WIZARD ADVENTURE GAME

By

Elias Sells

A capstone project submitted for
Graduation with University Honors

June 1, 2020

University Honors
University of California, Riverside

APPROVED

_____
Dr. Tamar Shinar
Department of Computer Science and Engineering

_____
Dr. Richard Cardullo, Howard H Hays Jr. Chair, University Honors

Abstract

This project originally began as a group project for my engineering senior design course. My group members were Chung Chin, Joshua Pedron, and Melissa Santos. The senior design course took place in the spring quarter of my 3rd year at UCR, so I used the extra time between then and the capstone deadline to continue working on the game and polishing it by myself. The reason I decided to make a video game for my capstone project was because game development is the job that I want to have in the future. I thought this project would be a good way to explore the process of game development and the flow of a game engine, as well as experiencing the challenges that come from working in a team and working alone and unguided on a project.

When I first got an in depth idea of what the capstone project was, I had no idea what kind of project I wanted to do. The honors faculty often mentioned that formal research projects were what students most commonly did, but I had no interest in that. As a computer science major, I've gotten the most enjoyment in my academic studies at UCR when I apply the skills I've learned towards a project where I get to create something. Designing a project, programming it, and testing it are what I would've preferred to do. Once I realized that my capstone could be just that, I knew for sure that I wanted to create a video game. I've been an avid fan of video games since I was four years old, and video games are still a major part of my life. Now that I'm an adult, I know that developing video games is what I want to do as a career. Creating a video game for my senior design project and the Honors Capstone felt like a great way to further my goal of becoming a video game developer. Creating a game would give me the chance to hone my programming skills, especially when it comes to gameplay scripting, which requires some skills I had not trained much in classes. It also would give me some insight on how a game engine operates, and the process of creating a world and entities within that world, and programming their behavior. I even would get to learn more about the artistic side of game development. Art, animation, and music are extremely important to the quality of a video game, but I've never been very talented in any of those areas, so it would be an amazing opportunity. After considering all of these potential benefits, I set out on a plan to develop a video game for my Honors Capstone.

The first stage in this plan was to start working on the game as an engineering senior design project. I have a hard time starting projects on my own and staying committed to them when there is a lack of guidance or near deadlines, so using the BCOE senior design class as a way to kick start this project would be the perfect solution for those problems. Senior design

projects are usually done in groups, so I got together with three friends of mine (Chung Chin, Joshua Pedron, and Melissa Santos) and we enrolled together in CS179N in Spring 2019. We all agreed that we wanted to make a video game, but figuring out the genre, the game engine to use, and the graphical style were all challenges we faced early on. The graphical style and engine decisions went hand in hand and weren't too hard to decide. None of us had much experience with 3D modelling, animation, or level design, so we decided to make a 2D pixel art game. Since we were working in 2D, we wanted an engine that would support it, so we went with an open source engine called Godot. Unlike more popular game engines 3D engines like Unreal Engine 4 and Unity, Godot has both a 3D engine/editor and a dedicated 2D engine/editor, so it was a natural fit for our project. Godot also has a very detailed documentation website and a helpful support forum, so those also helped us decide to use it. The final decision we had to agree on was what genre to choose for the game. Even when restricted to a 2D plane, there are hundreds of distinct genres to choose from. In the end, we decided on an Action/RPG/Platformer, also known as a "Metroidvania", which is a fan-made name combining the names of the popular video games series *Metroid* and *Castlevania*, which this genre was born from. Once the genre was decided, we started on the planning phase.

