RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

ENDLESS RUNNER IN 3D

Submitted by:
Pragatheesh K - 201501037
Lokesh K - 201501501
Jeevan MM - 201501504
Tinku vignesh S - 201501505

AI19P52 AI for Game Programming

Department of Artificial Intelligence and Machine Learning

Rajalakshmi Engineering College, Thandalam

BONAFIDE CERTIFICATE

This is to certify that the Mini project work titled **ENDLESS RUNNER IN 3D** done by PRAGATHEESH K 201501037, LOKESH K 201501501, JEEVAN M M 201501504 ,TINKU VIGNESH S 201501505 is a record of bonafide work carried out by him under my supervision as a part of MINI PROJECT for the subject titled AI for Game Programming AI19P52 by Department of Artificial Intelligence and Machine Learning.

Dr. Bhagavathi Priya S
HEAD OF THE DEPARTMENT
Artificial Intelligence and Machine Learning,
Rajalakshmi Engineering College,
College, Thandalam,
Chennai – 602 105.

Mrs. Tupili Sangeetha
FACULTY IN CHARGE
Assistant Professor
Artificial Intelligence and Machine Learning,
Rajalakshmi Engineering
Thandalam,
Chennai – 602 105.

This project report is submitted for practical examination for AI for Game Programming AI19P52 to be held on….....at Rajalakshmi Engineering College, Thandalam.

EXTERNAL EXAMINER **INTERNAL EXAMINER**

TABLE OF CONTENTS

ABSTRACT

Endless runner in 3D which is the endless running game it consist the player Joe who want to run as far as possible to catch the thief who is running front of the Joe .The player want to collect the coin which has been left by the enemy and avoid obstacles which is on the environment and run continuously to set the high score . The player can move left and right to avoid the obstacles when the player hits the obstacles the player fall down and the game gets over . Its very interesting to play the game it gives the realistic feel that we are running to catch the thief . Endless runner 3D game will increasing difficulty as the player run further. In Endless runner 3D the environment is designed more interesting like the street which contain the big building and the obstacles that like some destroyed van, barricades, old tires , and some road blockers which is used to block the lane and this environment road consist of two lane . Endless runner 3D developed for PC ,MAC,LINUX . This 3D game was made using the Unity game engine with C# as the programming language. Some players feel interested in playing, but there are challenges, a pretty difficult to the user.

CHAPTER 1

INTRODUCTION

In recent years, game industries have entered a stage of rapid development. More different types of games have appeared on a variety of new platforms, because relatively low cost and huge profits have motivated more people to get involved in this area. Earlier people only played games on consoles or computers until now people are more willing to play games on mobile phones. A variety of platforms brings more intensive competitiveness. On the other hand, in addition to playing the games, more people are willing to participate in the production process of the games. Nowadays, there are many game engines allow people to create their own games in an easier and more convenient way, and Unity is one of the most suitable game engine for the beginner. Unity3D used to develop video games for multiple platforms, such as Web, PC, MAC, IOS and PS3 etc. On the other hand, Unity3D supports a variety of programming languages, which enables the programmers to program with their familiar language. For this project, I will use the Unity engine to create an endless running game. During this project, developers will get familiar with most of the features of Unity engine, and use them as much as possible in our process. Since the game is not only made with game engine, I also need to explore the links between Unity and other game related software. The purpose of this project is to show the complete processes of an independent game. In this report, I will start with basic theoretical framework of the game design. Then I will introduce the Unity engine. After that, I will describe the implementation levels of the project. The implementation contains full details of the game process. Finally, I will make a conclusion of this project involving the future development. Through every stage of learning and exploring in this project, it allows us to have a better image of the game production procedure, Moreover, the project helps to determine the difficulties which may be encountered in the production process and how to solve those problems.

procedurally-generated and separated into distinct regions, each region having different sets of challenges. The ship's speed remains constant as long as the ship remains in direct sunlight, but if the ship falls into shadow or clips objects, the ship's speed will drop. Direct collision with an object will cause the ship to be destroyed and end the run. The player continually earns points as long as the ship is moving. Throughout each region are various collectable objects. The most common are yellow pyramids that, if five are collected without any collisions, will boost the player's scoring multiplier by one. Others give the player's ship a short speed boost, a single-time jump to clear obstacles, and a single-time use shield to prevent the ship's destruction on a direct collision.

CHAPTER 2

LITERATURE SURVEY

Games have long been a popular area of AI research, and become a fast growing software industry since the 1990s. Video and symbolic games have become a facet of our daily lives and a major way of entertainment. Despite the fast expansion, research on intelligent games with learning functions is still in its initial phase. A game that learns to evolve through interactive actions and plays will make the game more challenging, more attractive, and demand less for labor-intensive script design and planning when developing a game. A major hurdle to implement game learning is the game complexity due to high dimensional, numerical and statistical characteristics of the sensing and action space when simulating a virtual world with the high-realism. The recent research shows that Computational intelligence techniques, characterized by numerical learning, optimization and self-organization, may provide a powerful tool set to solve the difficulties in game learning.

Game Artificial Intelligence (Game AI) refers to the techniques used in computer and video games to produce the illusion of intelligence in the behavior of non-player characters (NPCs). Artificial intelligence in games is typically used for creating player's opponents. If the computer opponent always does the same thing or is too difficult or too easy, the game will suffer. The main goal of Game AI is to not beat the player, but to merely entertain and be challenging. Advanced AIs are not always fun as they end up making the game much harder. It is essential at this point to understand the difference between Game AI and AI in the academic field. Game AI is not totally on intelligence and shares very few of the objectives of the academic field of AI. Real AI addresses fields of machine learning and decision making based on arbitrary data input as opposed to game AI which only attempts to produce the illusion of intelligence in the behavior of NPCs.

The history of artificial intelligence in video games can be dated back to the mid-sixties. The earliest real artificial intelligence in gaming was the computer opponent in “Pong” or its variations, which there were many. The incorporation of microprocessors at that time would have allowed better AI which did not happen until much later. Game AI agents for sports games like football and basketball were basically goal-oriented towards scoring points and governed by simple rules that controlled when to pass, shoot, or move. There was a much better improvement in the development of Game AI after the advent of fighting games such as “Kung Foo” for Nintendo or “Mortal Kombat”. The moves of the computer opponents were determined by what each player was currently doing and where they were standing. In the most basic games, there was simply a lookup table for what was currently happening and the appropriate best action. Enemy movement was primarily based on stored patterns. In the most complex cases, the computer would perform a short minimax search of the possible state space and best action would be returned. The minimax search had to be of short depth and less time consuming since the game was occurring in real-time.

The emergence of new game genres in the 1990s prompted the use of formal AI tools like finite state machines. Games in all genres started exhibiting much better AI after starting to use non deterministic AI methods. Currently, driving games like “Nascar 2002” have computer controlled drivers with their own personalities and driving styles. The traditional method for implementing game AI was by mainly using tristate state machines. The complexity of the AI that could be implemented was restricted because of the tristate state machines. Traditional AI implementation methods also predominantly followed the deterministic implementation method. In the deterministic implementation method the behavior or the performance of the NPC’s and the games in general could be specified beforehand and it was also too predictable. There was a lack of uncertainty which contributed to the entertaining factor of the game. An example of deterministic behavior is a simple chasing algorithm.

The current methods which are used for implementing Game AI vary from neural network, Bayesian technique, genetic algorithms, finite state machines and Pathfinding. All these methods are feasible but not applicable in every situation given. Game AI is a field where research is still going on and developers are still perfecting the art implementing all these methods in any situation given..

