Analysis of a game in Unity Game Engine.

Pratik Prakash Waghmode
Lakehead University - Computer Science
Student ID: 1153301 email: pwaghmod@lakeheadu.ca

Abstract— **This paper discusses the implementation of a 3D Ball game developed using the Unity game engine. The paper will explain the key benefits of using Unity versus other popular platforms and how the various components of the Unity API and environment editor are used to create this game. This discussion is divided into sections covering the various methods of the ball game like moving the ball, collecting the coins, using a UI CANVAS for displaying the score while collecting the coins, rotating the coins and many more. This paper will also mention the interfaces of 2D game development in Unity game engine. I will also discuss how games can be used to make interesting educational games for a certain age group.**

KEYWORDS

Game Development, Unity Game Engine, Unreal Engine, 2-Dimensional game development, 3-Dimensional game development, Entertainment.

I. INTRODUCTION

The term "game development" refers to the process of designing, developing, and releasing a game. It may entail concept creation, design, construction, testing, and distribution. It's critical to consider game mechanics, incentives, player involvement, and level design while creating a game [1]. A game developer can work as a programmer, sound designer, artist, or designer, among many other jobs in the business. A big Game Development Studio or a single individual can work on game development. It may be as big or as tiny as you like. You may call it a game if it allows the player to interact with content and modify the game's aspects. You do not need to know how to code to participate in the game development process. Artists may produce and design items, whilst a Developer might develop a health bar. A Tester may be enlisted to ensure that the game functions as intended.

Here are some general terms to get familiar with when working with game development [2].

Since the century, game-related research has been rapidly increasing in popularity. For example, the annual publication of game documents (i.e. books, journal articles, conference papers, or chapters) increased from 900 to 3200 between 2006 and 2016 [3]. Despite the fact that the amount of academic work on games has been continually expanding, the procedures by which games are created remain mostly unknown [4].

But when so many game engines are available which one should you prefer? This is an individual's choice.Each type of engine has advantages and disadvantages, and selecting the correct package is highly dependent on the simulation's accuracy and intended output, and is typically a project-specific decision. For my project I have chosen to move forward with Unity Game Engine.

II. UNITY GAME ENGINE

Unity is a cross-platform game engine developed by Unity Technologies. It was initially introduced and launched as a Mac OS X-exclusive game engine in June 2005 at Apple Inc.'s Worldwide Developers Conference. Since then, the engine has been steadily expanded to accommodate a wide range of platforms, including PC, mobile, console, and virtual reality [5]. Unity combines a proprietary rendering engine with the Nvidia PhysX physics engine and Mono, Microsoft's open source .NET library implementation. When compared to engines like Unreal Engine and CryEngine, there are several advantages to utilizing Unity. This section breaks out what we consider to be Unity's most important features that make it such a great game engine.

A. Documentation

The Unity engine includes with comprehensive API documentation [6] and examples. When compared to competing engines like Unreal or Source, which only give limited documentation for non-paying users, this is Unity's major benefit and leads to greater productivity (mod developers).

B. Developer Community

There is a vibrant online developer community [7] that may frequently aid new users. Unity Technologies' engineers are also eager to add functionality to the engine based on user requests, something that would never happen if you choose a big-name engine like Unreal. There are different platform where questions related to the unity game engine can be answered by indie developers such as Quora, Reddit, Slack, Discord and many more.

C. Physics & Rendering

Objects may be given mass, drag, springiness, bounciness, and collision detection using physics characteristics, as well as constructed using a variety of joints. NVidia's PhysX engine [8], which is utilised in many AAA commercial games, simulates the physical characteristics. Shader and texture assignment are rendering properties that impact the appearance of visible objects. Unity's proprietary rendering engine employs a simplified shader language that, depending on the target platform, is compiled into DirectX 9 or OpenGL 2.0 shaders [10].