Planning for the project is probably where our group had the most room for improvement. It would be an understatement to say we vastly overestimated how much work we could get done during the 10 week senior design course. In reality, it was closer to 8 weeks of actually working on the project, since no work was done in week 1 and in week 2 we focused on figuring out the planning. To give a brief but great example: we planned for the "final" version of the game at the end of the quarter to have 12 levels. We ended up with 2. Thankfully, it didn't take us very long to realize we needed to reduce the scope of our project, so we were able to

focus on getting out something that was enjoyable to play. One of the main reasons we were able to quickly change the scope of our project and adapt to implementing new features was due to our use of what's known as Scrum. Scrum is a project methodology often used in modern software engineering that emphasizes being "agile" and allowing for changes in a project plan as needed. As we worked on the project and implemented new gameplay features, animations, or levels, we would keep track of it in our Scrum spreadsheet and compare what we've done to what we had set out to do for that week and our goals as a whole. At the beginning of every week, we discussed how the previous week had gone, and if we needed to adjust our goals for the project. After two or three weeks of developing the game, when we were still just figuring out how to accomplish tasks like scripting, level editing, and animating in the engine, we decided to tone down the goals for our project, and it turned out to be the correct decision. Scrum proved to be an invaluable tool for us in this project. It allowed us to produce a game that felt more complete and was fun to play, rather than appearing as a cobbled together mess. Here are some screenshots of our Scrum sheet, showing how we kept track of who completed a task, whether or not it was tested, and the priority/importance of that task:

Name	User Story	Acceptance Criteria	Story Points	Priority	Owner	Tester	Step	Status	Valid state
Player Sprites	Art	1. Idle Sprites 2. Animations	6.25	3	Melissa		2	not tested	OK
Player Movement	Player Behavior	The character's basic movement. The character should be able to move left and right as well as jumping.	5	1	Elias	Melissa	4	finished	OK
Backgrounds	Art	1. Generic Cave 2. Mushroom 3. Crystal 4. Slave Room 5. Wheat fields 6. The Gallows	6.25	3			0	not taken	OK
Set Pieces	Art	1. Armor pieces 2. Clothing 3. Platform animations	6.25	3			0	not taken	OK
Enemy Sprites	Art	1. Idle Enemies 2. Animation	6.25	3	Melissa	Elias	4	finished	OK
Melee Combat	Player Behavior	The player should be able to use a melee weapon for combat.	4	2	Elias		1	working	OK
Spell System	Player Behavior	The player should be able to cast a variety of spells.	4	5	Charlie		1	working	OK
Make level sketches	Levels / Map	The player should have several levels/maps to experience.	4	2.5	Josh		1	working	OK
Generic enemy	Enemy behavior	1. Sprite 2. Animation (movement / attacks) 3. Lifecycle	10	2	Melissa		1	working	OK

This first screenshot shows some of our planned features that never made it into the game, such as plans for 6 of the levels.