CHAPTER 3

GAME ARCHITECTURE

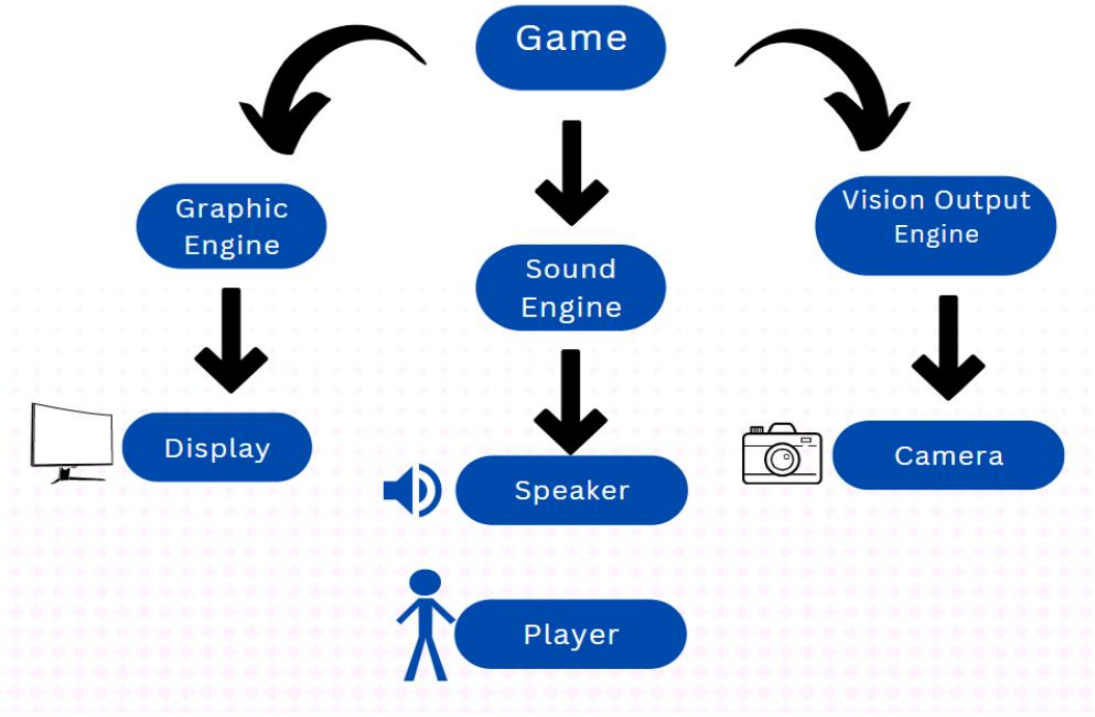


Fig 3.1

Fig. 3.1 represents our proposed game system architecture. The game architecture is divided into four categories. It contains the Main loop or the Game logic, Device API, I/O device, and Player. The main loop or the game logic contains the major component called the Game and the device API contains the Graphic engine, sound engine, and vision output engine. Display, Speaker, and Camera come under the category of I/O device. The game is divided into 3 different categories namely the graphic engine, sound engine, and vision output engine. A graphic engine is used by application programs to draw graphics on computer display screens. A sound engine is used to capture the sound and enhance the quality of the game for the players. The computer displays the output through the vision output engine. The Player gets to play the game with the help of I/O devices such as a Display, speaker, and camera.

CHAPTER 4
SOFTWARE USED

UNITY 2021.3.14:

Unity is a cross-platform game engine developed by Unity Technologies, first announced and released in June 2005 at Apple Worldwide Developers Conference as a Mac OS X game engine. The engine has since been gradually extended to support a variety of desktop, mobile, console and virtual reality platforms. It is particularly popular for iOS and Android mobile game development, is considered easy to use for beginner developers, and is popular for indie game development.

The engine can be used to create three-dimensional (3D) and two-dimensional (2D) games, as well as interactive simulations and other experiences. The engine has been adopted by industries outside video gaming, such as film, automotive, architecture, engineering, construction, and the United States Armed Forces.

The Unity game engine launched in 2005, aiming to "democratize" game development by making it accessible to more developers. The next year, Unity was named runner-up in the Best Use of Mac OS X Graphics category in Apple Inc.'s Apple Design Awards. Unity was initially released for Mac OS X, later adding support for Microsoft Windows and Web browsers.

Unity 2.0 (2007)

Unity 2.0 launched in 2007 with approximately 50 new features. The release included an optimized terrain engine for detailed 3D environments, real-time dynamic shadows, directional lights and spotlights, video playback, and other features. The release also added features whereby developers could collaborate more easily. It included a Networking Layer for developers to create multiplayer games based on the User Datagram Protocol, offering Network Address Translation, State Synchronization, and Remote Procedure Calls. When Apple launched its App Store in 2008, Unity quickly added support for the iPhone. For several years, the engine was uncontested on the iPhone and it became well-known with iOS game developers.

Unity 3.0 (2010)

Unity 3.0 launched in September 2010 with features expanding the engine's graphics features for desktop computers and video game consoles. In addition to Android support, Unity 3 featured integration of Illuminate Labs' Beast Lightmap tool, deferred rendering, a built-in tree editor, native font rendering, automatic UV mapping, and audio filters, among other things. In 2012 VentureBeat wrote, "Few companies have contributed as much to the flowing of independently produced games as Unity Technologies. More than 1.3 million developers are using its tools to create gee-whiz graphics in their iOS, Android, console, PC, and web-based games. Unity wants to be the engine for multi-platform games, period." A May 2012 survey by Game Developer magazine indicated Unity as its top game engine for

mobile platforms.

Unity 4.0 (2012)

In November 2012, Unity Technologies delivered Unity 4.0. This version added DirectX 11 and Adobe Flash support, new animation tools called Mecanim, and access to the Linux preview. Facebook integrated a software development kit for games using the Unity game engine in 2013. This featured tools that allowed tracking advertising campaigns and deep linking, where users were directly linked from social media posts to specific portions within games, and easy in-game-image sharing.[18] In 2016, Facebook developed a new PC gaming platform with Unity.[19] Unity provided support for Facebook's gaming platforms, and Unity developers could more quickly export and publish games to Facebook.

Unity 5 (2015)

The Verge said of 2015's Unity 5 release: "Unity started with the goal of making game development universally accessible. [...] Unity 5 is a long-awaited step towards that future." With Unity 5, the engine improved its lighting and audio. Through WebGL, Unity developers could add their games to compatible Web browsers with no plug-ins required for players. Unity 5.0 offered real-time global illumination, light mapping previews, Unity Cloud, a new audio system, and the Nvidia PhysX 3.3 physics engine. The fifth generation of the Unity engine also introduced Cinematic Image Effects to help make Unity games look less generic. Unity 5.6 added new lighting and particle effects, updated the engine's overall performance, and added native support for Nintendo Switch, Facebook Gameroom, Google Daydream, and the Vulkan graphics API. It introduced a 4K video player capable of running 360-degree videos for virtual reality. However, some gamers criticized Unity's accessibility due to the high volume of quickly produced games published on the Steam distribution platform by inexperienced developers.CEO John Riccitiello said in an interview that he believes this to be a side-effect of Unity's success in democratizing game development: "If I had my way, I'd like to see 50 million people using Unity – although I don't think we're going to get there any time soon. I'd like to see high school and college kids using it, people outside the core industry. I think it's sad that most people are consumers of technology and not creators. The world's a better place when people know how to create, not just consume, and that's what we're trying to promote."

Unity (2017–present)

In December 2016, Unity Technologies announced that they would change the versioning numbering system for Unity from sequence-based identifiers to year of release to align the versioning with their more frequent release cadence; Unity 5.6 was therefore followed by Unity 2017. Unity 2017 tools featured a real-time graphics rendering engine, color grading and worldbuilding, live operations analytics and performance reporting. Unity 2017.2 underscored Unity Technologies' plans beyond video games. This included new tools such as Timeline, which allowed developers to drag-and-drop animations into games, and Cinemachine, a smart camera system within games. Unity 2017.2 also integrated Autodesk's 3DS Max and Maya tools into the Unity engine for a streamlined asset sharing in-game iteration process.

Unity 2018 featured the Scriptable Render Pipeline for developers to create high-end graphics. This included the High-Definition Rendering Pipeline for console and PC experiences, and the Lightweight Rendering Pipeline for mobile, virtual reality, and augmented reality. Unity 2018 also included machine learning tools, such as Imitation Learning, whereby games learn from real player habits, support for Magic Leap, and templates for new developers. The C# source code of Unity was published under a "reference-only" license in March 2018, which prohibits reuse and modification.