D. *Multi platform Distribution*

The editor for the Unity engine runs on OSX, however Unity apps may be built for OSX, Windows, or as a Web-Player (which runs in a web browser through a plugin, similar to Adobe Flash). There are no limits on the distribution of Unity apps, and because Unity applications are not modifications of current games, the end user does not need to possess a copy of the game. Unity calls it" Build Once, Deploy Anywhere" [9].

E. *Easy User Interface*

Content is organised in a tree and dragged and dropped into an environment. Each object in the environment has its own tree, to which it may be assigned numerous scripts written in C#, a Javascript-like language, or Boo, as well as physics and rendering attributes. Script developers have complete access to the Mono API. Scripts can be used to provide interactive actions to objects, build user interfaces, or simply manage data.

F. *Reasonable costing*

The Unity engine is reasonably priced for a comprehensive game engine (although it is more expensive than free open source engines). The Indie version of the engine is free but there are 2 more versions plus and pro. For Plus it is $399/year and $1800/year for the Pro version (per seat). Finally the enterprise edition is available for $4000/month (per 20 seats). A detailed description of the services included is given on the official website [11]. This cost is comparable to Torque game engine, however Unity's Editor is more easier to use in our view. While this is more expensive than modifying an existing game, which generally just involves the purchase of a copy of the game, it gives you considerably more creative control.

III. Methods

I have divided my work into two parts. A 3D Ball game and a 2D RPG (Role Playing Game) game. Lets discuss the 3D Ball game.

A. *3D Ball Game*

The main objective of this game is to collect the coins and increase the score which can be displayed on the screen through a UI Canvas. The User (Player) has to control the ball with the "wasd" keys and use a "spacebar" to jump when needed. The Ball will be dropped on a thin platform and the player has to move along the platform. If the player looses the control and the ball falls off the platform the game is over and the player has to restart.

Now lets see step by step how this is implemented in the Unity.

1) Using Unity's build in 3D objects for Platform and Ball: We can use one of many Unity's build in 3d objects like Sphere, Cube, Capsule, Plain, Rag Doll, Cylinder and many more. To achieve the above results, I used a Sphere for a Ball and a cube for a platform and modified their position and scale according to need.

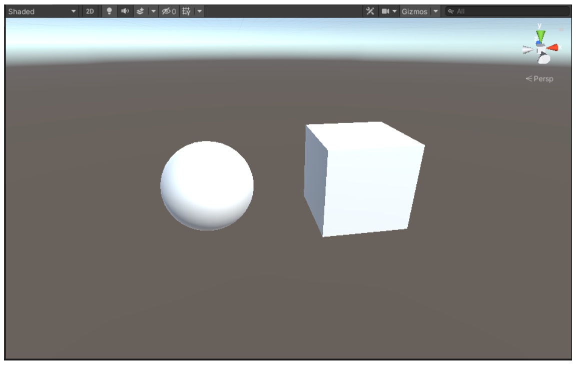
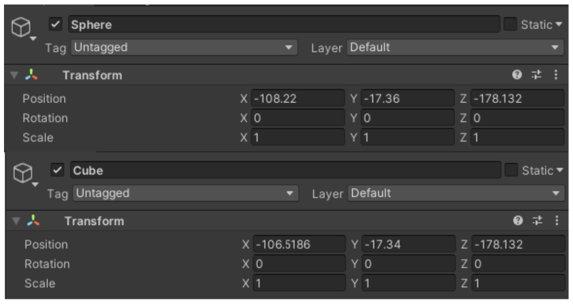


Fig. 1. An Example of 3D Game Objects

The below fig. 1 shows the 3D sphere and a Cube.

The Fig. 2 shows the transform attribute of both sphere and cube game objects which we can adjust to our needs.