Name	User Story	Acceptance Criteria	Story Points	Priority	Owner	Tester				
Fire spell	Player Behavior	The player should be able to cast a fire spell.	4	4	Charlie	Melissa	4	finished	OK	4
Water spell	Player Behavior	The player should be able to cast an water spell.	1.25	4	Charlie	Elias	4	finished	OK	1.25
Earth spell	Player Behavior	The player should be able to cast an earth spell.	1.25	5	Charlie	Josh	4	finished	OK	1.25
Backgrounds	Art	There is a generic forest background for a level.	2	2	Charlie	Melissa	4	finished	OK	2
Player Movemen	Player Behavior	The character's basic movement. The character should be able to move left and right as well as jumping.	15	1	Elias	Melissa	4	finished	OK	15
Player Movemen	Player Behavior	The character's basic movement. The character should be able to move left and right as well as jumping.	10	1	Elias	Melissa	4	finished	OK	10
Melee Combat	Player Behavior	The player should be able to use a melee weapon for combat.	5	2	Elias	Melissa	4	finished	OK	5
Lightning spell	Player Behavior	The player should be able to cast a lightning spell.	4	4	Elias	Charlie	4	finished	OK	4
Enemy HP	Enemy behavior	Enemies must have HP	5	3	Elias	Charlie	4	finished	OK	5
Enemy sprite	Enemy behavior	Enemy has a unique sprite	10	2	Elias	Josh	4	finished	OK	10
Defeat	Player Behavior	The game should result in a "defeat" when you run out of health.	5	6	Elias	Josh	4	finished	OK	5
Enemy attack	Enemy behavior	Enemies must attack player	2.5	3.5	Elias	Charlie	4	finished	OK	2.5
Spell touch-up	Player behavior	Boulder rolls and looks better visually	2.5	5	Elias	Charlie	4	finished	OK	2.5
Spell touch-up	Player behavior	Lightning has a "forked" effect and looks better visually	2.5	5	Elias	Charlie	4	finished	OK	2.5
Spell touch-up	Player behavior	Fire shoots in bursts and looks better visually	2.5	5	Elias	Charlie	4	finished	OK	2.5
Spell touch-up	Player behavior	Player regains health upon casting water spell	2.5	5	Elias	Melissa	4	finished	OK	2.5
Enemy lifecycle	Enemy behavior	Enemies spawn and respawn or stay dead	3	3	Elias	Josh	4	finished	OK	3
Background Music	Music / Sound	Game has background music at all times	5	3	Josh	Charlie	4	finished	OK	5
Backgrounds	Art	There is a generic mushroom/crystal background for a level.	4	2	Josh	Melissa	4	finished	OK	4
Backgrounds	Art	There is a generic cave background for a level.	2	2	Josh	Charlie	4	finished	OK	2
Make Tilesets fo	Levels / Map	Each stage has a unique tileset. (Level 1)	5	4	Josh	Melissa	4	finished	OK	5
Original Tile Sets	Levels / Map	Each stage has a unique tileset. (Level 2)	5	4	Josh	Charlie	4	finished	OK	5
Backgrounds	Art	There is a generic underground background for a level.	2	2	Josh	Elias	4	finished	OK	2
Enemy Sprites	Art	1. Idle Enemies 2. Animation	10	3	Melissa	Elias	4	finished	OK	10
Player Sprites	Art	1. Idle Sprites 2. Animations	10	3	Melissa	Elias	4	finished	OK	10
Enemy Sprites	Art	1. Idle Enemies 2. Animation	10	3	Melissa	Elias	4	finished	OK	10
Making platforms	Levels / Map	Player must move with the platform	2	1	Melissa	Josh	4	finished	OK	2
Enemy animatior	Enemy behavior	Enemy has animation for its attacks, movement, death, idle, etc.	3	4	Melissa	Elias	4	finished	OK	3
Set Pieces	Art	Player has their own unique clothing	6.25	3	Melissa	Josh	4	finished	OK	6.25
Platform animati	Art	Platforms have animations	4	3	Melissa	Elias	4	finished	OK	4
UI	UI	The player should have a display for important information, e.g. health, mana	10	2	Melissa	Elias	4	finished	OK	10

For the majority of the project, we gave ourselves specific roles so that we could become specialized in a particular field and become more skilled over the course of the project. This would allow each of us to excel in our specific tasks and create a better game than if all of us had to do a little bit of everything. The four main roles we split ourselves into were level creation, sprites/animations, gameplay scripting, and Scrum management/generalist. My role was gameplay scripting, so that's what I'll be focusing on here.

The Godot engine uses a proprietary scripting language called GDScript. It's very similar to Python, which is a programming language I was already familiar with, so thankfully learning the syntax of GDScript was very easy. Gameplay programming as a whole was very challenging, however. Even with three years of computer science courses at UCR under my belt, many aspects of gameplay programming proved to be difficult. The main difference between programming in computer science courses versus programming for a video game is that there's no correct way to do things. For example, in a lab project at UCR, typically you will have to solve a relatively boring programming challenge, such as sorting an array of numbers in a certain amount of time. There's an explicitly laid out goal, and it should be easy to know when you've reached that goal. With gameplay programming, this could not be further from the experience. Something as simple as programming the player's jump ability took me two weeks to finish, and even then it was a mess. There are what feels like countless aspects you need to consider when designing and implementing a jump function. What button should the player press to jump? What should happen if the player holds down the jump button versus letting go of it early? What should the maximum height of the jump be? How fast should they fall? How much should they be able to control their air speed? How can you make sure a player can only jump when their feet are planted on the ground? These are just some of the questions you have to ask yourself when

