Through Ball: A Multi-platform Game

Name: Md Oliuzzaman
ID: 2011-1-60-012

Undergraduate Project Presentation
14th January 2016
CSE Dept., EWU

Supervised By
Md. Shamsujjoha
Senior Lecturer,
CSE Dept., EWU

Overview

- Introduction
  - Motivation & Contribution
- Basic Definition
  - Game Engine
- Demonstration
  - Implementation and Others
- Conclusion
  - Summary & Future Enhancement

CHAPTER ONE

INTRODUCTION

1.1 Introduction

As time goes the computer/Mobile technology upgraded massively and all the technology giants' example Microsoft, Apple, Google, and Linux started developing software (various application) to increase function and features along with the computer parts and hardware. In the past few years application development sector turns out to be billion dollar business both for the developer and for the company application they are developing. Although there are few people out there who develop application not considering to earn few bucks they do these because they are passionate about this. The last few years have seen an unprecedented number of people rushing to develop mobile apps for iOS and Android. But looking at the installed user base on each platform and information on the payouts made by the different companies.

The consensus around the industry is that Google dominates the mobile market with 900 million users, while Apple follows with 600 million iOS devices purchased, and Microsoft comes in third place with an estimated 12 million Windows Phones sold.

Apple, at its Worldwide Developer Conference, talked about 1.25 million apps in the app store accounting for 50 billion downloads and $5 billion paid off to developers in the last year. To the company, it is a sign of pride to be able to pay this developer community moreover, Microsoft, meanwhile, has been claiming 160,000 apps in their store from 45,000 developers.

1.2 Motivations

Considering all the platforms I always thinking to develop any application which is made can be compatible to any platforms then I came up with my game "Pong" Which is developed using C# ad can be compatible in all the popular platform. Moreover I always very much passionate about games and developing them in user friendly way, besides that this game development project I am sure will be great help to take it further way in future for my career to develop something massive.

1.3 Objective

Upon completion of this course I will be able to:

- Build my first commercial game.
- Share my ideas and thoughts will others.
- Represent myself as a game developer(for multi-platform)
- Increasing C# programing knowledge, learning visual basic in details knowledge about Unity3D.
- Enrich my portfolio for my ahead career which is certainly helpful to set-up a new path.

1.4 Contribution

To develop the game all the requirement are collected. Software and hardware and game framework requirements are also discussed in the chapter to initialize a multi-platform application development environment. After that move into our project to discuss about it in details.

- Gathering necessary information about Android, IOS, windows, Linux etc.
- Learning Unity3D programming techniques.
- Gathering requirement for this project.
- Testing the application and it passed in all the method applied.

1.5 Organization of the Project Report:

In this chapter I discuss all the background before get into the application development sector. In my whole project I mainly focus on mobile based application development and I bring some issue relating Android, iOS and windows.

Chapter 2

This chapter discuss about some fundamental background issue evolution of smartphones how it became massively popular how mobile application development came in so large in various platform.

Chapter 3

Chapter 3 discuss all about unity how it came in the market the evolution of Unity why I choose unity as my developing platform and benefits of using Unity and how it works.

Chapter 4:

Chapter four illustrates about the game I created using Unity features of it and it played.

Chapter 5:

Finally in chapter five I turned the conclusion and discuss my future plan to expand this project.

CHAPTER TWO

Literature Review: General Discussion about few Platform

2.1 What is iOS

iOS is Apple's proprietary mobile operating system (OS) for its handheld devices, such as the iPhone, iPad and iPod Touch. The operating system is based on the Macintosh OS X. Apple iOS was originally called the iPhone OS but was renamed in 2010 to reflect the operating system's evolving support for additional Apple devices. Apple's portable devices like iPhone, iPad or iPod are known as iDevices. iOS does things on these devices such as allow user to set the brightness of the screen if you can't see it on a bright sunny day, it also secures the iDevice, if enter a pass code the system will automatically lock the screen after a certain amount of minutes and letting user connect to the WiFi network. Most of the settings anyone want to customize and use are all contained in the settings button, or App. The iDevice operating system (iOS) has the features and runs all the apps you want to use. Each time Apple adds features to iOS they call it a software update and usually include a version number. Apple's iOS is currently the main software that runs on all models of the iPhone, iPod Touch, iPad and iPad Mini mobile devices and now the all new Apple Watch.

2.2 History of iOS

The term iOS was originally known as iPhone OS was introduced in 2007 along with the first iPhone hardware device Apple released. It was the term used to describe the software that would operate the iPhone and is derived from the term OS X which is how Apple describes its operating system for the Macintosh computers. The X stands for 10 which is the newest version of the computer Apple created to operate the desktop and laptop computers they design. The iOS platform is a mobile device based software system that works like computer system, but on mobile like portable phones. It is designed to be smaller faster and use less power. It is also touch friendly user interface so it works better when a finger is used to interface with the system instead of a mouse that been used in past to interact with operating system. The iPhone has run on iOS since its release in 2007.

2.3 Building and Designing iOS application

At the highest level, iOS acts as an intermediary between the underlying hardware and the apps you create. Apps do not talk to the underlying hardware directly. Instead, they communicate with the hardware through a set of well-defined system interfaces. These interfaces make it easy to write apps that work consistently on devices having different hardware capabilities.

The implementation of iOS technologies can be viewed as a set of layers, which are shown in Figure I-1. Lower layers contain fundamental services and technologies. Higher-level layers build upon the lower layers and provide more sophisticated services and technologies.

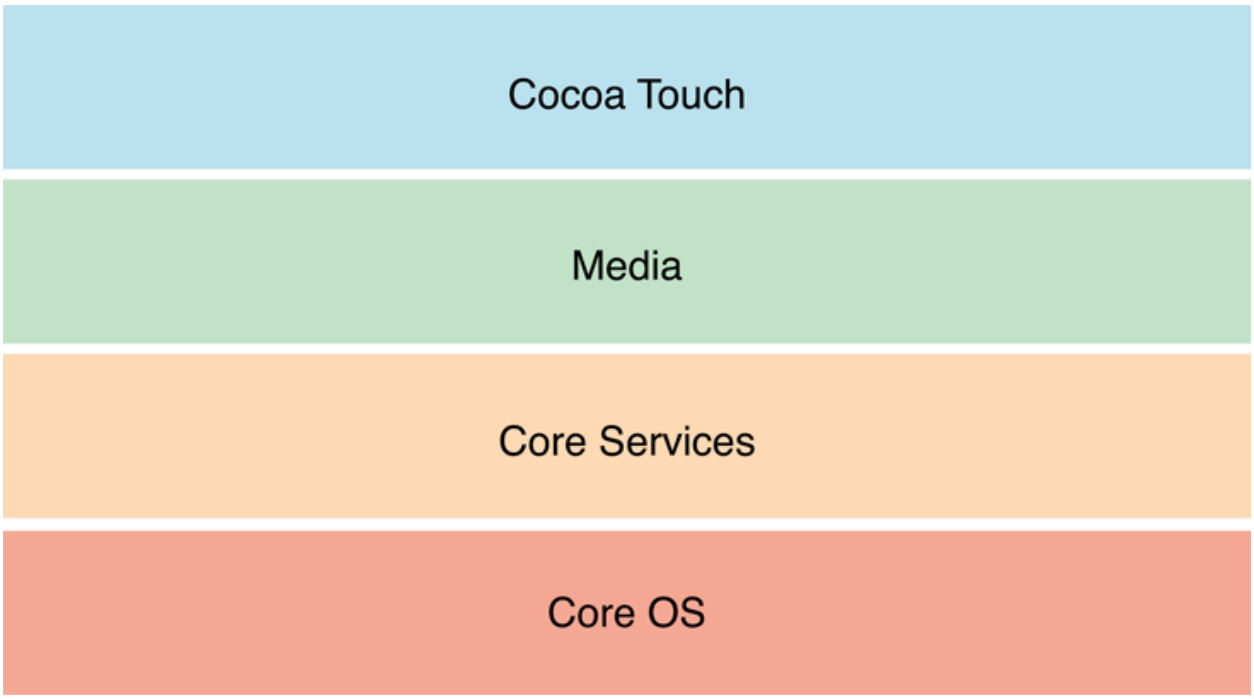


Figure 2.1: iOS Architecture

As developer write code, it is recommended that is prefer the use of higher-level frameworks over lower-level frameworks whenever possible. The higher-level frameworks are there to provide object-oriented abstractions for lower-level constructs. These abstractions generally make it much easier to write code because they reduce the amount of code you have to write and encapsulate potentially complex features, such as sockets and threads. You may use lower-level frameworks and technologies, too, if they contain features not exposed by the higher-level frameworks.

To develop on a device, have to sign up for Apple's paid iOS Developer program and then configure a device for development purposes. After sign up, getting a copy of Xcode and the iOS SDK at the iOS Dev Center.

2.4 iPhone SDK

The iPhone SDK is a software development kit that will allow third parties to create applications that can run directly on the iPhone and the iPod Touch. The kit is significant because Apple can't possibly anticipate, nor produce, all the applications that people might want to use on an iPhone.

And some of those applications will convince people who weren't sure about the iPhone to buy it. Anyone can download the SDK and develop an application, but you have to join Apple's iPhone Developer Program, and Apple is only accepting a "limited" number of applications at the moment. The application development process will be very similar to how applications are developed for Mac OS X.

