The Videogame Development Process

By: Anthony Sanchez

Submitted to the Department of Mathematics and Computer Science

School of SUNY Purchase in partial fulfillment of the requirements

for the degree of Bachelor of Arts

Purchase College

State University of New York

November 2021

Sponsor: Lee Tusman

Second Reader: Knarik Tunyan

Table of Contents

Abstract

The purpose of this project is to research and inform the average person of the video game development process from the perspectives of big/triple-A companies, indie/small companies, and individual developers. Additionally, I've created my own video game using Unity and Blender Software, described detailed steps, as well as compared my personal experience of game development to the above cases. The research of this topic is important because these experiences of the game development processes can ultimately help gain a better understanding of what it takes to make a game from an intermediate to a professional perspective. In hindsight, what does it take to make a game? How do these companies/developers make theirs? And how does my own game and its process compare to them?

General game development

The game development process, in general, varies based on many different factors including company size, budget, staff capability, current technology, target audience, and genre. No two games were made with the same intention. Based on my research, I found that there is a general pattern between developers across all magnitudes that the "I have an idea for a videogame" notion is the starting point to almost any game. Then, they begin elaborating more on their ideas or concepts of the game (Stefyn).

The concept or design process of game development is when developers begin to pitch different methods in which to portray their game idea physically, artistically, and/or

psychologically. Such examples would be early drawings of a character, environment, or level that would be in the game or a short story of the lore of the world to give a better idea of the atmosphere/characteristics of the game. If an idea is found satisfactory in being fun, innovative, and/or profitable, the developers would attempt to implement said idea into the game using software and programming to hopefully have this concept functioning as intended. After enough development into the game has gone underway, developers begin testing the game for problems. This is otherwise known as "debugging" (Stefyn).

Debugging is considered the lengthiest process of game development due to the inherent unpredictable outcomes of programming such as corrupted files/code, the game crashing for no clear reason, fixing one thing may break something else, etc. Not to mention to repeat this process on different programming languages used across different platforms. These problems alone make debugging the longest part of game development because making sure everything works is to look at every line of code to find out why the game would not be functioning properly. It only gets more complex based on how big the game in question is as well… Even with all this painstaking work, it does not necessarily mean everything has been fixed either. Many games still have game-breaking problems long after their initial release dates and developers work tirelessly to debug the game with an even tighter deadline to appease their player base (Stefyn). Despite everything, the game should (hopefully) be finished and ready to be played as intended. Though, as mentioned above, game development varies greatly based on many factors. As such the following will be the perspectives of some well-known (and lesser known) game companies and developers of their experiences in game development.

Chapter 1: Big Companies

AAA companies (Triple-A companies/Big gaming Companies) are the most well known in the gaming industry due to their success of maintaining a large player base as well as earning millions of dollars of income from their releases that brought them up so high in the industry. In this success, these companies have made the means to use a vast array of resources such as staff numbers, advertisements, quality assurance, the latest software/hardware, etc. However, this is considered a double-edged blade because games made by these companies cost on average several million dollars to make and there have been many cases in which the game failed costing the companies said several million dollars. So, the stakes are high, to say the least… Nevertheless, Let's glean a perspective of big game companies such a Square Enix and their popular MMORPG (Massive Multiplayer Online Role-Playing Game): "Final Fantasy XIV: A Realm Reborn".

Final Fantasy XIV was originally released on September 30, 2010, with a player base of around 600,000 in its first month. However, the game was ultimately deemed a failure… The game was riddled with bugs that were there since the beta tests, the UI was considered awful and clunky, players had a lack of activities to do, it had an unstable and confusing economy, recycled content from previous games, and overall lackluster gameplay... ("FFXIV Documentary 'One Point O'"). With all these issues, it would usually mark the grave of yet another game in the industry. But fast forward to 2021, Final Fantasy XIV is now the most played MMORPG today. Now how did this happen? How did a failing game suddenly become the most played MMORPG today?

Well, many would agree that it is due to the hiring of a new Producer/Director by the name of Naoki Yoshida (a.k.a Yoshi P). Naoki Yoshida along with a team of developers was tasked with fixing and improving upon the game before they would lose their entire player base. In the end, Yoshida and the team decided to shut down servers on November 11, 2012, to restructure the entire game by starting over from the beginning and working their way up from there. This was probably the first time in gaming history a game was developed this way. Mainly because what Yoshida and the team did was essentially make an entirely new game altogether ("FFXIV Documentary 'One Point O'").

