LET'S CAVORT TO LEARN

PROJECT REPORT

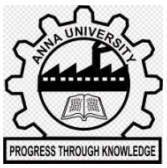
Submitted by

R.RATHINA MALA(17UITE027)
G.AISHWARYA (17UITE044)
S.RALINA BEGAM (17UITE050)

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY IN

INFORMATION TECHNOLOGY



DEPARTMENT OF INFORMATION TECHNOLOGY
KAMARAJ COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University, Chennai)

K. VELLAKULAM – 625701 (Near Virudhunagar)

APRIL 2021

KAMARAJ COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University, Chennai)

K. VELLAKULAM - 625701(Near Virudhunagar)

BONAFIDE CERTIFICATE

Certified that the project report **"Let's cavort to learn"** is the bonafide work of **"R.RATHINA MALA (17UITE027), G.AISHWARYA (17UITE044) and S.RALINA BEGAM (17UITE050)"** who carried out the project work under my supervision.

SIGNATURE

Dr. P. SUBATHRA, M.E., Ph.D.,

HEAD OF THE DEPARTMENT
Professor
Dept. of Information Technology
Kamaraj College of Engineering and Technology,
K.Vellakulam, Madurai- 625701.

SIGNATURE

Mrs.V.Deepapriya,
M.Sc.,M.Phil.,M.E.,(Ph.D),

SUPERVISOR
Assistant Professor
Dept. of Information Technology
Kamaraj College of Engineering and Technology,
K.Vellakulam, Madurai -625701.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

Game development is the overall process of creating video game. It's the art of creating games and describes the design, development and release of a game. It involves concept generation, design, build, test and release. Corona virus disease (COVID-19) is an infectious disease caused by a newly discovered corona virus. The title of our project title itself says " LET'S CAVORT TO LEARN". Cavort refers to 'play' .The main moto of our project is to develop a game which acts as a source of entertainment as well as to create awareness our minds. Playing this game helps to nurture imagination and give a child a sense of adventure. Through this they can learn essential skills. It gives both fun and gain. To make the game much more interesting the player can go on with the increase in level which makes the game much more lively. Rewards and bonuses are additionally provided to the player.

ACKNOWLEDGEMENT

First of all I thank the **Lord Almighty** for his abundant grace and countless blessings in making this work a great success.

We extend our gratitude to our Secretary **Thiru. S.P.G.C.Srimurugan** for all the facilities offered.

We also express our sincere respect and thanks to our beloved Principal **Dr. AnantAchary, M.Tech., Ph.D.,** for making us march towards the glory of success.

We are very much grateful to pay our sincere thanks to our Head of the Department, **Dr. P.Subathra, M.E., Ph.D.,** who is the source of inspiration, guidance and help.

We would like to express our sincere thanks to our project coordinators **Dr. R. Arthy, M.E., Ph.D.** and **Mr. D. Vendhan, M.E., Ph.D.,** for their valuable suggestions and motivation.

We are highly indebted to our project guide **Mrs.V.Deepapriya, M.Sc.,M.Phil.,M.E.,(Ph.D),**for her guidance and constant support for completing this project.

We would like to express our gratitude towards our parents and faculty members of Kamaraj College of Engineering and Technology for their kind Cooperation and encouragement which helped us in completion of this project.

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

GDBL	Game Development-Based Learning
GDF	Game Development Frameworks
AI	Artificial Intelligence

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

Game development is the overall process of creating video game. It's the art of creating games and describes the design, development and release of a game. It involves concept generation, design, build, test and release. While creating a game, it is important to think about the game mechanics, rewards, player engagement and level design.

If we think creating a video game is as easy as playing one, well it is not. There are many components while creating a game such as story, characters, audio, art, lightening, etc…that eventually merge together to create a whole new world in a video game. In a game project, the product is the game but here comes the point.

A game is much more than it is a software. It has to provide content to become enjoyable. Just like a webserver without content the server is useless, and the quality cannot be measured. This has an important effect on the game project as a whole, the environment of the game, the story, character, game plays, the art work**.**

1.1.1. Concept

Concept is a more detailed document than the pitch document. This includes all the information produced about the game. This includes the high concept, game's genre, gameplay description, features, setting, story, target audience, hardware platforms, estimated schedule, marketing analysis, team requirements, and risk analysis. Before an approved design is completed, a skeleton crew of programmers and artists usually begins work.

1.1.2. Game design document

Before a full-scale production can begin, the development team produces the first version of a game design document incorporating all or most of the

material from the initial pitch. The design document describes the game's concept and major gameplay elements in detail. It may also include preliminary sketches of various aspects of the game. The design document is sometimes accompanied by functional prototypes of some sections of the game.

1.1.3. Prototype

Writing prototypes of gameplay ideas and features is an important activity that allows programmers and game designers to experiment with different algorithms and usability scenarios for a game. Prototypes are often meant only to act as a proof of concept or to test ideas, by adding, modifying or removing some of the features.

1.1.4. Production

Production is the main stage of development, when assets and source code for the game are produced. Main stream production is usually defined as the period of time when the project is fully staffed. Programmers write new source code, artists develop game assets, such as, sprites or 3D models.

1.1.5. Design

Game design is an essential and collaborative process of designing the content and rules of a game, requiring artistic and technical competence as well as writing skills. Creativity and an open mind is vital for the completion of a successful video game. During development, the game designer implements and modifies the game design to reflect the current vision of the game. Features and levels are often removed or added.

1.1.6. Level creation

Designers and artists use the tools for level building, they request features and changes to the in-house tools that allow for quicker and higher quality

development. Newly introduced features may cause old levels to become obsolete, so the levels developed early on may be repeatedly developed and discarded. Because of the dynamic environment of game development, the design of early levels may also change over time.

1.1.7. Testing

At the end of the project, quality assurance plays a significant role. Testers start work once anything is playable. This may be one level or subset of the game software that can be used to any reasonable extent. Early on, testing a game occupies a relatively small amount of time. Testers may work on several games at once. As development draws to a close, a single game usually employs many testers full-time (and often with overtime). They strive to test new features and regression test existing ones. Testing is vital for modern, complex games as single changes may lead to catastrophic consequences.

1.2 INTODUCTION TO 3D GAME DEVELOPMENT

The discovery of 3D games has allowed us, enthusiast gamers, to experience video games in a whole new way. The experience of a 3D game makes the user feel that he is playing the game being a part of it, not just playing it on a TV. Such splendid experiences give rise to curiosity in users and these curiosities encourages game developers to develop games that would cater to the requirements and preferences of the users.

From smartphones to personal computers and consoles, there are various video games of varied characters and genres available in this present era for different devices. There are various websites that offer such games, both paid and free, which can be played online or can be downloaded on a gaming device, install and play.

Although being a 3D video game designer may sound like a dream come true, it is a hard job to accomplish.

Opposite to the false notion that video games are created overnight or in some weeks, it takes developers years to plan a game and develop it. Extensive tests are carried out on each development before releasing it out in the market. Such specialization areas include programming, animation, creating characters, sound effects, music etc.

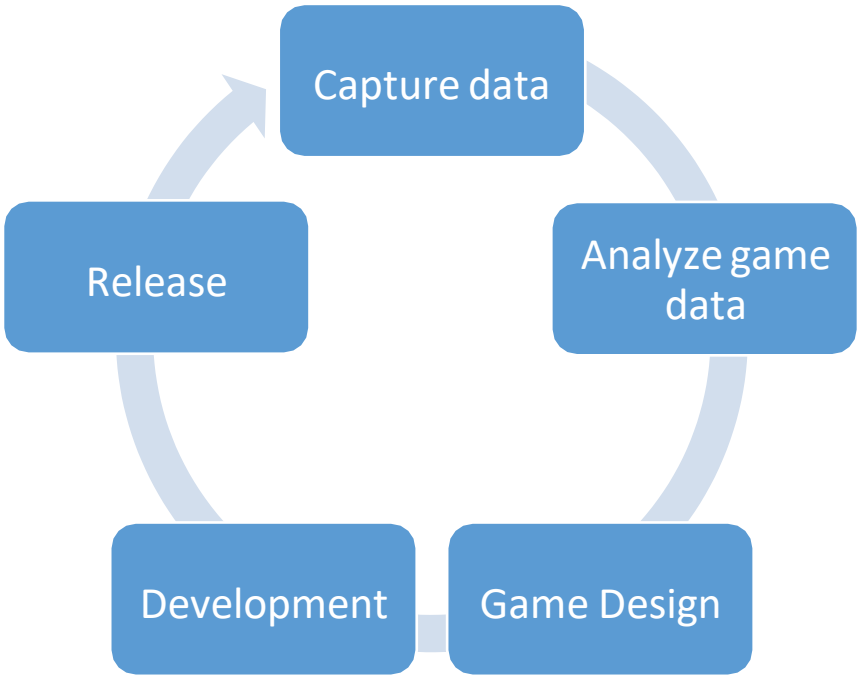


Figure 1.1: Level Creation

1.3. 2D REPRESENTATION

2D computer graphics is the computer-based generation of digital images mostly from two-dimensional models (such as 2D geometric models, text, and digital images) and by techniques specific to them.

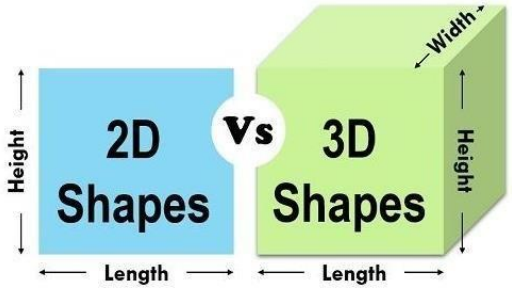


Figure 1.2:Shape Comparision

1.4 3D REPRESENTATION

In 3D computer graphics, 3D modeling is the process of developing a mathematical coordinate-based representation of any surface of an object (inanimate or living) in three dimensions via specialized software. The product is called a 3D model. Someone who works with 3D models may be referred to as a 3D artist or a 3D modeler. A 3D Model can also be displayed as a two-dimensional image through a process called 3D rendering or used in a computer simulation of physical phenomena. The 3D model can be physically created using 3D printing devices that form 2D layers of the model with three-dimensional material, one layer at a time. In terms of game development, 3D modeling is merely a stage in the entire development process. 3D modeling software is a class of 3D computer graphics software used to produce 3D models. Individual programs of this class are called **modeling applications**

Representation

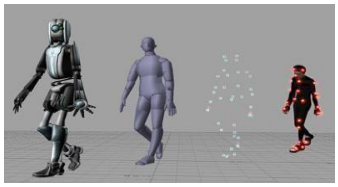


Figure 1.3: 3D Graphics

Almost all 3D models can be divided into two categories:

Solid – These models define the volume of the object they represent (like a rock). Solid models are mostly used for engineering and medical simulations, and are usually built with constructive solid geometry and Shell or boundary – These models represent the surface, i.e. the boundary of the object, not its volume.

