is also a beginner friendly platform , another advantage of using unity is it's developer base, Unity has a very vast developer base which is a crucial element in game development as many game developers who develop games also have doubts which is cleared by the community and it is a great way to also encourage the community to create sprites which can be a crucial part while developing games, Unity is also used by various companies such as Disney and other companies which requires unity to animate or to create games from it , unity also supports multi platform conversion so if we want we can also create a game for a mobile phone and create it .

Applications of Unity:

Architectural Visualization:

Video game engines like Unity are already capable of handling huge amounts of complex geometry and rendering realistic looking lighting and surfaces. Using a game engine like Unity to explore virtual building for engineering and architectural purposes is a no brainer. It's fairly easy to import data from Sketchup or Revit right into Unity, and then spice it up with Unity's high end graphical capabilities. Instead of just looking at concept renders and blueprints to clients, architects can use a game engine to create 3D experiences, that can be deployed to the client's cell phone, desktop or web browser.

Previsualization for Film:

This one is pretty similar to Animation, but not quite the same. Live action filmmaking requires a lot of pre-planning: in many cases, the more planning, the better. Drawing traditional storyboards can be a good starting point, but sometimes it can be useful to have a more detailed and realistic approximation of how these scenes will look. With Unity, the Asset Store, and The Cinema Suite, this becomes very possible. Like we talked about with Animation, Unity and Cinema Director (part of the Cinema Suite) can work together to make great animated sequences in Unity, so why not apply this same concept to previsualization?

Interactive Experiences:

Those touch screen devices are popping up everywhere. Traditional maps at shopping malls are being replaced by interactive touch screen maps, McDonald's has a touch screen experience for ordering food now, and large movie theatre chains let you buy tickets through a kiosk. All of these solutions need a rich multimedia platform to run on, and solutions like these aren't getting any less popular.

Unity's extensibility makes it possible to integrate into whatever existing back ends these companies already have, and Unity's powerful graphical capabilities make it easy to blow people away visually. Again, Unity's various export options make it possible to build to whatever platform the touch screen surface supports.

3.3.2 C# LANGUAGE:

C# is a object oriented programming language which was developed by the Microsoft and it runs on .NET Framework, c# is also a scripting language and it was released by Microsoft along with Visual studio code and .NET Framework and they kept it closed source for four years until they became an open source language , a person with knowledge in c++ and java will find C# Easy as it is extremely similar to them

Goals of c# Language:

- The language is intended to be a simple, modern, general-purpose, object-oriented programming language.
- The language, and implementations thereof, should provide support for software engineering principles such as strong type checking, array bounds checking, detection of attempts to use uninitialized variables, and automatic garbage collection. Software robustness, durability, and programmer productivity are important.
- The language is intended for use in developing software components suitable for deployment in distributed environments.
- Portability is very important for source code and programmers, especially those already familiar with C and C++.
- Support for internationalization is very important.
- C# is intended to be suitable for writing applications for both hosted and embedded systems, ranging from the very large that use sophisticated operating systems, down to the very small having dedicated functions.
- Although C# applications are intended to be economical with regard to memory and processing power requirements, the language was not intended to compete directly on performance and size with C or assembly language.

3.3.3 ADOBE PHOTOSHOP:

Adobe photoshop is a versatile tool that is used around the world for editing photos and also for drawing digitally, Photoshop is a beginner friendly tool that is easy to learn and use , it is not compulsory to use photoshop for game development but it is one of the best tool to create sprites for the game.

Tools used in photoshops:

Pen tool

Photoshop includes a few versions of the pen tool. The pen tool creates precise paths that can be manipulated using anchor points. The free form pen tool allows the user to draw paths freehand, and with the magnetic pen tool, the drawn path attaches closely to outlines of objects in an image, which is useful for isolating them from a background.

Clone stamp tool

The Clone Stamp tool duplicates one part of an image to another part of the same image by way of a brush. The duplication is either in full or in part depending on the mode. The user can also clone part of one layer to another layer. The Clone Stamp tool is useful for duplicating objects or removing a defect in an image.

Shape tools

Photoshop provides an array of shape tools including rectangles, rounded rectangles, ellipses, polygons and lines. These shapes can be manipulated by the pen tool, direct selection tool etc. to make vector graphics. In addition, Photoshop provides its own shapes like animals, signs and plants.