As of 2020, software built with Unity's game engine was running on more than 1.5 billion devices. According to Unity, apps made with their game engine account for 50 percent of all mobile games, and are downloaded more than 3 billion times per month, and approximately 15,000 new projects are started daily with its software. Financial Times reported that Unity's engine "powers some of the world's most lucrative mobile games", such as Pokémon Go and Activision's Call of Duty Mobile. In June 2020, Unity introduced the Mixed and Augmented Reality Studio (MARS), which provides developers with additional functionality for rules-based generation of augmented reality (AR) applications.Unity released Unity Forma, an automotive and retail solution tool, on December 9, 2020. Unity acquired Finger Food Advanced Technology Group in 2020, as it aimed to bolster its non-video game uses and offer additional design help to customers. The company went public in September 2020, to further expand use of its game engine into industries outside of gaming.

Unity 2021 brought multiple new features such as Bolt, Unity's Visual Scripting system, a new multiplayer library to support multiplayer games, improved Il2cpp runtime performance, Volumetric clouds for the High Definition Render pipeline. Shadow caching and Screen Space Global Illumination for HDRP. For the Universal Render Pipeline it added new features such as point light shadows, Deferred renderer and general core engine improvements and fixes.

For the year 2022, Unity has come up with some upgrades which involves speed integration to enter play mode and import files, visual search queries and multi-selection in the package manager. For 2D creators, they've focused on accelerating foundations, import, animation, and physics. They have added Sprite Atlas v2, support for PSD extension files and layer management in the 2D PSD Importer, and Delaunay tessellation for 2D physics.

UNITY ASSETS STORE:

The Unity Asset Store is a growing library of Assets. Both Unity Technologies and members of the community create these Assets and publish them to the store. There are various types of Assets in the store ranging from textures, animations and models to entire Project examples, tutorials and Editor extensions. There is a mix of free and affordable commercial Assets that you can download directly into your Unity Project. You can become a publisher on the Asset store and sell your Unity creations.

You can visit the Unity Asset Store in a couple of ways: either visit the website, or through the Unity Game Engine. To access the store via the Game Engine, open your Project and go to Window > Asset Store.

Note: It is faster and easier to download and import assets through the Unity Engine. However, there are more assets available on the website version.

What is a Unity Asset?

A Unity asset is an item that you can use in your game or Project. An asset may come from a file created outside of Unity, such as a 3D model, an audio file, an image, or any of the other types of file that Unity supports. There are also some asset types that you can create within Unity, such as an Animator Controller, an Audio Mixer or a Render Texture.

The Asset store is sorted into the different types of assets available. Here's a breakdown of what each asset type is:

3D assets

The 3D assets section includes vehicles, characters, props, vegetation and animations. Unity's humanoid animation retargeting means you can mix and match characters and animations from various sources.

Top rated 3d assets:

1. Nature Starter Kit 2 contains trees and bushes compatible with the built-in tree generator, so you can easily create all kinds of new variations of your own. Edit the shapes and colors of the plants right inside Unity!

2. Simple Town - Cartoon Assets is a collection of content that's suitable for nearly every purpose imaginable. It's everything you need to prototype, and can be used for any type of game genre.

3. FPS Weapons is a high quality collection of animated first person weapons. There are over 34 weapon types to choose from, each with fire, reload, hide and ready animations, plus matching sound effects. It also includes two characters with 3rd person animations, and a simple demo scene is set up to test the weapons out of the box.

2D assets

The 2D assets section includes sprites, textures, characters, environments, fonts, materials and UI elements.

Top rated 2d assets:

1. Skybox Volume 2 (Nebula) is a package of various high quality, beautiful nebulae of different colors and planets that's perfect for any space genre project.

2. Mighty Heroes (Rogue) 2D Fantasy Characters Pack is a project containing 6 characters with Unity animations. And lots of clothes and weapons for customization and creation of new characters.

3. 2D Forest Pack is a high quality, beautiful forest scene that comes with 120 recomposable sprites, 37 prefabs, 4 particle presets and a demo scene. This asset will make your scenes look

amazing.

Add-ons

Adds-ons are more advanced features you can import into your Project. Browse here for features such as Unity Ads, analytics, and in-app purchases.

Audio

Sound design is an important component to create an immersive and emotionally-charged game experience. Finding the perfect music to go with your game isn't easy, but you don't have to make it from scratch. Audio has a library of sound files that you can use to enrich the user experience of your project.

The Unity Asset Store has a variety of free and affordable audio assets, including ambient, music, and sound effects, so it's easy to find exactly what you're looking for.

Top rated audio:

1. Casual Game SFX Pack is a collection of free, original hand-crafted one-shot sound effects. It contains essential audio material covering most events of any casual game, from short blips for bonuses, juicy explosion sounds, tight snappy clicks for tile removal, ticking clocks for depicting the level running out of time, etc.

2. Universal Sound FX has over 5,000 sound effects you can use across any game genre.

3. Absolutely Free Music is a free collection of music in a variety of styles that you can use in all your projects. The package contains anything from trance, orchestral, instrumental, rock, symphorock, etc.

Templates
The Templates section allows you to download various tutorials and starter packs, a great section especially for beginners.

Top rated templates:

1. Tanks Multiplayer is an action packed multiplayer template running on Unity Networking or Photon Unity Networking. Play in one of four different teams with up to 12 players per room and compete for the highest team deathmatch score.

2. Corgi Engine - 2D + 2.5D Platformer is a complete platformer solution for Unity. It's a tight character controller for your game. It's very fast and works on desktop, mobile, and anywhere you want. This is the perfect asset to create the 2D + 2.5D platformer or run & gun game you want!

3. Top Down 2D RPG Kit lets you create your own top down 2D RPG with 100+ well documented, modular and fully customizable scripts.

Tools

Browse here for useful tools to help you create your project. There is a wide spectrum of options for your project needs, anything from AI to Visual Scripting.

Top rated tools:

1. GAIA is an all in one terrain and scene generation system for both artists and programmers. This 5-star tool allows you to easily create environments in minutes by stamping textures, mountains, rivers, trees, buildings and props, skies and lighting.

2. Final IK is a collection of Inverse Kinematics solutions which help provide beautiful motion for both humanoid and non-humanoid models in your projects.

3. Easy Roads 3D is a powerful road terrain generator which lets you create unique road networks directly in Unity based on your own models.

VFX
Visually enhance your Unity project with visual effects. These include particle effects and shaders.

Top rated VFX assets:

1. Unity Particle Pack is a set of sample particle assets to use in your games, and to help you understand how to achieve certain effects using the Unity Particle System component and modules.

2. Post Processing Stack is an über effect that combines a complete set of image effects into a single post-process pipeline.

3. Realistic Effects Pack 4 provides a diverse array of spectacular particle effects to bring your project to life.

Navigating the Unity Asset Store

You can access the Unity Asset Store through the website or the Unity Game Engine. To access in the Unity Game Engine, open your Project in Unity and go to Window > Asset Store. This opens a new tab in the Editor.

The steps below will help guide you through your visit to the Asset Store website.

Scroll down on the main page to find recommended assets, new assets, popular assets, and others, in clearly labelled sections.

Click the Plus/Pro button near the top of the page to only show assets marked 20% off for Plus or Pro subscribers.

Click the Impressive New Assets button to browse some recently created assets selected by the Asset Store Content Curators.

Click the folder icon to view the assets you have already purchased.

Each asset has a heart icon you can click on to save it as a favourite. Click the heart symbol at the top of the page to view the assets you have saved.

Click the bell symbol to see your notification feed. It contains useful information about packages you interact with.

Click the cart symbol to view the assets you have bookmarked to buy.

There are categories at the top of the page. Click on one of these categories to filter the assets.

Use the search bar to search through every asset in the Asset store. Click the All Assets dropdown to view the Asset categories. Click on one of the categories to narrow down your search.

A page like this appears after you click on one of the categories:

Click on one of the categories at the left to further narrow-down your search.

Also, you can use the dropdown menus under the banner to filter your search. For example, use the Price menu to look for assets within your price range by using the slider.

Use the Sort By dropdown menu to change the order of the Assets according to your preferences.