Advantages of wireframe 3D modeling over exclusively 2D methods include:

- Flexibility, ability to change angles or animate images with quicker rendering of the changes.

- Ease of rendering, automatic calculation and rendering photorealistic effects rather than mentally visualizing or estimating.
- Accurate photorealism, less chance of human error in misplacing, overdoing, or forgetting to include a visual effect.

1.5 INTRODUCTION TO LANGUAGE

Unity:

Unity is a cross-platform game engine developed by Unity Technologies, first announced and released in June 2005 at Apple Inc.'s Worldwide Developers Conference as a Mac OS X -exclusive game engine. As of 2018, the engine had been extended to support more than 25 platforms. The engine can be used to create three-dimensional, two-dimensional, virtual reality, and augmented reality games, as well as simulations and other experiences.

Unity gives users the ability to create games and experiences in both 2D and 3D, and the engine offers a primary scripting API in C#, for both the Unity editor in the form of plugins, and games themselves, as well as drag and drop functionality. Prior to C# being the primary programming language used for the engine.

C#:

C# is a general-purpose, multi-paradigm programming language encompassing static typing, strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines. C# was developed around 2000 by Microsoft as part of its .NET initiative and later approved as an international standard by Ecma (ECMA-334) in 2002 and ISO (ISO/IEC 23270) in 2003. It was designed

by Anders Hejlsberg, and its development team is currently led by Mads Torgersen, being one of the programming languages designed for the Common Language Infrastructure (CLI). The most recent version is 9.0, which was released in 2020 in .NET 5.0 and included in Visual Studio 2019 version 16.8.Mono is a free and open-source project to develop a cross-platform compiler and runtime environment (i.e. virtual machine) for the language.C# supports strongly typed implicit variable declarations with the keyword var, and implicitly typed arrays with the keyword new[] followed by a collection initializer.

C# supports a strict Boolean data type, bool. Statements that take conditions, such as while and if, require an expression of a type that implements the true operator, such as the Boolean type. While C++ also has a Boolean type, it can be freely converted to and from integers, and expressions such as if (a) require only that a is convertible to bool, allowing a to be an int, or a pointer. C# disallows this "integer meaning true or false" approach, on the grounds that forcing programmers to use expressions that return exactly bool can prevent certain types of programming mistakes such as if (a = b) (use of assignment = instead of equality ==).

C# is more type safe than C++. The only implicit conversions by default are those that are considered safe, such as widening of integers. This is enforced at compile-time, during JIT, and, in some cases, at runtime. No implicit conversions occur between Booleans and integers, nor between enumeration members and integers (except for literal 0, which can be implicitly converted to any enumerated type). Any user-defined conversion must be explicitly marked as explicit or implicit, unlike C++ copy constructors and conversion operators, which are both implicit by default.

C# has explicit support for covariance and contra variance in generic types, unlike C++ which has some degree of support for contra variance simply through

the semantics of return types on virtual methods.Enumeration members are placed in their own scope

1.6 CONTROL SYSTEMS

Game theory is the study of interacting decision makers, whereas control systems involve the design of intelligent decision-making devices Game theory is the study of interacting decision makers i.e. settings in which the quality of an actor's decision depends on the decisions of others.

Game theory is the study of interacting decision makers, whereas control systems involve the design of intelligent decision-making devices. When many control systems are interconnected, the result can be viewed through the lens of game theory. This article discusses both long standing connections between these fields as well as new connections stemming from emerging applications.

Game theory is the study of interacting decision makers , i.e. settings in which the quality of an actor's decision depends on the decisions of others. In commuting, the congestion experienced on a road depends on a vehicle's path, as well as the paths taken by other vehicles. In auctions, the outcome depends on one's own bid as well as the bids of others. In competitive markets, market share depends on both a firm's pricing as well as the pricing of its competitors.

While game theory traditionally has been studied within the realm of mathematical social sciences, there are also strong ties to control systems .A longstanding connection is the setting of zero-sum, or minimax, games. In zero-sum games, there are two players, and a benefit to one player is a detriment to the other. A classical example is pursuit-evasion games . A common perspective in control systems is that one player is the controller and the opposing player is an adversarial environment, e.g. exogenous disturbances or model mis-specification . The controller seeks to optimize a specified performance objective, whereas the

adversarial environment seeks to reduce achieved performance. There has been renewed interest in zero-sum games in the area of security, where security measures are to be taken against a variety of adversarial attacks ranging from intrusion to data corruption to privacy violation.

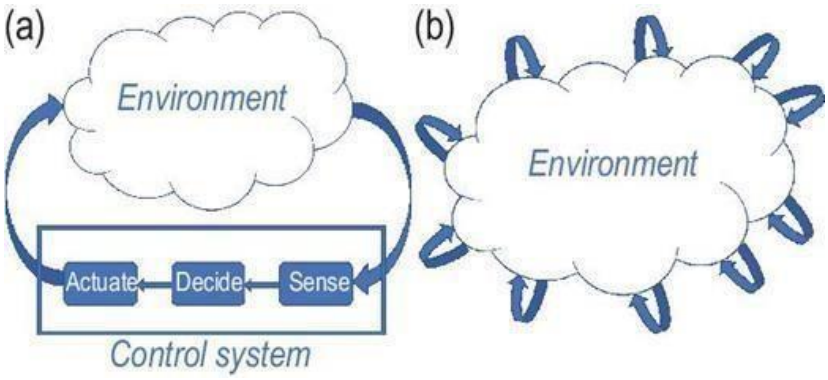


Figure 1.4:Control System

Keyboard and mouse:

The two primary, and most commonly used, devices for players to use when gaming on a computer are the mouse and the keyboard. While both are integral in the interaction of the game, their evolutionary track has not been equal.

The mouse, over the years, has had better adaptation and incorporation into gaming than the keyboard has. This could easily be attributed to the fact that the mouse is a much more simplified device. The mouse has had many advances to make it a much more adapted device for gaming. It has been upgraded from a rolling ball to an optical sensor, and the optical sensor has been upgraded to a laser. The results of these progressions have allowed players increased sensitivity and accuracy while in a game environment. The mouse has also been equipped with increasingly more buttons. Starting with two buttons, the mouse can now be found with up to seventeen buttons. Buttons have also become programmable, such that the player can perform a greater variety of actions with their mouse.

The keyboard has not seen as much advancement in terms of making it a formidable gaming device. The keyboard is mostly viewed as simply a

conglomeration of over a hundred keys that are placed and configured for typing efficiently, not for navigating a character through a virtual world. There have been some modifications made to keyboards to entice a gamer, such as adding macro buttons on the perimeter of the keyboard, or having keys that glow in the dark. Still today the shape and layout of a keyboard remains the same, optimised for word processing but not gaming.

Recently, certain companies have started to introduce mini-keyboards, or sub-keyboards, that are specifically designed to maximize the gaming experience. These are commonly referred to as gaming keypads.

Joystick:

A joystick is an input device consisting of a stick that pivots on a base and reports its angle or direction to the device it is controlling. A joystick, also known as the control column, is the principal control device in the cockpit of many civilian and military aircraft, either as a centre stick or side-stick. It often has supplementary switches to control various aspects of the aircraft's flight.

Joysticks are often used to control video games, and usually have one or more push-buttons whose state can also be read by the computer. A popular variation of the joystick used on modern video game consoles is the analog stick. Joysticks are also used for controlling machines such as cranes, trucks, underwater unmanned vehicles, wheelchairs, surveillance cameras, and zero turning radius lawn mowers. Miniature finger-operated joysticks have been adopted as input devices for smaller electronic equipment such as mobile phones.

1.7 SCOPE AND OBJECTIVE

- The main scope of our project is to create awareness among children and all age group of people.
- Game helps to simulate future imagination and gives to child a sense of adventure. Through this, they can learn essential skills.
- Awareness to prevent the spread of virus causing disease.

The objective focuses on:

- Realistic Environment
- To learn game development basics
- User friendly interface
- Viruses threaten that objective by attacking and damaging the player and bonuses exist along the way to make things more interesting.

1.8 APPLICATIONS

- Learn about new tools and best practices to support your development
- Design pattern: learn how to utilize design pattern by. making changes in an existing system
- Game design is the art of applying design and aesthetics to create a game for entertainment
- Games can be used within higher education in various ways to promote student participation, enable variation in how lectures are taught, and improve student interest.

CHAPTER 2
LITERATURE SURVEY

2.1 A Guideline for Game Development-Based Learning

This study aims at reviewing the published scientific literature on the topics of a game development-based learning (GDBL) method using game development frameworks (GDFs) with the perspective of (a) summarizing a guideline for using GDBL in a curriculum, (b) identifying relevant features of GDFs, and (c) presenting a synthesis of impact factors with empirical evidence on the educational effectiveness of the GDBL method.

They provided more cases in the diversity of GDFs methods used in teaching, which also presents the potential advantages of using GDF in education. They showed the development tendency of GDF related to other factors (e.g., times and technology).

Provide students degrees of freedom in developing their games for the platform of their choice and learn about the strengths and constraints of different platforms (e.g., user interface, viewing screen size, resolutions, resources such as memory and processor power, storage for saving/loading the game, and, etc.) in game development. Design activities are meaningful and engaging to students for exploring skills (analysis, synthesis, evaluation, revision, planning, and monitoring) and concepts to understand how they can be applied in the real world. Further, GDBL can be considered as a variant of several available construction activities.

The important issue is that the design and implementation of products are meaningful to those creating them and that learning becomes active and self-directed through the construction of artifacts. In the GDBL method, creating games with GDFs could be this artifact. This could be the fundamental concept to explain the pedagogical context of the GDBL. Special programming languages are not widely accepted and as useful as official programming languages in a long run if the students will do more software programming in the future.

2.2 Overview and Comparative Analysis of Game Engines for Desktop and Mobile Devices

Literature in the field has studied game engines focusing on specific needs, such as 3D mobile game engines or open source 3D game engines.