Sphere — Static
Tag Untagged — Layer Default
Transform
Position X -108.22 Y -17.36 Z -178.132
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Cube — Static
Tag Untagged — Layer Default
Transform
Position X -106.5186 Y -17.34 Z -178.132
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Fig. 2. Transfrom property of Sphere and Cube

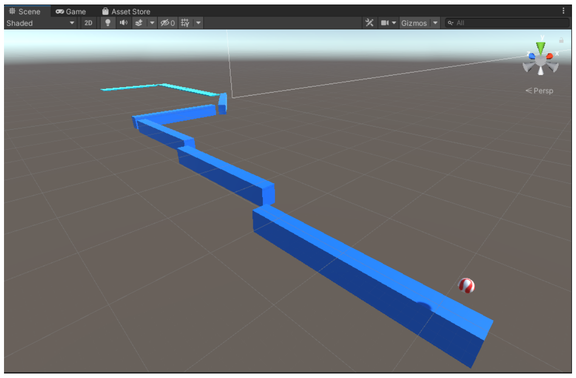
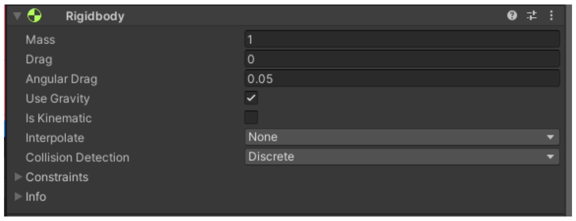


Fig. 3. Finished Environment for the game

2) Moving the Ball: Once the environment is set we need to move the ball. To do so we need to attach a rigid body to our sphere game object. When you add a Rigidbody component to an object, Unity's physics engine will govern its motion. If the appropriate Collider component is also present, a Rigidbody object will be dragged downward by gravity and will respond to collisions with arriving objects even if no code is included [12]. After adding the rigid body component to the sphere I used the AddForce() [13] method to add force to the ball in the specified direction using the Vector3 (This structure is used to pass 3D locations and directions around in Unity. It also has routines for doing standard vector operations.).



Rigidbody

Mass	1
Drag	0
Angular Drag	0.05
Use Gravity	✓
Is Kinematic	
Interpolate	None
Collision Detection	Discrete
Constraints	
Info	

Fig. 4. Example of Rigidbody

Jumping is one of the main ability of a player. We can again use the vector3 and specify the Y-component in the Vector3 and use AddForce() method to move the Rigidbody of the sphere along the Y-direction. Now the ball is jumping and moving in forward and backward direction.

3) Adding and Rotating coins: Coins in this game are collectible items and are used as a score counter. For coins we can use the cylinder 3D game object and use the transform property of the cylinder to change the shape and position of the coins. We can use any coin texture on our cylinder game object to make it look like a coin. Below is the figure (Fig. 4) of the texture I have used.

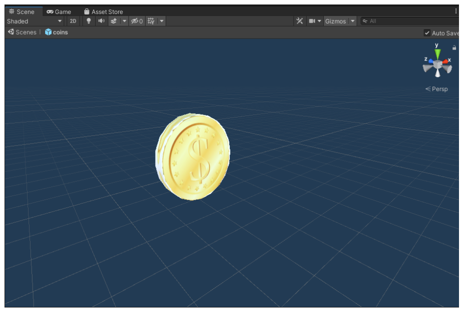


Fig. 5. Texture for Coins

Now once we done the above steps we have to place the coins on the platform as shown in the Fig. 6.

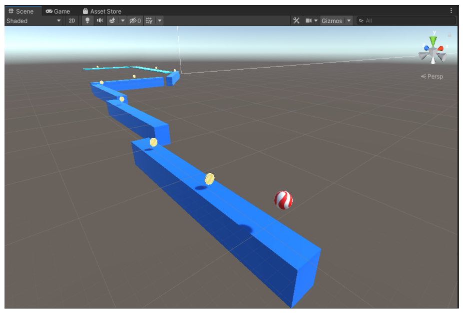


Fig. 6. Placing Coins on the Platform.