Selection tools

Selection tools are used to select all or any part of a picture to perform cut, copy, edit, or retouching operations.

Cropping

The crop tool can be used to select a particular area of an image and discard the portions outside the chosen section.

This tool assists in creating a focus point on an image and unnecessary or excess space. Cropping allows enhancement of a photo's composition while decreasing the file size.

The crop tool is in the tools palette, which is located on the right side of the document. By placing the cursor over the image, the user can drag the cursor to the desired area. Once the Enter key is pressed, the area outside the rectangle will be cropped. The area outside the rectangle is the discarded data, which allows for the file size to be decreased. The crop tool can alternatively be used to extend the canvas size by clicking and dragging outside the existing image borders.

Slicing

The slice and slice select tools, like the crop tool, are used in isolating parts of images. The slice tool can be used to divide an image into different sections, and these separate parts can be used as pieces of a web page design once HTML and CSS are applied. The slice select tool allows sliced sections of an image to be adjusted and shifted.

Moving

The move tool can be used to drag the entirety of a single layer or more if they are selected. Alternatively, once an area of an image is highlighted, the move tool can be used to manually relocate the selected piece to anywhere on the canvas.

3.6 VS CODE:

Vs code was released alongside c# language by Microsoft in their .NET project

Vs code is a code editor which is primarily used by both beginners and professionals who are into coding and learning to code, it's community is very vast and developers are continuously adding new and exciting extensions for the developer which makes it fun and future proof , vs code plays an important role in game development by giving us opportunity to edit the code and by always showing us the error we make.

CHAPTER 4 : EXPERIMENTAL OR MATERIAL AND METHODS

4.1 DESIGN METHODOLOGY

4.1.1 Traditional Method:

Traditional method involves in creating Scripts for each and every character in the game so that every gameobject is a new part but this method of approach is not efficient and it can lead to repeating the code again and again , this method also involves creating panels instead of scenes which makes games more complicated and while debugging it becomes an issue as a developer is confused by the hierarchy and has no idea what is above it this method is also leads to larger development time as it requires us to write the codes again.

4.1.2 Modern Method:

This method involves game developers to be more efficient and make sure that game is less complicated as possible , this method used to have an common script for common purpose , for example, there is a script called as enemy script which has methods such as health which will make sure that we don't have to re create health system again for the new enemies we will be adding and it will also destroy itself when it's health becomes zero without even writing any code so this method of game creation is being used in this game.

4.1.3 Rules of the Game:

In the beginning of the game we are giving the player weapons to defend himself and user can use mouse and keyboard to move around and use the weapon to kill the monsters which will be spawning around him,

so player has to kill all the monster and also do it without getting killed, player is allowed to use various weapons that monsters will be spawning and can revive his health, but once out he has to restart the level and play again.