To top it all off, Yoshida aimed to make the game by 2013 to be released in time for the Play Station 3 Era. Now to make an entirely new game in a year is unprecedented. It is surmised that it would generally take around 5 years to make an MMORPG so to cut that timeframe by 80% required a lot of project management. Many assets had to be created, implemented, and tested in such a short time. Fortunately, Yoshida created a system in which every project no

matter how small was to be timed for each developer to gain a better understanding of which developer can finish their tasks faster and in acceptable quality. Using this data, Yoshida created a schedule that best fits each developer based on their performance that best fits their skillsets. This system was surprisingly effective and on August 24, 2013, Final Fantasy XIV: A Realm Reborn was launched (Lawver).

The game was a massive improvement from its original release counterpart (which Yoshida dubbed 1.0), The UI was improved and designed to be more intuitive and to be used with a keyboard and mouse unlike 1.0 which despite being a pc game at the time, required a controller to play. The combat system was improved upon with better resource management, class/job actions, and balancing. Finally, there was a massive surge of content for players to enjoy unlike 1.0. ("FFXIV Documentary 'One Point O'")

All these changes resulted in the foundation for the future of the game and the release of award-winning expansions resulting in a total player base of around 34,000,000 today and still growing rapidly. So much so, that Square Enix is having difficulty with server management to accommodate the influx of players joining the game. As of 2021, there is a semiconductor shortage so there aren't enough servers to be bought for Square Enix to accommodate their growing player base despite being willing to pay above the market price (Lawver).

Ultimately, this game went through the general game development process from having an idea, implementing that idea, and debugging the game to improving upon it. In this case, Yoshi P had the idea to scrap the original game entirely and start back at square one,

implemented the idea with a series of micromanagement of his team, and debugged the game to be rid of all the technical difficulty within the micromanagement.

MMORPGs in general are considered an exclusive genre of games by big companies since they are one of the most expensive games to develop as well as maintain due to the population of thousands/millions of players that the game must provide service for. From a big company perspective, game development is usually more focused on profit rather than ambition (With Final Fantasy XIV being a very rare case), but at the same time, developers are given more financial flexibility to hire more staff and buy better software/hardware to improve upon the game. However, MMORPG games are constantly evolving in which content is added, updated, fixed regularly which in a development sense, is an infinite cycle that doesn't end so resources are constantly needed which leads to spending millions every year. Because of this, it isn't so outlandish to consider abandoning an MMORPG instead of trying to save it because it may end up being a lost cause if the company needs to spend even more money for the uncertainty of the game survival after. Ultimately, Final Fantasy XIV is a good representation of the high risk/high reward aspect of big game development because the resources, money, and manpower that go into making a successful game can magnify profits in the end.

Chapter 2: Small Companies

Unlike the big companies above, indie companies (small companies) do not have as many resources or as much funding to work with for game development. Generally, they tend to have teams of about two-dozen or less developers working on a single game, unlike the hundreds of developers Square Enix or Bethesda have. This sets a smaller boundary on how far indie companies' game development can go in terms of progress, quality, software, advertisement, and so on.

Despite all these limitations, several games from small companies have been successful both in terms of profit and player count. For instance, Moon Studios' hit series: Ori, which

includes ”Ori an the Blind forest” and “Ori and the Will of the Wisps.” The Ori series features games that adopted a Metroid mania(side scrolling maze game layout) 2d platformer games made by Moon Studios, a small indie company that partnered with Microsoft. At first, they consisted of just 10 developers at the release of their first game: “Ori and the Blind Forest,” which released on March 11, 2015. Eventually, They grew to 80 people by their second game: “Ori and the Will of the Wisps,” which released on March 11, 2020. Both games were regarded as huge successes, with an average of 90% review scores from both critics such as IGN, Metacritic, and gamers alike (Metacritic, IGN, 2019). The games sold millions of copies worldwide, widely due to the amazing storylines, artistic style, and satisfying gameplay. In a developmental perspective, Moon Studios is a good example that takes full advantage of the creativity and risk-taking that bigger companies are reluctant to do.

Both “Ori and the Blind Forest” and “Ori and the Will of the Wisps” took five years to develop, which is considered a long time in terms of development, as opposed to generally full-fledged videogames that take 2-3 years to develop. On top of the smaller developer team size, this was particularly due to the digitally hand-drawn art style the game had, which is much more time consuming and expensive to do than developing a game with CGI (Computer Generated Imagery) or 3D models. Despite this, the games have sold millions of copies worldwide and turned profits within the first week of release of their first game and within a few days for their second. This is due, in part, to the fact that Moon Studios partnered with Microsoft to release their titles on the Xbox (“Moon Studios Interview - Ori and the Blind Forest”). However, partnerships like this benefit both parties in which the smaller partner gains notoriety with the help of advertisement and/or benefits specific to consoles from big companies for a

cut of the profits. In this case, “Ori and the Blind Forest” and “Ori and the Will of the Wisps” were bundled with purchases of Xbox One console bundles, as well as new prioritized optimization for the Xbox Series X. Moon Studios benefits from selling their game and makes huge profits while Microsoft gets a cut of the profits.