designing a jump function. However, even when you've implemented the entire thing and it works the way you've planned, you might have to throw it all out because it doesn't "feel" right. In the end, the feeling of what you've programmed is the single most important thing for the game. If it doesn't feel fun to play, or if players are frustrated by the mechanics of the jump, it was a wasted effort. Luckily, it only took a small amount of trial and error before I implemented a jump function that felt good to use.

The jump function was just one of many features to implement, however. As time went on, we realized that we had bit off more than we could chew, so to speak. Just as we reduced the number of levels from 12 to 2, we decided as a group to cut back on some of the gameplay features and systems we wanted to include. For example, originally the player was going to be able to collect elemental crystals around the level. They could combine different types of these crystals together to create new spells that they could cast. For example, if a player finds two wind and one fire crystal, they could combine these into a fire tornado spell that they could unleash upon enemies. For a spell system like this to be fun to use and have enough different spells to create would be a lot of work; not just for me, the gameplay scripter, but also for the person in charge of creating the animations and visual effects for the spells. We decided to change this spell accumulation system to something much simpler. The player would have 4 spells given to them as soon as they start the game, and they could use them whenever they want, with some restrictions. However, this turned out to be a very fun spell system for players, despite being born out of impending deadlines. One spell in particular, which is my personal favorite, is a boulder that rolls around the level and bounces off of walls with semi-realistic physics, crushing any enemies that stand in its path. Learning to create fun, engaging, and quality gameplay and levels despite the small amount of time was one of the main things we learned

from this project.

At the end of 10 weeks, despite our initially lofty goals, we were able to cut back on enough features while keeping the ones that were important to make a fun game. The gameplay that is in the final version of our project can't really be called a Metroidvania, the genre we were originally going for. The action and platforming elements are there, but the player is not able to gain any permanent power-ups or new abilities that allow them to access new locations of the level. This is a key component in the Metroidvania genre that separates it from other Action/Platformer games, and without it we can't consider our game as a part of that genre. However, I think we created a good base that features could be added to in order to make a Metroidvania game.

The gameplay that we did get to implement focuses on the player exploring a level. As the player travels to new places, they will encounter platforming (jumping) obstacles, a variety of enemies to battle, and even some traps for the unsuspecting. The player fights against enemies by using a basic sword swing attack with low damage and range but has no cost to use, or they can choose to use powerful spells which cost mana points, a resource that the player has a limited supply of. As the player traverses through the level, they will come across a boss enemy that deals more damage and is harder to defeat. After defeating the boss, the player will gain access to the next level.

I look back on the project and I am fairly satisfied with what we were able to make in just 10 weeks. The only thing I would've liked to be different was to have more time like most other engineering senior projects, but that was never an option anyways. Another thing I could've done better in this phase was looking up more tutorials. Trying to brute force figuring out how to implement certain difficult features (like the jump function) would have been easier if I tapped