Nowadays, serious games as well as mobile games development are on the rise. Day by day, desktop and console games are replaced by games which run on mobile phones and tablets. Moreover, people involved in the design and development of serious games come from various fields. For example, pedagogists and domain experts with limited, if any, coding skills are involved in designing and developing serious games. So, the choice to develop a game by using solely a programming language, such as C++, C# or Java, is not really feasible for everyone. This has made the usage of game engines even more important. Game engines expedite the process of developing a game through existing templates and assets that can be reused, minimizing or completely extinguishing the need to have a deep knowledge of programming.

The results of the comparative analysis regarding the features falling into the category of audio visual fidelity are presented. Game Maker is the least developed game engine in terms of audio visual fidelity, since the only category in which it supports all the features is sound.

As far as the engine's usability and learning curve are concerned, Unity seems to be better in terms of a user-friendly interface and free assets for developing games, while it has fewer requirements for hardware. Specifically: Unity and Unreal Engine 4 both integrate all the necessary tools for game development, but Unreal Engine 4 has a more complex interface in comparison with Unity that offers all the necessary tools in a single window.

Unity does not support light mapping, morphing animation, blends, 2D sound, and bump mapping. Unity does not support just multiple textures, while on the other hand it supports all the other features and seems to outperform the

rest of the engines in animation along with Unreal Engine. In Unity normal mapping is actually not supported as was denoted in the comparative analysis of the game engines. However, there are many available shaders that can be used for avoiding errors. Another problem faced was a problematic joystick from the asset store. This problem was dealt with by writing code for programming the joystick using existing tutorials. In Unreal Engine 4 there is no variety of mobile shaders and errors occurred. The shaders had to be modified using the material editor, while new settings for lighting had to be made. Moreover, there is no simulator module and the user has either to use a mobile previewer or launch the game while being developed. Finally, the installation is not straightforward and requires preparing and running the appropriate bat file using a computer.

2.3 Developing a Driving Training Game on Windows Mobile Phone Using C# and XNA

It deals with the development of an educational game that intends to teach the rules of road driving on the Windows Phone platform, using C# and XNA languages.

In our educational game, we focus on several aspects of how to teach the user the basics of driving. For examples, we focus on speeding, on the road signs, driving without crossing the center line, etc.

The Windows Phone platform has offered a lot of opportunities for developing new kinds of applications and games. Every year in the United States, the number of deaths on the road is very high especially for young people; the development of an educational game that teaches the basics of road driving may be helpful to enhance road safety.

To make this driving game more attractive and enjoyable to players, there are many improvements that can be done in the visual design of the application. Also, different scenarios can be added to put the player in different driving situations.

2.4. Research on of 3D Game Design and Development Technology

Firstly, this paper puts forward to design concept, designs the main frame of 3D game. Secondly, through researching collision detection technologies, the improved collision detection method is proposed based on Aligned Axis Bounding Box according to some disadvantages of collision detection methods. The improved collision detection method effectively reduces operation time, advanced game speed. Thirdly, through researching artificial intelligence for 3D game, path search method is improved according to the needs of system. The improved idea uses restricting search range of A * search arithmetic with the shortest path. It can get intelligent path search efficiently and factually. Finally, the 3D game system is realized based on 3D engine technologies. The system is rendered smoothly, authentically, manipulated easily.

Collision detection algorithm: in this procedure, the action responding to collision is that virtual character slides along the edge of the object colliding so as to avoid obstacle. In this process, you have to pay attention to the situation that d is O. This experiment adopts AABB collision detection algorithm and improved algorithm.

The experiment shows that the improved collision detection algorithm could effectively reduce the calculation amount of system and promote operational speed, and calculation amount reduces 11.85%, the operational speed of game promotes 5.33%. For the defects of large storage space, large calculation for node and long time for searching, this article puts forward the strategy of grading path search and improve 765 A * algorithm. The improved algorithm could achieve optimal path search when the game map is not very complex. From the experimental data, we could see that the improved A * path search algorithm could make visitor volume for inner storage reduce 34.76% and the rendering frequency for game promote 2.56%.

Path Search Collision detection technology and artificial intelligent technology could be applied in the development of game, as well as the areas of education, virtual reality, historic preservation, geo-information system, scientific research, entertainment and business. Improvement idea mainly includes the following aspects: 1) apply BSP to discompose space of scene to reduce the calculation amount for collision detection; 2) optimize bounding volume; 3)put forward segment collision detection algorithm to promote the speed of AABB collision detection.

2.5 A Unity 3d Framework for Algorithm Animation

The goal of this paper is to outline a framework in which the Unity 3D game engine can be used to the fullest in developing new and modern ways to bring compelling learning experiences to new computer science and engineering students. Additionally, the same type of visualization and animation technology could be utilized by established programmers as a sandbox for determining the effectiveness of various algorithms and heuristics for determining a best solution for a target problem.

The line and map graph project serves as a basic learning framework for teaching the basic concepts relating to graphs and graph-related search algorithms. The first implementation of a Unity 3D framework for algorithm animation is a project which generates an X, Y grid (the user may define width and height variables X and Y as integer inputs). A maze generation algorithm may then be run on the grid any number of times in order to demonstrate the many unique graph structures that might result from such a generation. Hexagon project aims to immerse the learner in an interactive experience in which he or she can easily and directly change the state of the graph.

This study is an exciting one, because it opens many doors to lots of potential future work. The framework for algorithm animation established in this

study could easily be expanded to include many other data types commonly found in computer science and engineering (e.g. trees, 3D graph representations, etc.)

In addition to this, a number of convenient features could be added to improve the overall quality and usability of this software. One such useful feature is serialization and deserialization of user-created graph structures and configurations that could be easily saved and loaded from file. This could help a prospective learner or researcher to save his or her work incrementally and use this software for long-term studies and features. Beyond added graphs and product features, the visualization and animation quality could be further improved to make better use of the stellar graphical capabilities of the Unity 3D engine. More media types and multimedia elements could be implemented in order to aid learning (e.g. audio, video cues).

2.6 Research on the Elimination of Game Based on Unity 3d Technology

Through the study of various problems in the elimination game, with unity 3D technology which has the advantages of convenient interactive graphical development environment and the cross-platform features, and C# scripting language is simple, safety, efficient, this paper carefully investigated and described the candy elimination of categories in mobile Android game algorithm design, AI design and the game development process. To achieve the level of the game options, multiple candy elimination, special effects, scores of statistics, and other functions.

Candy elimination game system is by the Unity3D engine and the C# programming language released on the PC side, Android platform, it is made through the using of C# development language and the Visual Studio 2013 compiler development platform, implemented based on Unity3D technology to develop the elimination game, for beginners learning to develop games and

develop such games have a certain guidance and reference. The game implements the initial interface described above, select level, game interface, all the features of victory and fractional statistics, there are five modules. At present, the candy elimination mobile game has been released in Android market, downloads reached 1000 times. Elimination game has successfully achieved the basic functions. The game is interesting and rich, get the players praise, and for leisure travelers provides an entertainment program in spare time.

Through the study of the elimination game in series of algorithms and structural design, in the each game, the game interface, select levels, each level AI design, different cancellation algorithm, fractional statistics, etc. Using Unity3D Technology to design a Android platform for the elimination of candy type mobile phone games, which has the certain guidance and reference for the development of these games or other games.

The Array List array is used to store the information of each candy location information, and the detection and elimination are tested by the whole array traversal, the method of time complexity is O(n), to achieve a fast game goal. This game development for beginners and researchers eliminate some of the problems in games play a guiding role, but also has some practical value. However, it should be noted that. Although mentioned in this paper candy elimination game is relatively complete, but the specific details of the detail remains to be studied, such as how to detect whether can exchange two candies after update candy game scene, also need more in-depth study and reflection.

2.7 Using Computer Game to Develop Advanced AI

These games populate the environments with both human and computer controlled characters, making them a rich laboratory for artificial intelligence research into developing intelligent and social autonomous agents. Indeed, computer games offer a fitting subject for serious academic study, undergraduate education, and graduate student and faculty research. Creating and efficiently rendering these environments touches on every topic in a computer science curriculum. The "Teaching Game Design" sidebar describes the benefits and challenges of developing computer game design courses, an increasingly popular field of study

In computer games, designers can use AI to control individual characters, provide strategic direction to character groups, dynamically change parameters to make the game appropriately challenging, or produce a sports game's play-by-play commentary. Computer games offer an inexpensive, reliable, and surprisingly accessible research environment, often with built-in AI interfaces. Moreover, computer games avoid many of the criticisms leveled against research based on simulations: Because researchers do not develop them, the games avoid embodying preconceived notions about which aspects of the world designers can simulate with ease and which aspects they can simulate only with difficulty. These games constitute real products that create real environments with which millions of humans can interact vigorously.

The factors they have listed is combined with the layout of the level's topology, make the game tactically complex, which becomes obvious when watching an expert play

Although we have found computer games to be a rich environment for research on human-level behavior representation, we do not believe that the future of AI in games lies in creating more and more realistic arenas for violence. Better AI in games has the potential for creating new game types in which social interactions, not violence, dominate. The Sims9 provides an excellent example of how social interactions can be the basis for an engaging game. Thus, we are pursuing further research within the context of creating computer games that emphasizes the drama that arises from social interactions between humans and computer characters.

CHAPTER 3

3.1 SYSTEM METHODOLOGY

SYSTEM ARCHITECTURE

We have two levels in this game. Once the player is entered, Player can have a four options. They can play the game or they choose the options for sound settings. They can also have the options to buy the character with earned coins. They have to increase points for buying character or unlock character. Player have to collect mask, soap and sanitizer. Player should not be affected by virus. If Player is affected by virus, health bar get reduced. Player can increase our life span to collect mask, soap and sanitizer. In level 1 we have the target. When the target (Vaccine) is collected, the Level 2 will be unlocked.

If the game was kept as simple always the player would get bored to play the game repeatedly. Thus the complexity of the game has been increased which make the game much more eager to play. To make the game much more complex, more and more Obstacles has been created to achieve the target.

In level 2, we have a time limit and also the target to complete the level. We have to collect 5 sanitizer within the time limit. If the target is achieved, the corona virus awareness tips will be given. Through this, we can create awareness among the users.