4.2 ARCHITECHURE DIAGRAM

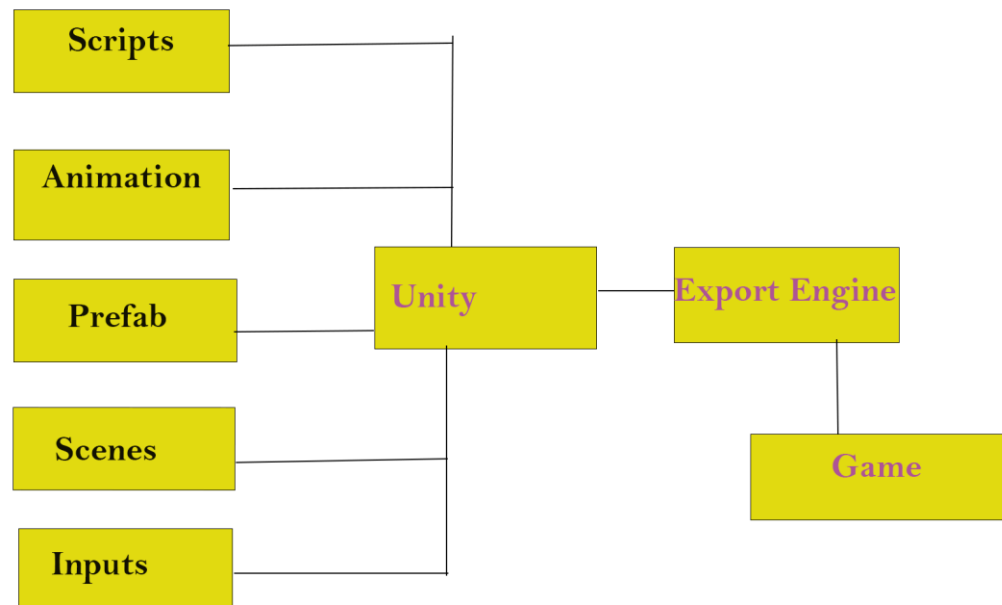


Fig 4.1 Architecture diagram

4.3 MODULE EXPLANATIONS :

4.3.1 Scripts:

Historically in the world of cinemas script writing is important and It is a skill that script writer has to master to determine the success or failure of the movie that he/she is working on .

For example:

Actor 1:“cries”, I don’t know what to do with my life

Actor 2: “looks to console”, don’t worry it will pass.

Here Script written by us as an example is used by the actors and they are asked to read it carefully and act accordingly.

In Game Development it is similar to the cinema scripting we saw before this line , in this project we are using c# as a scripting language to aid us to implement functionality, scripting plays a crucial role in game development as it controls each and every functionality of the object it is attached to it.

Script can access various methods and properties of the object to which it is attached , we can use this property of the script to modify or change properties such as it's position and other properties to suite our expectations we can also use script to access other game objects in the game to get their direction and command our enemy game object to follow the player continuously so without scripting in game development game objects will stand and do nothing while time goes .

There are two methods in scripting that will be given to us in default

- 1) Start
- 2) Update

Start:

In this method , the code that is inside this method will be executed only once and It will be executed only after the

Update:

In this method whatever code is inside it will be executed every second in the game , is it ideal if we want to maintain the health of the player and also used to send player's location continuously to the enemy so that it can follow him around.

Player script:

Player is a very important part of the game , so we start by creating variable such as speed and rigid bodies , animators , vectors ,etc , in unity engine we drop the respective objects and we are getting input from the user and we are changing the value of it's position through script which is a good technique to add the movement, since the we use raw method to get the player's input we are getting input in integer form so that accuracy is increased in the game object which makes the player's movement faster which is crucial in this genre of game, and also by using if else statement we are triggering player's animation which is the reason for it's animation movement and stop movement in the game , we are also creating a new function which aims at reducing the health of the player so that after it reaches death , scene can be transformed and user gets a reset game option and also a quit game option, we are also creating a weapon swap function to implement the weapon swap in the game. So in order to make some changes to player , enemy object will have to access the player script function and implement change for that purpose we are declaring all the major functions as public and we are also declaring variables as public so that it can be used by the developer in the unity to implement some useful changes to the game.

Enemy script:

Enemy script is a common script in this game that will help new enemies to inherit default methods and it also extends the player script which initiates a chain reaction that will grant the enemy to be able to access the important methods in player script. There are 5 types of enemies in this game and they inherit enemy script in their respective scripts to enable them to inflict damage to the player and also implement health system as a default feature to enable it to get destroyed.

Melee enemy script:

This is an unique script developed for the melee enemy which has to get close to the player and inflict damage to the player and to do that we are using various inbuilt methods to implement it , we are using a method from vector2 called move towards and it accepts three values one is current position and second is player position and third is how fast we want that enemy game object has to reach the player so we are multiplying public speed variable with time delta time function to implement the smooth and same frame rate along all the different platform the game will be running and for tweaking purposes we are using a variable called stop distance and using if and else condition to control the enemy's distance from the player ,after the else part executes it access the player health and reduces it and also produces sound while attacking.

Skeleton enemy:

This enemy is also like the melee enemy and the only difference is it explodes on impact and dies instantly but the damage done is 2 hearts so don't underestimate this low health and fast monster , only change from melee enemy is the destruction of the skeleton after attack script execution.

