

The Falling Up Game

Student Number: 3933889

# Project Overview

As someone who is fond of games and gaming in general, I have always wanted to be able to make my very own game and so I wanted to use the opportunity of this project to do exactly that. Therefore, I have decided that for my project I will be creating my very own interactive game and I aim to build it successfully and even go on to possibly publish it. I also want to make sure that once I build the game, I regularly try to improve every aspect of it from the mechanics of the game to the graphics of the game, I want to keep on building it until I am satisfied before I think about publishing it.

I have decided to build my game using a cross platform game engine called Unity. There are several reasons as to why I chose Unity in particular. First, Unity is free of any costs, and it also supports multiple different platforms which enables the developers to download and manage their creation from different devices and gadgets. Unity is also known for having great high-quality graphics and it also allows you to create a game without having a great deal of coding knowledge or experience. The game will be available on both IOS and Android.

There are thousands and thousands of games out there both on android and IOS yet I still wanted to make one of my own games because my personal gain in knowledge and experience in a field I have no experience in. I want to finish this project taking away new and beneficial skill sets and getting comfortable with using Unity and learning a new language. My main objective is to get familiar with using Unity and to be able to comfortably create another game using the skills I learn in this project, I do not want this project and everything I learn during it to go to waste.

# Methodology and Requirements

When deciding on the development of the game itself I decided to use an agile development approach. In gaming development there are three main agile practices; Scrum, XP and Kanban. I looked into all three methodologies and decided that I cannot use the Scrum approach of development because it is an approach which heavily relies on having a team and coordinating different roles of development to each member and since I am developing the game by myself, I counted out this approach of development. After some research on Kanban and XP development, I decided that for my gaming development I would go ahead with the XP approach over Kanban. There are a few reasons as to why I chose to do so, mainly because XP is more about embracing changes during development rather than following rigid plans which I think is most suited for my gaming development. I did not use Kanban because of its inability for development in iterations and building software in iteration is one of the most important aspects for development processes. Being able to develop the game in iteration using the XP approach is important because it reduces the number of risks in the programming and also helps to keep the game bug free.

**Requirements**

As I have decided to go forth and use the agile development approach to develop my own game and due to iterative development, there will be gradual and constant addition of new requirements after every increment of my program therefore I will only be specifying the fundamental features and characteristics of the game at the beginning stages.

My main requirement elicitation technique I decided to use while developing my game is looking into an existing game and monitoring all the features it offers and using it as a guide for my own development. The existing game which I felt I got my inspiration from and helped me come up with my own idea for a game is a game called ‘Doodle Jump’. After doing some research I manage to gather information on all the necessary requirements for my game development. The game should be simple to use and the home screen should be straightforward, giving the users different options and buttons such as a play button or a settings button. As well as offering some level of simplicity to the users the game should also be made fun and easy to play to make sure to get the users well engaged. As my target audience is mobile phone users the game should have a fast loading time to make the game popular and keep users engaged otherwise a long loading time could result in an uninstall. Another helpful method to an efficient development of the game is user feedback, this is important because it shows me as the developer what is convenient for my target users and what changes are required for further development of my game.

Functional requirements for the game

* User can press the play button
* User can choose character
* User can access settings menu
* User can adjust audio volumes
* User can view high score

Non-functional requirements for the game

* User must have access to Apple store or Play store
* Game must take 3-4 seconds to load
* User needs at least 300MB space

# Technical Review

When it came to the actual development of the game, amongst the many gaming development software out there I had narrowed my choice of software to using Unity or Unreal Engine. Unreal Engine is a game engine developed by Epic Games which was initially developed for first-person shooters but has recently been successfully used in a variety of other games. (Engine, 2022) The engine is used to deliver very high quality and high-performance game projects. On the other hand, Unity is cross-platform game engine developed by Unity Technologies and it supports more than 25 platforms giving the users the ability to adapt and create games in 2D, 3D and virtual reality. (People, n.d.)

When deciding which game engine I would then go on to use for my own game development, I compared a few aspects of each of the two game engines to make the right decision, choosing that which is most suitable for my project.

