The Coin Thief
And
A Hidden Hex

A look on the framework and tools Used to create games

By:

Zachary Crisostomo

School of Natural & Social Sciences

Purchase College

State University of New York

First Reader: Athar Abdul-quader

Second Reader: Lee Tusman

Table of Contents

Abstract

This paper explores the different stages of how one develops and creates a video game from scratch. It delves upon the techniques and knowledge learned using the Unity game engine. Exploring the many tools and utilities that are used in creating assets in which all the objects in the game are made with. This paper also examines the various aspects of game design which is the culmination of different ideas and knowledge to help piece together a game. As a video game is not just one idea upon itself but a mix and mash of many ideas that have melded together through trial and error.

The goal of this paper is to understand and experiment using a current game engine to help and develop a platformer. Using techniques and technology to implement mechanics and interactions from my imagination into reality for a game called Coin Thief and a Hidden Hex

Introduction

I have wanted to make a game ever since I was a young child. I always played games and sparked my interest in the question " Am I able to create a game myself?". With that in mind I started to research the questions that would seem the most important in learning how to create one. Which programs are used? What is a game engine? Which coding language is most commonly used in game developing? I am aware of some terminology used in game creation such as "NPC" , "A.I" , "Sprite" or "Multiplayer". I had no idea of how they are implemented in a game or even created. It is quickly turning out that this is a very broad and open topic with many working parts that I was not familiar with.

After researching I discovered that coding of games can be coded through multiple types of languages such as Processing, Javascript, and Python. But that the main languages used for programming games are C# and C++. Many of the major games that are created today are coded using them and that software such as Unity and Unreal engine are used. Unity which is the game engine used for popular titles such as Pokemon Go and Overcooked, uses the programming language of C#. It is commonly used for various projects outside of game development but is still recommended by many as one for beginners. Hosting an assets store, Unity helps solidify and manage the amount of work you are able to find and develop within its reach as you are able to find things such as art, environment, and music which can assist you in creating a varied game with some assistance on some elements. Unreal Engine uses C++ which was used to create popular games such as Fortnite. Warhammer, Tekken and many more, the larger studios mainly use Unreal due to the high visualization engine compared to Unity.

Literature Review

Many of today's games are coded in language based upon C which is a computer programming language developed in the early 1970's by computer scientist Dennis M. Ritchie. The purpose was to be used in writing operating systems for micro computers which contained limited memory compared to main stage computers at the time. Based on the Combined Programing Language or B's restoration which used features of the UNIX operating system eventually and eventually creating C. The language due to the American National Standards Institute has then amended and standardized the language this helped it become one of the most used and common programming languages used for writing other system software or applications.(William L Horch)

Unity is a 2D/3D based engine and framework that gives you a system to design and create games or app scenes that works through the use of C# coding language. The software is used to facilitate many troubles that developers have when wanting to create virtual spaces therefore giving them tools such as physics and input controls tools which are basic elements that help create one from the ground up.(Ward, 2008). This is very important within the world of game creation as one of the key features is to create virtual worlds in which people can experience. Unity's extensive website which contains detailed documentation of its uses and code is another reason for its popularity amongst users . As proper information being available helps

those who are new and wanting to learn a simple and easy to reach source for their needs through the Unity Manual.

Within Unity there are two genres of games which are often made using this software and they would be 2D games and 3D games. Generally speaking there are vast differences that set them apart from one another which dictate the tools used and development altogether. For starters in comparison to a 3D game 2D games are more simplistic in nature and are very linear. This is due to the lack of exploration and creative capabilities that they have access to only using the x and y axis. An example of this would be some controls used, as your character doesn't have access to a full range of 3D motion which only allows for a few possibilities of interaction. Within 2D games this only allows your character to move horizontally and vertically between the two ends of the screen. In comparison 3D games include the movement through three-dimensional planes giving them a wider range of movement. The ability to turn 360 degrees and see multiple sides of objects give the feeling of length, height and depth. Another example would be the camera systems that would be used with each respective genre. In 2D games a very simple camera is used that follows the player although a creative mechanic is used to help add depth and that is called a parallax. A parallax creates an illusion of depth as it imitates the feeling of movement, this is done by moving the camera and background at different speeds. Whereas the camera used in 3D games provides more complexity through the use of perspective game view; allowing you to see things from different angles and lead to a plethora of different game mechanics that can be added with the addition of more complexity.(Ben Stegner 2020)

Mainly focusing on the 2D form of game development Unity gives access to the use of visual components, interactive elements and a basic UI that allows for the use of images such as sprites. Sprites are pixelated animations or images that are two-dimensional which are then

overlaid into a static element such as a background. These images and animations were very common in the earlier generations of computers as it allowed for more flexibility to the developers since RAM was fairly expensive and made the developers have to use creative means to provide a gaming experience.(Jacob Sobolev).