Applications will be distributed through Apple's newly announced App Store, which will be built in to the iPhone but is also accessible through iTunes. Apple plans on personally approving every application destined for the iPhone.

The applications are wirelessly downloaded to the iPhone over either EDGE or Wi-Fi. Developers name the price of their applications themselves and get 70 percent of the revenue from sales of their apps; Apple gets 30 percent. Free applications will be listed for free on the App Store and iTunes. The SDK will only work on Macs.

2.5 Android OS

The Android OS is an open source operating system primarily used in mobile devices. Written primarily in Java and based on the Linux operating system, it was initially developed by Android Inc. and was eventually purchased by Google in 2005. The Android operating system is symbolized by a green colored Android Robot logo.

2.6 Google Play Store

Google Play, formerly known as the Android Market, is the official app store for Android smartphones and tablets. Google makes software applications, music, movies and books available for purchase and download through the store. The Google Play store, which comes pre-installed on Android devices, allows users to purchase, download and install applications from Google and third-party developers. Google Play is the only way to download and install applications on Android devices without changing the device's application setting to "allow the installation of non-market applications."

To combat the spread of malware through the Google Play store, Google introduced Bouncer in 2012, which scans apps for threats before making them available for download. Android's Google Play store has just officially reached over 1 million apps and it is now finally outgrown the Apple App Store and its 900 000 applications. That's a huge achievement for Android, the platform that launched as the underdog, after Apple first started the app revolution. Now, the tide has clearly changed. Google has made great advances in paying its developers more and they are now receiving 2.5x the money. App downloads have also grown to over 50 billion.

2.7 Android Architecture

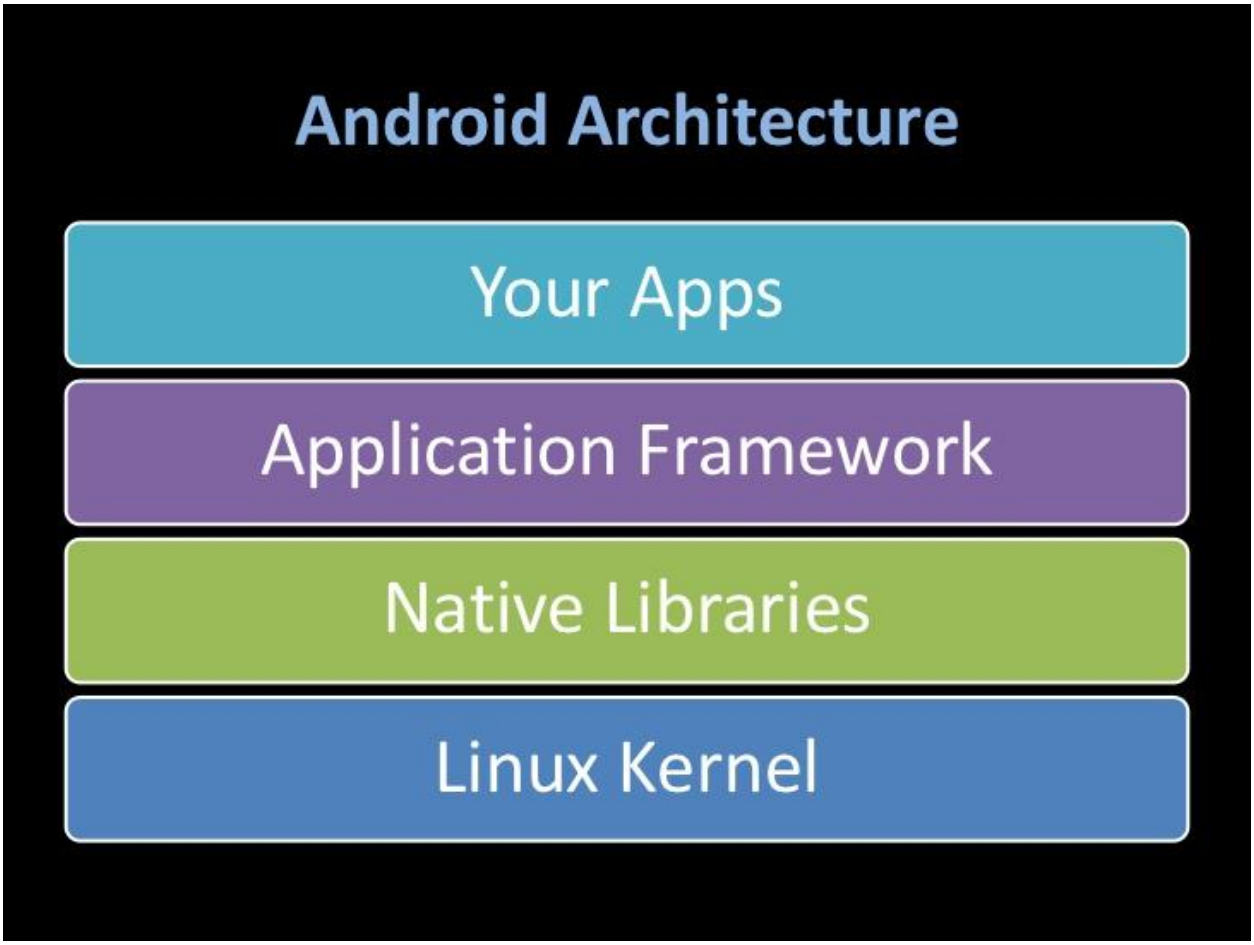


Figure 2.2: Android Architecture

2.8 Libraries

This layer holds the Android native libraries. These libraries are written in C/C++ and offer capabilities similar to the above layer, while sitting on top of the kernel.

2.9 Kernel

The Android OS is derived from Linux Kernel 2.6 and is actually created from Linux source, compiled for mobile devices. The memory management, process management etc. are mostly similar. The kernel acts as a Hardware Abstraction Layer between hardware and the Android software stack.

2.10 Android SDK

The Android SDK (software development kit) is a set of development tools used to develop applications for Android platform. The Android SDK includes the following:

- Required libraries
- Debugger
- An emulator
- Relevant documentation for the Android application program interfaces (APIs)
- Sample source code
- Tutorials for the Android OS

2.11 API

An application programming interface (API) is a software program that facilitates interaction with other software programs.

An API allows a programmer to interact with an application using a collection of callable functions. An API can be general or specific. The full set of a general API is bundled in the libraries of a programming language. With a specific API a specific term is meant to deal with a specific problem.

2.12 Integrated Development Environment

An Integrated Development Environment (IDE) is software that facilitates application development. In the context of .NET-based applications, Visual Studio is the most commonly used IDE. Some of the key features included are:

- Single IDE for all .NET applications. Therefore no switching required to other IDEs for developing .NET applications
- Single .NET solution for an application which has been built on code written in multiple languages
- Code editor supporting Intelligence and code refactoring
- Compilation from within the environment based on defined configuration options
- Integrated debugger that works at source and machine level
- Plug-in architecture that helps to add tools for domain specific languages
- Customizable environment to help the user to configure the IDE based on the required settings
- Browser that is built-in within the IDE helps to view content from internet such as help, source-code, etc. in online mode.

Figure 2.3: Android, iOS, Windows Phone

CHAPTER THREE

UNITY GAME ENGINE

3.1 Introduction

In the early 2000s, three young programmers without much money gathered in a basement and started coding what would become one of the most widely used pieces of software in the video game industry. A decade later, untold numbers of developers have used Unity3D to make thousands of video games for mobile devices, consoles, browsers, PCs, Macs, and even Linux. The existence of Unity3D and similar products helped democratize game development, making the kinds of tools used by the world's largest game companies available to developers at little or no cost. This has helped developers focus less on creating a video game's underlying technology and more on the artistic and creative processes that actually make games fun to play.

Unity is a cross-platform game engine with a built-in IDE developed by Unity Technologies. It is used to develop video games for web plugins, desktop platforms, consoles and mobile devices.

Unity Technologies is the creator of Unity, an intuitive and flexible development platform used to make wildly creative and intelligently interactive 3D and 2D content. The "author once, deploy everywhere" capability ensures developers can publish to all of the most popular platforms.

Unity Technologies boasts a thriving community of over 1.2 million registered developers including large publishers, indie studios, students and hobbyists. Unity Technologies aggressively re-invests in its award-winning 3D development tools and democratization initiatives, such as the Asset Store digital content marketplace and Union game distribution service, in order to remain at the forefront of innovation.

Unity Technologies is headquartered in San Francisco and has offices in Canada, China, Denmark, Lithuania, Sweden, the United Kingdom, Japan and Korea.

3.2 What Is the Unity Game Engine?

The Unity game engine has often been referred to as the best video game engine for under a million dollars. It was created by Unity Technologies in 2004 as a development tool for their game, *GooBall*. It was later launched in 2005 at Apple's Worldwide Developers Conference. Today, the Unity game engine flies under the banner of "democratizing game development and enabling everyone to create rich interactive 3D content" according to the Unity website. It is estimated that there are over 1.3 million registered Unity developers (do names like Cartoon Network, Coca-Cola, Disney, LEGO, or NASA ring a bell?) and that there are over 300,000 active developers monthly. A 2012 survey conducted by video game magazine Game Developer states that 53.1% of mobile developers reported using Unity to make games.