One of the most important features I compared between the two engines is the coding language that each engine uses, in most cases the language you code in is the determining factor for developers. In Unity, the code that is used is C# while Unreal Engine uses C++. The table below shows the main differences between the two;

|  |  |  |
| --- | --- | --- |
| Basis | C++ | C# |
| Language Type | C++ comprises both High-level language and Low-level language as it is built directly over C. It is an intermediate language | C# is a High-level programming language |
| Compilation Process | C++ compiles programs to machine code | C# compiles programs to Common Machine Runtime or CLR that is interpreted by Just In Time(JIT) in ASP.NET |
| Memory Management | Memory management is done manually | Memory management is done automatically |
| Difficulty | C++ includes complex features that are hard to understand | Hierarchy in C# is simple and is easy to understand |
| Platform | C++ is used in multiple platforms such as Linux, Windows, macOS, etc. | C# is rarely used outside Windows Operating System |
| Object-Oriented | C++ is not a completely object-oriented language. | C# is purely an object-oriented language |
| Project type | C++ is used for projects that focus on the access of the hardware and better performance | C# is used in modern application development |

Looking at the table I can see that both languages are helpful in their own ways but I think the language I would prefer to use in my project is C# this is because it is a fairly easy language to learn which is important for me because I don’t have prior knowledge on either language.

The next feature of the two game engines I decided to consider is the versatility. Both engines have great 2D and 3D capabilities however Unity has a much larger focus and toolset which makes it efficient to use. Considering my target audience, I wanted to also find out which game engine is most suitable for creating mobile games. Unity gets the edge on the flexibility department as its very well suited to creating mobile first experiences. (Jordan, n.d.) Unreal Engine is more powerful however it is somewhat inferior for 2D mobile development. Most mobile games these days don’t really require the most cutting-edge graphics meaning that Unity will be the better choice. Unity is also already the most used and considered the best engine for mobile. (Jordan, n.d.)

That concludes the comparison of the two, both have their own advantages and disadvantages however at the end I decided to use Unity for my project because I feel it is much more suitable taking into consideration the type of game I am looking to make and my target audience also considering the time span I have to complete the project.

# Design and Development

A screenshot of a computer

Description automatically generated with medium confidence

**1.**

When we first open up Unity this is the user interface which we are presented with, there are a lot of components to play around with and this was all something completely new to me. I went on to do some research in order to get familiar with the application and to get started with the creation of the game. So the first step of the development for me was to know what each component in the user interface does and how I could use each one to help me create my game.

A screenshot of a computer

Description automatically generated with medium confidenceAfter some research and playing about with the application I imported some assets which are sort of like objects which you can add into your scene and give it different components and edit it. These assets are what I will be using to create a basic idea of the game which I plan to make. Right now I have two scripts on there but I can import all sorts of existing object on there. I will be talking more about what these existing scripts are.

**2.**

**3.**

A screenshot of a computer

Description automatically generated with medium confidenceThis is the basic scene which I am starting off with and in the center of the screen there is a Maincamera which is a box which shows what will be in the frame once the game is run, you can see a little box on the bottom right which displays a preview of the camera shot as I am adding new objects and components into the frame. The right hand panel is where we can see the different options to make any changes to the selected object, here is where I will be adding any new components to the player model and to the platforms.

**4.**

A screenshot of a computer

Description automatically generated with medium confidenceScreenshot 4 here shows a closer look into the right panel of the user interface. This screenshot specifically show the information of our selected object which here is the main player on the screen. At the top we can see the transform panel where we can control the positions and size of the selected object and at the bottom we can see the components which I have added to my player character. The first component I added is the Box Collider 2D which is added and used to handle the physical collisions of the object. I also added a Rigidbody 2D component which allows us to control the physics of the object. At the bottom you can see a script component which is where the external coding takes place and I will explain this in detail with the next screenshot.