Within Unity the software houses tools called “Sprite Tools” which are an assortment of tools for proper implementation of sprite based objects to be used and incorporated within a given project. One of these key “sprite tools” is called the Sprite editor. This is a tool whose purpose is to extract an image or images from a graphic, this tool is commonly used on large files where multiple images are located where then the user is able to section off each image into their own sprite and then slice the images apart creating multiple singular images. This is done by the tool allowing the user to manually create rectangular selection around the individual elements and the areas will isolate that from everything else within the page and turn them into individual images.

Image of sprite editor slicing images into individual images

Another tool is the sprite renderer. The main function of this tool is to control how the "Sprite" visually appears; done through the use of a menu that is able to see certain properties of the sprite. For instance the variables such as the name of the sprite being used, the sorting order in which sprites go and lastly the materials which counts for textures attached to the given object that is normally set to sprite while being used in 2D creation.

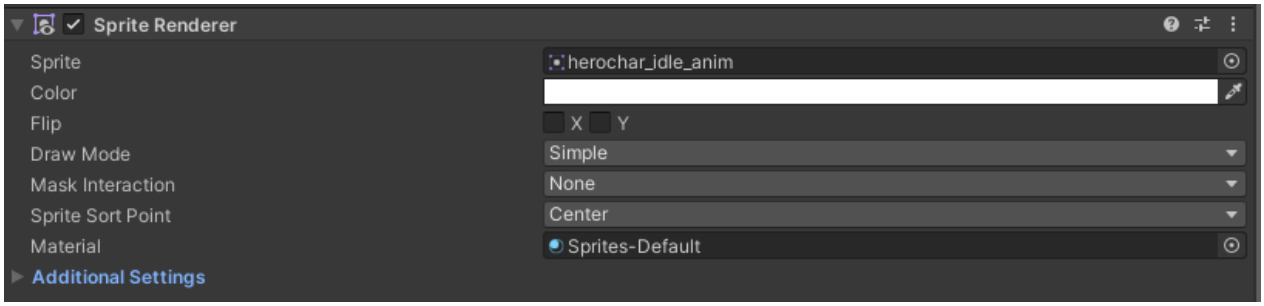


Image of sprite renderer used in project

The addition of physics creates an effect in games that allows for interactions between objects and changes how one would normally affect the other. In normal circumstances objects without physics would have no physical involvement with one another. But through the use of the built in systems within unity one is able to add gravity, interaction , positional movement giving objects more of a physical feeling when they move. This is done through the use of the Rigidbody 2D, a component that is assigned to an object and gives it the properties of the physics engines. How this system works is that when an object is created it is given a transformation component that defines how it is positioned, scaled, and rotated. Once changed this will then update other components which will in turn update other components like where things render or colliders are positioned. This all occurs through a method in which the physics engine is then able to communicate the interactions between the colliders back to the original transformation component; this needed connection is the role that the rigidbody plays.

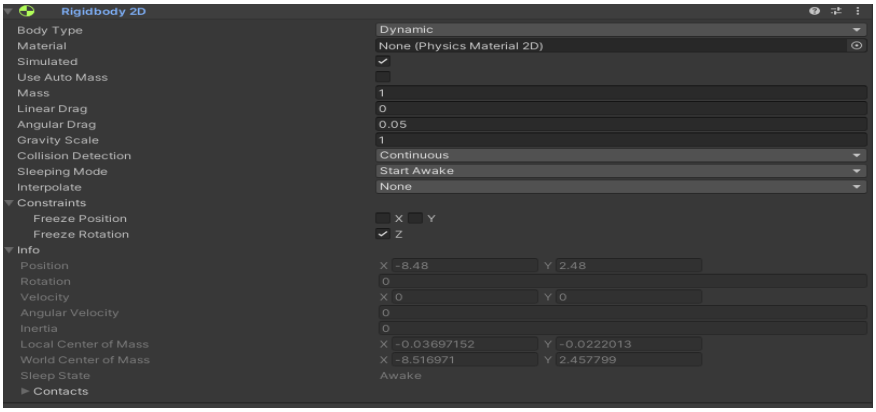


Image of RigidBody Settings

One of the important components of Unity's physics engine is that of colliders though in our case we will mainly focus on those that are used which would be Collider 2D. These are the components which will define the shape of the object for the purpose of physical collision. These are the invisible boundaries which dictate what the player or objects will interact with, although most times when used on an object or characters the exact shape of the object doesn't have to be perfectly matched by the collider, only a rough estimate.