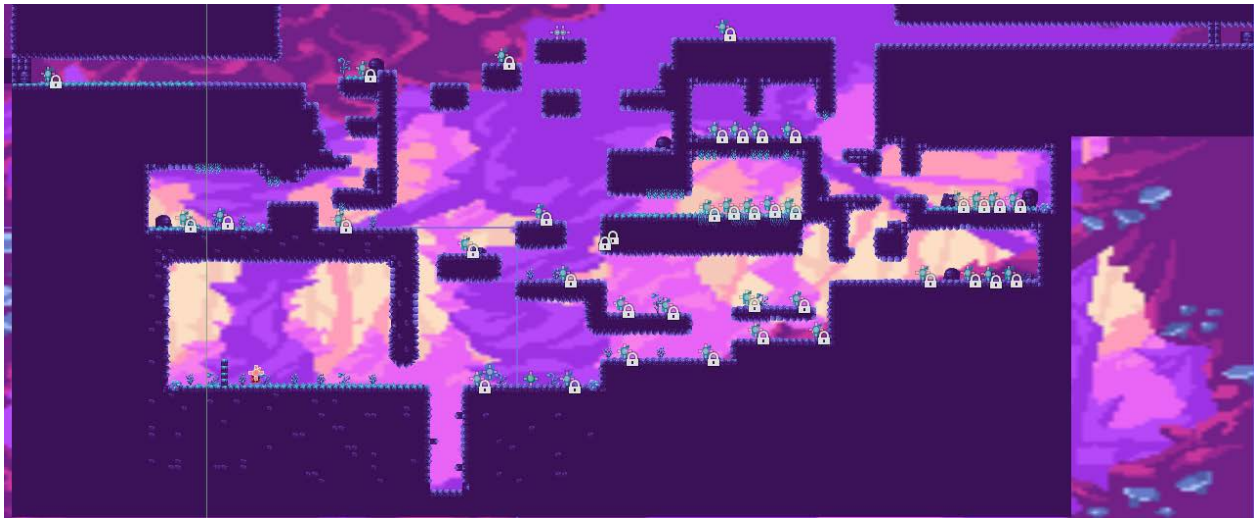
into the vast amount of resources that there are online.

Below are some screenshots of the game. Some of them were taken recently after additional changes (which I will talk about next), but the look of the game has not changed since we finished the senior design class. Some screenshots will also be of some development tools and level editing.

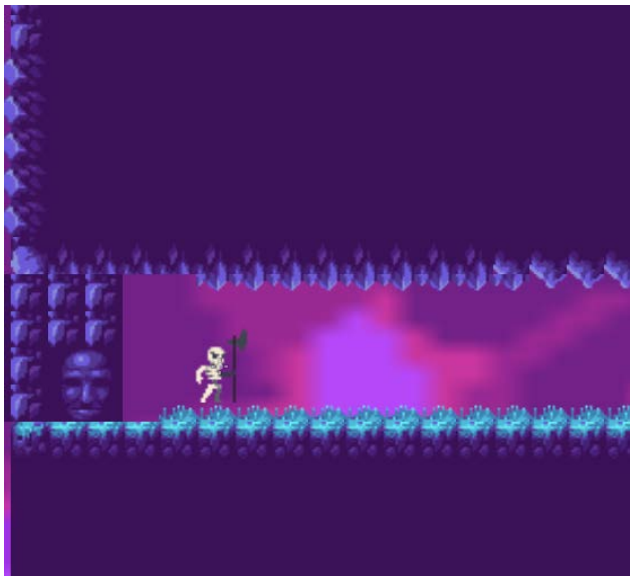
The boulder spell in action, barreling towards an unaware skeleton. It can’t properly be shown in a picture, but the boulder uses a clever sprite rotation effect that gives the appearance of rolling as it moves across the screen. When it hits a wall, it will slow down slightly and start “rolling” and moving in the opposite direction.

A spectral ghost enemy shoots a shadowy projectile at the player who swiftly jumps to avoid it. Some enemies will approach the player to use a melee attack, while some, like this one, will launch projectiles from far away. The player must use all of the tools at their disposal to defeat the enemy monsters.