To rotate the coins along the Z-coordinate we can use the Transform property of this game object. As this is a 3D object we can use the Vector3 (WE USE VECTOR3 TO ACCESS THE X,Y AND Z COORDINATES.) and the transform.Rotate().

4) Canvas: Canvas here is used to display the score on the in-game screen. All UI components should be contained within the Canvas. All UI components must be children of a Canvas, which is a Game Object with a Canvas component on it [14]. The Text component of the Canvas is being used to display the text we want(Score).
A capsule collider [15] is by default attached to all the coin(cylinder) objects and as discussed above in the ”Moving the ball” section we have attached a sphere collider to the Sphere game object. Which will detect a collision with the coins and when the collision is detected the score will be added. We can use the OnTriggerEnter() function to state the action(ADD POINTS TO THE SCORE) when the collision is detected.

B. *2D RPG(ROLE PLAYING GAME) GAME*

As I am New In Game Development field, I am learning this 2D game from Udemy.com [16]. A course instructed by James Doyle. Currently I am studying all the aspects of Unity around 2D development.

1) Game Objects: Every item in our game is a GameObject, from people and treasures to lighting, cameras, and special effects(for example). A GameObject, on the other hand, can't accomplish anything on its own; to become a character, an environment, or a special effect, it requires properties.

2) Rigidbody: A Rigidbody is the fundamental component that enables physical behaviour in a GameObject. If the item is linked to a Rigidbody, it will respond to gravity quickly. An object's physical collisions are handled by an unseen form if one or more colliders are engaged.

3) Colliders: Collider components are used to determine the form of a GameObject in terms of physical collisions. The mesh of the GameObject does not have to be the same form as a collision that is not visible. In Unity3D, there are numerous sorts of colliders. Box Collider, Capsule Collider, Sphere Collider, MeshCollider, Terrain Collider, and Wheel Collider are a few of them. These colliders are also available in two-dimensional variants.

4) Character Controller: You may use the Character Controller as a component in your player. Its purpose is to move the player in accordance with the surrounding environment (colliders). The character controller is an actor class for player objects that provides collision-based physics and additional customisation for game characters (players or NPCs). It's a common choice for both first-person and third-person games.

5) Sprite Sheet: A sprite sheet is a bitmap image file that contains a tiled grid of many small images. You may use multiple graphics in Animate and other apps while only loading a single file by merging them into a single file. To separate all of the pictures from a single sprite sheet, the sprite sheets are modified in unity's sprite sheet editor.

Figure 7. Picture credit kidscancode.com [17] shows an example if sprite sheets.

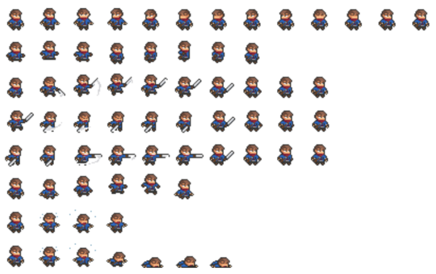


Fig. 7. Example of SpriteSheet

6) Animator Controller: For a character or object, an Animator Controller helps you to organise and maintain a set of Animation Clips and accompanying Animation Transitions. In most circumstances, having many animations and switching between them when certain game conditions arise is normal. But this can get confusing so to avoid this we use Blend Trees.

7) Blend Trees: Blending two or more motions that are similar is a common challenge in game animation. The most well-known example is the mixing of walking and running animations based on the character's pace. Another example is a character tilting to the left or right when it turns during a run. It's mostly utilised to make transactions go smoothly.

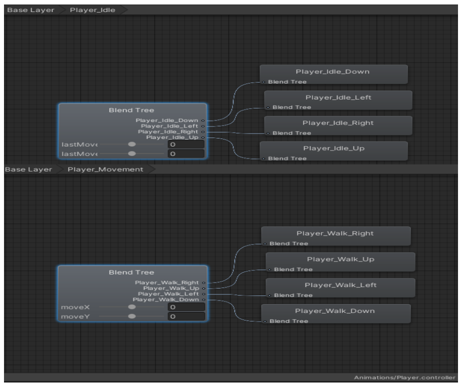


Fig. 8. Example of Blend Trees from the RPG game

IV. Results

A. 3D Ball Game

As we can see above the results of the game with screenshots have been given. The objective of the game is to collect all the coins on the platform and making sure not to fall off the platform. The results of the current score can be seen at the in-game screen as shown in Figure 9.