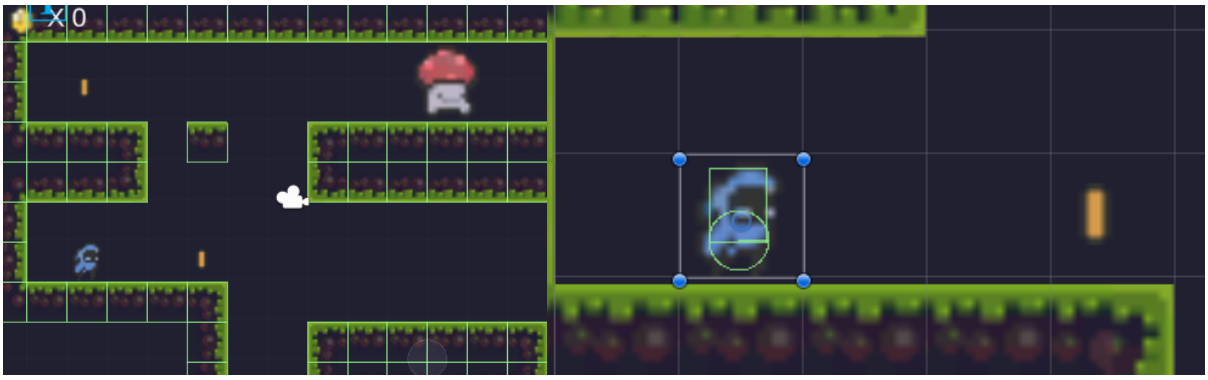


Image are of tile colliders Image is of circle & box collider on the player

Throughout the history of this development game design has played an important part in which it helps those who play the game experience a type of involvement. This is the "flow" and immersion that is expressed when one plays your game as they feel a sense of emotion, nostalgia, or wonder as they play through it.(Michailidis 2018). The psychology behind game

design helps in the design of levels and allows for different aspects and mechanics of the game to shine forth in order to combat the psychological boredom. This has the effect on those who design games and are forced to think outside the box on what they try to achieve and implement. One aspect of this is level design; this is the layout of the level which would include platforms, obstacles, enemies, and overall environment of the game. Within this a lot of the pace, tone and involvement between the player and game is created. When the gameplay or level design is very repetitive and uninspiring, reusing the same tricks over and over throughout continuous levels breaks the engagement between the player. Meaning that to retain the attention of the player games should incorporate different aspects and usages of already known and existing mechanics and systems that would further engage the user rather than the latter. Another game design idea that those who want to develop games is to stray away from predictability meaning that there is nothing interesting mechanically about what you are developing this can go all the way from a story plot or even characters as tropes have been overused throughout the history of gaming it then becomes the duty for the developers to find a way to improve those already given tropes to create interesting interactions. As this is an issue what developers need to do is keep and retain an element of surprise and make players think of doing what's not normally expected and reward them for it, simple things such as happy accidents or discovering something hidden or a different outcome rather than what's expected really goes a long way with player interaction.(Collin 2019)

Editor

The code editor used in the making of Coin Thief and a Hidden Hex is Visual Studio which is an IDE built by Microsoft. This IDE has support for many languages including C#, C++, Java, Python also with the addition of runtimes such as .NET and Unity(Microsoft,2019). Since Visual Studio is very popular there are many online resources and tutorials that would help customize it towards my needs. In this case for the use of Unity there are installations which one can download that would add in certain parameters and phrasings within Visual Studio called Unity Tools. What this does is it adds IntelliSense or in simpler terms code-completion making it fast and easy to implement Unity API messages which also includes their parameters. Included inside this package are also many debugging and suggestion settings which would help manage your code and implement best practices for performance.(Microsoft,2019)