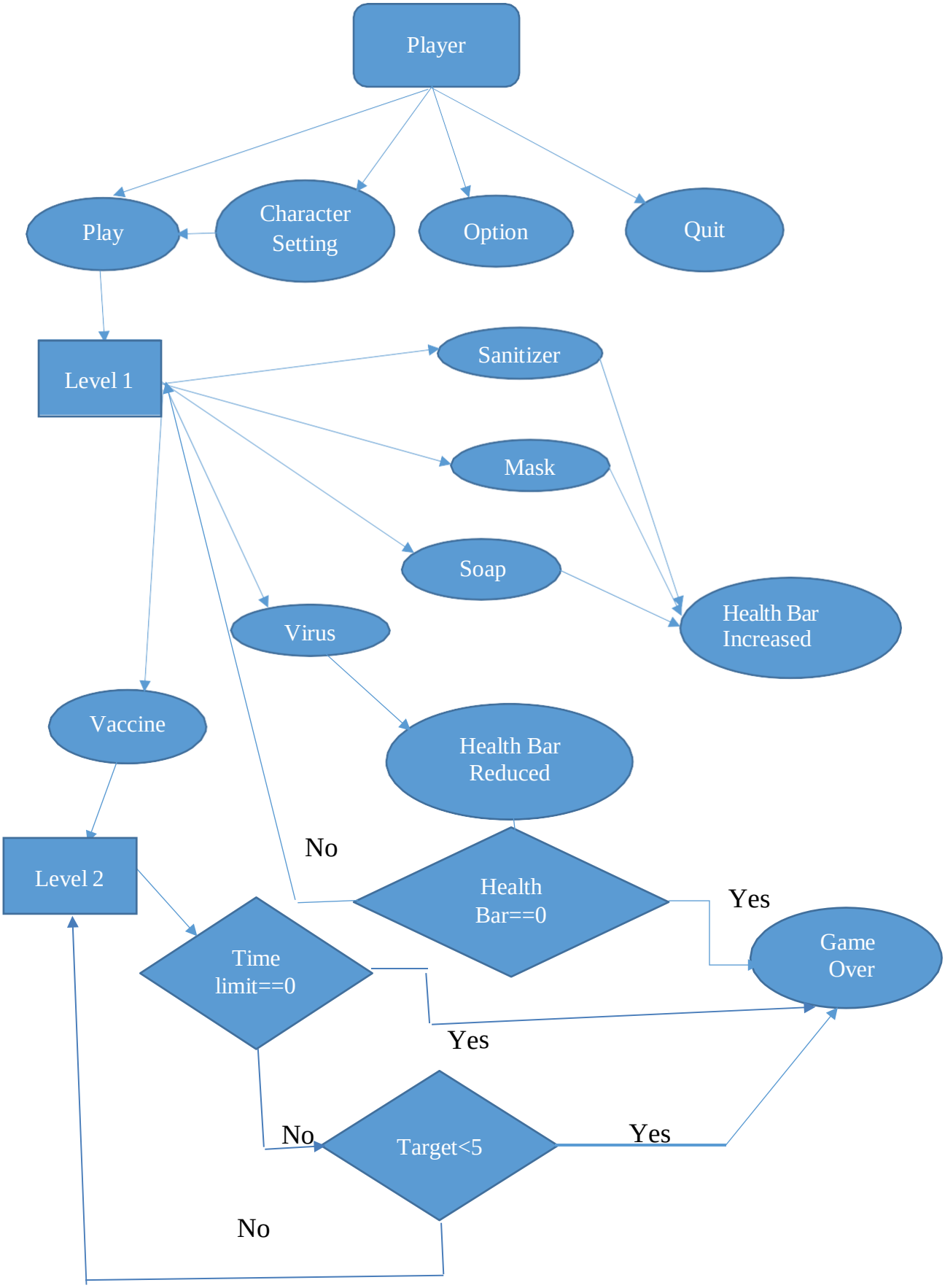


Figure 1.5: System architecture

3.2 Module 1 - Initiation of game environment and characters

Fixture of Environment

A game environment is also called as game environment is a specialized development environment for creating video games. The environment art should also support the game play. Environment is the major aspect which is required for a game. To set the suitable game environment assets are downloaded from online asset store

Character Setting

Three narrative elements we might consider while creating or analyzing a video game character are the setting of the game, the backstory of the character, and the structure of the plot. Characters and assets are placed in scene view. Movements were given to the characters developed-walk forward, right turn, left turn.

3.3.Module 2 – Completion of Level-1 and Setting up of user interface

A graphical user interface (GUI) is a type of user interface through which users interact with electronic devices via visual indicator representations. It provides the user the capability to intuitively operate computers and other electronic devices through the direct manipulation of graphical icons such as buttons, scroll bars, windows, tabs, menus, cursors and the mouse pointing device. Setting up of interface has been initiated by writing code in c sharp. In addition to that, Mini map has been implemented, health bar has been set. Materials collection such as Soap, masks, sanitizers needs to be collected by the player while playing the game.

3.4.Module 3 – Complex level of the game

If the game was kept as simple always the player would get bored to play the game repeatedly. Thus the complexity of the game has been increased which make the game much more eager to play. To make the game much more complex, more and more Obstacles has been created to achieve the target.

CHAPTER 4

4.1 SYSTEM IMPLEMENTATION

4.1.1 Hardware Requirements:

- Windows :10
- Processor : Minimum 1GHz;Recommended 2GHz or more
- Ethernet connection(LAN)or wireless adapter(Wi-Fi)
- Hard Drive : Minimum 32 GB; Recommended 64GB or more
- Memory(RAM) : Minimum 1GB; Recommended 4GB or above

4.1.2 Software Requirements:

Unity

- Unity is so much more than the world's best real-time development platform
- Unity is excellent for cross-platform development and multi-platform games.

C #

- Compatible frameworks and great tools, improve C#'s game-building capacity.

4.2 WORK BREAKDOWN STRUCTURE

Review 1

Resource Collection

The overall idea of this game development process has been revised and resources from various means were collected

Review 2

Setting Environment and creating movements

The game environment has been set and the character required for the game was created and movement for the characters were given. Movements include running and walking (move right, front, back, left)

Review 3

Virus Creation, Mask collection, AI Implementation

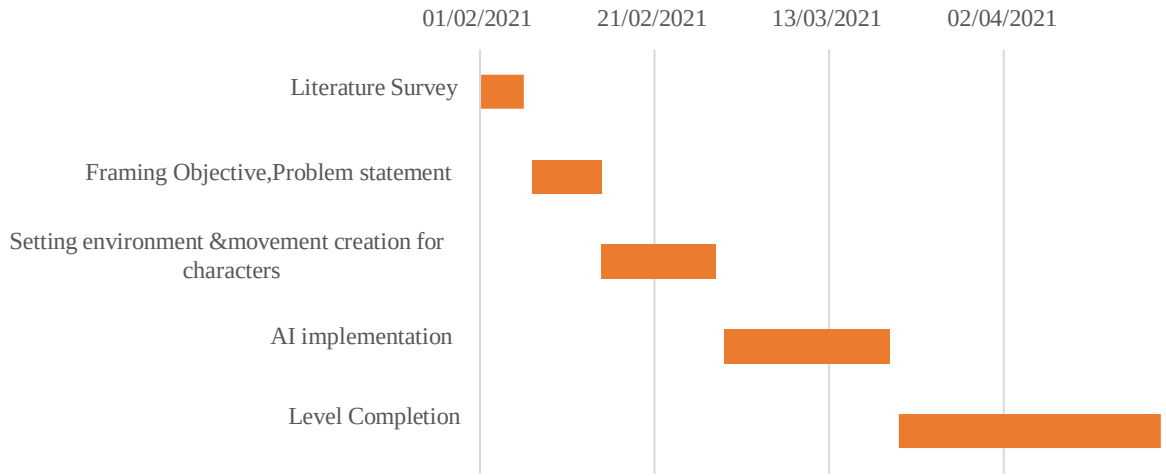
Virus are the major aspect to keep up the game alive. Thus viruses has been created .These virus attacks the player, by which the player has to escape from the viruses. Collection of masks for the player to increase the life span has been implemented using Artificial intelligence.

Review 4

Level Completion

The overall game was developed and the levels where tested. The game consists of two levels.

4.3 GANTT CHART



4.4 COST ESTIMATION

COCOMO	a	b	c	D
Organic	2.4	1.05	2.5	0.38
Semi Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

LOC = Lines of code

1KLOC = 1000 LOC

Effort = a * (KLOC)^b persons per month

Schedule = c * (Effort)^d Months

Persons = Effort / Schedule Persons

E = 2.4 * (2)^1.05 = 2.4 * 2.07 = 4.96

S = 2.5 * (5)^0.38 = 2.5 * 1.84 = 4.60

P = 5 / 5 = 1

CHAPTER 5

PO RELEVANCE

Vision - To make the department of Information Technology the unique of its kind in the field of Research and Development activities in this part of world.

Mission - To impart highly innovative and technical knowledge in the field of Information Technology to the urban and unreachable rural student folks through Total Quality Education.

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

PO1 - Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2 - Problem Analysis: Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3 – Design/ Development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4 - Conduct Investigations of Complex Problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5 - Modern Tool Usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6 - The Engineer and Society: Apply reasoning informed by the contextual

knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7 - Environment and Sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8 - Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9 - Individual and Team Work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10 - Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11 - Project Management and Finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12 - Life-Long Learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSO)

Engineering Graduates will be able to:

PSO 1 - Design and develop applications in the field of Information Technology by applying the knowledge acquired from Computer Organization & Engineering, Networking, Software Engineering & Programming, Data Analytics & other allied topics.

PSO 2 - Demonstrate an ability to analyse, design, and develop software solutions to cater the needs of diversified business sectors.

PO RELEVANCE APPLICABLE TO PROJECT

PROJECT TITLE: LET'S CAVORT TO LEARN

PO RELEVANCE	**DESCRIPTION**	**LEVEL**
PO1-Engineering Knowledge	Engineering knowledge is applied in all phases of the project	High
PO3-Design/Development of Solutions	Sight change in the design of the IDS is proposed	Medium
PO4-Conduct Investigations of complex problems	Investigation for more complex levels of the game	Low
PO7-Environment and sustainability	Every environment and sustainable support is there for this work	High
PO8-Ethics	Ethical practices were adopted	Ethical practices were adopted
PO9-Individual and	Yes students did both	High

team work	individuals and team work	
PO10-Communicaton	Their presentation is good	High
PO11-Project Management and Finance	They manage the works related to project	High
PO12-Lifelong Learning	Lifelong learning is possible with their project	High
PSO1-Design and develop Application	Students developed the algorithm	High

CHAPTER 6

RESULT AND DISCUSSION

RESULT AND DISCUSSION

6.1 RESULTS

6.1.1 DATA SET

- Data set is not applicable.
- Inputs are given only through keyboard.