Ranged enemy:

This enemy is different as it requires two scripts to work , one is to control the enemy and shoot targets at the player and second is to be added to the bullet of the projectile so that is launches and goes along it's path so when the distance is satisfied enemy instantiates a bullet and that bullet follows the script and goes in a straight line and after successful collision it executes a function and health is reduced for the player

.4.3.2 Animation:

Unity engine has a dedicated module called as a Animator which is used to add animation to the character or to the enemies , we can also create an animation like screen is turning dark and use it when we are switching scenes , using Animation window we can set up triggers which will be used by our script when doing a certain process , for example , if we want our player to have a move animation we have to use the animator and create an animation however we want to see that only when the player is running and while standing we need another animation so by creating another animation we can create a constraint under which the animation only plays when we move and vice versa, so this feature of the unity is extremely useful.

Player:

For the player's animation we are having two animations , one is for when the player is in idle state and another one is for when the player is running , so we create two animation using animation window and in the animator window we are using a Boolean value as a trigger to enable and disable running animation which comes handy , so in the player script we can find a if else statement that sets the trigger and changes accordingly.

Enemy:

For the enemies we are animating like the player and we are also setting trigger in each and every enemy script to enable and disable their animation.

4.3.3 Prefabs:

It refers to game objects that has a duplicate version of itself on the game storage which makes it easier to access and use it in the game, prefabs of the enemies can be used to spawn them on the map if not we can spawn a lot of them by using the prefab so for unity prefabs are crucial elements in creating game to create a prefab we simply have to drag and drop the game object into the file system and we can find the prefab of the game created and ready to be reproduced into our game.

Player:

Player parts are sliced in the unity and are assembled in the scene window , we are also making sure that we are also changing the pivot part to the center of the body to enable it to work while we implement the rotation function because we want the weapon to rotate with respect to the center of the body to enable it to stay less confusing , we are now parenting the parts of the body and after checking the precedence level in the inspection window .

Enemy:

Enemy follows same methodology of the player and they are also parented , but for the enemies we have to change it to a prefab so that by using the spawn script we can create more enemies out of the prefab to create an army of enemies who will be ready to kill the player, it is also to be noted that health is very important aspect and also the placement of the rigid body and box collider to enable the destroy function that will eventually destroy the game object.

There are types of enemies in the game:

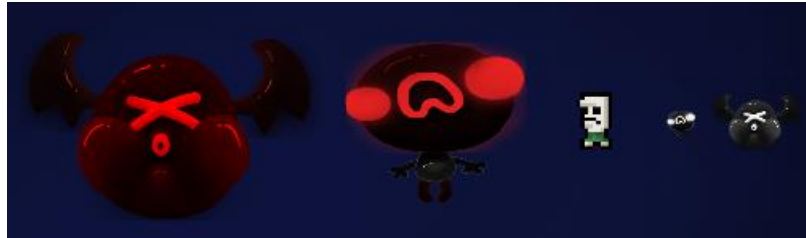


Fig 4.2 : Enemy

Big Enemy:

This category has two monsters that is upscaled and faster and stronger version of their normal variants

Mille Enemies:

This enemy aims at reducing the health of the player in a faster and annoying way , the downside is this enemy is extremely weak and can be instantly killed by any weapons.

Skeleton enemy:

This enemy is fast and can deal a lot of damage to the player but it is also weak and it is designed to catch the player off guard and take away his health

Ranged Enemy:

This angry enemy shoots bolts at the player from a distance to avoid collision and make the player irritated to the point when the player is on the verge of giving up, they spawn health pickup on their death

Weapons:



Fig 4.3: weapon

There are three types of weapons:

Sword of the Morning:

This weapon is designed to make more damage so that boss enemies are easy to kill but it is slow in shooting

Fire Head:

This weapon is designed to be a fast shooting and also less damage so that it is balanced

Eye warrior:

This weapon is the best weapon in the entire game as it is super fun to use and it also spawns the projectile faster.

Projectiles:

There are three types of projectiles in this game, they are:



Fig :4.4 Bullet

Star projectile:

This projectile is created from the eye warrior, it is medium fast and has limited range but it causes a lot of damage.

Diamond projectile:

This projectile is used in the sword of the morning and it causes a lot of damage, it can one shot any enemies except the boss variant

Fireball:

This is one of the basic bullets in the game, hence they spawn faster and also travel faster, but it causes less damage compared to other bullets in the game.