3.3 Why Unity?

As a game developer, you have a lot of options when it comes to choosing a game engine. The selection runs the spectrum from simple 2D engines to fully featured 3D powerhouses. Likewise, game engines' costs can range from free to millions of dollars. With so many options it can be difficult to choose the engine that is right for any project and for me I am more than happy to be using Unity Game Engine.

3.4 Pricing and Licenses

Many engines today come with extreme price tags or unintelligible payment plans. Many engines don't have any prices listed and long quote discussions must take place to determine actual figures. Unity doesn't do any of that and instead just comes in two basic flavors: Unity Free and Unity Pro. Unity Free is of course free (my favorite number) and is not some watered down, gimped version of the engine. It comes with all of the features you would need to make and sell games commercially. If you would like some of the more professional features like LOD Support, Path-finding, or IK Rigs, you can purchase the Unity Pro license for a mere $1,500. That may seem like a lot to an individual just getting into the field, but in the land of game engines (especially those packed with similar features) that amount is tiny. If you're not sure which license is right for you, the Unity Free license comes with a 30-day Pro trial, as well as a 30-day Android add-on trial.

3.5 Installation

Installation is a painless two-step process. First, download and run the Unity installer.

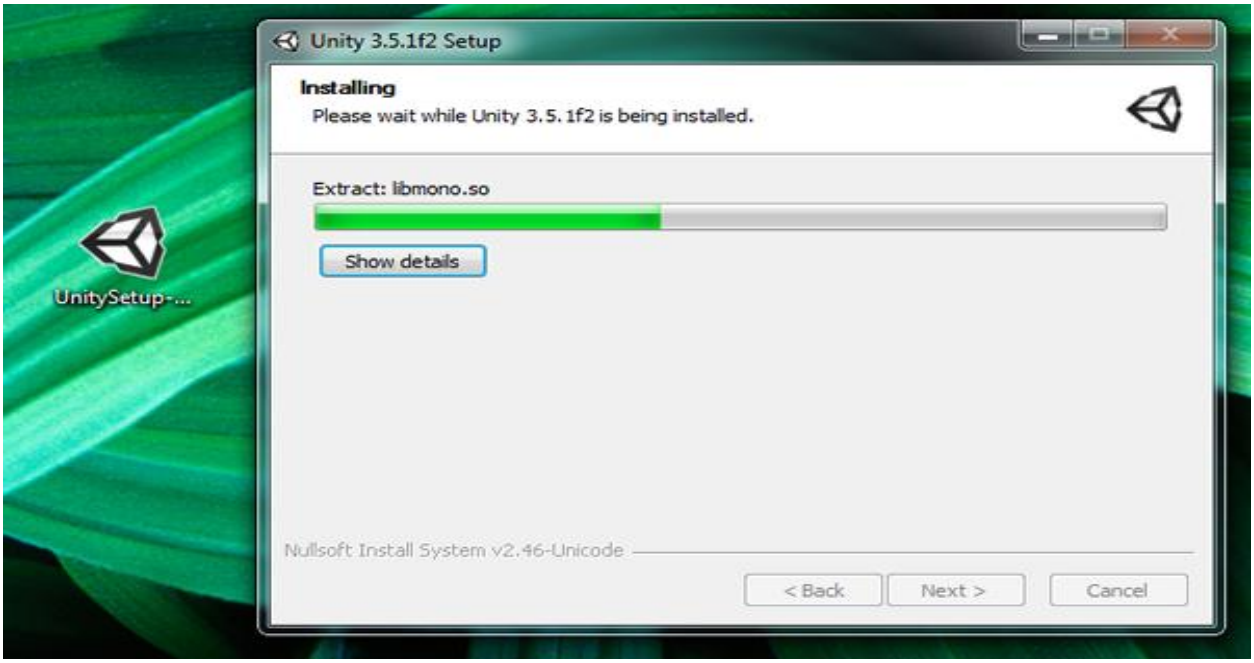


Figure 3.1: Unity Installation

Second, when you start Unity for the first time, it will open a web browser and prompt you to register using your email address. It will let you choose which version you want to run. You can select either the free version or a Pro trial that will fall back to the free version after 30 days.

3.6 The Application

The Unity application is a complete 3D environment, suitable for laying out levels, creating menus, doing animation, writing scripts, and organizing projects. The user interface is well organized and the panels can be fully customized by dragging and dropping.

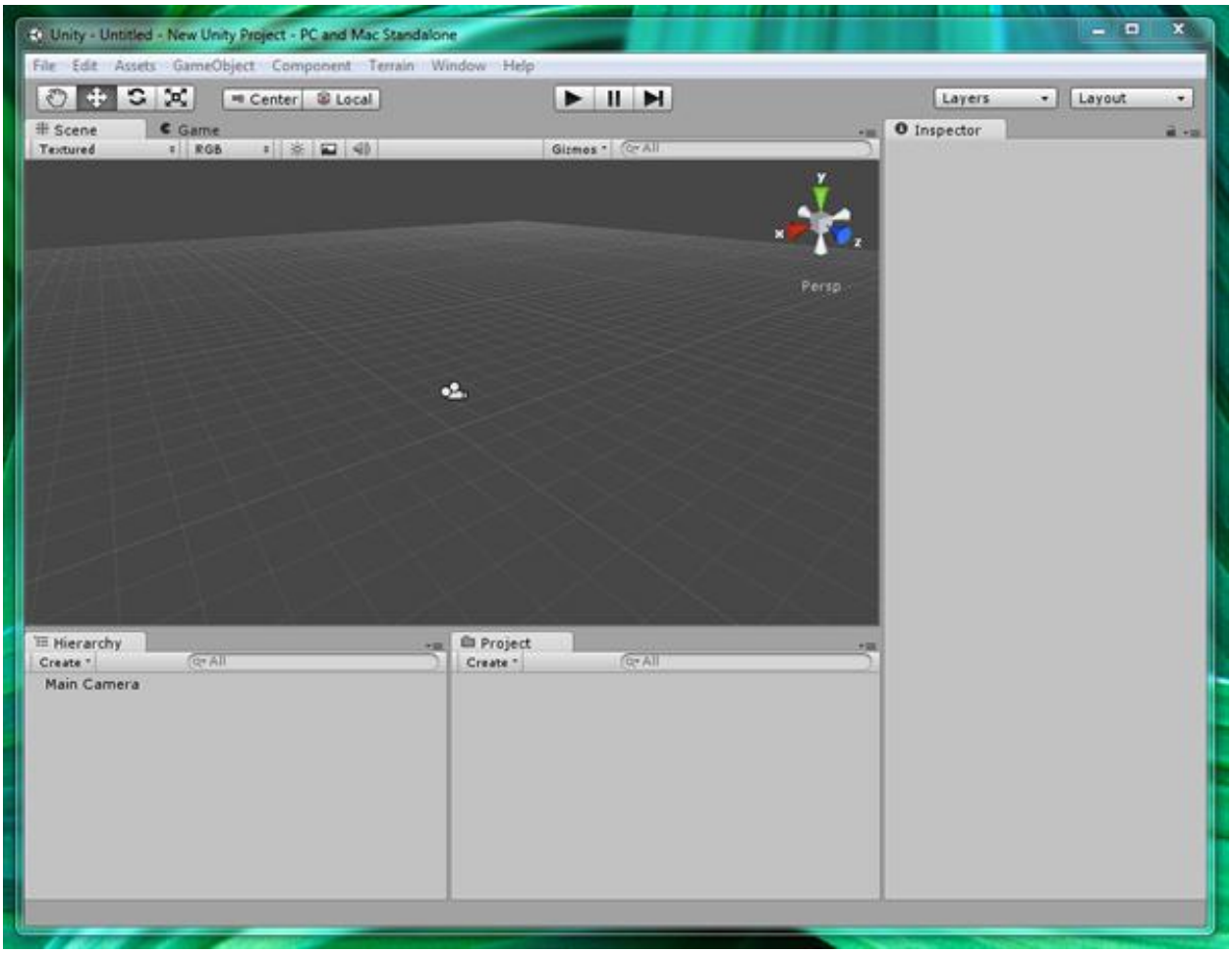


Figure 3.2: Unity3D interface

The Project panel is where all the assets within a project are stored. When assets are imported, they will first appear here.

The hierarchy panel is where assets are organized in a scene. Assets from the Project panel can be dragged into the Hierarchy panel to add them to the current scene.

The Inspector panel lets you inspect and adjust all the attributes of a selected asset. Everything from its position and rotation, to whether it's affected by gravity or able to cast a shadow.

The Scene panel is a 3D viewport where you can physically arrange assets by moving them around in 3D space. You can navigate the viewport by panning, rotating, and zooming the view. If you've used Maya at all, you should find these hotkeys familiar:

When it comes to running your game, it couldn't be simpler. Just press the play button. To stop it, press the play button again. You can even pause your game during play to inspect your scene.

3.7 Unity Projects

A Unity project is an ordinary folder containing every resource that belongs to your game. Creating a new project is a straightforward affair.