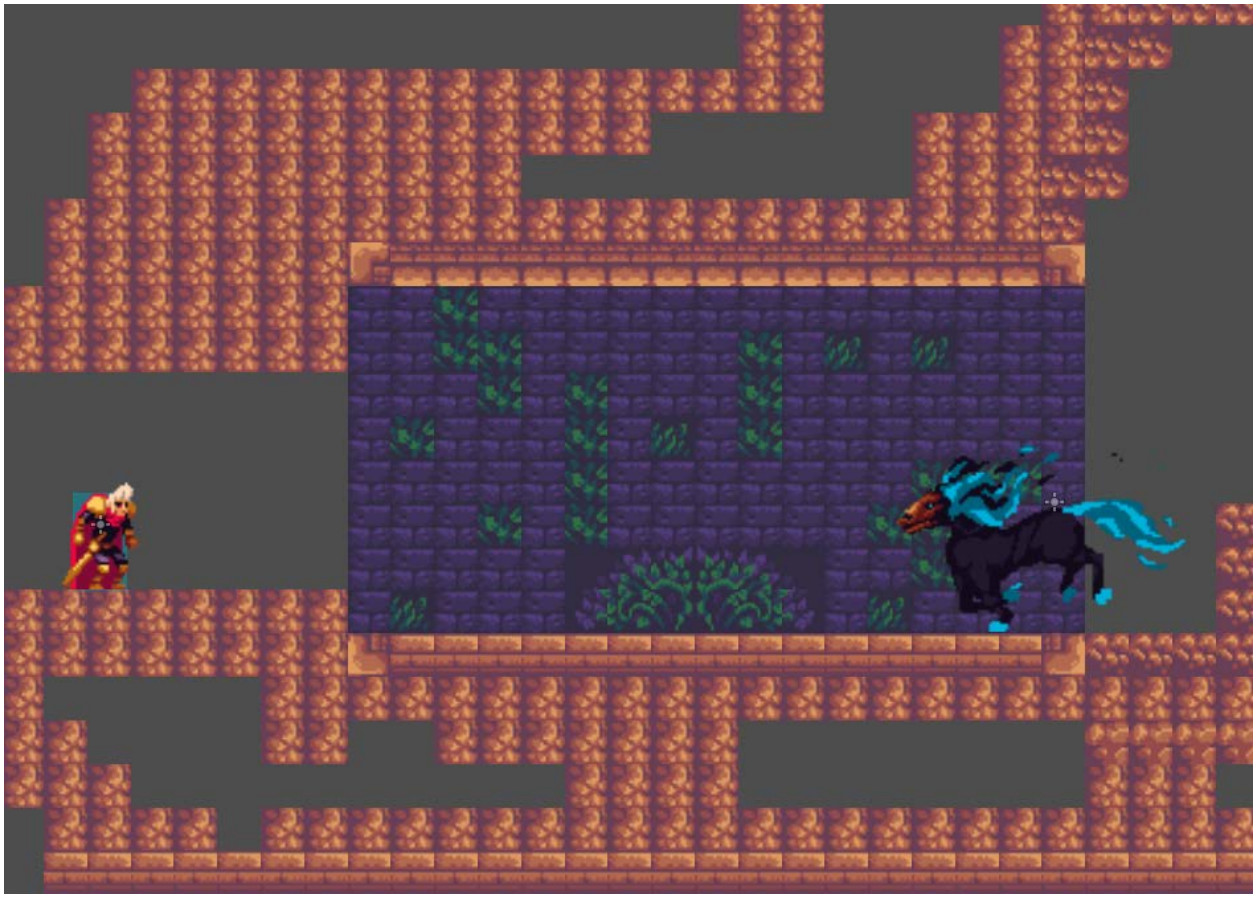
The player walks into a trapped room full of skeletons and dies as they attack him. You'll need to be careful to avoid this sort of thing as you play the game.



A full overview of how the first level looks inside the engine/editor. The orange + symbol in the lower left corner is where the player begins, and each blue + symbol with a lock next to it is where an enemy is located.



The “Super Skeleton” boss of the first stage. He looks like any other skeleton at first, but players will understand the threat he poses when they notice his faster movement and higher damage. Behind him is a portal that leads to the next level.



An in-editor screenshot of an in-development level. We weren't able to fully create this level by the time we had to submit the project, so we scrapped it, hence the unfinished look. It was planned to have different enemies from the first level, such as the "Hell Horse" pictured here across from the player.