Mixamo

Mixamo is a 3D computer graphics technology company. Based in San Francisco, the company develops and sells web-based services for 3D character animation. Mixamo's technologies use machine learning methods to automate the steps of the character animation process, including 3D modeling to rigging and 3D animation.

Mixamo is spun-off of Stanford University and has raised over $11 million from investors Granite Ventures, Keynote Ventures, Stanford University and AMD Ventures. Mixamo was acquired by Adobe Systems in 2015.

Mixamo was founded in 2008 by Stefano Corazza and Nazim Kareemi as a spin-off of Stanford University's Biomotion Lab, and started out as a cloud-based service offering animations and automatic character rigging. It launched its first online animation service in 2009. In 2010, Mixamo worked with Evolver to provide characters to customers. Later Autodesk acquired Evolver and made it proprietary, but Mixamo had already begun work on its own character creation service.

Mixamo released its automatic rigging service in 2011. That was followed by the launch of its real-time facial animation product, Face Plus, in 2013, and the official launch of its Fuse 3D character creator software in March 2014. In August 2014, Mixamo launched a new pricing structure.Mixamo was acquired by Adobe Systems on June 1, 2015

3D Character Animation

Mixamo's online services include an animation store featuring downloadable 3D models and animation sequences. The animations were created at Mixamo using motion capture and cleaned up by key frame animators. All its animations work with characters created in Fuse and/or rigged with Mixamo's AutoRigger.

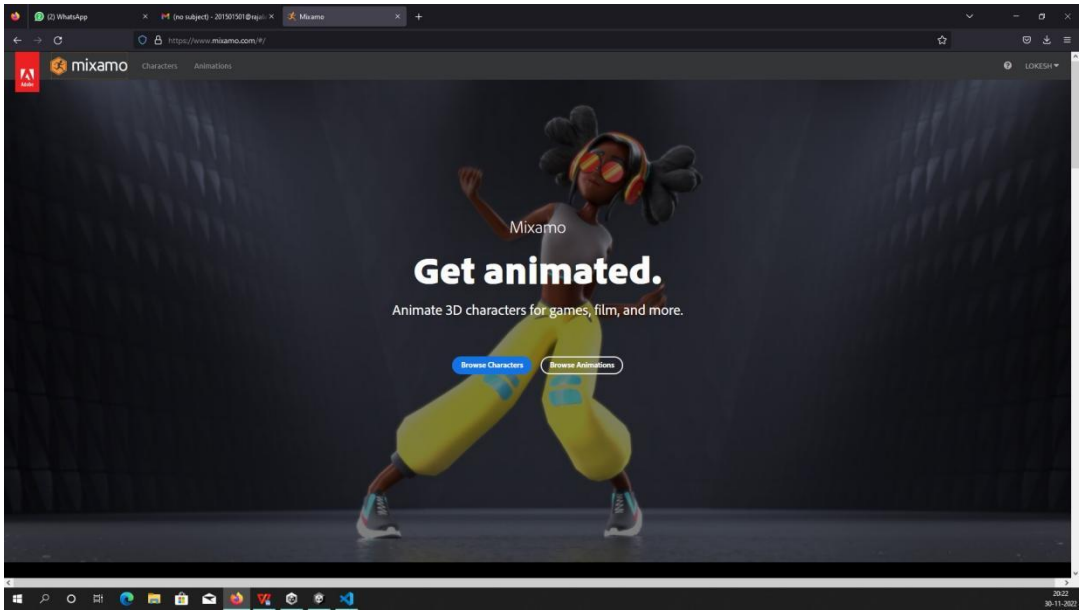
Real-time facial animation

August 2013, Mixamo released Face Plus, a game development tool that allows users to record facial animation data of themselves using a standard webcam and apply the animation to a character inside the Unity game engine in real-time. Face Plus was briefly included in the keynote presentation at the Unite conference in Vancouver. The technology was developed in collaboration with AMD. and uses GPU acceleration.

The animated short "Unplugged" was created using Face Plus technology and as of April 2014 has won several awards.

Face Plus has been discontinued and according to a September 2017 Adobe Forum post by Adobe staff JMathews, no further work on the product was being contemplated at that time