Chapter 3: Individual developers

Figure 1 Markus Persson (a.k.a Notch), the creator of Minecraft and founder of Mojang Studios

The most unique case of game development is when the entirety of a game is made by one developer. Games like these are usually passion projects that don't take much time and/or resources to develop and don't cost as much to buy. They're also widely unknown, due to the lack of funding for its promotion. In special cases, games like these become popular through word of mouth (but in this case an internet setting). Such games that become popular can be bought out by bigger companies so that the individual developer can make big profits.

A good example of a single person developing an entire game is Markus Persson (a.k.a Notch), the creator of *Minecraft*. Minecraft is a sandbox survival game in which you play as a human trying to survive and create in an infinitely generating world. Though this sounds simple enough, Minecraft is one of the most played games today. Though this was originally the work

of an individual developer, Markus sold Minecraft to Microsoft in 2014 in which they took the reins of development. Despite this, Minecraft is still considered an indie game since it was originally made by one person but it was just bought out by a big company.

Minecraft first released as an early alpha testing phase in June 30th, 2010, by a sole Java developer and founder of Mojang studios, Markus Persson, known better as "Notch." He made the game as an inspiration project based on roguelike games such as *Infiniminer* and *Dwarf Fortress*, but with a sandbox/rpg element. After about only a week of development, *Minecraft* was born. Notch was used to creating a whole plethora of simple games in short spans of time, constantly creating a game, and then moving on to the next. Minecraft was one of the games that stuck with him because he was inspired by the concept of building your own structures, and he eventually further developed the game in the form of patches in which every now and then, more content was created expanding the game as time went on (Cheshire).

The game started out with nothing but a fixed sized island of nothing but dirt, stone, and the player with a player base of about 6,000. As time went on, new content patches were implemented to bring features such as wood, water, trees, ores, animals, monsters and so much more. This was the typical development cycle for Notch during his early days of Minecraft. Eventually, Markus made Minecraft procedurally generated to infinitely create randomized terrain and structures making each world for each player unique. There were many games that capitalized randomized generation. These were known as "rouge likes", which were described as games with the purpose of deviating from a set path or story and allowing the player to create their own styles of gameplay while still retaining a theme. What Minecraft did to capitalize on this further is offering complete control of the world with no set story allowing

players to do whatever they desired with the only limit being their imagination. Coupled with the simple building block style the game has, this was an easy to pick up concept for demographics of all ages. And in time, the game steadily grew in popularity from a few thousand players to millions of players worldwide at around 2014. That same year, Microsoft bought Minecraft from Notch for 2.5 billion dollars (Cheshire).

Another aspect to glean from this is that like in Minecraft's early days, simple or small games are made in the thousands by many other individual developers. However, so few of them are noticed by the public for many reasons such as little/no advertising, being an experimental but ultimately unappealing new idea, or just being one game in a sea of thousands. With Minecraft being those few that end up being noticed and gaining a popularity, even fewer can rise higher either by gaining the interest of big companies with the partnerships, being bought out, or simply building upon their success themselves. Ultimately, Minecraft is a great example of how a small and simple game that used very little time and resources to develop can become one of the most popular and profitable games today through outside help(In this case, Microsoft).

Though Minecraft is now owned by a multibillion-dollar company, it is still considered an individual developer's creation. This is a common occurrence in the gaming industry. Bigger companies often spend millions, or even billions, making their games. But even then, they aren't immune to failure. Though $2.5 billion dollars is quite a hefty sum of money, the projected profit of Minecraft was well worth it as shown in their third quarter financial results in which Minecraft has resulted in a $15.5 billion net profit (Cheshire).

In hindsight, big companies often can't afford to take risks or try new ideas as smaller companies or individual developers can. Big companies usually stick with what is sure to make them money. But said companies can also buy the games from smaller companies/individual developers to save money from trying to build a new game from the ground up, improving upon what was made instead. In *Minecraft's* case, Microsoft basically took the reins of the direction that Minecraft would be developed in many ways. They made versions of the game that use different coding languages such a C#/C++ to be used on mobile devices as well as console devices such as Xbox since Minecraft was initially made exclusively on PC with Java as the language. There has also been the implementation of microtransactions, DLC, and games with the likeness of Minecraft such as *Minecraft: Dungeons* and *Minecraft: Story Mode*. Of course, there are still content patches as the main staple of Minecraft's longevity. But with Microsoft as the new owner of the game, it has come a long way from a simple game made in a week to an infinite expanse of creativity.