Click File > New Project

Click the Create New Project tab

Browse to a suitable folder

Click Create

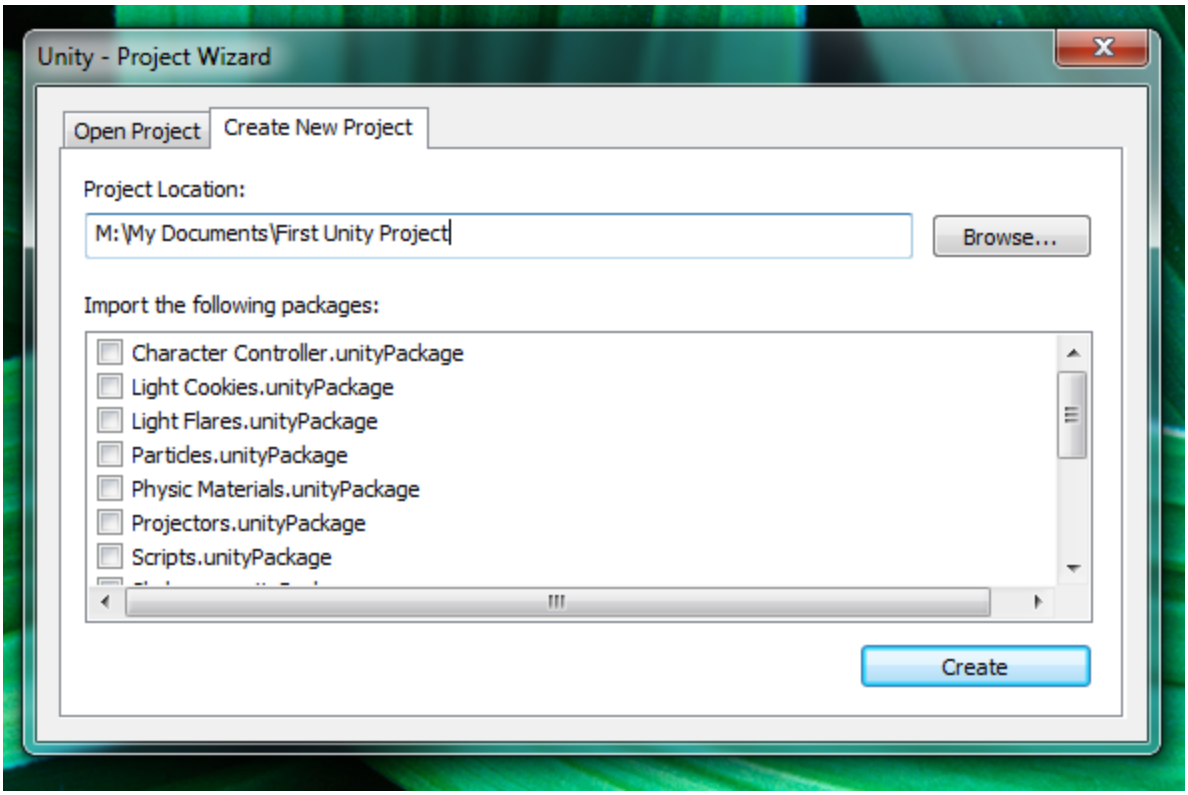
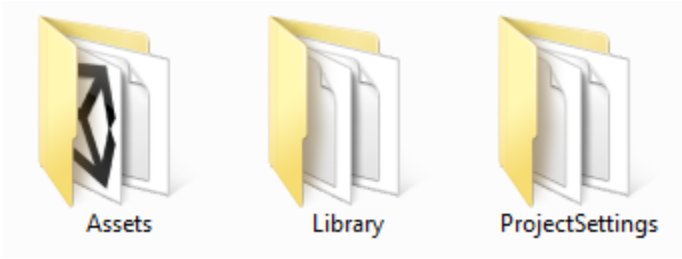


Figure 3.3: Project Wizard

The result is a project folder containing subfolders named Assets, Library, and Project Settings.



3.8 Assets

Assets are any resource your game uses. These include 3D models, materials, textures, audio, scripts, and fonts, to name a few. Other than a few simple objects such as cubes and spheres, Unity can't actually create most of these assets. Instead, they must be created externally using 3D modeling applications and painting tools and then imported into Unity.

Thankfully, Unity's asset importing is robust and intelligent. Traditionally, 3D game engines have usually been finicky things and are very particular about what files you give them, forcing developers to carefully convert all their files. Not Unity. It will accept all popular 3D file formats

including Maya, 3D Studio Max, Blender and FilmBox with all the rigging, materials and textures intact. Unity also supports all common image file formats, including PNG, JPEG, TIFF and even layered PSD files directly from Photoshop. When it comes to audio, Unity supports WAV and AIF, ideal for sound effects, and MP3 and OGG for music.

A complete list of all the formats Unity can import can be found here:

http://unity3d.com/unity/editor/importing

Let's import an asset so we have something to work with:

- Download boxboy.zip
- Unzip it to your desktop
- Drag the boxboy folder (containing boxboy.fbx and texture.png) from your desktop into the Project panel
- Drag the boxboy asset from the Project panel into the Hierarchy panel
- Select boxboy in the Hierarchy panel
- Press F to focus the Scene panel on the boxboy

Note: Unity has an Asset Store where you can purchase 3D models, characters, textures, sound effects, music, tools, and even scripts. The Unity Asset Store has quickly become an invaluable resource for game developers and a money making venture for artists and tool developers.

3.9 Scenes

Scenes are where you can drag in project assets and arrange them to make levels and game screens. The Hierarchy panel represents the contents of the current scene in a tree-like format. While the Scene panel is ideal for arranging your scene's assets in 3D space, the Hierarchy is where you'll spend most of your time actually organizing your scenes and keeping them tidy.

When you start a new project, Unity automatically creates a new scene for you. Scenes start out with nothing but a camera. If you were to run the game now, you won't see anything but the background color. To give us something to look at:

1. Drag the boxboy asset we imported from the Project panel into the Hierarchy panel

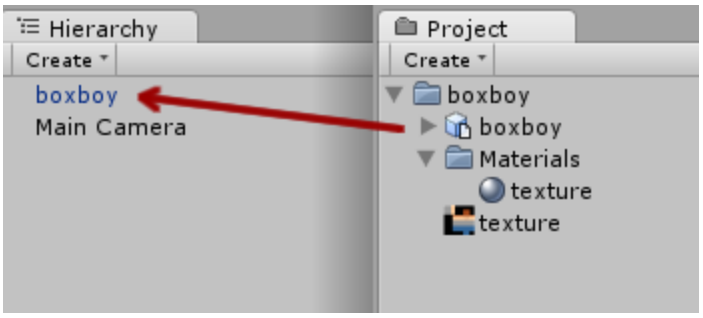


Figure 3.4: Hierarchy Panel

2. Select the box boy asset in the Hierarchy panel
3. In the Inspector, find the Transform component and adjust the position so that X, Y, and Z are all set to 0. This will ensure your asset is at the exact center of the 3D world.

Figure 3.5: Transform Interface

4. The default camera position isn't very good, so let's give it a better angle. Select the camera, then reposition it using the move and rotate tools.

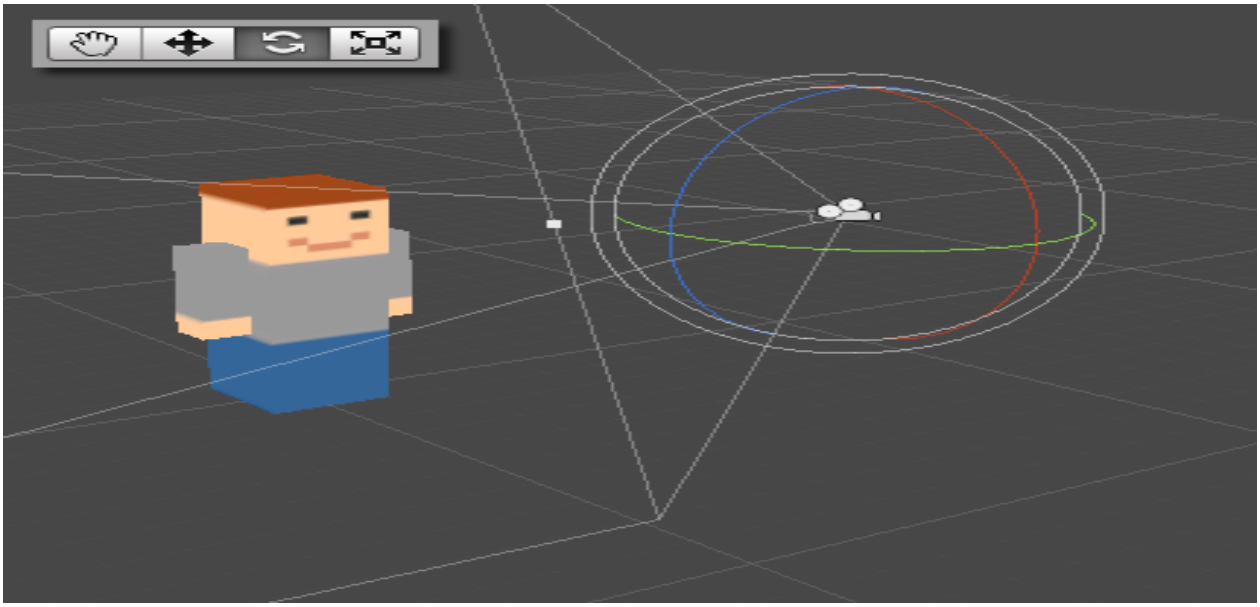


Figure 3.6: 3D interface

Scenes are assets and should be saved in your project just like other assets. To save your scene:

1. Click File > Save Scene
2. Navigate to your project's Assets folder
3. Name your scene Main
4. Click Save

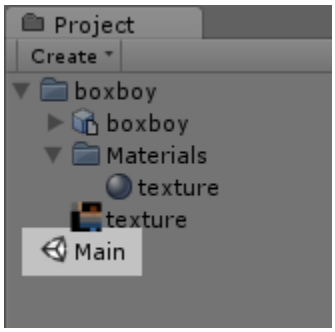


Figure 3.7: Project file

3.10 Scripting

Scripts, known in Unity as behaviors, let you take assets in your scene and make them interactive. Multiple scripts can be attached to a single object, allowing for easy code reuse. Unity supports three different programming languages; Unity Script, C#, and Boo. Unity Script

is similar to JavaScript and Action Script, C# is similar to Java, and Boo is similar to Python. Depending on your background you may feel more comfortable with one or the other.

