Project Spellda

James Nail
Sharon, USA
jamgnail@ut.utm.edu

Logan Brown
City, Country
e-mail address

Steven Gray
City, Country
e-mail address

ABSTRACT
Our project is Project Spellda, a two-dimensional, top down role playing game developed in Godot. Our game will minimally include 3 different levels: a tutorial level to introduce the mechanics of the game, followed by an overworld and dungeon that are intertwined. At the beginning of the game, the player will choose two of the four in-game elements: Fire, Water, Air, and Earth. The player then explores the surrounding area, looking for upgrades to their default spells. At any time, the player can dip into the dungeon to test their abilities on the stronger enemies that await them, with the end goal being to get to the center of the dungeon and slay the boss. The game is made entirely using Godot's built-in scripting language GDScript.

Author Keywords
Authors' choice; of terms; separated; by semicolons; include commas, within terms only; this section is required.

1. INTRODUCTION
Project Spellda is a top-down 2D arcade style RPG game, mixing elements from the likes of Legend of Zelda among other dungeon crawlers and RPGs of the era. The style of the game came from the games "Thomas Was Alone" and "Illumine". The game will have an overworld with a couple different enemy types, along with several offshoot dungeon-like areas and one main dungeon existing under the overworld. The dungeon will be relatively harder than the overworld, with better loot and harder bosses, as well as mechanics used in the dungeon-like offshoots found scattered in the overworld. The overworld will have varied difficulties across numerous zones, with several of these difficulties tying directly to difficulties found within the dungeon.

The main feature of the game is the lighting style, which keeps anything that isn't in direct line of sight pitch black. This makes for interesting ways to explore areas around you, never knowing what is around the next corner.

2. TECHNICAL SPECIFICATIONS
We are using the Godot game engine[**?**], with much of our scripting done in a built in language built specfic to the engine; GDScript (Godot Script). This language is very similar to Python, and has everything you need to use the engine to its fullest. This language even has specific commands for just the engine, which makes things very customizable.

For Spriting, most of that was made in MS Paint. The simple measuring tools made it easy to have properly sized sprites that stuck to the simplistic art style we were shooting for in this project.

Some of our knowledge came from a wonderful playlist on youtube (from the youtuber "jmbiv"[**?**]), showing how to do certain things so we could translate that knowledge to our project.

3. MOTIVATIONS
Why did we choose to make a video game for our final project? Well, the answer is simple. Each of us had little experience in complete game design, but we were all avid gamers that were eager to learn what goes into the titles we all know and love. We saw this as a common point of interest and pursued it. Though, this wasn't the first idea that was floated around.

Early on in deciding what to make, there were a few thoughts. Minecraft modding was a possibility, but we all agreed that coding into something that already exists could be more trouble than it's worth to figure out. Once that was ruled out, outright game developement was suggested and was a favorite. We eventually settled to use the Godot game engine, but only knew roughly that we wanted a game with magic in it.

4. GAME OVERVIEW
Once we had a general idea of what we wanted to use for game-making, we started to iron out details. So, we made an outline of what we wanted in the game.

The first thing we wanted was three different areas that the player could explore; A tutorial, an overworld, and a dungeon. The tutorial area would be, of course, a small place that players could learn in and test controls in. The overworld would be a larger area, in which the player would explore and get into all kinds of trouble. The dungeon was to be the final place the player would encounter, and have trials fitting for someone who had made it thus far.

Secondly, we wanted a minimum of 4 types of enemies. We generally categorized this further by saying we wanted two melee enemies and two ranged enemies. While we had no specific ideas, we were thinking of having it so that enemies would be designed as simple shapes.

Third, we wanted the player to have the option of 4 different elements to use in spellcasting. To clarify; the player can only have two spells at a time, but they get to choose two from a pool of four. This was just our starting point, as the four basic elements (water, fire, earth, air) were the first things to pop to mind. We considered that in the future we could add different, not-so-common types of magic to the game.