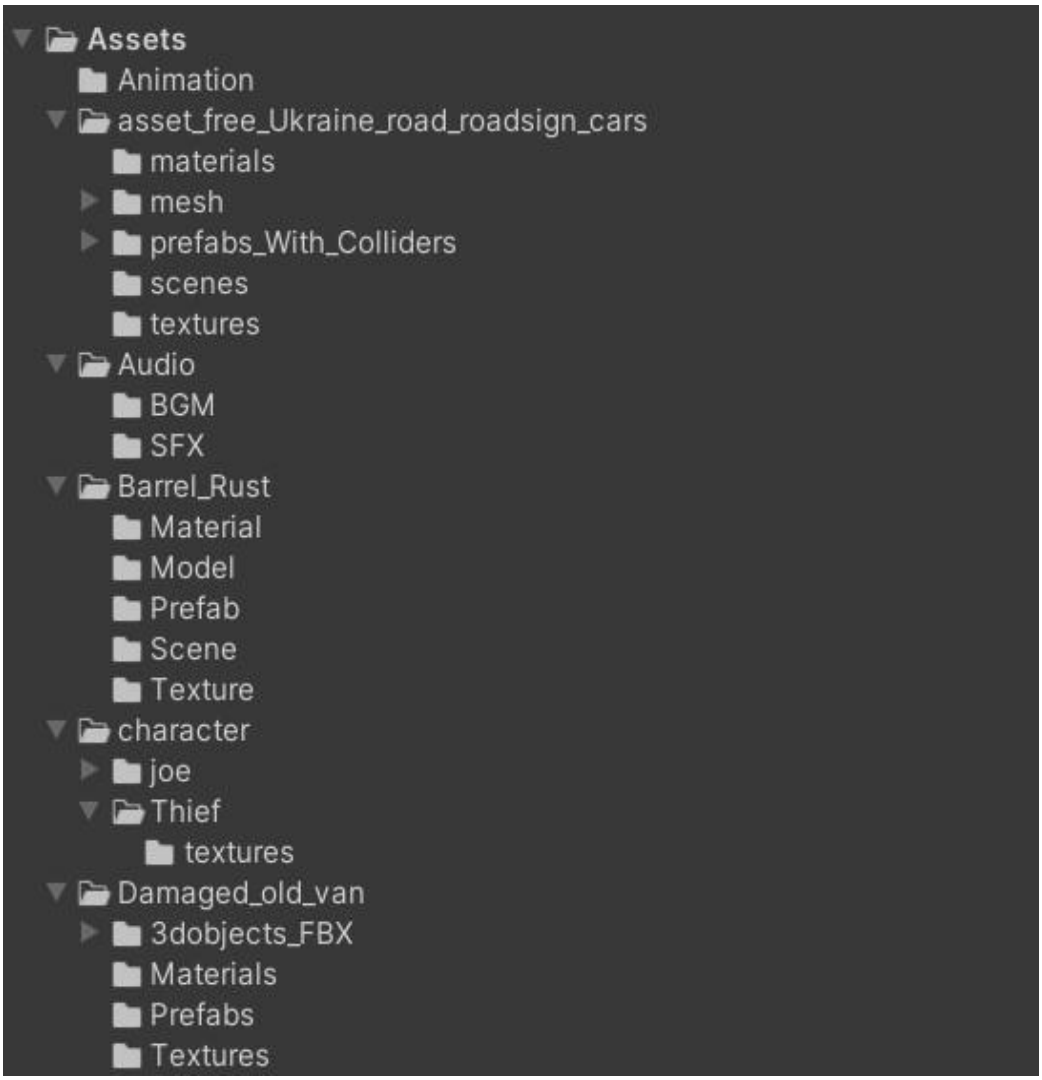
https://www.mixamo.com/#/

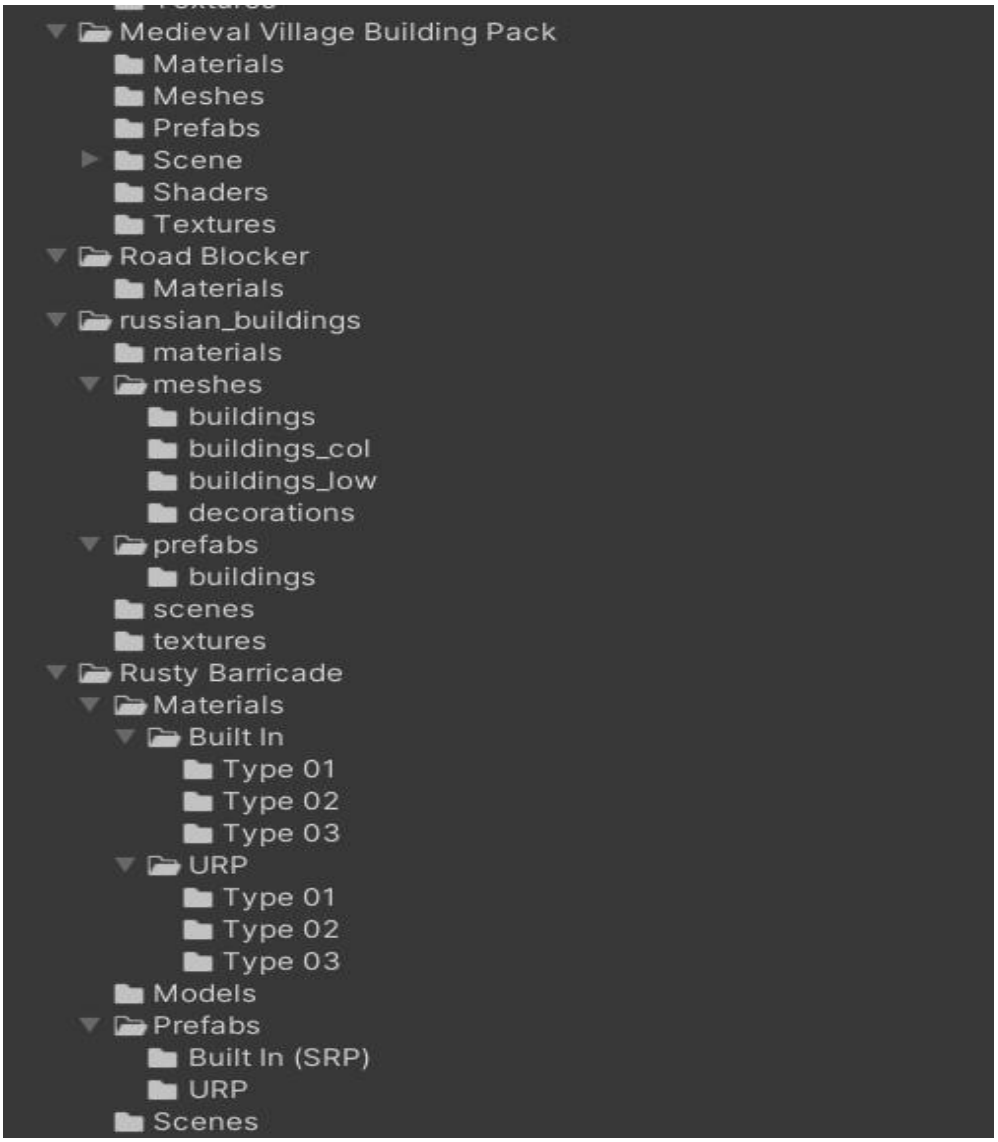


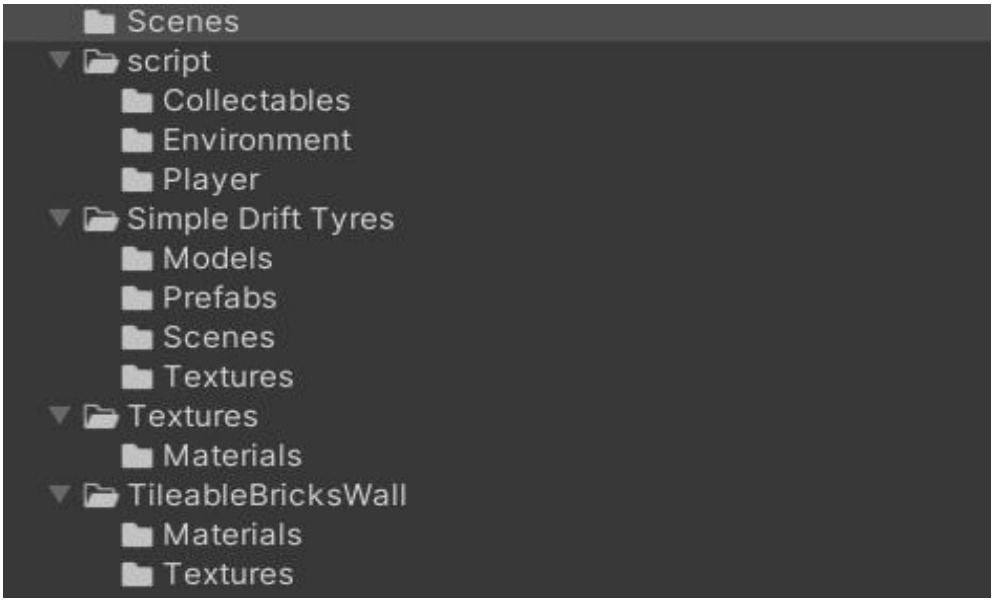
CHAPTER 5
IMPLEMENTATION and ALGORITHM

Game Assets

The game's visual experiences to the players are often achieved through the game assets. The game assets are often divided into art assets and script assets. The game art assets refer to the game models, model textures, game background music and etc. These art assets make up the in-game objects and user interfaces. Instead, scripts assets are the core parts of the game. These assets are programmed with the code which is responsible for the administration of the gameplay and rules of the game. Also, clouds and mountains have different speed. Moreover, the script assets are essential to the game project as well. The scripts are the key to make the gameplay and performance in the right way. During this project, I will use C# to create the scripts.