6.1.2 KEY GENERATION

We use the following keys to play our game:

KEYS	PURPOSE
W	Move front
D	Move Right
A	Move Left
Shift+w	Run front
Shift+D	Run Right
Shift+A	Run Left

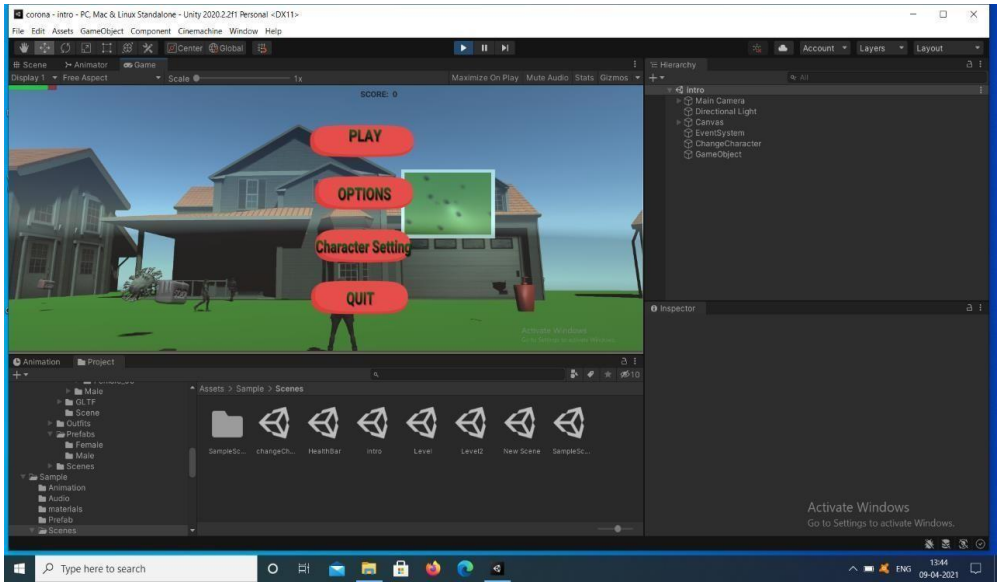
6.2 DISCUSSION

This is the overall structure of our game.

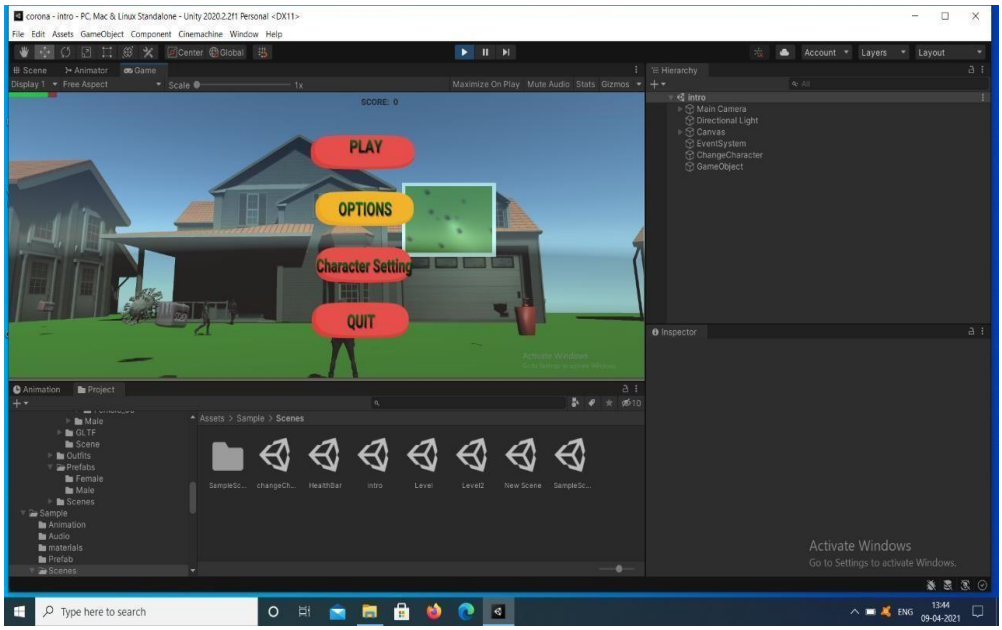
It consists of 4 options

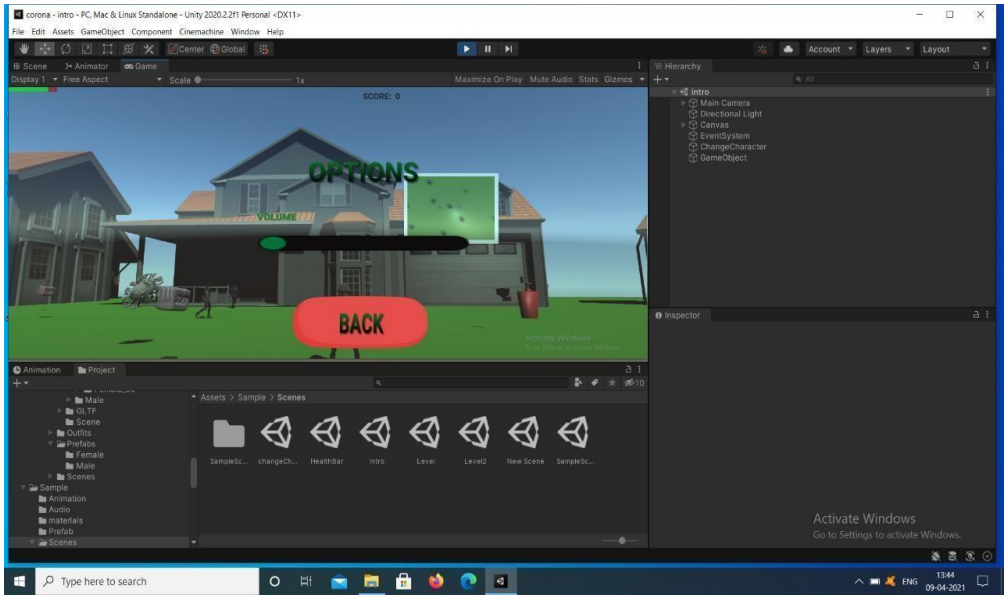
They are:

1. Play
   - Start the game and move on with the consecutive levels
2. Options
   - Music
   - Back to home page
3. Character setting
   - Choosing and Setting up of characters
   - One free character for initiation of game
   - Collect coins by playing the game to unlock more characters
4. Quit
   - Quit from the game and returns to home page.

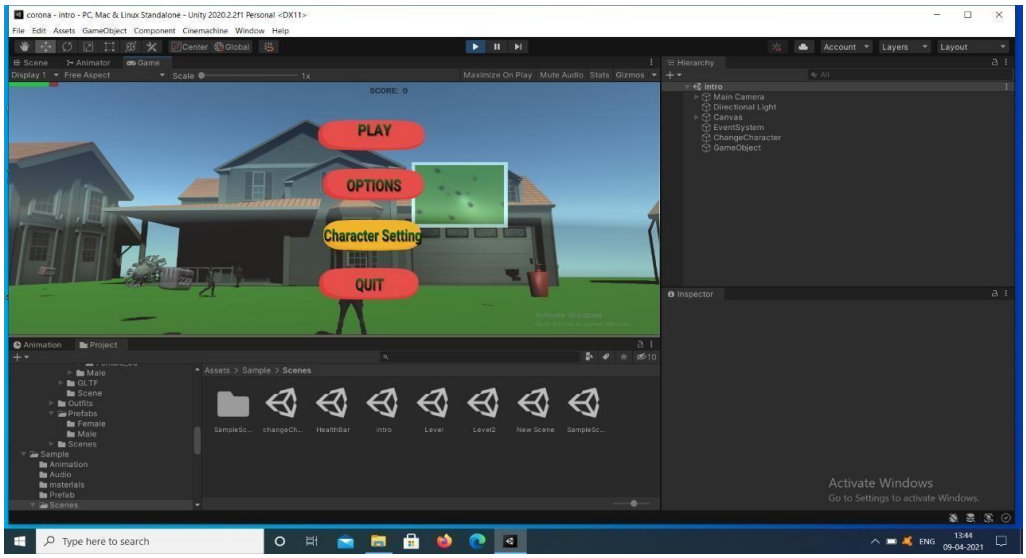


There will be a four options Play, Options, Character settings and Quit. Player can choose any of these options.