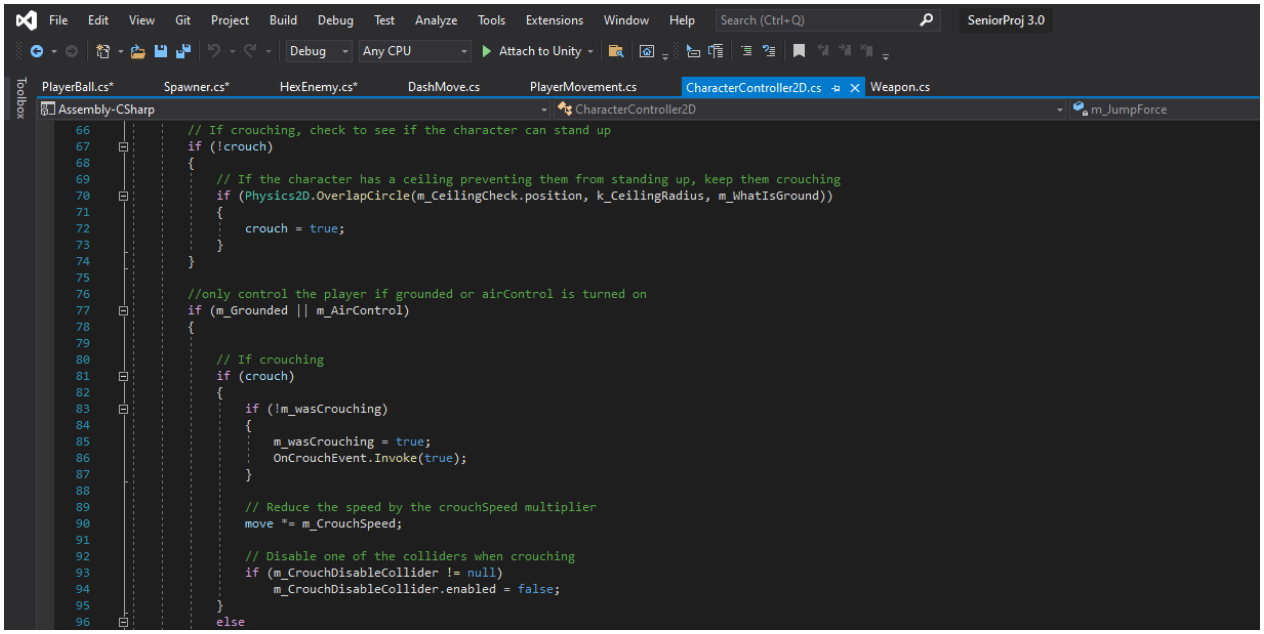
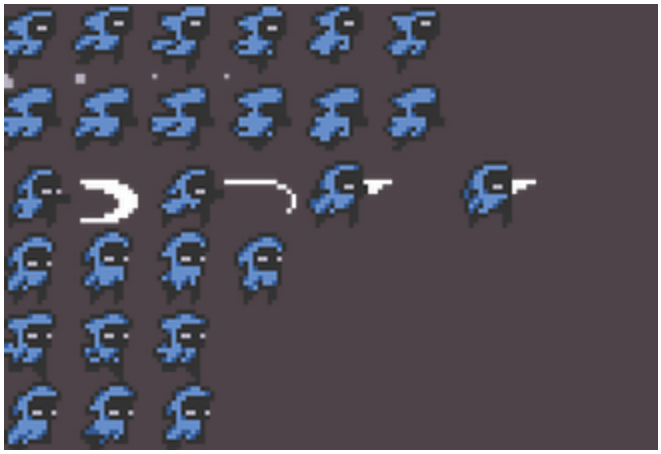


Image of Visual Studio

Assets

Within Unity's system one is able to obtain assets for their development through multiple sources that vary in quality and in cost. Primarily one is able to obtain assets through personally creating them yourself, this is normally done through using software such as Adobe, PiskelApp, Pixie, Gimp etc.... Regarding creation as Unity and the overall subject of pixel art is very popular there are many tutorials that go step by step on how to create "Sprites" . The other options for obtaining assets are the Unity asset store and 3rd party websites, while both having their pros they also have cons. Starting with the Unity asset store one of the pro's is that there are a range of offers in the store that are free which is accessible to everyone making it great for beginners. A con is that while free they are more generalized and more intricate assets cost money. The 3rd option is the use of 3rd party websites. These can be very random depending on where you look, there are some websites/creators that offer their services for a given price. They are also costly if one is on a given budget. There are also websites/creators that have them for free and usually will ask for credit if used as a form of compensation. Out of the three options obtaining assets through a 3rd party source might be the most optimal choice. Having access to custom assets while getting the flexibility of free time to spend on the coding and testing rather than creating is very beneficial. There are some cons such as cost and searching time but I don't think they offset the pros of this method.