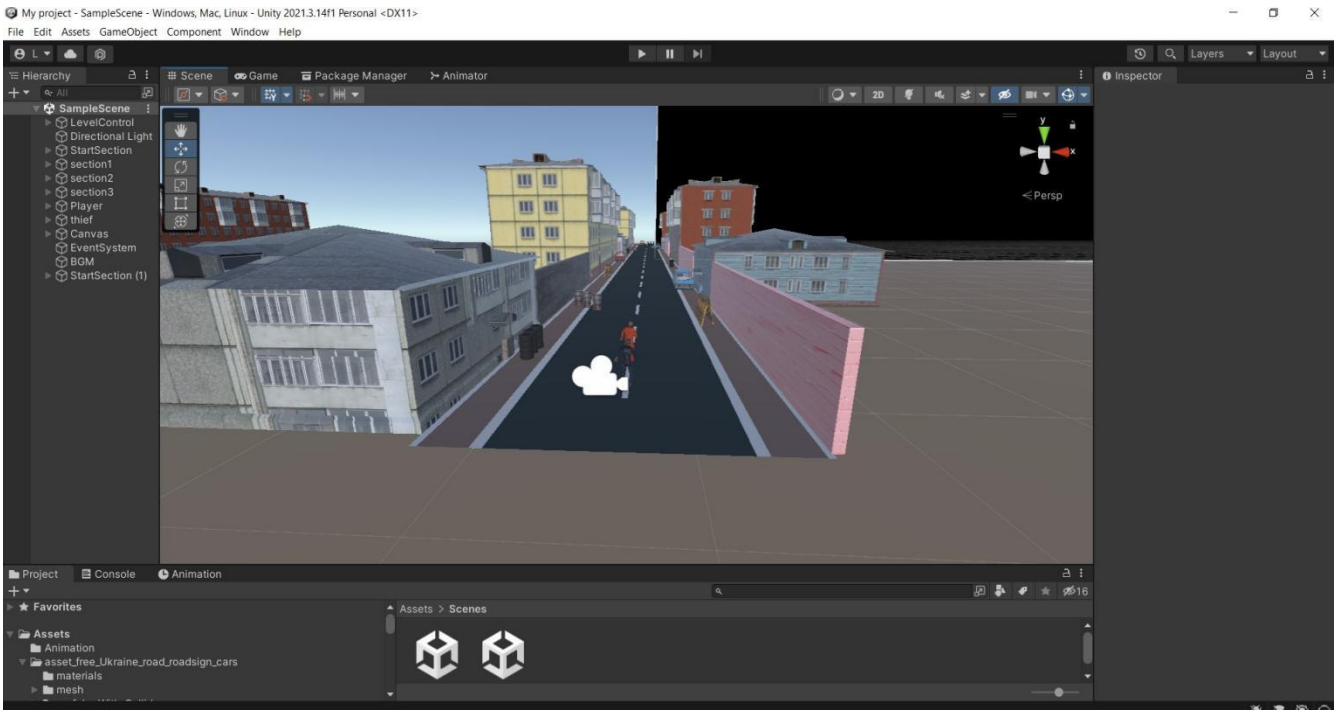


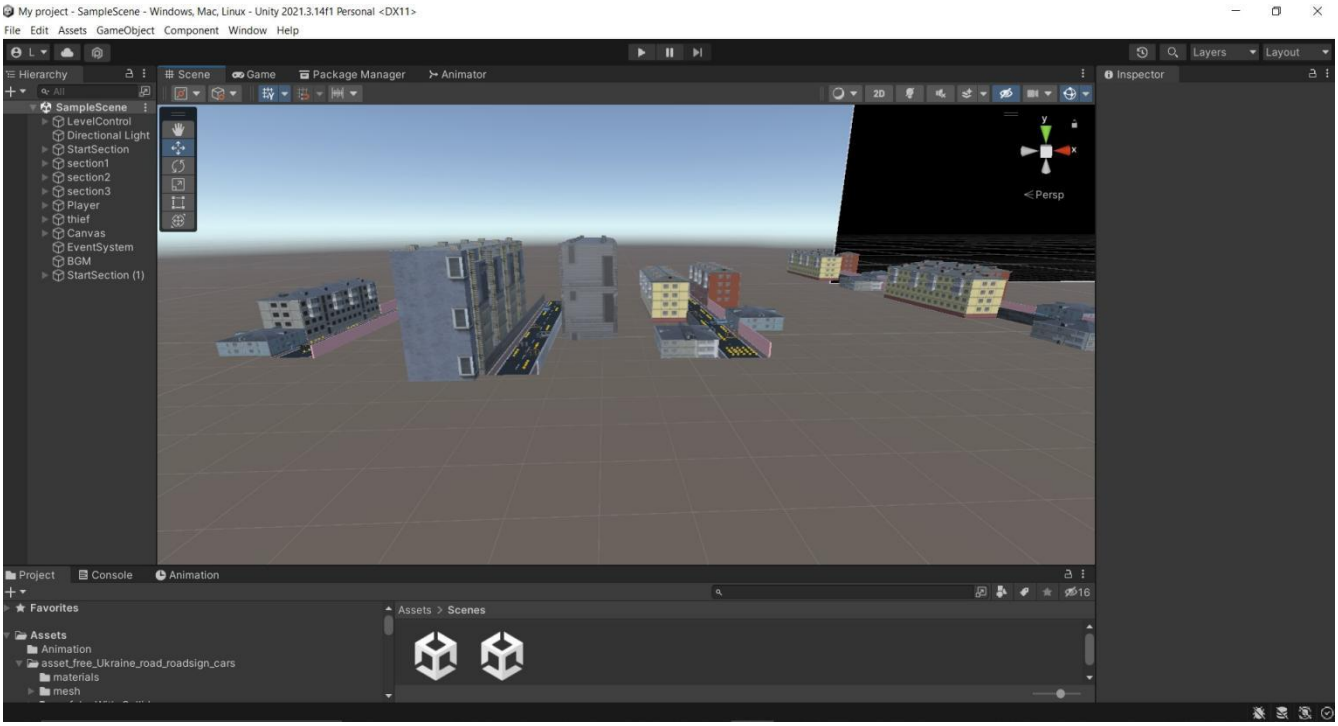

Medieval Village Building Pack
Materials
Meshes
Prefabs
Scene
Shaders
Textures
Road Blocker
Materials
russian_buildings
materials
meshes
buildings
buildings_col
buildings_low
decorations
prefabs
buildings
scenes
textures
Rusty Barricade
Materials
Built In
Type 01
Type 02
Type 03
URP
Type 01
Type 02
Type 03
Models
Prefabs
Built In (SRP)
URP
Scenes


Scenes
script
Collectables
Environment
Player
Simple Drift Tyres
Models
Prefabs
Scenes
Textures
Textures
Materials
TileableBricksWall
Materials
Textures

Starting with the ground

Starting the game stage, we first need to create the basic ground to let the main character walk on it. First, I need to edit these art assets of the grounds, so that it can fit into the game. One of the most important things is to add a 2D collider to the ground, so that the ground can contact with the main character. Also, the 2D collider can let the character stand on the art assets in the physical environment. In this project, all the settings of the game asset are edited in the inspector section. This section allows the user to change the initial values of the object, which contains the basic values such as Transform and Renderer, and we can add new components later. Also, the inspector allows the user to change the values in the game operating mode. The ground assets do not need editing its transform, but for the 2D collider I have changed its offset and the collider size to let the collider fit into the game assets, The basic settings of the ground prefab Also, we need to create the prefabs of the grounds for later use. The main object of using prefabs is to allow the game objects and resources to be reused during the game process. Prefabs can improve resource utilization and efficiency of the development in this project. Code 1 shows this process. C ODE 1. Generating the prefabs from the Resources folder.21 At this point, the ground assets can be made into a complete road. Each of the ground with 2D collider can allow the character to runn on it. In all the endless running games, the game scene often moves in a horizontal or vertical way. With the movement of the scene, the level of the ground will change frequently. In this game, the ground will change the height and width randomly. Also, the width of the gap between the grounds are randomly different. First, the grounds and backgrounds need to be in motion. Since the game is in two dimensional perspective, we only need to configure the axes X and Y. Also, the game is scrolled in horizontal movement and then we should only configure the Transform position of X axes.. The middle ground will be generated during the process, and the number of the middle grounds should be in a random range, so that the game will not look so boring. Moreover, if the last ground is the right ground, it means that the blank ground will be the next generated ground. Then, the scene needs to set the counts of the blank ground in a random range





GenerateLevel.cs

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

public class GenerateLevel : MonoBehaviour
{

    public GameObject[] section;
    public int zPos = 79;
    public bool creatingSection = false;
    public int secNum;

    void Update()
    {
        if (creatingSection == false)
        {
            creatingSection = true;
            StartCoroutine(GenerateSection());
        }
    }

    IEnumerator GenerateSection()
    {
```

```
        secNum = Random.Range(0,3);
        Instantiate(section[secNum], new Vector3(0,0,zPos), Quaternion.identity);
        zPos +=79;
        yield return new WaitForSeconds(2);
        creatingSection = false;
    }
}
```

LevelBoundary.cs

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

public class LevelBoundary : MonoBehaviour
{
    public static float leftSide = -5.6f;
    public static float rightSide = 0f;
    public float internalLeft;
    public float internalRight;

    void Update()
    {
        internalLeft = leftSide;
        internalRight = rightSide;
    }
}
```

PlayerMove.cs

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

public class PlayerMove : MonoBehaviour
{
    public float moveSpeed = 10;
    public float leftRightSpeed = 6;

    void Update()
    {
        transform.Translate(Vector3.forward * Time.deltaTime * moveSpeed, Space.World);

        if (Input.GetKey(KeyCode.A) || Input.GetKey(KeyCode.LeftArrow))
        {
            if (this.gameObject.transform.position.x > LevelBoundary.leftSide)
            {
                transform.Translate(Vector3.left * Time.deltaTime * leftRightSpeed);
            }
        }
```