Finally, we wanted different puzzles and "offshoot" areas that would encourage the player to explore around the maps and really find everything they can. This was really from our own experiences as players, making sure that there was plenty to do that was interesting in this game.

5. INSPIRATIONS

The main inspirations for our game were Thomas Was Alone, Illumine, and Legend of Zelda. Each contributed a different mechanic, with all three tying together in the end rather nicely.

Thomas Was Alone contributed a majority of the sprite design to the game. We decided to go with the minimalist art style, using colors and shapes to define the world instead of drawn or illustrated sprites that had depth. The strict focus opened up a new dimension to play with, that being color denoting areas or objects of interest in the world, with shapes denoting more complicated encounters or mechanics based on the shape and color combination.

Illumine contributed lighting and shading, and lots of it. We initially did not know the direction we wanted to go in when designing the particle effects and lighting, and this showed early on. As we developed the game, however, we quickly realized we could leverage lighting to our advantage, and used it to hide all sorts of stuff from the player, such as hidden passageways and enemy traps. This lighting bled from the dungeon, where it originated, into the overworld, allowing some crazy overworld designs, such as writing words on the map and literally using it as an obstacle. The player would never know the difference because they can't see all of it.

Finally, Legend of Zelda contributed the map style. The map style of Legend of Zelda played very nicely with our lighting and shadow style, allowing for almost hidden feeling rooms, dark dungeons, and otherwise visually interesting areas. This primarily made its way into the overworld, with the dungeon being a more linear straight shot through, but traces of the Legend of Zelda dungeon style can still be seen in the dungeon layout.

6. CHANGES FROM INITIAL DESIGN

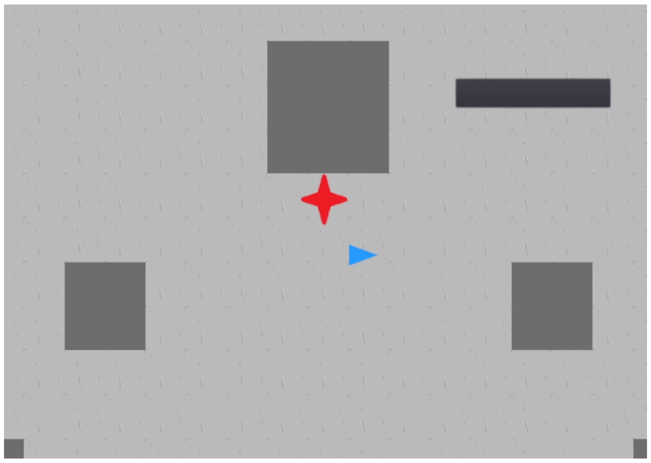
We initially planned on this to be combat-centric, focusing on the spells and systems therein. This changed drastically once we realized what was possible with our given inspirations, and we combined the three in a different way. What began as a spell-caster morphed into an exploration game with combat on the side, with lighting growing to be the main focus. The combat just wasn't anywhere near as interesting as the lighting, nor was it as eye-popping as some of the visual effects we experimented with along the way.

Lighting played a huge role once we decided on shapes and colors. Dim or bright lights began to take on meaning, while colors began denoting friend or foe. This was the point during which the game experienced the largest growing pain, that being the shift in focus from combat to lighting. This required the offshoot areas, part of the overworld, and the dungeon to go back to the drawing board for a bit. While this did set us back a little, we were still able to deliver the project in a timely manner.