Sprites used in project

Personally I had learned this the hard way as initially my approach to getting my custom assets was that of creation following a guide made by Brackeys on basic pixel art creation in photoshop. During which I had spent hours creating multiple sprites of different players, objects, enemies, and terrain. While putting things together I suffered a technical issue regarding my storage which involved me losing all my created assets and was unable to recover them. Now running low on time I decided to search online to find new assets that were free and that of a certain aesthetic to what I had planned to make. This is when I discovered a creator named o_Lobster's itch page which held assets for a platformer and ended up using this. Personally I think if given more time one should create their own assets but for practicality one should try the 3rd option first above all else if time is involved.

Design

So following the standard platforming genre the game Coin Thief and a Hidden Hex contains elements of standard movement used in 2D games which are vertical and horizontal movement. The initial start of the game is followed by a menu that gives the user the choice of quitting and choosing to play, although the quit button is used and coded to exit the application it currently has no use as of yet.

Image of Main Menu

The choice to have a play and quit option plays into the account of immersion as giving the ability to choose creates an interaction with the user from the start. As platformers are a simple concept meaning they have simple objectives it is always good to give your users a clear understanding of what they should do in order to accomplish a task. How the user acts upon these objectives is based on their own choices and should not be punished for.

Image of instruction menu.

The structure of this game is very simple: a player is placed in a starting point once the game starts and they are free to do as they wish. At this point in time the player has many options from what they could do based on the instruction screen and player choice. Once in the game there is a score counter present on the upper left of the screen which increases as the player runs over coins. To reach these coins they would have to do jumping puzzles and destroy enemies that block jumping points to progress. Some coins have been placed in harder locations than others to provide a challenge to the user making them have to perform what is called a “frame perfect jump” which is the idea of pressing the button on the exact frame or pixel in which it is only possible for the jump to be accomplished. This is used as a way to challenge the player in order for them to feel as though there is some complexity to what they are playing.

Image of a simple jump

Image of a frame perfect jump

The level design for The Coin Thief and a Hidden Hex is laid out in a manner that would cause the player to explore and choose different options that lead to other paths. For example in many parts of the map one is able to see holes inside solid units of blocks and then walk on them therefore the player has the notion that all of these types of blocks are solid but further into the level there are some coins within these blocks which will then cause the player to think differently. This increases engagement and changes the old perceptions of earlier jumping puzzles and solid block units as through enough attempts the player will then realize some blocks are fake. This causes the approach of the player to change and will make them revisit older jumping puzzles to verify if there were other ways to solve them.

Image of character on solid block

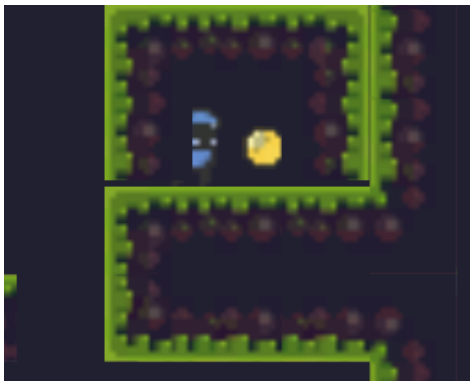


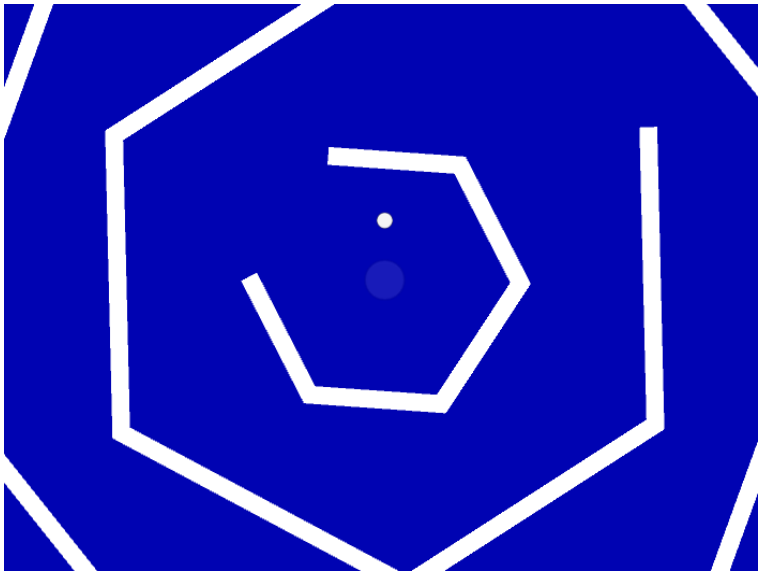
Image of character behind a fake block

While currently the game has no potential for the player to lose there is the ability to end the level and try again trying out alternate routes and exploring what level has to offer. When reaching the end the player is presented with a play again screen in which if chosen takes them back to the main menu. If the player remembers the name of the game and associates with the mechanics of hidden things, they will notice a hidden button within the main menu which leads to an alternate game. This is a way to reward the player's curiosity and keep them surprised that not only is there a literal meaning to the game but also a new game to play.