Currently, according to our editor analytics statistics, the percentage of scripts created in the three languages break down like this:

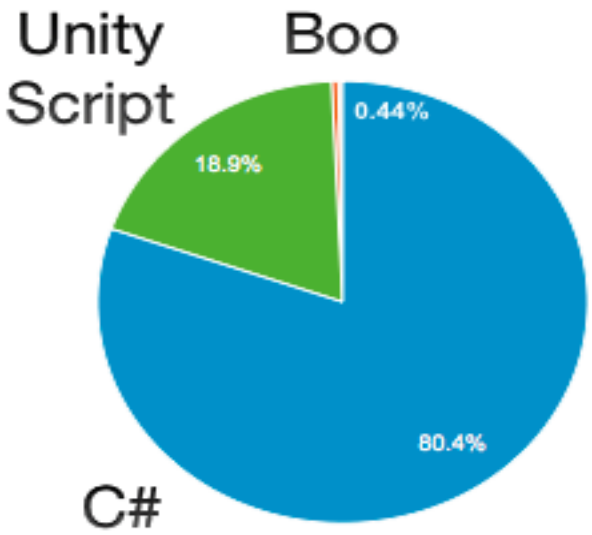


Figure 3.8: Scripting language preference

Let's create a C# script:

1. Click Assets > Create > New C# Script
2. Rename the new script in the Project panel to PlayerScript
3. Double click the script to open it in MonoDevelop

The script should look just like this:

```
01      using UnityEngine;
02      using System.Collection;
03
04      public class PlayerScript: MonoBehaviour {
05            // Use this for initialization
06            Void  Start () {
07            }
08
09          // Update is called once per frame
10            void Update()  {
```

```
11              }
12          }
```

Note: C# class names must be the same as their file name and are case sensitive. Make sure your class name matches the file name exactly, excluding the file extension.

All scripts have a `start()` method and an `update()` method. The `start()` method is run once when the object is first created, while the `update()` method run once per frame. Our script needs to be constantly checking for arrow keys being pressed, so we'll add the following code to the `update()` method.

```
01  void Update() {
02          float horizontal = Input.GetAxis("Horizontal");
03          float  vertical = Input.GetAxis("Vertical");
04          Transform.Translate(horizontal, vertical, 0);
05  }
```

Now that our script is done, we need to assign it to our asset. Naturally, Unity makes this a simple affair:

1. Drag the script onto the boxboy asset in your scene

With the script assigned to our boxboy asset, we can run the game and move BoxBoy around by pressing the arrow keys.

3.11 Publishing

Unity is able to publish to Windows, OS X, and the web via the Unity Web Player. The Web Player is a browser plugin that works in all major browsers and offers the same performance available on the desktop.
You can download the Unity Web Player here:

http://unity3d.com/webplayer/

Not surprisingly, Unity Pro can publish to even more platforms, including iOS, Android, Wii, Xbox 360, Playstation 3 and even a Flash version of the Web Player.

To publish our game for the Web Player:

1. Click File > Build & Run
2. Select Web Player from the list
3. Click Build And Run

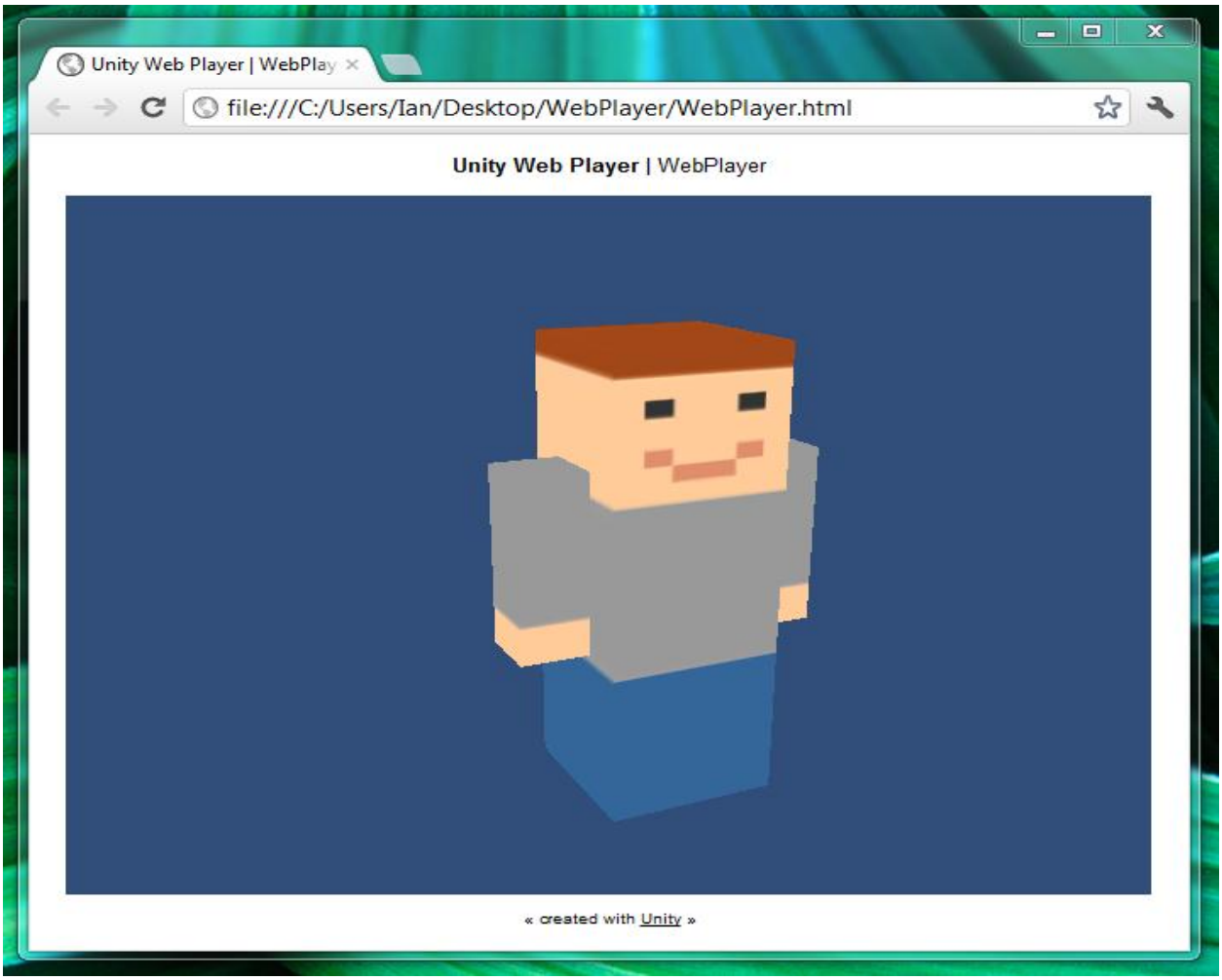


Figure 3.9: unity Web player background

3.12 Conclusion

This article barely scratches the surface of what is possible with Unity. If this introduction has whet your appetite for more 3D game development, be sure to check out the following resources:

- Unity Answers
- Unity Scripting Reference

CHAPTER FOUR

IMPLEMENTATION

4.1 Concept of the game

The game can be played up to 2 player at a time also it can be played against computer.

- There will be two paddle in two opposite side.
- In the center there is ball for hitting in the opposite size.
- If any player scored then next ball serve will be on its opposite direction.
- In the menu panel number of player can be fixed by the user.
- Number of score to win the game can also be selected from one to ten.

4.2 System Requirements for Unity 5.3

OS: Windows 7 SP1+, 8, 10; Mac OS X 10.8+.

Windows XP & Vista are not supported; and server versions of Windows & OS X are not tested.

GPU: Graphics card with DX9 (shader model 2.0) capabilities. Anything made since 2004 should work.

The rest mostly depends on the complexity of your projects.

4.3 Additional platform development requirements

iOS: Mac computer running minimum OS X 10.9.4 version and Xcode 6.x.

Android: Android SDK and Java Development Kit (JDK).

Windows 8.1 Store Apps / Windows Phone 8.1: 64 bit Windows 8.1 Pro and Visual Studio 2013 Update 2+.