```
        if (Input.GetKey(KeyCode.D) || Input.GetKey(KeyCode.RightArrow))
        {
            if (this.gameObject.transform.position.x < LevelBoundary.rightSide)
            {
                transform.Translate(Vector3.left * Time.deltaTime * leftRightSpeed * -1);
            }
        }
    }
}
```

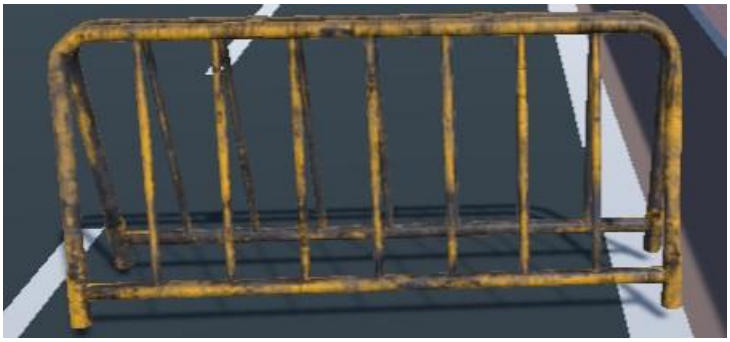
In-game objects

In the game, the character does not just jump through the gap to reach the other grounds. There are also enemies. During the game process, the character also needs to avoid the monsters on the road. The monster has the basic components such as Renderer and Collider2D. In addition, Collider 2d is set as a trigger so that the monster has the trigger events with the character. The trigger event code is in Code 6. When the character contacts the monster, the trigger will be activated. On the other hand, the monsters need to dynamically be generated in the game. The monsters should on the grounds and move together with the grounds. Also, only one monster will be generated in the platform in order to avoid the game being too difficult for players. Always one monster in the platform (the white box) The position of the monster on the grounds is generated randomly with the codes. Moreover, the monsters are always generating on the middle grounds to protect the game balance. Since if the monsters are generating on the first or the last ground, then the character cannot pass through the monster to reach the other side. Another very important in-game item is the reward coins. The character is not just earning points by moving forward to reach other grounds. Also, the reward coin can give a high score. The reward coins have some 28 very good bonus points to reach ciples compare d it and then basic printo the monster. But the coin will disappear when the character collected the right side. These features are scripted with it will be respawning frrom code in Code. generate t Therefore, he coins and the reward feature players need to control the character to get the coins as much as they can. At the same time, they also need to dodge the monsters to avoid death. All these in objects can make the game more flexible and fun.

Player

Obstacles

Barricades



Tires

Barrels

Old van

Coin

CollectCoin.cs

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

public class CollectCoin : MonoBehaviour
{
   public AudioSource coinFX;

   void OnTriggerEnter(Collider other)
   {
      coinFX.Play();
      CollectableControl.coinCount +=1;
      this.gameObject.SetActive(false);

   }

}
```

RotateObject.cs

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

public class RotateObject : MonoBehaviour
{
   public int rotateSpeed =1;

   void Update()
   {
      transform.Rotate(0, rotateSpeed , 0, Space.World);
   }
}
```

CollectableControl.cs

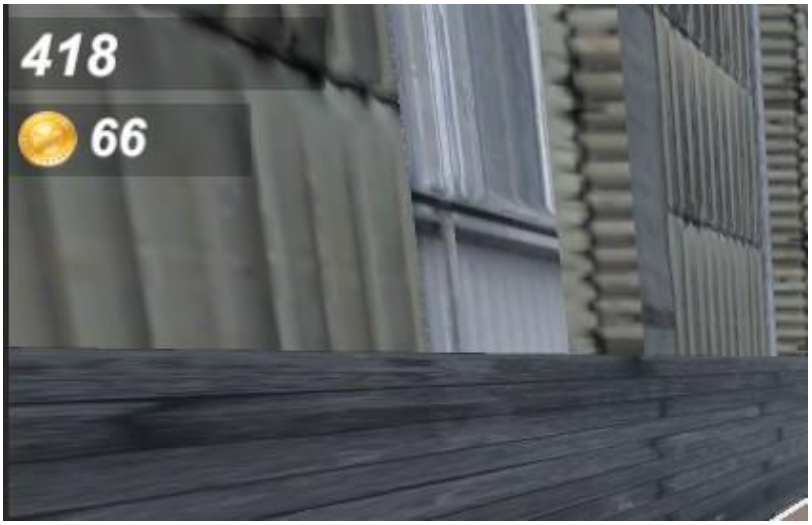
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class CollectableControl : MonoBehaviour
{
   public static int coinCount;
   public GameObject coinCountDisplay;

   void Update()
   {
      coinCountDisplay.GetComponent<Text>().text =""+ coinCount;
   }
}
```

High score

In this game , the records high Also, the score is to measure the distance that the player coins give bonus points score record database will replace the old C ODE 8. Replacing thgame has reached so far. . When the player reaches a new higher score, the high with the new score, as in Code 8. e old score to a new high score29 In this game, it is necessary to use GUI to display the score and high score on the platform. The GUI is implemented with codes and the basic interface should also be edited in the script. In Code 9, there are two parts of the score, including the current score and the high score. The current score is to tracking the current score that player achieved during the game process. The high score is to show the highest score so far. CODE 9. GUI settings in script When the game starts, the current score will start counting the points that the character achieved. Also, the high score record will keep the previous high score until a new one replaces it.



LevelDistance.cs

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

public class LevelDistance : MonoBehaviour
{

    public GameObject disDisplay;
    public int disRun;
    public bool addingDis = false;
    public float disDelay =0.2f;

    void Update()
    {
        if (addingDis == false)
        {
            addingDis = true;
            StartCoroutine(AddingDis());
        }