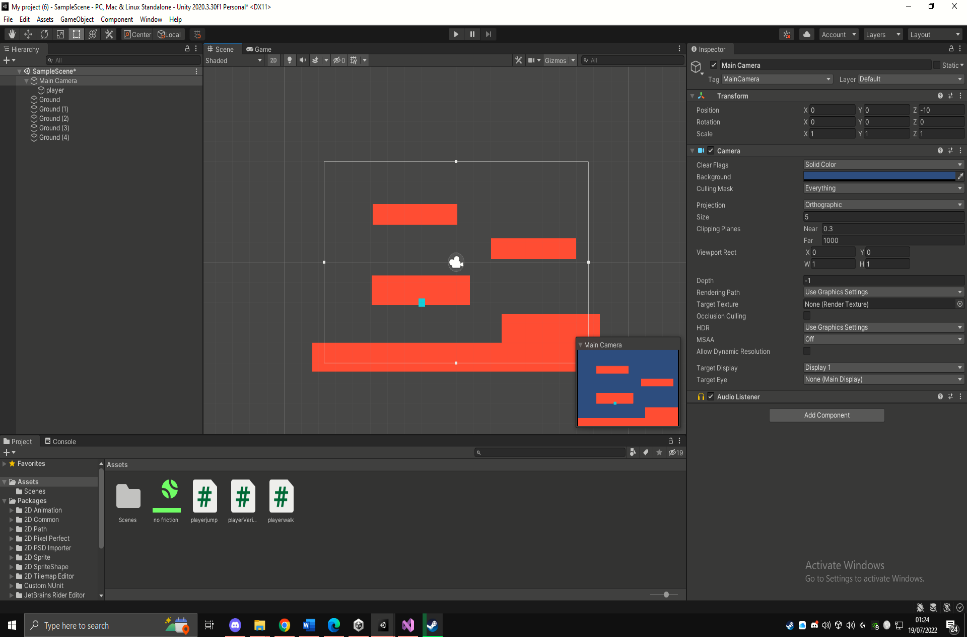
**5.**

A screenshot of a computer

Description automatically generated

Once you add the script component to the particular object, you then add and edit the code in an external application and not in Unity itself and so I used Microsoft Visual Studio to carry this out. The screenshot 5 shows the code I have as of now for the playable in game character. By defaul when you create a script there will be a public class with the name you named the script, in this case its called playerwalk. Firstly, on line 7 I added a Rigidbody2D and and named it rb and this is where I will be storing the players rigid body. Then on line 8 I added a float named moveInputH and this used to store and detect the user’s input. Line 9 as we can see is another float which I have added in for the speed of the player and I can change this speed to whatever I wish to in Unity itself and simply modify it by giving it different figures. Following that I set the Rigidbody2D variable equal to the the rigid body component of the player as you can see on line 14 inside the void Start function. After this I moved on to the update function and this is where I added the player movement keys so that the player moves as I press the arrow keys. I added this in line 23 and because I added the the horizontal plane for now only the right and left keys control the player and then in line 24 I set the horizontal velocity according to the speed I put and the values of the velocity are used here to determine the direction which the character moves.

**6.**



After successfully getting the player to move left and right by using the arrow keys I then duplicated the ground and added platforms within the scene, this means that I must now make it so the player can jump on to the platforms. In order to do this I must add a new script and code in the instructions for the player, the script I made I named it playerjump which can be seen in the assets panel.

**7.**

A screenshot of a computer

Description automatically generated

In screenshot 7 we can see the final code which successfully allows the player to jump by using the arrow up key. Now first off all I started off by creating some variables the first of these being a variable for the rigid body of the player in line 8. The next variable is float which will be used to determine the jump speed and this is followed by a Boolean variable which will be used to determine whether the player is on the ground or in the air. The last of the variables is a layermask variable which will be used for the ground and the purpose of this is to allow the player to identify whether he is on the ground and is ready to jump again otherwise the player can have simultaneous jumping ability. In line 20 I made the variable ‘rb’ equal to the player’s Rigidbody2D which allows me to actually make changes to the player’s physics. After this I created a new function calle OnDrawGizmosSelected and this is used as a drawing tool and in my game I will be using it to draw the layer mask which will be used to identify whether the player is on the ground or not. After creating the new function I then went back into the update function to make to Boolean isgrounded variable do the drawing of the cube when the player is touching the ground and on line 28 I added an ‘if’ statement to add instruction for the arrow up key which can only be used if the player is on the ground hence the ‘&& isGrounded’ is a condition inside the ‘if’ statement.

**8.**

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generatedThe player is now successfully able to jump using the arrow keys and it can also strafe in mid air which allows it to change directions while jumping. However, one issue I noticed is that the camera is stationary and so once the player jumps up from the top platform it disappears from the scene. So I made sure that the camera uses the player as a guide for its center point which will allow it to follow the player wherever he goes. To do this I had to install a new package from the Unity registry name Cinemachine, which is shown on the little screenshot on the right, and after creating a 2D camera it now follows the player at all times making sure the player is at the center of the screen.



# Critical Evaluation

What I have present thus far in my development and design part of the game is a very basic look to give you an idea of the general gameplay. I have used very basic shapes to represent the player and platform with solid colors and the real game will be nothing of such and I hope to develop the game further by spending more time and effort on it. The big and necessary changes I hope to make are that the game starts with a home screen and instead of shapes I hope to add an actual person like figure and improve the graphics of the game in general. I would also like to add a score counter which tracks and displays the players progress, saving the highest score and also would like to add a penalty element to the game where if the player falls off the platforms they are forced to restart.

# Bibliography

Engine, U., 2022. *Unreal Engine.* [Online]   
Available at: https://www.unrealengine.com/en-US/blog/unreal-engine-4-27-released  
[Accessed 03 2022].

Jordan, J., n.d. *Is Unity A Good Tool for Mobile Game App Development?.* [Online]   
Available at: https://narrasoft.com/is-unity-a-good-tool-for-mobile-game-app-development/  
[Accessed 03 2022].

People, U. o. t., n.d. *What Is Unity Game Development? How To Get Started.* [Online]   
Available at: https://www.uopeople.edu/blog/what-is-unity-game-development/  
[Accessed 03 2022].