WebGL: Mac OS X 10.8+ or Windows 7 SP1+ (64-bit editor only)

4.4 For Scripting

For scripting purpose I used visual studio which is integrated with Unity 5.3 and I used C# as scripting language although there is other option to choose language like JavaScript and Boo.

- I added three different script with unity.
- Paddle Script for moving paddle and playing with it.
- Ball Script to move the ball automatically maintaining Rigid Body.
- Finally the game script for handling score, restart game and selecting menu.

Besides all that I also had to consider ball mass paddle speed and few fixing some direction and angle for the ball to move freely.

4.5 Unity Game Engine Interface

Here is the game engine interface where I tested and finalize the game using visual Basic.

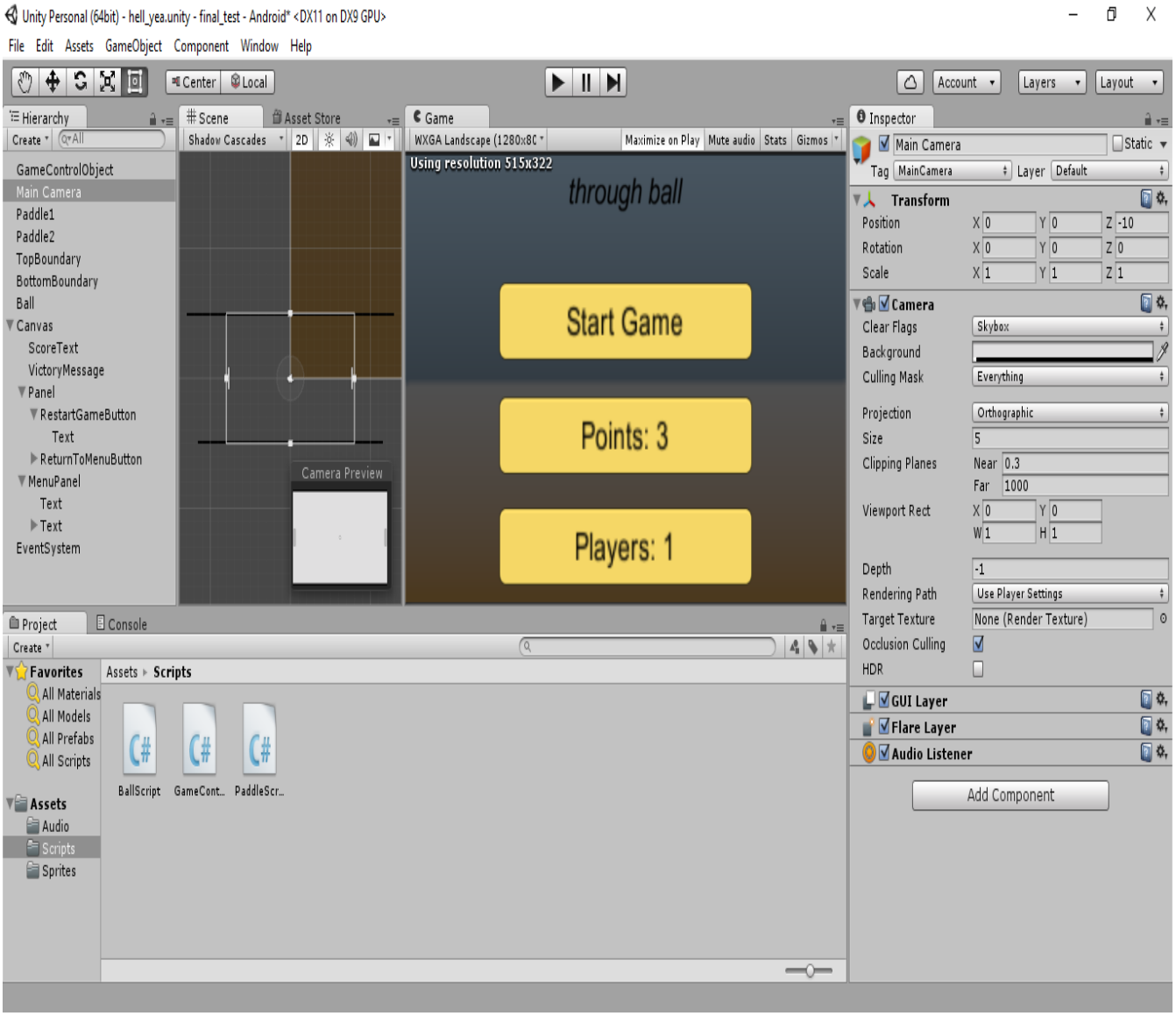


Figure 4.1: Unity Interface

4.6 Visual Studio Interface

This is the visual studio interface what is integrated with Unity 5.3 and scripting all the logic for make the game complete using C#.

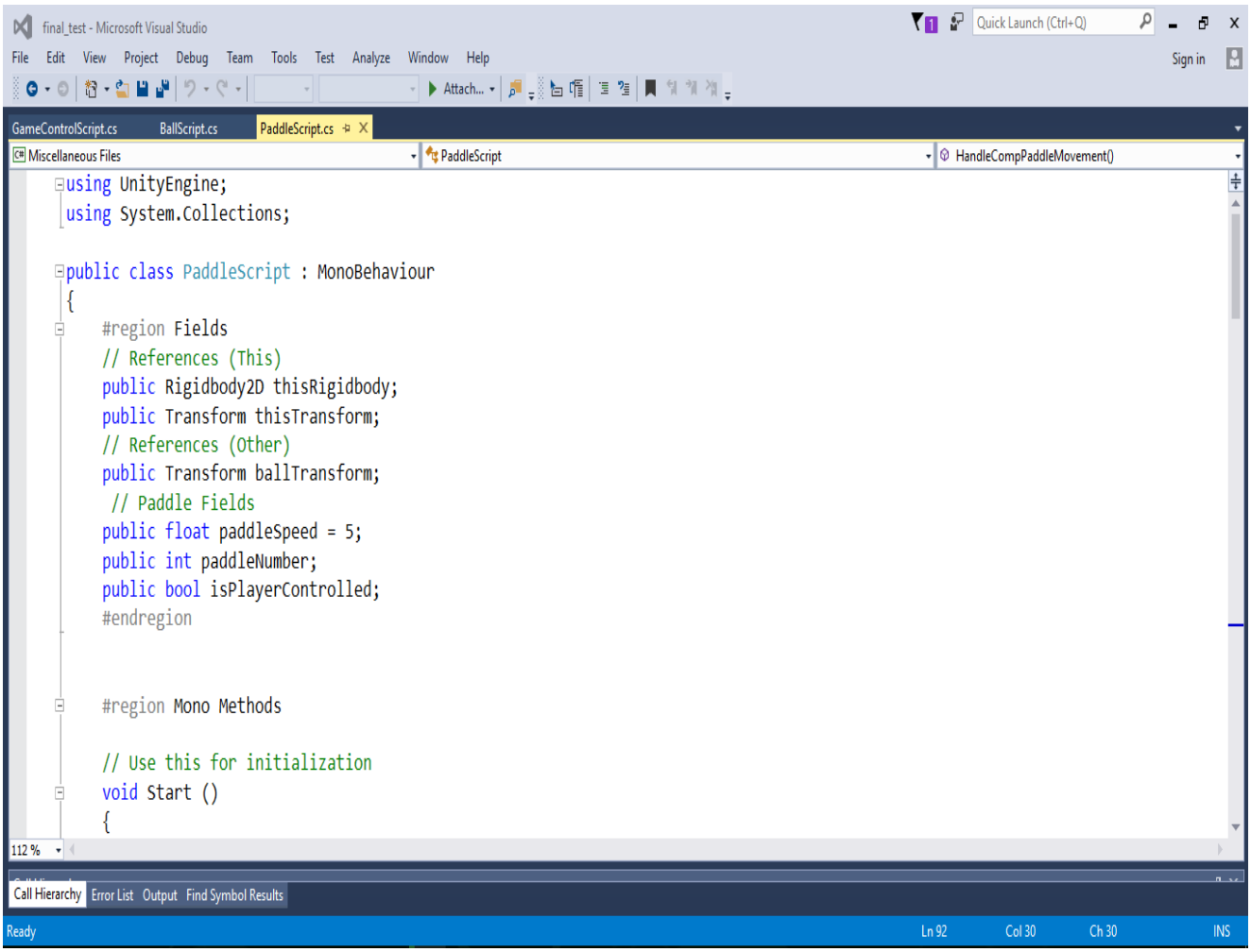


Figure 4.2: Visual studio interface

4.7 Game Interface

Below showing the starting game interface which come when the game run by its user.

Figure 4.3: Menu Panel

At first number of point to win the game will be 3 but user can change it from one to ten by clicking the button.

Figure 4.4: Number of points to win the game can be changed

By pressing the player button user can also change the no of player button or can play with the computer.

Figure 4.5: Number of player can also be changed

Game is on Play and each player can move the paddle up and down following the balls movement.