Chapter 4: Making my game

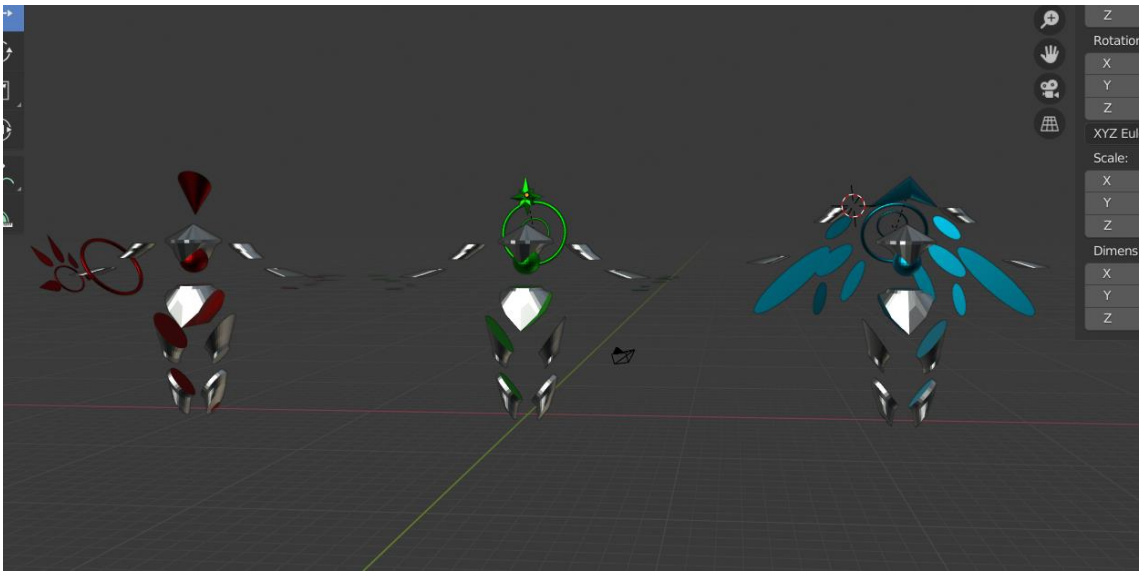

Figure 2 Models made with Blender

There were many different cases of game development as described above. Now it is time to initiate the development process of my very own game.

I started with the common question for a developer: “what kind of game do I want to make?” I began by making game models with Blender, a 3D modeling tool, and ended up with three different animated models, each with slightly varying characteristics. One with an arm cannon, one with wings, and one with a spiked head. Each variation was designed with a specific genre (shooter, collector, flyer) in mind to pinpoint what kind of game I was looking for. In the end, of the three models I made, I chose the flying version because I felt like shooter/collection type games are common. I believe flying games are not as common these days, so it was a clear choice.

Additionally, I found myself interested in infinitely generating building-based games like Minecraft. However, I realized that making such a game was difficult in terms of controlling assets, optimizing processing power, and managing memory with my current skills in C#. As a result, the survival/building aspect was scrapped. I still wanted to keep the infinitely generating concept though.

For inspiration, I turned to a mobile game called *Temple Run* and found satisfaction in its example. *Temple Run* is an infinite runner game where you try to survive for as long as possible while running on an endless path that throws random obstacles at you. I found this to be a good basis for reference because, unlike *Minecraft*, it keeps memory on how the player interacts with the world. In *Minecraft*, even if the player travels far into the game, they’re still able to return to previous areas where their past actions, such as building a house or destroying

a forest, are still prevalent. But games like T*emple Run* infinitely generate the world ahead of the player, while also deleting the world behind the player to save processing and memory power. The only drawback is that you cannot backtrack/return to previous areas since they no longer exist. My game intends to do the same but in a different style.