4.3.4 Scenes :

Scenes in unity is used to create various parts in the game which will help in enhancing the game experience , We can create a new scene and we can also add screen transition methods to make scenes transit which means that we can also create a level using a scene and by screen transitioning we can achieve a game that has multiple levels to be present which is a great feature that we will be adding to this game. There are 10 scenes in this game:

1) Level1:

This scene is the first level we will be creating, we are creating a green background to indicate that this is just a warm up level and worst is yet to come , this scene has totally 4 waves and player has to pass all the wave to proceed to next level.

2) Level2:

This scene is the second level we will be creating, we are creating a violet background to indicate that this is middle level and worst is yet to come , this scene has totally 5 waves and player has to pass all the wave to proceed to next level.

3) Level3:

This scene is the first level we will be creating, we are creating a blue background to indicate that this is the end level , this scene has totally 4 waves and player has to pass all the wave to end the game, so successfully surviving this level and also by killing all the enemies we can beat this game, this level is hard we are using the pickup prefabs and we are adding them to the game map so that player can select the desired weapon and complete the game.

4) Loose levels:

These scenes are executed when the player losses , and they have individual buttons to reset or to restart the level , they also have an option to go to main menu of the game.

5) Win screens:

These are the scenes that is displayed to the player after successful completion of the game , so this scene has button that will take it to next level and in case of the last win screen we can reset the game and replay the entire game again which is amazing feature to have.

6) Main menu:

This scene is important as using this scene player can start the game and continue playing it so this part of the game is important , it has a option to quit or to start the game and this scene is decorated with the rules of the game.

4.3.5 Inputs:

Inputs are crucial part in a game, it gives us an opportunity to interact with the game , unity has various input engines that is designed to make input easier and flawless, so we can access the input system in the game and we can also make unity script to be able to access the input system of the unity and implement the input methods

Player:

Player uses input and horizontal and vertical method to get the input from the user and also uses angle methods to be able to rotate the weapon by using simple techniques.

Enemy:

All enemies need is the player's location and we are giving it to the enemies by using methods and asking it to play according to the functions they are coded to, they are for example required to be closer to the player and inflict damage or by staying in a distance and inflict the damage, whatever the case, they will keep receiving the player's location which is vital for its functionality.

Enemies with special abilities such as ranged enemy requires bullet prefab so that they are spawned in regular intervals to inflict damage to the player.

4.3.6 Export Engine:

Export engine is a part of the unity which helps in extraction of the game from the unity and after export engine work on it, game can be launched and played by everyone.

Build Settings:

Build Setting in unity is used to extract the game out of the engine and actually convert it to the application which can be shared and can be played by anyone with pc, so we go to the file and click on build settings and give build option.

4.4 PREDICTION:

Our prediction is that game after building and executing will run flawlessly and be fun and simple to play and run in a low end pc.

CHAPTER:5

RESULTS AND PERFORMANCT ANALYSIS:

5.1 GAME SELECTION:

Name	Date modified	Type	Size
FlameGameConcept_Data	10-03-2022 08:49 PM	File folder	
MonoBleedingEdge	10-03-2022 08:49 PM	File folder	
FlameGameConcept	23-11-2021 07:39 AM	Application	639 KB
UnityCrashHandler64	23-11-2021 07:40 AM	Application	1,204 KB
UnityPlayer.dll	23-11-2021 07:40 AM	Application extens...	27,559 KB

Fig: 5.1 Game Folder

In the build settings we wanted the game in the folder and after building successfully game is ready to be used by the players .

5.2 GAME EXECUTION:

When we click on the FlameGameConcept game runs and gives us the main menu scene .



Fig 5.2 Main Menu

When we click on play First level starts and game will progress accordingly.