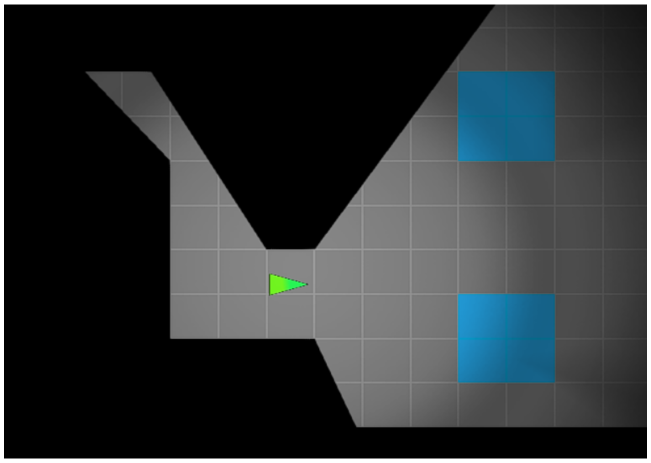
Once we had lighting figured out, shadows were a natural next step, and it did not disappoint. The ability to combine creative shapes and colors with shadows to hide most of it created an environment the player wanted to explore, rather than an environment the player had already visually explored upon immediately entering any given room. We tossed around the idea of having light be an environmental thing and the player casting these shadows, but this was quickly scrapped due to implementation difficulty and lack of interesting visual choice.

The puzzles were a massive paradigm shift from the original combat idea. We went from being a game about combat, elemental interactions, and spell-slinging, to a focus on visual design, exploration, and detail. This was, by far, both the biggest change in the project as well as the easiest. We had a very easy time making the transition, even though it was sudden and required us to return to the drawing board for large chunks of the game.

Here is what our design looked like before the consideration; Dull and flat, with little room for improvement in the direction we were going.



Here is the game with updated lighting, which we considered a massive improvement upon what we were planning originally.



Another leap in development occurred with particles, shortly after lighting. Particles were something we wanted in some capacity, since they were usually good indicators and good player feedback, but we weren't sure how or where we wanted them. Eventually we started with just basic door destruction particles, then went to plate activation particles, then on to damage particles. This provided a sense of feedback for the player so they knew when they did something meaningful to the surrounding world.

A lesser discussed aspect that changed was a lot of back-end implementational details. We fundamentally changed how the spells worked at least once or twice behind the scenes, and the spells were their own separate nightmare to get working. Once we got spells working, though, not too much else got reworked side from one major code overhaul for the dungeon and its mechanics.

7. BUMPS WE HIT ALONG THE WAY

One bump we hit was lighting. Even though it was a central focus of the game, it caused a fair amount of problems early on in implementation thanks to how godot handles lighting and shadows. In godot, lighting and shadows are handled based on lighting layers, which is not uncommon. The strange bit, however, is the occlusion detection - it does not automatically occlude light when you define occlusion boundaries. You must tell the light it is specifically allowed to cast a shadow, then define occlusion boundaries - this was not made abundantly clear in the documentation during creation of the lighting systems within the game, which set us back a tiny bit. You also must use a CanvasModulate node, which applies a filter over the camera/viewport, to darken the scene so the shadows will actually show up and the light will have an effect. This, also, was not made abundantly clear in the documentation when designing the lighting. These options are extremely flexible and allow for very fancy lights and shadows, but attempting to get them working was a minor hassle.

Another hurdle to get over was vector-based movement. Like said earlier, game developement was something we all were not very experienced in; And vector-based movement is vital to almost every project that goes beyond a simple, non-stop side scroller. When this was initially approached, we wanted to implent this into our spellcasting for the projectile movement. With a little research done for general game developement, the first attempt was making a Vector variable that was the product of a directional vector and any arbitrary speed value we chose and then just trying to apply it directly to an instance of our projectile (or in other words, a projectile that we spawn as part of spellcasting). Once we played around with it and tried different things, we actually settled on simply changing the *local* y-coordinates every frame by a set amount, until the projectile collided with something or was far enough away. With that taken care of, we moved on to different things, including enemies. Enemies were its own bag of fun, but ultimately we could not compromise on vector-based movement here. This took way longer than it should of, but eventually with the help of the magic we call "team work", we were able to implement this. I will save the trouble of all the details, but we were fairly close to what our initial idea of how vector based movement should work. Godot just has a particular way that it likes to handle this type of movement in a 2D space, and it was simply a matter of getting a better understanding of our tools available to us.