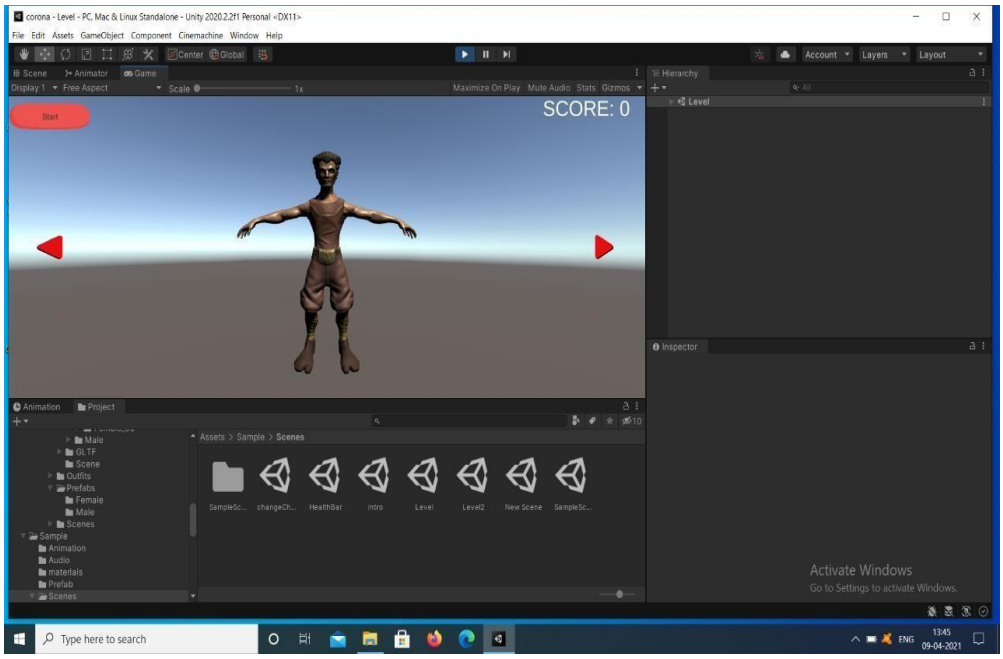




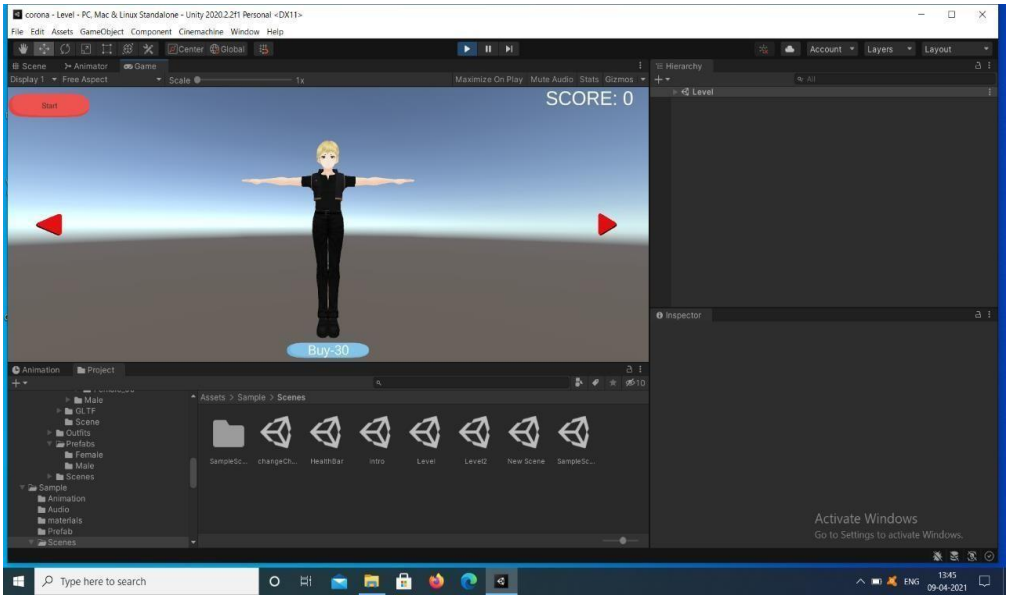
We have a options buttons to control the volume. User can increase or decrease the volume using this button. They also have the back button. Back to home page.



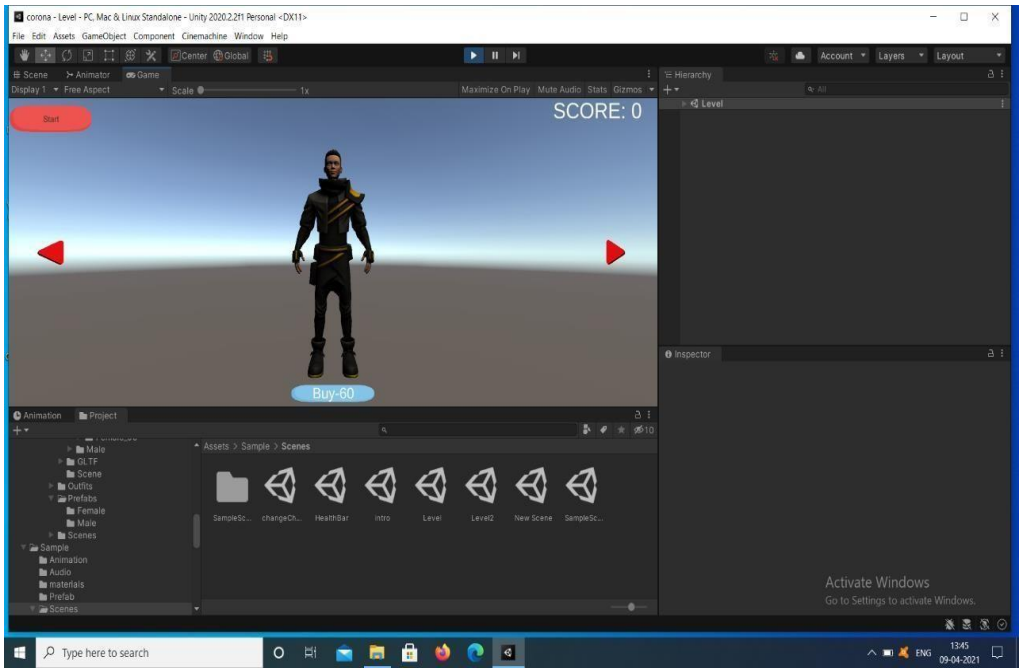
We have a character setting options. We have four characters. There will be one free character. Remaining characters will be locked. We have to unlock the character using points. User can increase points by collecting mask, soap and sanitizer.



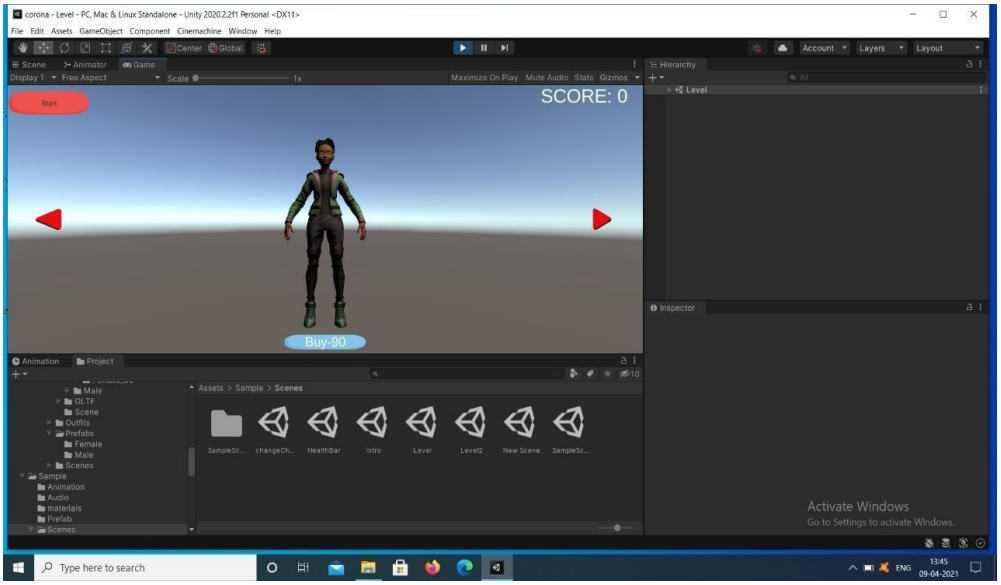
This is the free character.



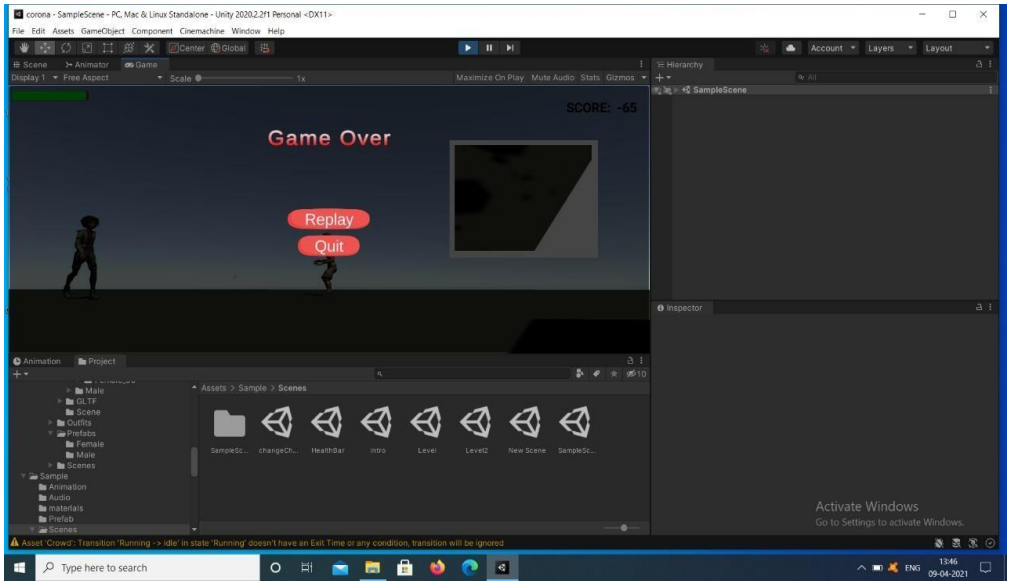
This character is locked. So the buy button will be disabled. User can enable the button after scoring 30 points. After that user can use this character. After unlock this character, button colour will be changed to red.



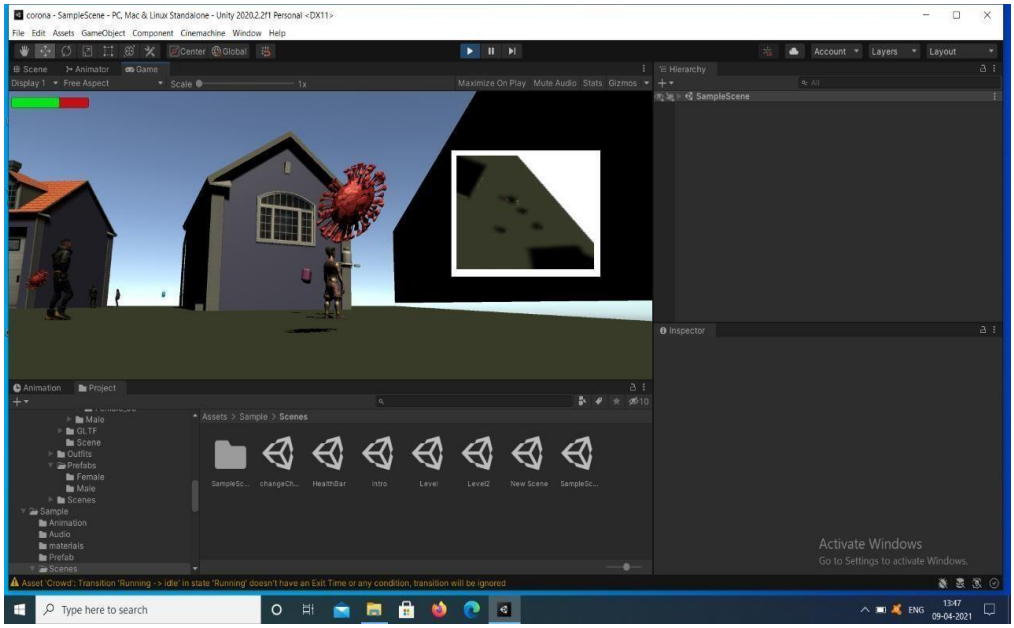
This character is locked. So the buy button will be disabled. User can enable the button after scoring 60 points. After that user can use this character. After unlock this character, button colour will be changed to red.