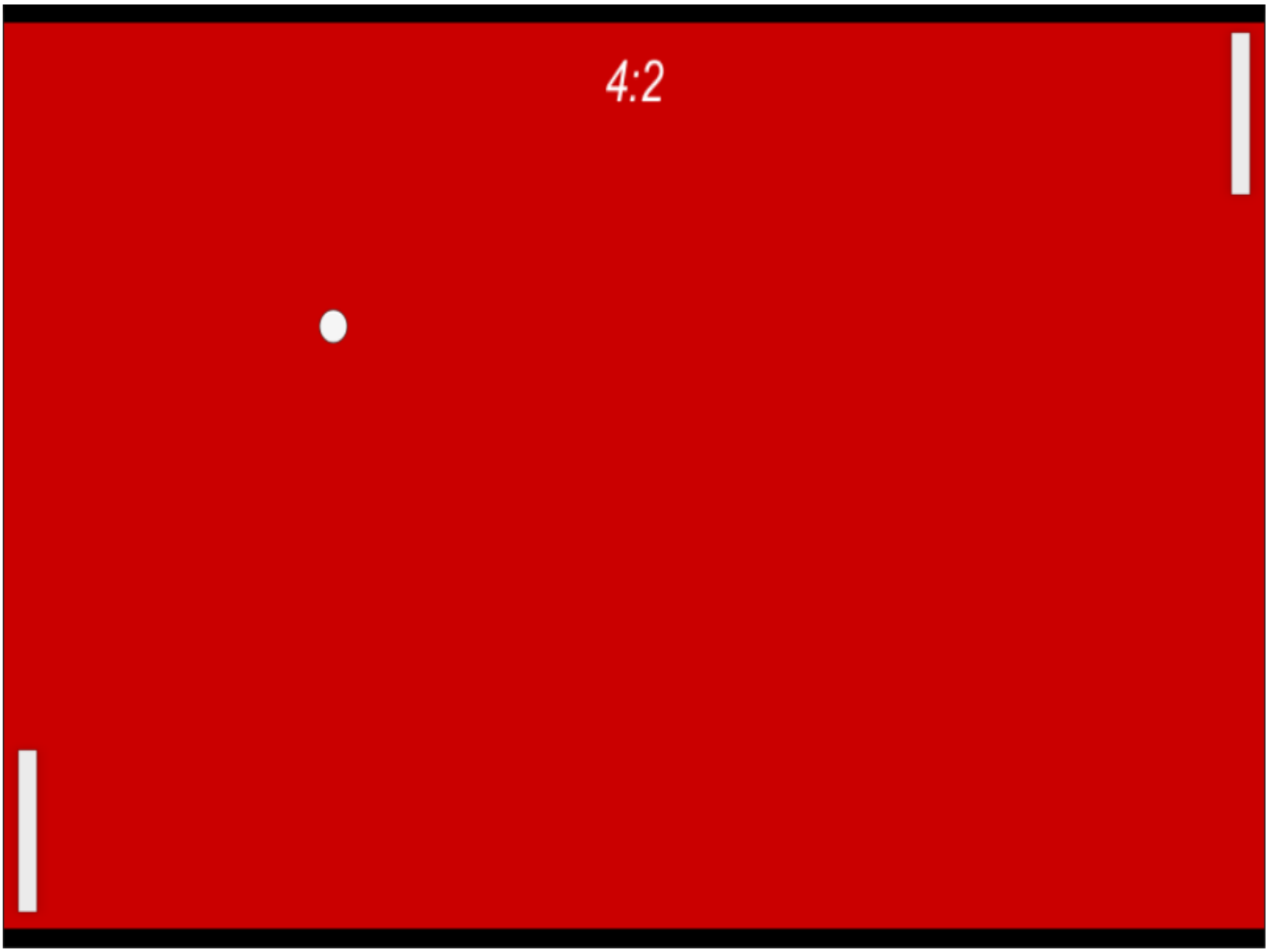


Figure 4.6: Game play interface

Following is the end game option which is showing “player 2 win” user can restart the game if he wants to play again also can press on the return to menu button to get access to the menu.

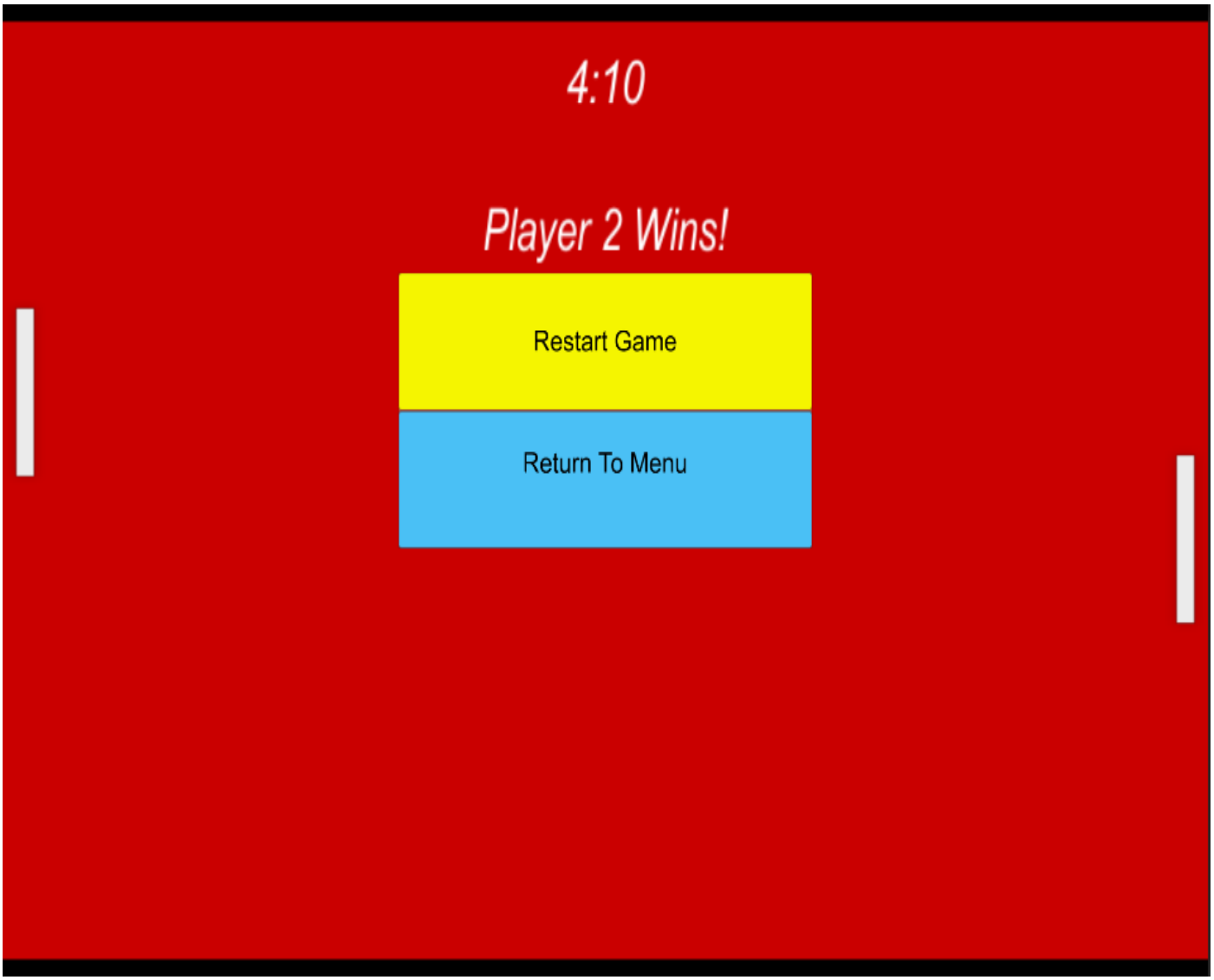


Figure 4.7: End game menu panel

Finally if user wants to leave the game he simply can click on the quit button on the menu it will take him out of the game.