Hidden Hex Button

Hidden Hex refers to a second game hidden within the first one. It is a simple but fun game where the player is a circle who can only move in a 360 degree circle around a knob and not hit any walls. How this game works is very simple as the player moves left and right using the A and D keys they revolve around the knob and while this is occurring hexagons with a

missing side appear one after another and shrink to a central point until disappearing. This acts as a constant source of danger to the player causing them to fully pay attention and immerse themself or die as dying once takes you to the end screen. My approach to this game was very simple, which was to find a way to keep the player constantly engaged with the game. By not giving them any breathing room and to constantly challenge themselves stops bordemon.



Game of Hidden Hex

```
using UnityEngine;

Unity Script | 0 references
public class HexEnemy : MonoBehaviour
{

    public Rigidbody2D rb; // creates  a public rigidbody to be assigned

    public float shrinkSpeed = 3f; // sets a shrink speed
    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        rb.rotation = Random.Range(0f, 360f);// sets the rotation for the object from 0-360 degrees this created the variation on which side is missing
        transform.localScale = Vector3.one * 10f;
    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        transform.localScale -= Vector3.one * shrinkSpeed * Time.deltaTime;

        if(transform.localScale.x <= .05f)
        {
            Destroy(gameObject); // once the game object is small enough destroy
        }
    }
}
```

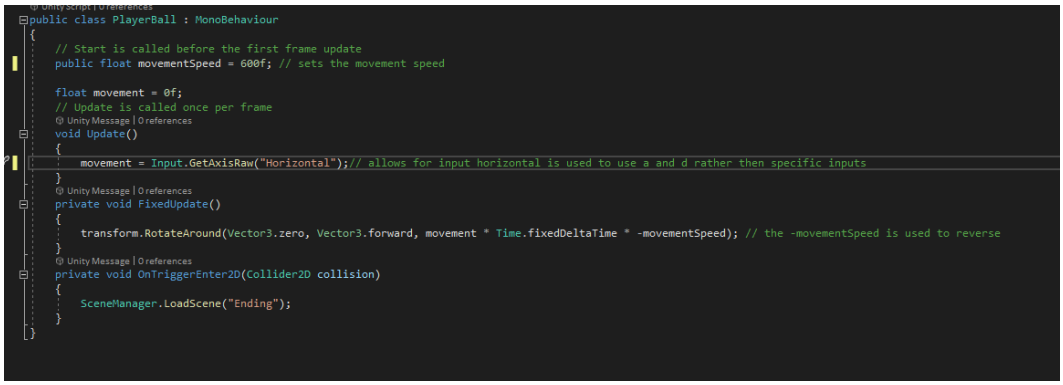
Hexagon Code

This set of code is one created for the hexagon; it allows the use of a rigidbody to be given 2D physics allowing it to have collisions and movement. In this code it shows you how the different sides of the hexagon are missing. This is done through the rotation of the object within the given scale by allowing it to have a rotation range of 0 to 360 degrees . Then allows the transformation of the current size to shrink times a certain variable by setting the primary transformation of the object to be that number multiplied by a constant viable. Lastly then we create an if statement that destroys the object once it has reached a certain size.

```
void Update()
{
    if(Time.time >= nextSpawn)
    {
        Instantiate(HexPrefab, Vector3.zero, Quaternion.identity);
        nextSpawn = Time.time + 1f / spawnRate;
    }
}
```

Spawner Code

How the hexagon operates is based on this spawner code which works off of the built in time system in Unity and this just states that if the time is equal to the variable which is nextSpawn create this object. Then what follows is that the object will then follow the previous hex code and start to shrink and the cycle continues until the player dies.



```
public class PlayerBall : MonoBehaviour
{
    // Start is called before the first frame update
    public float movementSpeed = 600f; // sets the movement speed

    float movement = 0f;
    // Update is called once per frame
    void Update()
    {
        movement = Input.GetAxisRaw("Horizontal");// allows for input horizontal is used to use a and d rather then specific inputs
    }
    private void FixedUpdate()
    {
        transform.RotateAround(Vector3.zero, Vector3.forward, movement * Time.fixedDeltaTime * -movementSpeed); // the -movementSpeed is used to reverse
    }
    private void OnTriggerEnter2D(Collider2D collision)
    {
        SceneManager.LoadScene("Ending");
    }
}
```