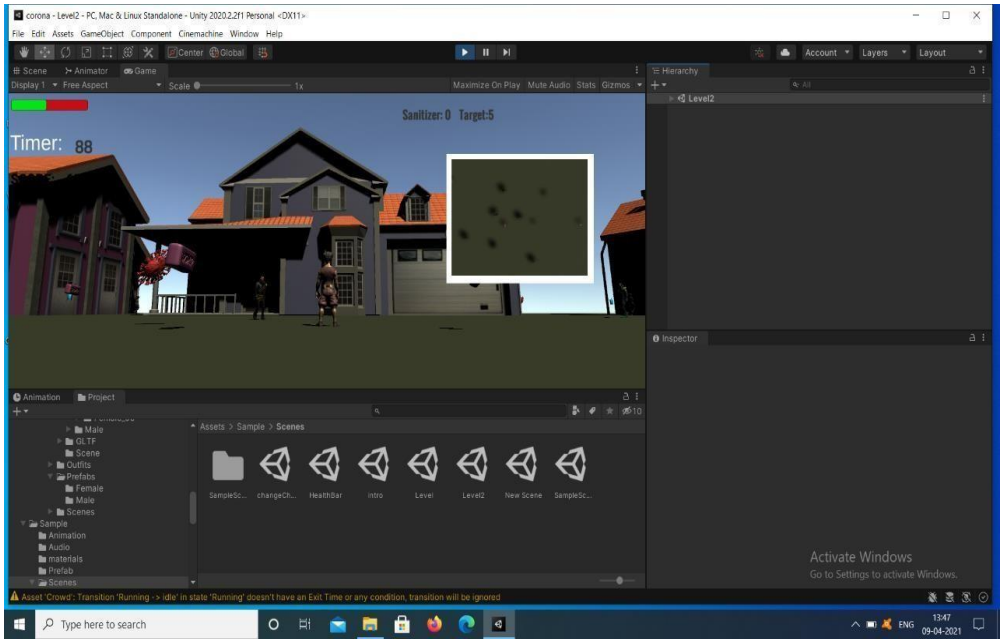
This character is locked. So the buy button will be disabled. User can enable the button after scoring 90 points. After that user can use this character. After unlock this character, button colour will be changed to red.



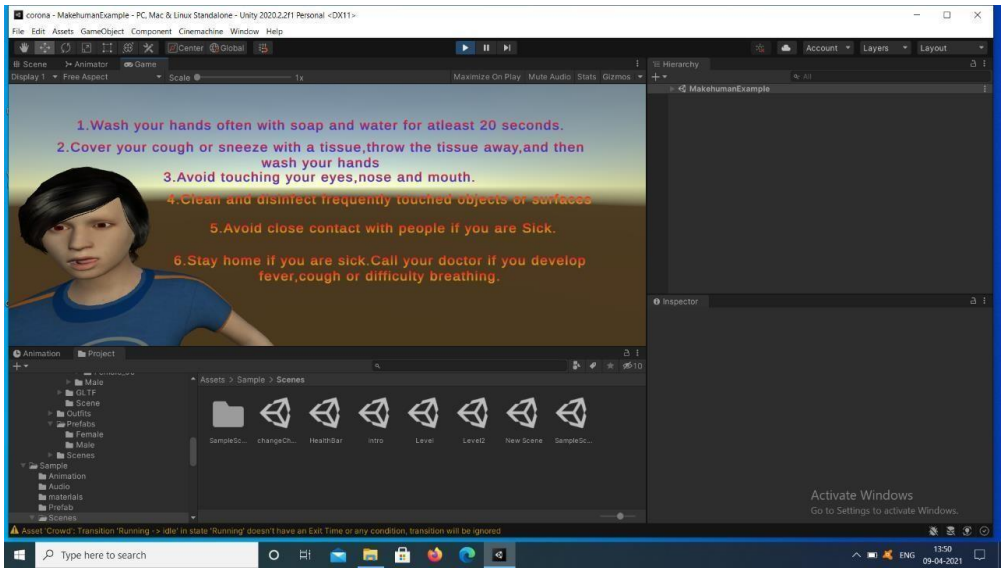
This is the game over page. When user failed to achieve the target, this page will appear. We have replay and quit options in this page. We can exit the game using Quit button and we can play again the game using replay button.



Player have to collect mask, soap and sanitizer. Player should not be affected by virus. If Player is affected by virus, health bar get reduced. Player can increase our life span to collect mask, soap and sanitizer. In level 1 we have the target. When the target (Vaccine) is collected, the Level 2 will be unlocked.



In level 2, we have a time limit and also the target to complete the level. We have to collect 5 sanitizer within the time limit. If the target is achieved, the corona virus awareness tips will be given. Through this, we can create awareness among the users.



After completing the two levels, the character will be explaining the awareness tips to prevent the spread of corona virus.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 Conclusion

Thinking of the game as a part of a bigger educational process is really in the core mind-set that this project wants to promote. Games can do many things very well, but they certainly cannot do everything at once. Especially not without solid supporting structures around them. Throughout the project and the case studies we built this was tr ue. This project aimed as much at using alternative and innovative methods to teach through coding digital games and playing games as part of learning. We have developed our project in such a way that is becomes the best awareness game among common people and children. It is a game which invokes children's interest.

7.2 Future Enhancement

In future we are in the idea of, to create more complicated levels. Conversion of desktop game to mobile application. Implementation of social distancing and isolation. Provision of gifts and daily bonuses. Connect this game via social media as muliple players game.

APPENDICES

SAMPLE CODE

AIController

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class AIController : MonoBehaviour{
    GameObject[] goalLocations;
    UnityEngine.AI.NavMeshAgent agent;
    Animator anim;
   void Start()
   {
       goalLocations=GameObject.FindGameObjectsWithTag("Goal");
        agent=this.GetComponent<UnityEngine.AI.NavMeshAgent>();
        agent.SetDestination(goalLocations[Random.Range(0,goalLocations.
Length)].transform.position);
        anim=this.GetComponent<Animator>();
        anim.SetTrigger("isWalk");
   }
   void Update()
   {
       if(agent.remainingDistance<1)
        {
        agent.SetDestination(goalLocations[Random.Range(0,goalLocations.
Length)].transform.position);
        }
   }
}
```

AudioScript

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class AudioScript : MonoBehaviour
{
    public AudioSource backgroundAudio;
   void Start()
   {
      backgroundAudio.Play();
   }
}
```

ChangeCharacter

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class ChangeCharacter : MonoBehaviour
{
  public void btn_change_character(string scene_name)
  {
       SceneManager.LoadScene(scene_name);
  }
}
```

CharacterBlueprint

```
[System.Serializable]
public class CharacterBlueprint
{
  public string name;
```

```
  public int index;
  public int price;
  public bool isUnlocked;
}
```

CharacterSelector

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class CharacterSelector : MonoBehaviour
{
  public  int  currentCharacterIndex;
   public GameObject[] characters;
   void Start()
   {
        currentCharacterIndex=PlayerPrefs.GetInt("SelectedCharacter",0);
      foreach(GameObject character in characters)
                character.SetActive(false);
    characters[currentCharacterIndex].SetActive(true);
   }
}
```

Coins

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
public class coins : MonoBehaviour
{
    [SerializeField]  TextMeshProUGUI  ScoreText;
```

```
    void Start()
    {
         Score.theScore1=PlayerPrefs.GetInt("Coins");
    }
    void Update()
    {
        PlayerPrefs.SetInt("Coins",Score.theScore1);
        ScoreText.text="SCORE:  "+Score.theScore1;
    }
}
```

CountdownTimer

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class CountdownTimer : MonoBehaviour
{
    float currentTime=0f;
    float  startingTime=100f;
     public GameObject player;
    [SerializeField]  Text CountDown;
    public static bool gameOver;
    public GameObject GameOver;
    void Start()
    {
      currentTime=startingTime;
    }

   void Update()
```

```
    {
        currentTime-=1*Time.deltaTime;
         CountDown.text=currentTime.ToString("0");
         if(currentTime<=0)
         {
                currentTime=0;
                gameOver=true;
                GameOver.SetActive(true);
         }

    }

}
```

EndTrigger

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class EndTrigger : MonoBehaviour
{
    public GameManager gameManager;
    public AudioClip collectSound;
    void OnTriggerEnter()
    {
         AudioSource.PlayClipAtPoint(collectSound,transform.position);
         gameManager.CompleteLevel();
    }
}
```

Events

```
using UnityEngine.SceneManagement;
```

```
using UnityEngine;
public class Events : MonoBehaviour
{
    public void ReplayLevel()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildInd
ex);
    }
    public void QuitGame()
    {
        Application.Quit();
    }
```

LevelComplete

```
using UnityEngine;
using UnityEngine.SceneManagement;
public class LevelComplete : MonoBehaviour
{
   public void LoadNextLevel()
   {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildInd
ex+1);
   }
}
```

MainMenu

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class MainMenu : MonoBehaviour
```

```
{
    public void PlayGame()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildInd
ex+1);
    }
    public void QuitGame()
    {
        Debug.Log("QUIT!");
        Application.Quit();
    }
}
```

maskPicker

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class maskPicker : MonoBehaviour
{
    public AudioClip collectSound;

    private void OnTriggerEnter(Collider other)
    {
        if(other.tag=="Player")
        {
        AudioSource.PlayClipAtPoint(collectSound,transform.position);
                    PlayerManager.health += 20;
                    Score.theScore+=20;
                    Score.theScore1+=20;
                    Destroy(gameObject);
```

```
        }
        if(PlayerManager.health==0)
       {
              FindObjectOfType<GameManager>().EndGame();
        }
   }
}
```

Minimap

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

public class Minimap : MonoBehaviour
{
   public Transform player;
   void LateUpdate()
   {
        Vector3 newPosition=player.position;
        newPosition.y=transform.position.y;
        transform.position=newPosition;
        transform.rotation=Quaternion.Euler(90f,player.eulerAngles.y,0f);
   }
}
```

Movement

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Movement : MonoBehaviour
{
```

```
Animator animator;
float velocityZ=0.0f;
float velocityX=0.0f;
public float acceleration=2.0f;
public  float deceleration=2.0f;
public float maximumWalkVelocity=0.5f;
public float maximumRunVelocity=2.0f;
    int VelocityZHash;
    int VelocityXHash;
public float playerSpeed=10.0f;
public float playerRotationSpeed=100.0f;
void Start()
{
   animator=GetComponent<Animator>();
    VelocityZHash=Animator.StringToHash("Velocity Z");
    VelocityXHash=Animator.StringToHash("Velocity X");
}
void changeVelocity(bool forwardPressed,bool leftPressed,bool
rightPressed,bool runPressed,float currentMaxVelocity)
{
    if(forwardPressed &&  velocityZ< currentMaxVelocity)
    {
        velocityZ +=Time.deltaTime *acceleration;
    }
    if(leftPressed && velocityX> -currentMaxVelocity)
    {
        velocityX  -=Time.deltaTime*acceleration;
    }
    if(rightPressed && velocityX< currentMaxVelocity)
```