Once all the coins are collected the game ends and the user is prompted with 2 options either to restart or to exit the game. The future work is going to be adding more competitive levels and making the game more interesting and interactive.

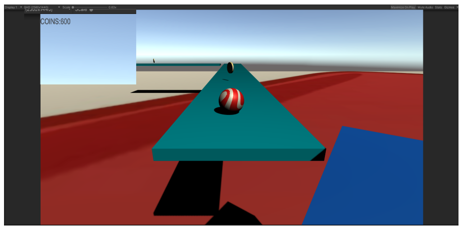
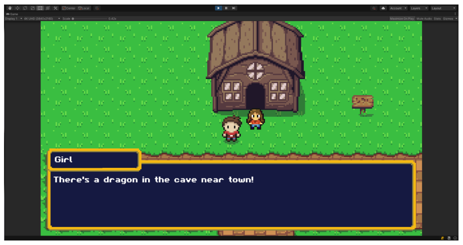


Fig. 9. Example of Score

B. 2D RPG GAME

The objective of this game is to beat the enemy using the game inventory. The game has a dialog system through which we can use a story-line as per requirement.


Girl

There's a dragon in the cave near town!

Fig. 10. Dialogue Box

The game environment can be made from ”Tile Pallet” [18]. It is a drag and drop system as shown in Figure 11.

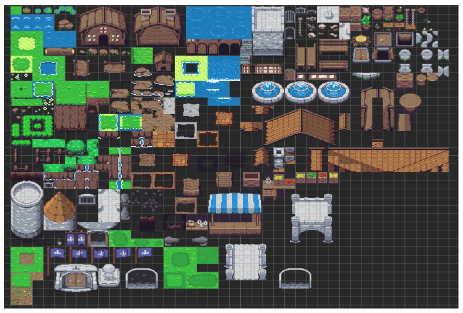


Fig. 11. An Example of Tile Pallet

There are many enemies in the game the instructor James Doyle has demonstrated the game with one of the enemies as shown in the Figure 12.

My future work for the 2D RPG Game is going to be adding more enemies and modifying the story line.

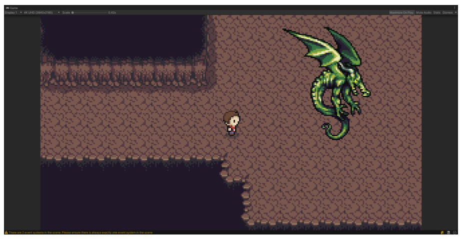


Fig. 12. Dragon Enemy

V. Discussion

The image of game development painted by the 7 reviewed paper is complicated and ambiguous: games are developed in a tangled web of content development linked with production processes, in which technological and artistic needs may clash but also provide new opportunities. The current research is lacking in the field of education. According to me, playing games is a proven method to keep students' attention, but creating games may be equally as engaging. Game creation and development improves art, math, and coding abilities while also providing a platform for students to demonstrate their knowledge. But during the SARS-CoV-2 (Covid-19) coronavirus pandemic, the use of ICT, the Internet, and some Industry 4.0 technologies in education increased. As a result of these technological advancements, education offered remotely through the Internet as part of the e-learning formula became quickly popular.

VI. Conclusion

The Unity game engine's many capabilities or components in both 2D and 3D, such as GameObjects, Rigidbody, Colliders, Character Controllers, Sprite Sheets, Animator, and Blend trees, are discussed in this article. The major objective of this research project is to learn about the Unity Game Engine's different parts and functions. I'll explore at how to build animations utilising mix trees, sprites, player physics, player movement, scene management, and other techniques in this project. Unity makes use of the C# programming language. Unity may use Python and JavaScript in addition to C# with the help of Boo. Unity is more accessible to developers than other engines since it supports a range of programming languages.