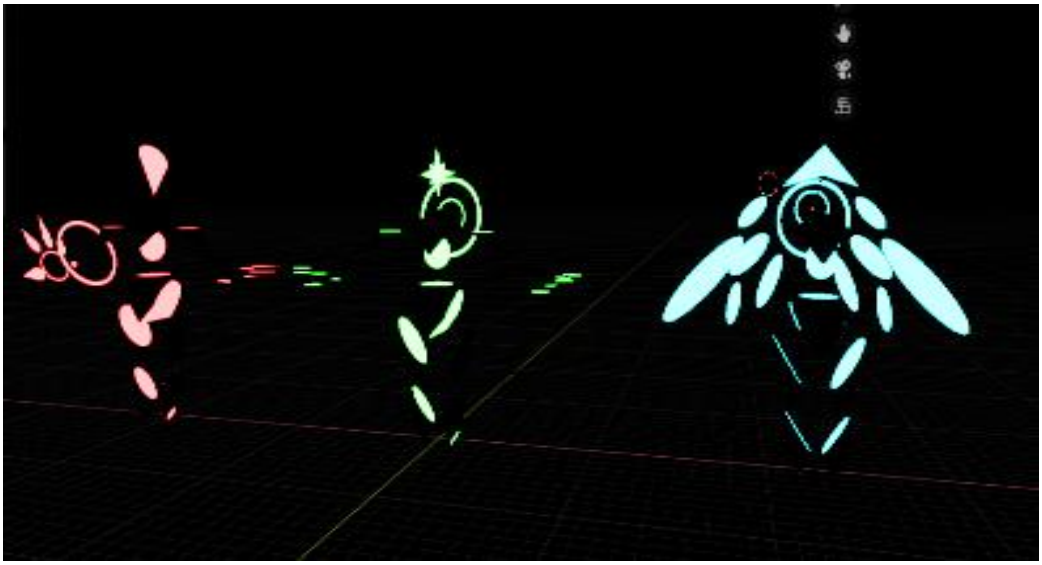

Figure 3 Models with glow effect

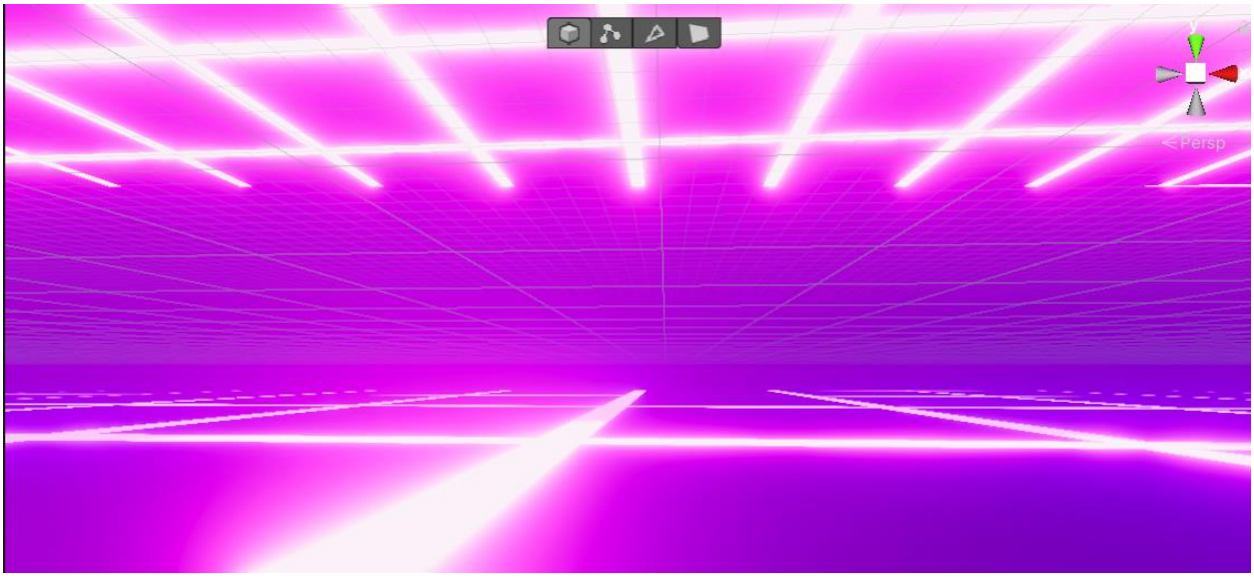

Figure 4 World Space for the game made with Unity

As for said style, I made a cyber-based theme to fit the player models. The game is set inside a computer in which you play as a personified program to fly through an infinitely generating virtual world. I decided to go with bright lights because I found them to fit well into a program like virtual game, akin to the designs of *TRON*. After working with Unity's shader tool to achieve the bright digital aesthetic, I was able to create a base world that the game would reside in. This was also the point when I made up a name for my game: *Light Flyer*.