```
        {
            velocityX   +=Time.deltaTime*acceleration;
        }
        if(!forwardPressed  && velocityZ >0.0f)
        {
            velocityZ -= Time.deltaTime *deceleration;
        }
        if(!leftPressed  && velocityX <0.0f)
        {
            velocityX   +=Time.deltaTime*deceleration;
        }
        if(!rightPressed  && velocityX >0.0f)
        {
            velocityX  -=Time.deltaTime*deceleration;
        }
    }
    void lockOrResetVelocity(bool forwardPressed,bool leftPressed,bool
rightPressed,bool runPressed,float currentMaxVelocity)
    {
        if(!forwardPressed  && velocityZ <0.0f)
        {
            velocityZ=0.0f;
        }
        if(!leftPressed && !rightPressed && velocityX!=0.0f
&&(velocityX>-0.05f &&  velocityX <0.05f))
        {
            velocityX=0.0f;
        }
```

```
        if(forwardPressed && runPressed && velocityZ >currentMaxVelocity)
        {
            velocityZ=currentMaxVelocity;
        }
        else if(forwardPressed  && velocityZ > currentMaxVelocity)
        {
            velocityZ -= Time.deltaTime *deceleration;
            if(velocityZ > currentMaxVelocity && velocityZ < (currentMaxVelocity+0.05))
            {
                velocityZ=currentMaxVelocity;
            }
        }
        else if(forwardPressed && velocityZ < currentMaxVelocity && velocityZ >(currentMaxVelocity-0.05f))
        {
            velocityZ=currentMaxVelocity;
        }
        if (leftPressed && runPressed && velocityX < -currentMaxVelocity)
        {
            velocityX = -currentMaxVelocity;
        }
       else if(leftPressed && velocityX < -currentMaxVelocity)
        {
            velocityX += Time.deltaTime*deceleration;
            if (velocityX < -currentMaxVelocity && velocityX > (-currentMaxVelocity-0.05f))
            {
```

```
                velocityX = -currentMaxVelocity;
            }
        }
        else if(leftPressed && velocityX > -currentMaxVelocity &&
velocityX < (-currentMaxVelocity+0.05f))
        {
            velocityX=  -currentMaxVelocity;
        }
        if (rightPressed && runPressed && velocityX >currentMaxVelocity)
        {
            velocityX=currentMaxVelocity;
        }
        else if(rightPressed  &&velocityX>currentMaxVelocity)
        {
            velocityX-=Time.deltaTime*deceleration;
            if                              (velocityX>currentMaxVelocity
&&velocityX<(currentMaxVelocity+0.05))
            {
                velocityX=currentMaxVelocity;
            }
        }
        else          if(rightPressed          &&velocityX<currentMaxVelocity
&&velocityX>(currentMaxVelocity-0.05f))
        {
            velocityX=currentMaxVelocity;
        }
    }
    void Update()    {
        bool    forwardPressed=Input.GetKey(KeyCode.W);
```

```
        bool leftPressed=Input.GetKey(KeyCode.A);
        bool rightPressed=Input.GetKey(KeyCode.D);
        bool  runPressed=Input.GetKey(KeyCode.LeftShift);
         float translation=Input.GetAxis("Vertical") *playerSpeed
*Time.deltaTime;
        float rotation=Input.GetAxis("Horizontal") *playerRotationSpeed
*Time.deltaTime;
        transform.Translate(0,0,translation);
        transform.Rotate(0,rotation,0);
        float currentMaxVelocity=runPressed ? maximumRunVelocity :
maximumWalkVelocity;
        changeVelocity(forwardPressed,leftPressed,rightPressed,runPressed,c
urrentMaxVelocity);
        lockOrResetVelocity(forwardPressed,leftPressed,rightPressed,runPres
sed,currentMaxVelocity);
        animator.SetFloat(VelocityZHash,velocityZ);
        animator.SetFloat(VelocityXHash,velocityX);
          }
}
```

PlayerManager

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class PlayerManager : MonoBehaviour
{
    public static int health=100;
    public GameObject player;
    public Slider healthBar;
```

```
    public static bool gameOver;
    public GameObject GameOver;
    void Start()
    {
        InvokeRepeating("ReduceHealth",1,1);
    }
    void ReduceHealth()
    {
        health=health-1;
        healthBar.value=health;
        if(health<=0)
        {
                player.GetComponent<Animator>().SetTrigger("death");
                gameOver=true;
                GameOver.SetActive(true);
                health=100;
        }
    }
}
```

Score

```
using UnityEngine;
using UnityEngine.UI;
public class Score : MonoBehaviour
{
    public GameObject scoreText;
   public static int theScore;
   public static int theScore1;
   void Start()
    {
```

```
        theScore1=PlayerPrefs.GetInt("Coins");

    }

 void Update()
    {

        scoreText.GetComponent<Text>().text="SCORE: "+theScore;
        PlayerPrefs.SetInt("Coins",theScore1);
    }
}
```

ScoreManager

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class ScoreManager : MonoBehaviour
{
    public static ScoreManager instance;
    public  GameManager  gameManager;
    public Text CountText;
    int count=0;
     private void Awake()
     {
         instance=this;
    }
    void Start()
    {
       CountText.text="Sanitizer: "+count.ToString();
```

```
    }
    public void AddPoint()
    {
        count+=1;
        CountText.text="Sanitizer:   "+count.ToString();

        if(count==5)
        {
                gameManager.CompleteLevel();
        }
    }

}
```

ShopManager

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using TMPro;
public class ShopManager : MonoBehaviour
{
    public int currentCharacterIndex;
    public GameObject[]  characterModels;
    public CharacterBlueprint[] characters;
    public Button BuyButton;
    void Start()
    {
        foreach(CharacterBlueprint character in characters)
```

```
        {
                if(character.price==0)
                {
                        character.isUnlocked=true;
                }
                else
                character.isUnlocked=PlayerPrefs.GetInt(character.name,0)==0
? false: true;
        }
        currentCharacterIndex=PlayerPrefs.GetInt("SelectedCharacter",0);
       foreach(GameObject character in characterModels)
                character.SetActive(false);
        characterModels[currentCharacterIndex].SetActive(true);
    }
    void Update()
    {
       UpdateUI();
    }
    public void ChangeNext()
    {
        characterModels[currentCharacterIndex].SetActive(false);
        currentCharacterIndex++;

        if(currentCharacterIndex==characterModels.Length)
                currentCharacterIndex=0;

        characterModels[currentCharacterIndex].SetActive(true);
        CharacterBlueprint      c=characters[currentCharacterIndex];
        if(!c.isUnlocked)
```

```
            return;
    PlayerPrefs.SetInt("SelectedCharacter",currentCharacterIndex);
}
public void ChangePrevious()
{
    characterModels[currentCharacterIndex].SetActive(false);
    currentCharacterIndex--;

    if(currentCharacterIndex==-1)
            currentCharacterIndex=characterModels.Length-1;

    characterModels[currentCharacterIndex].SetActive(true);
    CharacterBlueprint      c=characters[currentCharacterIndex];
    if(!c.isUnlocked)
            return;
    PlayerPrefs.SetInt("SelectedCharacter",currentCharacterIndex);
}
public void UnlockCharacter()
{
    CharacterBlueprint    c=characters[currentCharacterIndex];

    PlayerPrefs.SetInt(c.name,1);
    PlayerPrefs.SetInt("SelectedCharacter",currentCharacterIndex);
    c.isUnlocked=true;
    PlayerPrefs.SetInt("NumberOfCoins",Score.theScore1-c.price);
}
private void UpdateUI()
{
    CharacterBlueprint    c=characters[currentCharacterIndex];
```

```
        if(c.isUnlocked)
        {
            BuyButton.gameObject.SetActive(false);
        }
        else
        {
            BuyButton.gameObject.SetActive(true);

        BuyButton.GetComponentInChildren<TextMeshProUGUI>().text="B
uy-" +c.price;
            if(c.price   <PlayerPrefs.GetInt("Coins",0))
            {
                BuyButton.interactable=true;
            }
            else
            {
                BuyButton.interactable=false;
            }
        }
    }
}
```

Sanitizer

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class sanitizer : MonoBehaviour
{
   public AudioClip collectSound;
```

```
  private void OnTriggerEnter(Collider other)
   {
       if(other.tag=="Player")
       {
       AudioSource.PlayClipAtPoint(collectSound,transform.position);
                   PlayerManager.health += 15;
                   Score.theScore+=15;
                   Score.theScore1+=15;
                   Destroy(gameObject);
       }
       if(PlayerManager.health==0)
      {
            FindObjectOfType<GameManager>().EndGame();
       }
   }
}
```

Soap

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class soap : MonoBehaviour
{
   public AudioClip collectSound;
   private void OnTriggerEnter(Collider other)
   {
       AudioSource.PlayClipAtPoint(collectSound,transform.position);
       if(other.tag=="Player")
       {
                   PlayerManager.health += 10;
```

```
                    Score.theScore+=10;
                    Score.theScore1+=10;
                    Destroy(gameObject);

        }
        if(PlayerManager.health==0)
       {
              FindObjectOfType<GameManager>().EndGame();

        }
    }
}
```

Virus

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class virus : MonoBehaviour
{
    public AudioClip collectSound;
    private void OnTriggerEnter(Collider other)
    {
        AudioSource.PlayClipAtPoint(collectSound,transform.position);
        if(other.tag=="Player")
        {
                    PlayerManager.health -= 25;
                    Score.theScore-=25;
                    Score.theScore1-=25;
                    Destroy(gameObject);
```

```
        }
        if(PlayerManager.health==0)
       {
              FindObjectOfType<GameManager>().EndGame();
        }
   }
}
```

REFERENCES

[1] https://www.c-sharpcorner.com/UploadFile/asmabegam/unity-3d-game-creation-using-C-Sharp-script/

[2] https://scottlilly.com/learn-c-by-building-a-simple-rpg-index/

[3] https://www.codeproject.com/Articles/1016088/Unity-D-Game-Creation-using-Csharp-Script

Journals

Name:Research on 3D game based Unity Technology

Author: Senhao Jia

Year: 2016