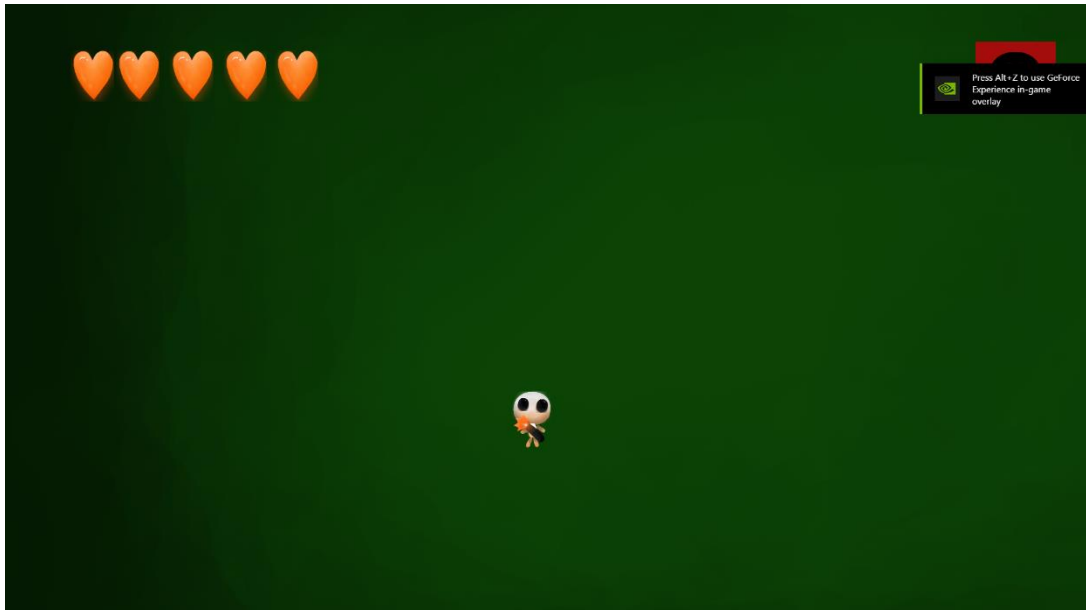


Fig 5.3 First scene

After we finish the game it shows the end screen

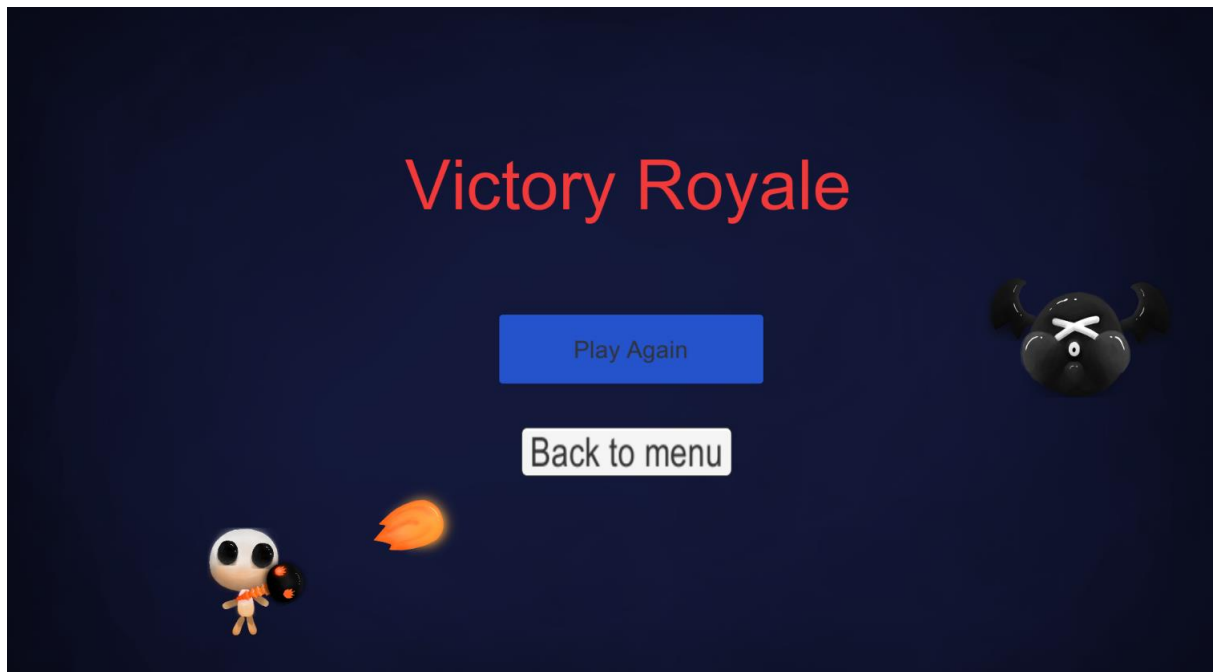


Fig 5.4 Win screen

Game also has various functions such as pause menu and resume menu and re-setting the levels also helps in the game , all the levels in the game transitions and exhibit the behaviour we wanted , enemies are strong and powerful and the game is interesting so far.