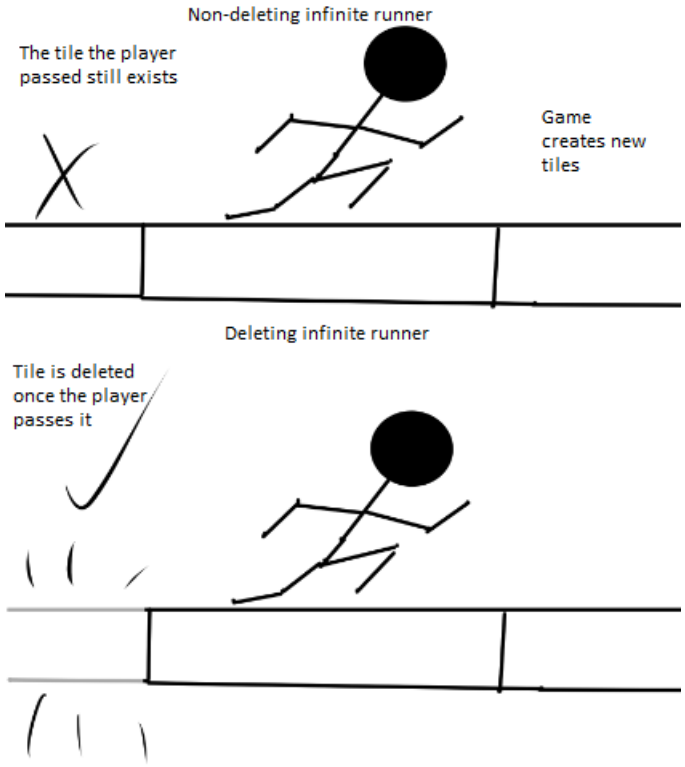

Figure 5 Poorly drawn representation of infinitely generating simulations

Now that I had the models and the world space, I needed to make the actual game. To make an infinitely generating game, I implemented a script to manage the tiles (the area that the player will go through) to infinitely generate terrain in front of the player and delete terrain behind them once they pass through it. If we don't delete the tiles when they are no longer

needed, the script will keep making more until the computer can't handle all the game objects. This will slow down, or possibly crash, the game. To fix this, I coded the script to delete the tile when the player passes in front of it. This saves processing power and ultimately creates the illusion of an infinite landscape. Finally, I added a life system, so the game won't continue forever. I decided to go with the classic three heart system. When you collide with an obstacle, you lose a heart. Once three hearts are lost, the game is over. Afterwards, the player is given a prompt to exit the game or try again. And with that, I completed the basis of my video game.

However, the game was not necessarily finished. In a perfect world, my game would have functioned as intended from the beginning, but there were far more problems than anticipated. One of the first was asset compatibility between Unity and Blender. You see, Blender's compatibility is based on how complex the asset in question is. Apparently, Blender can only successfully export animations and base model shapes but fails to export VFX (visual effects)/shader assets because Unity has its own version of these assets that are not compatible with blender however, I came to a solution in which I recorded a video of the aurora and exported that video to be played at an angle to give the illusion of a 3d object in the sky:

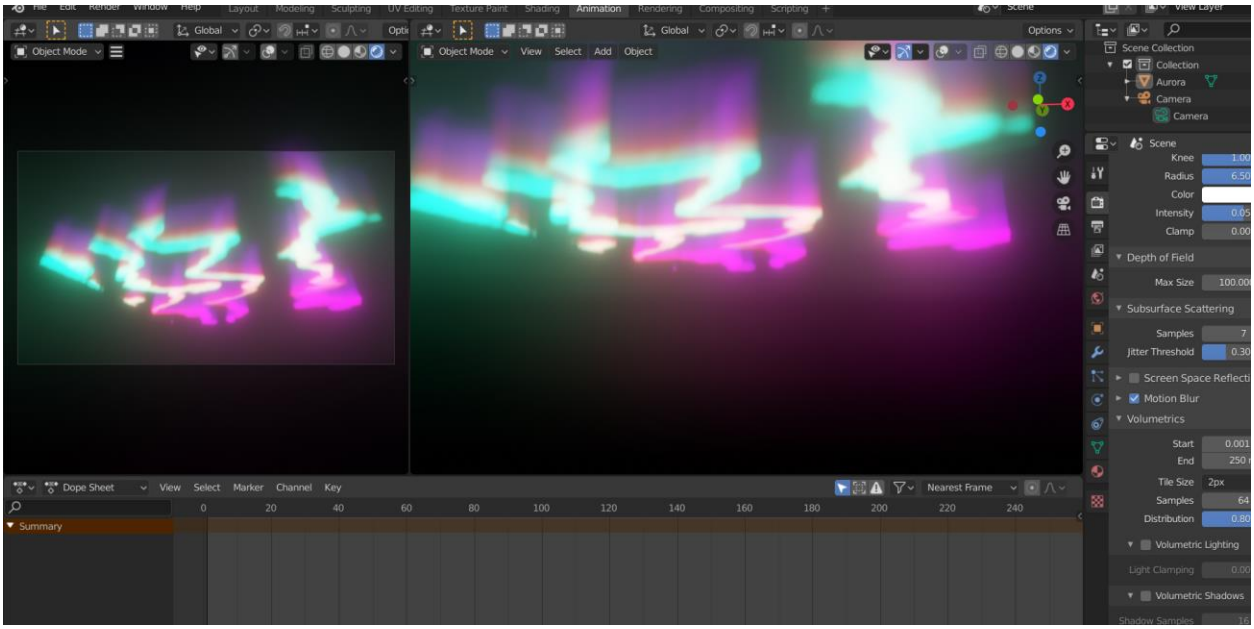

Figure 5 Aurora made in blender.

