Overview

Abysswalker is a 2D Survival game with Dark Fantasy themes.

This project was inspired by dark fantasy games such as Bloodborne or Dark Souls due to my fascination with the genre like Lovecraftian horror. My vision was to create a 2D Survival Dark Fantasy themed. I was also inspired by the survival waves in Terraria.

The core gameplay revolves around the player surviving waves of enemies that spawn on either side of the screen, cornering the player. Players can get Abyssal Essence from slain enemies which can then be used to upgrade the player's stats at the end of each wave. Each wave lasts for about a minute with a total of 10 Waves and a Boss spawning at the end. The player needs to manage which stat to level up first in order to beat the final boss at the end.

What Went Right

The core basics of the game was implemented around a timespan of a week, creating a solid foundation. Like the basic character movement of moving left or right as well jumping and dodging. But when it came to the simple upgrade system and the initial framework for player UI health and stamina, they were implemented towards the end of the development. The implementation of the player input for movement and mapping it to keyboard and controller was easy and direct. I was able to get the character playable on the screen which gave me a sense of motivation because of the hard work I see on screen.

I focused on mechanics that felt right. I tested and adjusted different settings such as how fast sprite frames would play as the player slashes his sword, how long the frames of the sprite lasts, the delay before the slashing again, how quickly it responds to the control key and how it works with other movements like rolling and jumping until everything feels good. Previous versions where the mechanics felt awkward and clunky. So I made sure that the player would be in states where certain actions such as attacking cannot be overridden while also doing certain actions like jumping.

The process of implementing and animating sprites went well. This is because I used Figma, Although there were some challenges in the initial creation of the sprites, using Figma to standardize the individual sprites into a single sprite sheet, like importing the unified sliced sprites and grouping them into individual frames for animations like jumping, idle, walking,

attacking, rolling and so on. Additionally GIMP was used to group background assets into bigger tiles.This method helped the game to stop lagging in busy sections, the assets are loaded faster and the player moves smoothly through the environment elements. I ensured all frames made it possible for the character/entities and environment background elements to work, avoiding animation jittering. This greatly improved the visual aesthetic and play of the game.

What Went Wrong

I experienced a lot of issues as I was developing projects with many sleepless nights. At first it was how to get the sprites up and going but once I overcame the problem, another one would arise. Not to mention the biggest thing that went wrong was time. I did not manage my time properly during the development of this project as I was still trying to set up the GP Framework during the holidays and doing IGP slipped my mind. I would have also started with the features first rather than the sprites because I wasted a lot of time trying to make the game look visually good rather than a lot of features. Also looking for assets for my game was also simply too time consuming and most of the good animated sprites were locked behind a paywall online. And in a genre such as Dark Fantasy, it would be hard to find a sprite sheet that is animated and looks good with my desired colour palette. I also encountered issues with player physics in game as in the early stages of my game, the player would infinitely gain velocity after they jumped which is obviously not what i had in mind.

Also due to time constraints, I had to cut some content such as player skills, abilities and I was thinking of having the game to be a platformer at first but I just couldn’t really get it to work. Cut content that I had removed was a necromancer feature where the player would be able to summon enemies that they have killed, up to 5/5 max summons and they would be 20% of their original strength. I did not realise that testing and debugging when it comes to big projects like these took so much time even for testing player movement or enemy movement because they would break when you least expect it. Now I understand why most game companies would take years or sometimes delay the release of the game, simply because of the fact that it took too much time.

A lot of issues, some at the start when I was trying to get a feel with sprites, mostly in the middle of development, were managing the sprite assets. The previous attempts of integrating and

I had too many layers of background images slowing things down so I used GIMP to have them all in one image optimizing the background assets, keeping things organized to reduce lag. I also had an issue where when I used background images exported from Figma, they would be in a lower quality so I used GIMP and I saw that there was no quality loss when using GIMP.

Lessons Learnt

Lessons I learnt is that I underestimate the amount of time needed for development, for debugging and for polishing the game. I also had too many features in mind that I wanted to add but I simply just did not have the time for them.

Start simple, build smart. I learnt the best approach is to first make the basic system work, only then to add more complex features. This way I will always have a working version and future issues will be easier to fix early, meaning time is saved from not wasting time on complex systems that you would eventually want to scrap.

Simple basic game mechanics can have unexpected bugs. I learnt to always have extra time for testing and fixing bugs. For instance small tweaks like how the player attacks the entity can take longer than anticipated. In the long run these things can consume your precious time.

Beneficial learning of specific tools like Figma and GIMP to solve asset-related problems led me to also gain skill in using these tools and more knowledge about quick problem-solving.

Conclusion

In conclusion, the development of Abysswalker was difficult yet very rewarding. I encountered during the project struggles mostly with physics implementation and initial asset implementation, the process of problem solving which provided me great learning opportunities as to what could be done better. Also having the good stuff that succeed, got the basics first and learnt to utilize Figma and GIMP to standardize messy sprites. The tough part taught me to use time wisely, that physics is way harder than expected, as movement can be heavy or floaty and worse clipping through environments and with a new added element is a new, leading to hours spent on debugging.