VII. Acknowledgements

I would want to convey our heartfelt gratitude to Prof. Dr. T Tomesh of LAKEHEAD UNIVERSITY'S MS in Computer Science department for his assistance and direction on this project, as well as the time and effort he put into assisting us in completing the project successfully. This is to express our heartfelt gratitude and admiration for the lecturer. I'd want to convey our gratitude to the faculty and department heads at LAKEHEAD UNIVERSITY for their assistance. Finally, I'd want to express my gratitude to everyone who has directly or indirectly assisted us in making our initiative a success.

References

[1] freeCodeCamp.org. (2021, April 28). What Is Game Development? https://www.freecodecamp.org/news/what-is-game-development/
[2] Technologies, U. (n.d.). Game Development Terms. Unity. Retrieved August 9, 2021, from https://unity.com/how-to/beginner/game-development-terms
[3] Martin, P. (2018). The intellectual structure of game research. Game Studies.
[4] Petrillo, F., Pimenta, M., Trindade, F., Dietrich, C. (2008). Houston, we have a problem. . . : A survey of actual problems in computer games development. In Proceedings of the ACM symposium on applied computing (pp. 707–711).
[5] Wikipedia contributors. (2021, August 4). Unity (game engine). Wikipedia. https://en.wikipedia.org/wiki/Unity_(game_engine)
[6] Unity - Manual: Unity User Manual 2020.3 (LTS). (n.d.). Unity3d.Com. Retrieved August 9, 2021, from https://docs.unity3d.com/Manual/index.html
[7] Technologies, U. (n.d.). Unity Community. Unity. Retrieved August 9, 2021, from https://unity.com/community
[8] PhysX SDK. (2021, August 6). NVIDIA Developer. https://developer.nvidia.com/physx-sdk
[9] Technologies, U. (n.d.). Multiplatform. Unity.Com. Retrieved August 9, 2021, from https://unity.com/features/multiplatform
[10] Technologies, U. (n.d.). Unity - Manual: Physics. Unity. Retrieved August 9, 2021, from https://docs.unity3d.com/Manual/PhysicsSection.html
[11] Technologies, U. (n.d.). Compare Unity plans: Pro vs Plus vs Free. Choose the best 2D - 3D engine for your project! Unity Store. Retrieved August 9, 2021, from https://store.unity.com/compare-plans
[12] Technologies, U. (n.d.). Unity - Scripting API: Rigidbody. Unity. Retrieved August 13, 2021, from https://docs.unity3d.com/ScriptReference/Rigidbody.html
[13] Technologies, U. (n.d.). Unity - Scripting API: Rigidbody.AddForce. Unity. Retrieved August 13, 2021, from https://docs.unity3d.com/ScriptReference/Rigidbody.AddForce.html
[14] Canvas — Unity UI — 1.0.0. (n.d.). Unity. Retrieved August 13, 2021, from https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/UICanvas.html
[15] Technologies, U. (n.d.). Unity - Manual: Capsule Collider. Unity. Retrieved August 13, 2021, from https://docs.unity3d.com/Manual/class-CapsuleCollider.html
[16] Learn to Create An RPG Game In Unity Tutorial. (n.d.). Udemy. Retrieved August 15, 2021, from https://www.udemy.com/course/unity2drpg/
[17] Spritesheet animation :: Godot Recipes. (n.d.). Kidscancode. Retrieved August 18, 2021, from https://kidscancode.org/godot_recipes/animation/spritesheet_animation/
[18] Introduction to Tilemaps. (n.d.). Unity Learn. Retrieved August 18, 2021, from https://learn.unity.com/tutorial/introduction-to-tilemaps