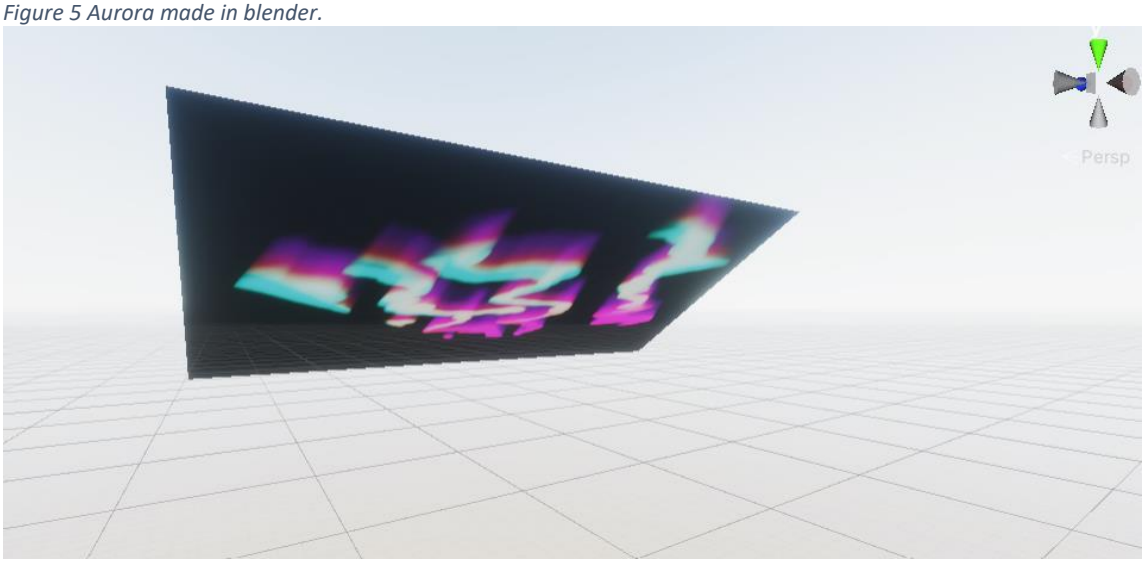

Figure 6 Video of Aurora in Unity

One of the biggest problems I faced was scripting, which leads me to the debugging section of the general game development process. I made various scripts: one to make the world infinitely generating, one that makes a health system and displays a game over screen when dead, one to control the player movements and speed, and more. Unfortunately, none of

them worked the first time. With an infinitely generating script, the game would crash from so many objects being generated at once. The movement script would not move the player at all. The health bar script would think the player was dead already. These scripting problems were some of just many problems I encountered throughout my game development process. I fixed them to the best of my ability and after about six months of debugging/fixing, I would like to say that I have finally finished my first build of the game. Afterwards, I sent a few copies out to some friends to see if the game would be functional. Thankfully, it functioned as intended, so now I can say I finished my game.

However, I intend to further improve and add content over time just as Minecraft did. So, this would be considered the "first build"(or 1.0) in which this would be the first released version of the game with more content being added in the future(2.0,3.0,etc.). Essentially, I start the general process all over again and come up with an idea, implement it, then debug any problems that occur. However, at this point it would be easier to make and implement content since I now have a base game to reference and improve upon instead of starting with nothing and working from there.

You can try out/view my game with the following:

(Link to download game WINDOWS ONLY)

https://drive.google.com/file/d/1xlNGzs5Eh-oeKOYT8c6tgn-anT7OriX4/view?usp=sharing

Instructions:

1. Download and extract from archive
2. Run the .exe file (Programma.exe)
3. Enjoy!

Controls (REQUIRES MOUSE):

- Move your mouse to control steering of the player and avoid obstacles

(Link to beta test video)

https://www.youtube.com/watch?v=rFGldYARwpk

Chapter 5: Conclusion

In terms of general game development, I believe I followed the general process to the letter. I essentially thought of an idea, implemented it using Blender and Unity, and debugged any problems that occurred when implementing these ideas in Unity. Though between specified processes they developed their games much differently than the general form. Such an example would be FFXIV that went backwards by general development standards by debugging their old game, dismantling it when debugging failed, and then rebuilding the game from the ground up. In hindsight, the general process should not be taken as the only way games are made but instead as a start for aspiring developers such as I to work with and improve upon to their specific needs since one process may only work for my game but not someone else's. Despite this, there are correlations of development between games and in my case and unsurprisingly, I felt the most aligned with the development process of individual developers since I am one now.

The time it takes for one person to make a game is astounding, but I feel proud about it now that I have made my own. This must be how Notch felt when he looks at how successful *Minecraft* became after all that time and hard work he put into development. As for small companies and/or big companies, I feel that my game would have been more complex and more in-depth in terms of content and graphics if I had a team to work on each aspect of the game individually. Additionally, having the funds to buy better hardware, software, and promotion for my game would be a big help in improving the notoriety and quality for my game.