    }
    IEnumerator AddingDis()
    {
        disRun +=1;
        disDisplay.GetComponent<Text>().text ="" + disRun;
        yield return new WaitForSeconds(disDelay);
        addingDis = false;
    }
}
```

CHAPTER 6

RESULTS AND DISCUSSIONS

During this project, all the basic features were achieved completely endless running game on a PC platform. All the game assets were created by myself with Photoshop software. In addition, all the assets in the game have some components to contact with other objects in the game. For the game play, the game player can control the character to jump and reach other grounds by right-clicking the mouse. The game points are increasing when the game goes on. Also, there are coins that can give bonus points and the monsters on the grounds that the player should dodge. The character dies, if he touches the monster. Then the score will be settled. If the new score is higher than the score record, the previous high score will be replaced by the new one. The main goal in this game is to get more points and break the old high score records. However, the way of running this game out of the Unity 3D engine is to run the exe file which is created from the Build & Setting tool.

OUTPUT SCREENSHOTS:


13
0
3


22
0
2

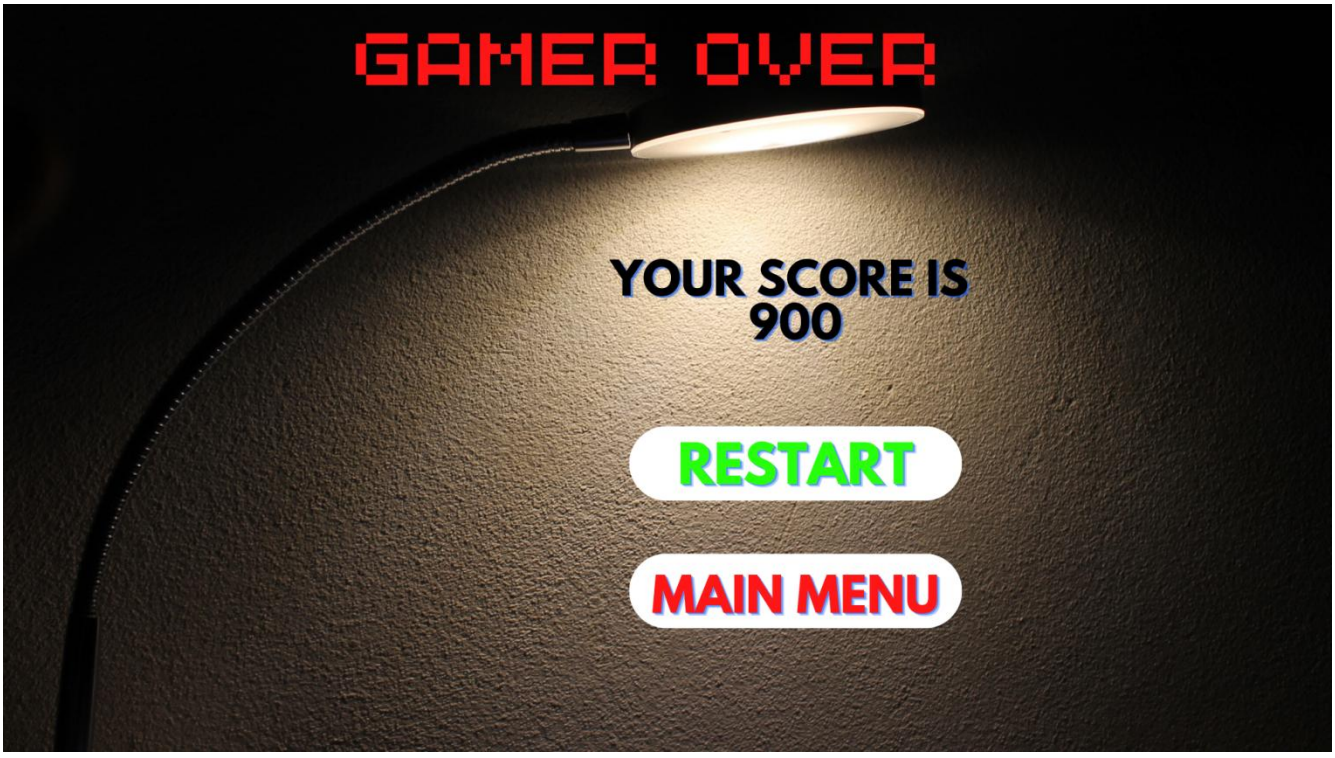

36
0
1


46
0
Runn..


7
0


619
136


786
178


GAMER OVER
YOUR SCORE IS
900
RESTART
MAIN MENU

CHAPTER 7
CONCLUSION

Developing AI techniques that can deal with the complexity of computer games is a big challenge, but has the potential to have a big impact in several areas including entertainment, education and training. The digital game industry is experiencing a shift to persistent games, business models emphasizing game ecosystems, more support for game communities, and new considerations for incorporating real world context into game worlds. We see these advances as positive signs of growth and maturity in the game industry. We also see these advances pushing the bounds on the scalability of game development practices. Artificial intelligence, computational intelligence, and machine learning have always excelled at addressing problems of scalability by automating tasks and dynamically adapting system behavior. Game AI has always been an integral part of computer game development. AI Actors have enhanced player experiences by supporting players' suspension of disbelief and dynamically managing dramatic contexts. AI Designers have supported and augmented the development of individual games through procedural content generation.

We believe that computer game AI will be the next revolution in the gaming industry. After the impressive advances in the audiovisual presentation and the networking capabilities, the next step in computer games is to incorporate advanced AI techniques that can achieve the goal of having truly adaptive games, increasing the level of believability and immersion. To achieve this goal, the gaming community needs new techniques, approaches and tools that allow them to easily specify, develop, and incorporate AI in their games.

The aim of the study was to create an 2D endless running game with the Unity 3D game engine. During this project, I gained a deeper understanding of the features of the Unity3D game engine and fundamental knowledge of game programming. Also, the game engine is an important tool for the game designer to start building a game. The Unity 3D game engine is a powerful game engine and suitable for beginners. But, to become a better game designer, there are many more functions of this game engine to be discovered. The theory backgrounds helped to understand the environment of programming and game engines. The game designer will become more familiar with the work environment. In this game, there are two scenes when the game starts. The game start scene is the in-game main menu. The player will switch to the gameplay scene from this scene. During this project, every step of building this game was fairly successful. The gameplay is working very smoothly. The in-game objects are doing their duties properly during the game process. On the other hand, there are still many features that can be added into this game; more in-game objects and game level design are necessary to make the game more interesting. Also, the current in-game objects and the main character can 32 add more features, such as more frames for the animation of the character to make it look more natural. This project still has a very large development space. The main objective of this project was to demonstrate the process of create the 2D game with the Unity 3D game engine. There were few problems and difficulties during the implementation of the project, such as the style of

programming in the new version of the Unity 3D game engine which had some changes. The game assets also involved some small issues. However, these problems and difficulties were solved during the project. By going through this project, I have improved my knowledge and skill of working with the Unity 3D game engine. Also, I consolidate my knowledge of programming with the C# language. For further development, there are still a lot of elements can be added into this game. Lack of elements can make players lose interest fast. There are many details not taken into consideration. None optimization was made for the code in this project. All the work was done only based on what I had learned from the websites and books. With more in-depth study of programming with the C# language and the environment of the Unity 3D game engine, this game could be better. The most important thing of designing this game is to discover how to develop a game and imporve it. There might be more things to add into this project in the further development. During this project I have learned how to work independently and solve all the problems and difficulties on my own. This project illustrated the whole process of making an endless running game with the Unity 3D game engine and C# language.

REFERENCES

- Fischer, Fabian. 2014. Criteria for Strategy Game Design. WWW-doucument. http://www.gamasutra.com/blogs/FabianFischer/20141201/231243/Criteria_for_Strategy_Game_
- Design.php. Updated 12.1.2014. Referred 10.4. 2016. Francis, Chris. 2014. The important of
- Gameplay Balance. WWW-doucument. http://www.theoryofgaming.com/importance-gameplay-
- balance/. Updated 14.5.2014. Referred 16.4. 2016. Masters, Mark. 2014. Unity, Source 2, Unreal Engine 4, or CryENGINE - Which Game Engine Should I Choose?. WWW-doucument. http://blog.digitaltutors.com/unity-udkcryengine-game-engine-choose/. Referred 10.4. 2016.
- Hiscott, Rebecca. 2014. 10 Programming Languages You Should Learn Right Now. WWW-doucument. http://mashable.com/2014/01/21/learn-programming-languages/#gl6J3bZOskqL.
- Updated 21.1.2014. Referred 16.4. 2016. Geldonyetich. 2012. MMO Game Balance. WWW-doucument. http://geldonsgaming.blogspot.fi/2012/07/the-impossible-goal-of-balanced-
- skyrim.html. Updated 18.7.2012. Referred 16.4. 2016. Unity Documentation, 2016. Unity Manual, Working in Unity, Creating Gameplay, Scenes. WWW-doucument.
- http://docs.unity3d.com/Manual/CreatingScenes.html. Referred 12.4. 2016. Unity Documentation, 2016. Unity Manual, Working in Unity, Creating Gameplay, GameObjects. WWW-doucument.
- http://docs.unity3d.com/Manual/GameObjects.html. Referred 12.4. 2016. Unity Documentation, 2016. Unity Manual, Working in Unity, Creating Gameplay, Transform. WWW-doucument.
- http://docs.unity3d.com/Manual/Transforms.html. Referred 12.4. 2016. 34 Unity Documentation, 2016. Unity Manual, Physics, Physics Overview, Colliders. WWW-doucument.
- http://docs.unity3d.com/Manual/CollidersOverview.html. Referred 12.4. 2016. Unity Documentation, 2016. Unity Manual, Unity Graphics, Graphics Overview, Materials, Shaders &
- Textures. WWW-doucument. http://docs.unity3d.com/Manual/Shaders.html. Referred 12.4. 2016. Unity Documentation, 2016. Unity Manual, Animation, Animation Reference, Animation
- Controller, Animation States. WWW-doucument. http://docs.unity3d.com/Manual/class-
- State.html. Referred 12.4. 2016. Pearson, Charles. 2014. Learning NGUI for Unity. Birmingham: Packet Publishing. Alex, Parker. 2013. Mobile Game UI. WWW-doucument.

- **http://www.riaxe.com/downloads/game-asset/. Updated 21.10.2013. Referred 17.4. 2016. Wenderlich, Ray. 2012. Introduction to AI Programming for Games. WWWdoucument.**

- **https://www.raywenderlich.com/24824/introduction-to-ai-programmingfor-games. Updated 27.12.2012. Referred 17.4. 2016**