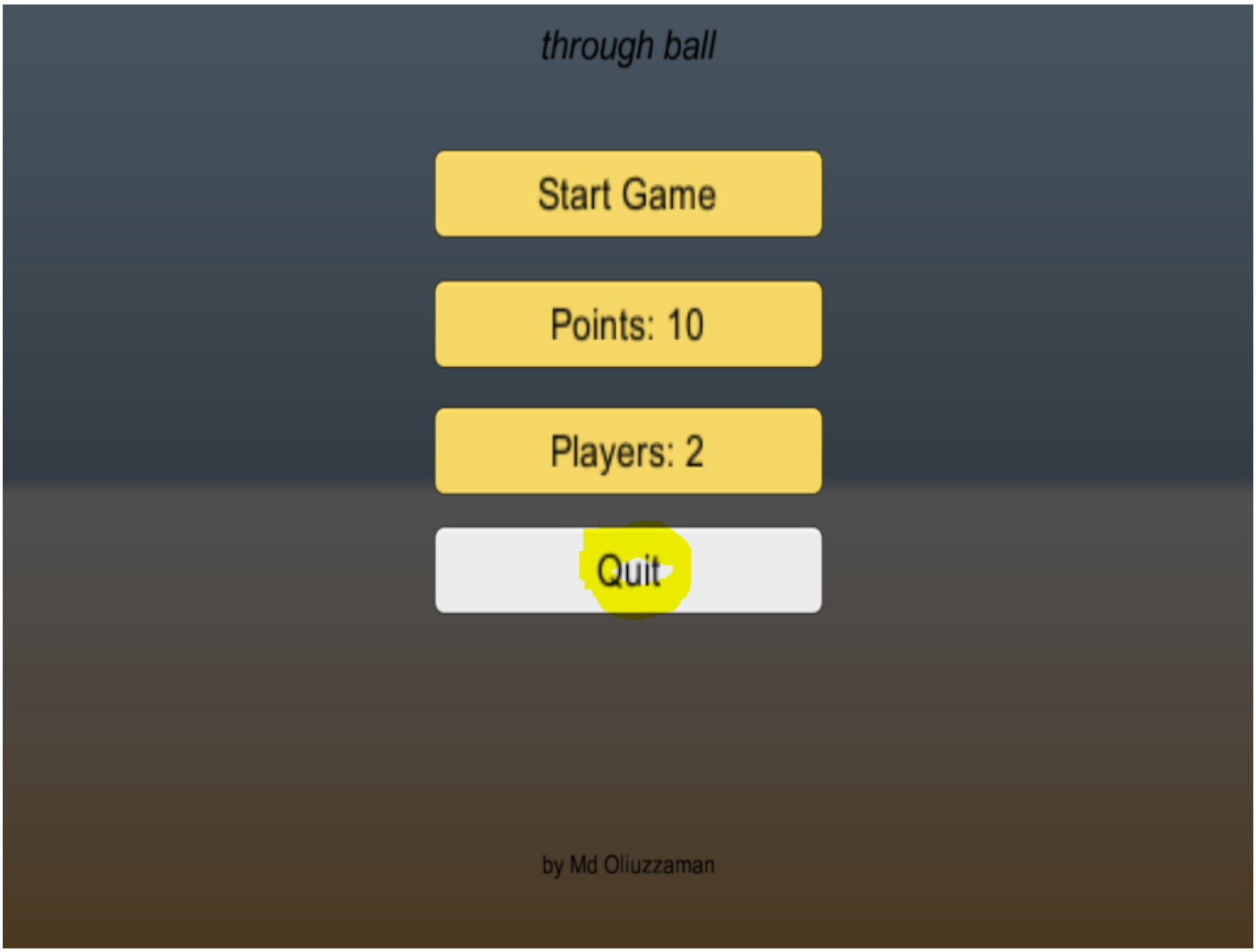


Figure 4.8: Press quit to exit the game

This is the build setting screen in Unity where the game can be built for Android, PC , MAC, iOS,, web player, Blackberry, windows, Apple TV, Linux, XBOX 360, XBOX one, PS3, WebGL, Samsung TV, PS4,Tizen and windows store.

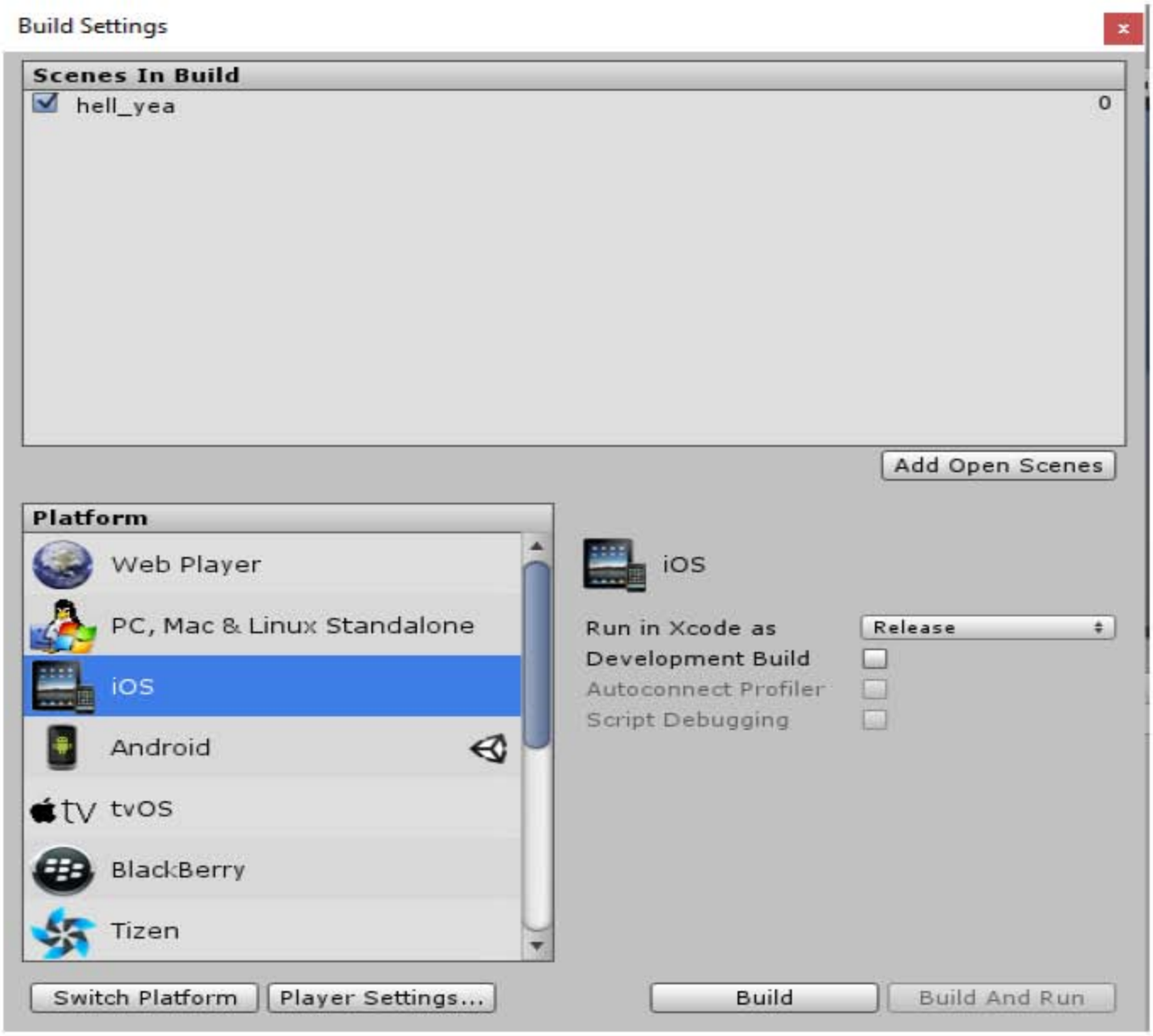


Figure 4.9: Built setting interface

CHAPTER FIVE

Conclusion

5.1 Conclusion

At the end of the day may be this project may not seems so big but it gave me a lot of courage to develop more application in future with unity and set up a path for me. However unity3D is a great tool to develop new game for both in 2D and 3D and it's very easy to learn for the beginners who has fundamental knowledge of JavaScript or C#. Although through my whole project I mainly forced on the mobile based application because not only the mobile devices itself is getting smarter day by day but also the inter application development era has be modernize massively comparing with the desktop or web based application. Mobile applications have come a long way. Just 15 years ago people were lucky to have the game Snake installed on their phone – today, mobile apps have become so integral to both working and daily lives that it's hard to believe people were happy with such limited options. Organizations must now invest in their mobile infrastructure and prepare for how they are going to implement a strong, cutting-edge development strategy that prepares developers business for the future.

5.2 Limitation

Although the application I created is working so well but still I realize there is few limitation on it. In the game play there is a little issue over the ball and it can get stuck between two player and the ball can't change its direction by itself and the ball following the same path. Secondly when the game start ball serve too fast between two player or there is very little time the players get to prepare itself for the next serve after finish previous serve.

5.3 Future Enhancement

For near future I would like to upgrade few things on this game.

- Game level can be included.
- Difficulty level (easy, medium, and hard) can be added when it's playing against the computer.
- Few graphical interface can be changed by the user when he start the game.
- Ball movement can be improved and more natural.
- Serve timer issue can be fixed so.
- Make it 3D.

CHAPTER TWO Reference

https://en.wikipedia.org/wiki/IOS

https://en.wikipedia.org/wiki/Mobile_application_development

http://www.hongkiat.com/blog/ios-history/

https://en.wikipedia.org/wiki/App_Store_(iOS)

https://www.objc.io/issues/13-architecture/viper/

http://android.stackexchange.com/questions/11400/what-are-the-names-of-the-various-versions-of-the-android-os-and-how-are-these

http://android.stackexchange.com/questions/11400/what-are-the-names-of-the-various-versions-of-the-android-os-and-how-are-these

http://www.pewinternet.org/2015/11/10/an-analysis-of-apps-in-the-google-play-store/ November 2015

http://developer.android.com/guide/components/fundamentals.html

https://github.com/android10/Android-CleanArchitecture

https://en.wikipedia.org/wiki/Windows_Phone

https://dev.windows.com/en-us/downloads/sdk-archive

Chapter Three Reference

https://msdn.microsoft.com/en-us/library/dn507457(v=pandp.30).aspx

http://code.tutsplus.com/tutorials/introduction-to-unity3d--mobile-10752

http://www.makeuseof.com/tag/programming-game-unity-beginners-guide/

http://insights.dice.com/2013/06/03/how-unity3d-become-a-game-development-beast/

https://unity3d.com/unity/whats-new/unity-5.0

http://blog.digitaltutors.com/unity-udk-cryengine-game-engine-choose/

A History of the Unity Game Engine an Interactive Qualifying Project by John Haas