5.3 LOW END LAPTOP TEST:

Since this project is about simplicity and we also wanted this game to be played by everyone without any Hardware barrier, so for testing our game we used the following specification laptop:

- 1. Processor : Pentium IV**
- 2. RAM : 4 GB**
- 3. Processor : 2.4 GHz**
- 4. Main Memory : 4 GB RAM**
- 5. Hard Disk Drive : 256 GB**
- 6. Keyboard : 104 Keys**

Despite we picked a low end laptop for this test , the frame rate was consistent and game ran flawlessly which achieves our objectives and it is also a smooth experience and the cpu fan noise was less indicating that game is running easily by the laptop. Thus this game is free from Hardware barrier and can be enjoyed by everyone with a laptop or a pc.

CHAPTER :6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION:

We are enlightened that every script we used in this game follows DRY principle and it is also extremely fun to play the game , this game is also non age restrictive game with little to no splash effect or blood smearing part is available which makes it a good game to play, screen transition was great and game transitioned to another level after we finished the level further adding fun to the process of the game.

6.2 FUTURE ENHANCEMENT:

We would like to explore the unity more and we are willing to increase the animation and effects while playing the game which makes it interesting and we are also looking forward to create new enemies and challenges which makes the game alive and fun to play. We are also expecting to release this game which will give us valuable reviews and we will use those reviews to plan and create the game accordingly .

REFERENCES:

- [1] "Rich Hickey Q&A by Michael Fogus". Archived from the original on January 11, 2017. Retrieved January 11, 2017.
- [2] Borenszweig, Ary (June 14, 2016). "Crystal 0.18.0 released!". It's heavily inspired by Ruby, and other languages (like C#, Go and Python).
- [3] "Web Languages and VMs: Fast Code is Always in Fashion. (V8, Dart) – Google I/O 2013". YouTube. Archived from the original on December 21, 2021. Retrieved December 22, 2013.
- [4] "Unity - Download Archive". Unity. Retrieved March 3, 2022.
"Unity 2022.1b". Unity. Retrieved March 3, 2022.
- [5] "How Unity3D Became a Game-Development Beast". Retrieved August 28, 2020.
- [6] Bradshaw, Tim; Kruppa, Miles (August 12, 2020). "Epic and Unity rev their engines for the next era of entertainment". Financial Times.
- [7] Dealessandri, Marie (January 16, 2020). "What is the best game engine: is Unity right for you?". GamesIndustry.biz. Gamer Network.

- [8] Oreskovic, Alexei (September 14, 2018). "Why EA's former boss believes the 3D tech that powers video games will make way more money outside of gaming". Business Insider. Archived from the original on September 15, 2018. Retrieved October 17, 2018.
- [9] "Keep Adobe Photoshop up to date". Retrieved March 15, 2022.
- [10] "Photoshop system requirements". Adobe Inc. Retrieved November 13, 2019.
- [11] Anderson, Wade. "Evolution of Visual Studio Code Insiders". code.visualstudio.com. Microsoft. Retrieved 24 September 2020.
- [12] GitHub repository microsoft/vscode, Microsoft, 2020-12-20, retrieved 2020-12-20
- [13] "Visual Studio Code Display Language (Locale)". code.visualstudio.com. Microsoft. Retrieved 2021-03-19.