However, there are things to address about what I have learned in this experience. The first is preparation. One of the biggest reasons why this project took so long to make is my lack of prior experience. I knew nothing of game development beforehand, so most of my development was trial and error, along with research of base knowledge of how C# and Unity work. Additionally, my resources were limited without money. Unity allows the purchase of assets such as game models, textures, VFX, and more. However, I did not intend to spend money on this project. Therefore, I had to make my own assets, resulting in longer development times. I also should have done more research of the compatibility between Blender and Unity. Had I known beforehand that VFX assets in blender are not compatible with Unity, I would have instead tried to make my aurora borealis in Unity instead of using my work around. Finally, I should have made a better schedule between working on my writing and developing my game. I chose to work on both at the same time which ended up being hectic during many occasions… and had I been pressed for time like most development companies are pressured to do, I would have ultimately failed my assignment. But we all start somewhere, so I am confident this was the case for many fledgling developers, myself included.

The game development process came in many forms, from the biggest and smallest magnitudes, to even my own perspective, in which I found this to be an enlightening experience. Though it seemed like a relatively short/simple description of my game development process, my game took roughly six

months to make. In this time frame, I modeled and animated assets in 3D, created a world space, implemented shaders, scripted in C#, tested to make sure everything works, sent copies of my game to friends to test compatibility, and debugged problems with the game, nearly all while having little to no prior knowledge of how to do so.

Despite everything, I still feel proud of my work, even its lackluster appearance and/or simple concept. I never realized how lucrative/complex making a video game was until I started researching this topic and I feel like I have a better appreciation of game development as a whole now that I've dipped my toes into this relatively new industry. In the end, I hope to have encouraged others to try making a game of their own!

Works Cited

"[Interview] Moon Studios on the Process of Bringing Ori and the Will of the Wisps on Switch; iam8bit Talks COLLECTOR'S Edition, Reception, More." *Nintendo Everything*, 13 Dec. 2020, nintendoeverything.com/interview-moon-studios-on-the-process-of-bringing-ori-and-the-will-of-the-wisps-on-switch-iam8bit-talks-collectors-edition-reception-more/.

"Moon Studios Interview - Ori and the Blind Forest." *YouTube*, 11 Mar. 2017, www.youtube.com/watch?v=aAnXe2g1_MY.

Lawver, Bryan. "'FFXIV: A Realm Reborn' Director Yoshi-P Reflects on 5 Major Milestones." *Inverse*, Inverse, 27 Aug. 2021, www.inverse.com/gaming/ffxiv-realm-reborn-anniversary-naoki-yoshida-interview.

https://www.youtube.com/watch?v=Xs0yQKI7Yw4&t=2630s

"FINAL FANTASY XIV Documentary Part #1 - 'One Point O.'" *YouTube*, 23 June 2017, www.youtube.com/watch?v=Xs0yQKI7Yw4&t=2630s.

Larsson, Daniel Goldberg and Linus. "The Amazingly Unlikely Story of How Minecraft Was Born." *Wired*, Conde Nast, 5 Nov. 2013, www.wired.com/2013/11/minecraft-book/.

Cheshire, Tom. "Changing the Game: How Notch Made Minecraft a Cult Hit." *WIRED UK*, 15 Sept. 2014, www.wired.co.uk/article/changing-the-game.

Aleem, Saiqa, Luiz Fernando Capretz, and Faheem Ahmed. "Critical Success Factors to Improve the Game Development Process from a Developer's Perspective." *Journal of computer science and technology* 31.5 (2016): 925–950. Web.

Banks, John. *Co-Creating Videogames*. 1st ed. London: Bloomsbury Publishing, 2015. Web.

Gizem Boyraz, and Pinar Kirci. "Constructing A 3d Game With Unity 3d Game Engine." *Proceedings of the XXth Conference of Open Innovations Association FRUCT* 28.2 (2021): 554–557. Web.

"Gaming the System: How User Innovation Has Transformed the Video Game Industry." *Strategic Direction (Bradford, England)*, vol. 36, no. 5, 2020, pp. 31–33, doi:10.1108/SD-01-2020-0017

Stefyn, N. (n.d.). *How video games are made: The game development process: CG Spectrum*. How Video Games Are Made | The Game Development Process | CG Spectrum. Retrieved November 17, 2021, from https://www.cgspectrum.com/blog/game-development-process.

Ori and the blind forest: Definitive edition. Metacritic. (2019, September 27). Retrieved November 18, 2021, from https://www.metacritic.com/game/switch/ori-and-the-blind-forest-definitive-edition.

Ori and the Blind Forest Review. IGN. (n.d.). Retrieved November 18, 2021, from https://www.ign.com/articles/2015/03/10/ori-and-the-blind-forest-review.