After the senior project was finished, I still had a whole year left before graduation. I knew I wanted to use this game as my capstone, but with all the extra time, I decided that I wanted to continue working on the game and improve it. There were a couple different ways to improve the game, and I had to pick one to do. I could go back and look at some of the things we cut from the original project plan and try to implement them. This would include some features such as an expanded spell system, one or two additional levels, and maybe even some more enemy variety. The other route was to refine what was already there. I would be taking a look at the code I had written and making it cleaner and more efficient, and, most importantly, it would *feel* better in gameplay. I'd also take a look at some of the unfinished or buggy visual aspects of our game, such as certain animations not working properly. I decided to go with refining what we already had made. The first reason I decided to do this was because I thought it would give me more experience in relevant skills such as programming. Taking "first draft" code and re-writing it to make it better is an important aspect of programming, so I wanted to take this chance to grow as a software engineer.

I had an internship during the summer of 2019, so I didn't start working on this project on my own until around fall quarter. Once I did start, the first step was to figure out what aspects of the project I wanted to improve. As I thought back several months to the senior design class, immediately a memory came back to me: the jump function. I knew that it would have to be the first thing I looked at improving. With that, I knew what I wanted to do first, but I didn't know how to start. Just trying to read through my code from not even half a year prior made me scratch my head and wonder what I was thinking back then. It was unoptimized, hard to read, and it took up over one hundred and fifty lines of code, which is a lot for a simple jump function. To figure out how to fix this, I decided to look at some tutorials, as well as the Godot engine

documentation. I came across a YouTube channel called Game Endeavor which is dedicated to making game programming tutorials in Godot. It was the exact thing I needed in order to figure out how to improve my code and make something that felt better to play. The channel even had a video about making a jump function. It was exactly what I needed to learn how to make the player's jump feel so much smoother. After working on it for a few days, the jump function I created was easy to read, optimized, and I even made it so I could enter a specific jump height for the player, while the old function had me guessing numbers until I stumbled upon a jump height that worked. It was also only about fifty lines long, as opposed to the previous one hundred and fifty line long version. Most importantly, it felt great to use. The jump felt like it had momentum to it and it was easier to make a precise jump.

Once I had successfully implemented a better jump for the player, my confidence grew and I started to tackle some other aspects of the project that I wanted to improve. I mostly focused on the player character, but I also did some work on the enemy you see the most, the skeleton. For the player, I implemented some additional animations that went unused in the final senior design project, such as a dedicated animation for attacking while mid-air and attacking while crouching. These helped gameplay feel more fluid and it made the actions that the player would try to perform work as they would expect. It also helped me learn about pixel art animation and how it works in a game engine like Godot. These changes allowed me to make a game that I felt was even better than the one we originally made for the senior design class, and I'm glad I did them.

Overall, I'm very happy with how this project turned out. In terms of learning experience, I was able to get everything I wanted out of this project. I became a better programmer, I learned how to work using Scrum methodology, I got a chance to work both in a team and on my own,

so I was able to see how these experiences are different in game development. While we weren't able to meet our original expectations for the project, we still created something that was fun to play and that I am proud of.

Link to game executable on Google Drive:

(Windows)

https://drive.google.com/file/d/1_Ztgra65fhYzYd3pAuuhyPZlHwpg3TXn/view?usp=sharing

Controls:

- Left/right arrow keys to move
- Up arrow key to jump
- Down arrow key to crouch
- 'X' key to melee attack
- 'Z' key to cast spell
- '1', '2', '3', '4', keys to switch between the four spells.

Works Cited

"What is Scrum Methodology?" *CollabNet*, 2020, https://resources.collab.net/agile-101/what-is-scrum

Game Endeavor, *Youtube*, https://www.youtube.com/channel/UCLweX1UtQjRjj7rs_0XQ2Eg/featured

Godot Community, *Godot Docs*, 2020, https://docs.godotengine.org/en/stable/index.html