Code for the ball player

In the code shown above it shows the simple movement of the ball player within Hidden Hex. This shows how the ball moves with the A and D keys within the Unity System there are input managers that can assign buttons to a range of in this case the input Horizontal relates to the A and D keys meaning that if either is pressed the object will go in a horizontal direction based on each key. After this the input is read causing an action to happen to the ball this is called RotateAround what this does is it referencing and rotates the stated object around by angles and degrees this allows for specific directions to be used and in majority of the movement uses within unity the term Time.fixedDeltaTime refers to interval seconds in which the physics are updated on a fixed framerate.

Conclusion

There are many reasons why those who create and develop games choose the Unity game engine as their main tool. From the freedom they are given due to the tools provided to them which allow them to not just start from the ground up but also a range of creative landscapes in which they are able to work in. From the viewpoint of many it is very accessible and friendly for those who would begin to start their developer journey using this software. Using standardized and the simple coding language of C-Sharp(C#) helps those who are even semi familiar with any C language get a decent start and understanding of the language but also the accessibility to knowledge will attract those who are new or rather learning as this will help them achieve goals in a more efficient fashion.

In game development what matters more than anything is the experience of the user and should be taken into consideration throughout all of development. This means how they will interact and engage with the game in a way that retains their interest and makes them feel as though the time they take to engage in your product is worth it. Examining some of these key points provide some insight on how to prevent boredom from occurring as players choose to take their time to play and experience games that people develop.

Programming within Unity and developing a game requires a lot of time and effort. Although to use Unity one does not need to have an extensive knowledge on coding in this language it does help and alleviate some of the burden. The most challenging thing within Unity is the logic behind some of the code and the various syntax used inside unity which is unlike that of many other languages. It is important to understand how each code works with one another in

order to make proper mechanisms and interactions to happen that you want to design. While working within the framework of Unity the engine will take most of the heavy burden on certain tasks like physics, input, rendering, graphics and many more. It is still very pivotal to understand how each works independently and together. Just acquiring the tools doesn't give you the ability to create or design one must learn how to work on and improve the use of these tools and systems to benefit oneself. In the scope of this project the proper planning of development and understanding the limitations set upon can work in your favor if you set minor details to learn about the more broader strokes can help you accomplish large tasks in a given time.

Bibliography

- “2D Physics.” *Unity Learn*, Unity Technologies, 14 Mar. 2019, learn.unity.com/tutorial/2d-physics.
- “Applying 2D Colliders for Physics Interactions.” *Unity Learn*, 14 Jan. 2021, learn.unity.com/tutorial/applying-2d-colliders-for-physics-interactions.
- Brackeys, director. *1. How to Make a 2D Platformer - Basics - Unity Tutorial*. *YouTube*, YouTube, 26 Apr. 2014, www.youtube.com/watch?v=UbPiCgCkHTE.
- Brackeys, director. *MELEE COMBAT in Unity*. *YouTube*, YouTube, 15 Dec. 2019, www.youtube.com/watch?v=sPiVz1k-fEs&t=370s.
- Brackeys, director. *PIXEL ART in Photoshop (Tutorial)*. *YouTube*, YouTube, 22 Nov. 2017, www.youtube.com/watch?v=rLdA4Amea7Y&t=99s.
- Brackeys, director. *TILEMAPS in Unity*. *YouTube*, YouTube, 31 Jan. 2018, www.youtube.com/watch?v=ryISV_nH8qw&t=1s.
- Buckley, Daniel. “Unity vs. Unreal – Choosing a Game Engine.” *GameDev Academy*, 13 May 2021, gamedevacademy.org/unity-vs-unreal/.
- Castaneda, Lisa. “Beyond Programming: The Power of Making Games.” *THE Journal*, 18 Feb. 2015, thejournal.com/Articles/2015/02/18/Beyond-Programming-The-Power-of-Making-Games.aspx?Page=3.
- Collin. “27 Jul Why Video Games Get Boring: The Psychology of Fun.” *Game Design Lounge Video Game Level Character Story Design*,

gamedesignlounge.com/why-video-games-get-boring/#:~:text=Video%20games%20get%20boring%20when,t%20fun%20from%20the%20beginning.

