Max Valle

CITA 212

December 9, 2024

“Forest Frenzy” Documentation

Game Link: Forest Frenzy

Github: https://github.com/Valle94/2DRPG.FinalProject

This game is a top-down 2D RPG demo with a strong emphasis on combat, an inventory system, and visual feedback. It was made using Unity 2D.

Primary Features:

- Player character that has three different usable weapons, each with its own attributes and animations. The sword is a melee weapon that slashes in a hitbox in front of the player. The bow is a long-ranged weapon that shoots an arrow projectile. The magic staff is a medium-ranged weapon that shoots a laser that deals more damage than the sword or staff.
    - Movement is done with WASD, and aiming and shooting are done with the mouse. Use SPACE to ‘dash,’ which is actually just a momentary increase in speed. There is limited stamina, however, which both refreshes gradually over time, and can be regained instantly through stamina orbs.
- Inventory system contains all three usable weapons and allows swapping between each of them. This system uses combination of ActiveInventory and ActiveWeapon classes to handle switching between both the visuals and the logic of the weapons.

- Spawnable pickups: a health orb that restores health, a stamina orb that restores stamina, and coins, which don't currently serve a purpose. These pickups spawn randomly when destroying grass or enemies.
- Multiple enemy types: a slime, a ghost, and a grape. The slime is a basic enemy that just moves and damages the player on contact. The ghost has a special 'shooter' class attached that makes this enemy shoot repeatedly in an arc. The grape enemy 'lobs' a single projectile

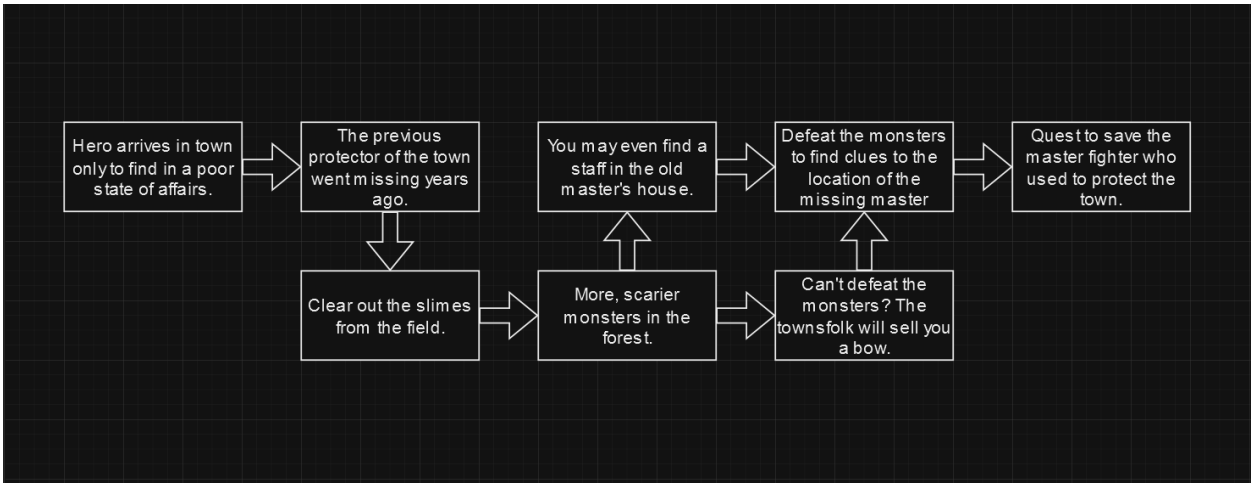
Additional Functionality

- Multiple scenes and transitions that include specific 'entrances and exits.'
- Using a singleton class, both the player and camera attributes are persistent through scene transitions.
- Tilemaps and layers for environmental effect. There is a 'base' layer containing grass, a 'foreground' layer that has the ledge terrain that acts like a wall that can't be shot through, a 'water' layer with water that can't be walked through but can be shot over, a 'cosmetic' layer that has all the paths which serve no functional purpose other than looking nice and indicating where to go, and finally a 'canopy' layer that has the green canopy that uses a parallax effect.
- Visual effect for almost everything: Attacking, dealing damage, taking damage, scene transition

Creative Choices

- I didn’t make a ton of creative choices for this project, but I did design my own weapon sprites. The sprites I created are mySword, myBow, myArrow, myStaff_bottom, and myStaff_top.
- The game is designed to allow for many creative choices in the future. The tilemaps and prefabs are all structured in a way that making new custom levels in the future is very easy. All you need to do in a new scene is drag in the prefabs for camera, managers, and the UI, and start adding tiles or environmental objects from there.

The storyline is pretty basic and is as follows:



Known Bugs:

- In the Unity Editor, when leaving play mode there is an error about a coroutine not being able to start because the canopy is inactive. As far as I can tell, the canopy *is* active in both the editor and in game. However, there is some unexpected behavior that I think is relevant: if you are standing behind an object that can fade, like the canopy, and then

attack, the canopy stops being transparent, and the scripts that handle the transparency start acting differently.

- The second known bug is with the grape enemies. They should only shoot when the player is close to them. However, when entering a scene where that enemy exists, they will shoot once at the player when they spawn into the scene no matter where they are relative to each other in the scene.
- The last bug isn't so much a bug as something I neglected in level design. Each of the areas where you can move to a new scene doesn't include walls, so enemies are able to just walk off the map if their random movement takes them in that direction. Fixing this is as simple as adding a little bit of terrain off-screen that the enemies will collide with.