APPENDIX

SOURCE CODE:

Player script

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

public class Player : MonoBehaviour
{
    public float speed;
    private Rigidbody2D rb;
    private Animator anim;
    private Vector2 moveAmount;
    public int health;
    public Image[] hearts;
    public Sprite fullHeart;
    private SceneTransitions sceneTransition;

    public string sceneLoseTransaction;
    public Sprite emptyHeart;
    private void Start() {
        anim=GetComponent<Animator>();
        rb=GetComponent<Rigidbody2D>();
        sceneTransition=FindObjectOfType<SceneTransitions>();
    }

    private void Update() {
        Vector2 moveInput=new
Vector2(Input.GetAxisRaw("Horizontal"),Input.GetAxisRaw("Vertical"));
        moveAmount=moveInput.normalized*speed;
        if(moveInput!=Vector2.zero){
```

```

        anim.SetBool("isRunning",true);

    }
    else{
        anim.SetBool("isRunning",false);
    }

}

private void FixedUpdate() {
    rb.MovePosition(rb.position+moveAmount*Time.fixedDeltaTime);
}

public void TakeDamage(int damageAmount){
    health-=damageAmount;
    UpdateHealthUI(health);
    if(health<=0){
        Destroy(gameObject);
        sceneTransition.LoadScene(sceneLoseTransaction);
    }
}

void UpdateHealthUI(int currentHealth){
    for(int i=0;i<hearts.Length;i++){
        if(i<currentHealth){
            hearts[i].sprite=fullHeart;

        }else{
            hearts[i].sprite=emptyHeart;
        }

    }
}

public void Heal(int healAmount){
    if(health+healAmount>5){

```

```

        health=5;
    }else{
        health+=healAmount;
    }
    UpdateHealthUI(health);
}

public void ChangeWeapon(Weapon WeaponToEquip){
    Destroy(GameObject.FindGameObjectWithTag("Weapon"));
    Instantiate(WeaponToEquip,transform.position,transform.rotation,
transform);

}
}

```

Enemy script:

```

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

public class Enemy : MonoBehaviour
{
    public int health;
    public float speed;
    public float timeBetweenAttacks;
    public int damage;
    public int pickupChance;
    public GameObject[] pickups;
    //public GameObject deathEffects;

    [HideInInspector]public Transform player;
    public void TakeDamage(int damageAmount){
        health-=damageAmount;
        if(health<=0){

```

```

        int randomNumber=Random.Range(0,101);
        if(randomNumber<pickupChance){
            GameObject
randomPickup=pickups[Random.Range(0,pickups.Length)];
            Instantiate(randomPickup,transform.position,transform.rotation);
            //Instantiate(deathEffects,transform.position,Quaternion.identity);

        }
    }

}

public virtual void Start() {
    player=GameObject.FindGameObjectWithTag("Player").transform;
}

}

```

Health ui:

```

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

public class healthPickup : MonoBehaviour
{
    Player playerScript;
    public int healAmount;
    public GameObject effects;
    public GameObject sound;

    private void Start() {

        playerScript=GameObject.FindGameObjectWithTag("Player").GetComponent<Player>();
    }
}

```

```

    }
    private void OnTriggerEnter2D(Collider2D other) {
        if(other.tag=="Player"){
            playerScript.Heal(healAmount);
            Instantiate(effects,transform.position,Quaternion.identity);
            Instantiate(sound,transform.position,Quaternion.identity);
            Destroy(gameObject);

        }
    }
}

```

Menu script:

```

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

public class SceneTransitions : MonoBehaviour
{
    private Animator transictionAnim;
    private void Start() {
        transictionAnim=GetComponent<Animator>();
    }
    public void LoadScene(string sceneName){
        StartCoroutine(Transition(sceneName));

    }
    public void QuitGame(){
        Application.Quit();
        Debug.Log("QUitter");
    }
    IEnumerator Transition(string sceneName){

```

```

        transitionAnim.SetTrigger("end");
        yield return new WaitForSeconds(1);
        SceneManager.LoadScene(sceneName);

    }

}

```

Projectile :

```

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

public class Projectile : MonoBehaviour
{

    public float speed;
    public float lifeTime;
    public GameObject explosion;
    public GameObject soundFireball;
    public int damage;
    private void Start() {
        Invoke("DestroyProjectile",lifeTime);
        Instantiate(soundFireball,transform.position,transform.rotation);
    }
    private void Update() {
        transform.Translate(Vector2.up*speed*Time.deltaTime);
    }
}

```

```
void DestroyProjectile(){
    Instantiate(explosion,transform.position,Quaternion.identity);
    Destroy(gameObject);
}

private void OnTriggerEnter2D(Collider2D collision)
{
    if(collision.tag=="Enemy"){
        collision.GetComponent<Enemy>().TakeDamage(damage);
        DestroyProjectile();
    }
}
}
```