- Dealassandri, Marie. “What Is the Best Game Engine: Is Unity Right for You?” *GamesIndustry.biz*, 16 Jan. 2020, www.gamesindustry.biz/articles/2020-01-16-what-is-the-best-game-engine-is-unity-the-right-game-engine-for-you.
- Dollar, Brandon. “Types of Game Perspectives.” *Mozzastryl*, 20 Jan. 2013, mozzastryl.wordpress.com/2013/01/20/types-of-game-perspectives/.
- Foxman, Maxwell. “United We Stand: Platforms, Tools and Innovation With the Unity Game Engine - Maxwell Foxman, 2019.” *SAGE Journals*, 21 Nov. 2019, journals.sagepub.com/doi/full/10.1177/2056305119880177#.
- Hosch, William L. “C.” *Encyclopædia Britannica*, Encyclopædia Britannica, Inc., 20 May 2009, www.britannica.com/technology/C-computer-programming-language.
- “Introduction to Sprite Animations.” *Unity Learn*, Unity Technologies, 7 Nov. 2020, learn.unity.com/tutorial/introduction-to-sprite-animations.
- “Introduction to Sprite Editor and Sheets.” *Unity Learn*, Unity Technologies, 10 Jan. 2021, learn.unity.com/tutorial/introduction-to-sprite-editor-and-sheets.
- “Introduction to the Sprite Renderer.” *Unity Learn*, Unity Technologies, 2 Oct. 2020, learn.unity.com/tutorial/introduction-to-the-sprite-renderer.
- “Introduction to Tilemaps - 2019.3.” *Unity Learn*, Unity Technologies, 24 Feb. 2020, learn.unity.com/tutorial/introduction-to-tilemaps-2019-3.
- K., Dustin. “The 6 Best Programming Languages for Game Design.” *The Ultimate Resource for Video Game Design*, 18 May 2021,

www.gamedesigning.org/career/programming-languages/#:~:text=The%20two%20most%20common%20languages,a%20type%20of%20systems%20programming.

- Lee, Terry G. “Overview of Visual Studio.” *Overview of Visual Studio | Microsoft Docs*, 19 Mar. 2019, docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019.
- Medeiros, Pedro. “How to Start Making Pixel Art #1.” *Medium*, Pixel Grimoire, 8 Jan. 2019, medium.com/pixel-grimoire/how-to-start-making-pixel-art-2d1e31a5ceab.
- o-lobster. “PIXEL ART METROIDVANIA ASSET PACK by o_lobster.” *Itch.io*, o-lobster.itch.io/platformmetroidvania-pixel-art-asset-pack.
- “Recorded Video Session: 2D Platformer Character Controller.” *Unity Learn*, Unity Technologies, 3 Nov. 2020, learn.unity.com/tutorial/live-session-2d-platformer-character-controller#5c7f8528edbc2a002053b68e.
- Samuel Axon - Sep 27, 2016 11:15 am UTC. “Unity at 10: For Better-or Worse-Game Development Has Never Been Easier.” *Ars Technica*, 27 Sept. 2016, arstechnica.com/gaming/2016/09/unity-at-10-for-better-or-worse-game-development-has-never-been-easier/.
- “Setting Up the Project.” *Unity Learn*, Unity Technologies, 28 July 2020, learn.unity.com/tutorial/setting-up-the-project.
- Sobolev, Jacob. “What Are Sprites And How They Work In Games?” *Gaming Shift*, 17 Sept. 2020, gamingshift.com/sprites-in-games/.
- Stegner, Ben, and Ben Stegner (1696 Articles Published) . “2D Games vs. 3D Games: What Are the Differences?” *MUO*, 7 Oct. 2020,

www.makeuseof.com/2d-games-vs-3d-games-differences/.

- Technologies, Unity. “Animator Controllers.” *Unity Learn*, 13 Oct. 2019, learn.unity.com/tutorial/animator-controllers-2019-3#5f85e343edbc2a001fcb8fcc.
- Technologies, Unity. “Unity User Manual (2020.1).” *Unity*, docs.unity3d.com/2020.1/Documentation/Manual/.
- Tuliper, Adam. “Unity - Developing Your First Game with Unity and C#.” *Unity - Developing Your First Game with Unity and C# | Microsoft Docs*, 1 July 2015, docs.microsoft.com/en-us/archive/msdn-magazine/2014/august/unity-developing-your-first-game-with-unity-and-csharp#what-unity-is.
- “Unity 2018 UI Fundamentals.” *Unity Learn*, Pluralsight Company, 11 Feb. 2020, learn.unity.com/project/unity-2018-ui-fundamentals.
- Ward, Jeff. “What Is a Game Engine?” *Game Career Guide Article*, 29 Apr. 2008, www.gamecareerguide.com/features/529/what_is_a_game_.php.