Ruby RPG
A 2.5D game in Godot Engine

Lariza Julia Bucao Rodrigo
19718171

Final Year Project – 2023
B.Sc. Single Honours in
Computer Science and Software Engineering

Department of Computer Science
Maynooth University
Maynooth, Co. Kildare
Ireland

A thesis submitted in partial fulfilment of the requirements for the B.Sc. Single Honours in Computer Science and Software Engineering.

Supervisor: Ralf Bierig

Table of Contents

Declaration

I hereby certify that this material, which I now submit for assessment on the program of study as part of **B.Sc. Computer Science and Software Engineering** qualification, is *entirely* my own work and has not been taken from the work of others - save and to the extent that such work has been cited and acknowledged within the text of my work.

I hereby acknowledge and accept that this thesis may be distributed to future final year students, as an example of the standard expected of final year projects.

Signed: Lariza Julia Bucao Rodrigo Date: 27/03/2023

Acknowledgement

I would like to thank my supervisor, Ralf Bierig, for approving this project proposal and granting me the opportunity to work on this idea as part of my final year project. And additionally, I am grateful for his guidance and time given to me to improve on my ideas.

Abstract

Game development is a complex field that involves a deep understanding of the target audience's preferences. It requires a rewarding enough experience to make a successful game, involving the creativity in game mechanics, engaging storyboards and descriptive character designs. The topic offers a completely different way of interacting with technology and exploring new worlds and features, allowing the players to learn within an environment that will bring them to the final conclusion of what the game has to offer. This requires a careful balance between the level designs, story, aesthetics and true understanding of the audiences of interests. This report will go through all the steps, from concepts, designing and testings to get a full working draft of the game of interest. The final project being an experimental draft design of a world, with a working quest and story system of a classic role playing genre that has existed since the beginning of the gaming industry. While game development is a complex, long and challenging process, it allows many ways to bring an idea and imagination to life. With the right approach and consistent effort, the creativity presented can lead to a truly amazing experience for the players.

Chapter one: Introduction

Summary

This project is a unique idea I have presented to my supervisor. The idea was developed earlier in 2022 , after playing games that have given me a really exciting experience. I designed a pixel animation of the character of ‘Little Red Riding Hood’, and this led me to developing a game as part of my final year project. The goal of the project is to make an engaging world for the player, and allow the player to interact with the environment to reach the end game conclusion, using the mechanics throughout the game and learning about the main character and how the story revolves around her.

1.1 Topic

The type of game that will be made is a 2.5D Role Playing Game (RPG). I will be using the ‘Godot Engine’ to make a working environment of the game specifics and features. The player will accompany the main character throughout her day and follow a short storyline through interacting with the other characters in the game, with a monster catching and turn-by-turn battle system inspired from the Pokemon and Octopath game series.

1.2 Motivation

Video gaming takes a large aspect of our modern society. Moving onto 2023, around 80% of the world’s entire Internet population plays video games with the lowest populations in the countries, Belgium and Japan, with only below 70% according to the survey in Statista. This shows that the video game industry will continue to grow, bringing in more opportunities in game developing. Making games has become increasingly accessible to the point that even beginners like myself can jump in anytime.

There are many engines currently available and many different ways to reach a solution. In this project, I have chosen the gaming engine ‘Godot’ and decided to learn the fundamentals of game development through the YouTube videos available and the coaching of my supervisor. The main reason I chose to do this project is because it will give me an opportunity to make a working game as a project to show in any gaming company interview I may pursue in the future. The game is inspired by two of my favourite games, Pokemon and Octopath Traveller.

1.2.1 Game Inspiration

An aspect of Pokemon I liked was the monster catching system for the project. This will allow the player to catch creatures, level up them and fight other tamers in the game. The mechanic will have the player travel across different areas to find new monsters in their adventure. The part of Octopath I wanted to imitate was the quest system. You speak to

other people and collect items for them on their behalf to unlock a route. I plan to make a monster catching system with these quest systems in between where you cannot leave the level without the quest. I also liked the mix of the 3D-made environment and the 2D pixel characters in Octopath and I would like to replicate the feel in my project.

1.2.2 Little Red Riding Hood

One of the main inspirations of this game was the story of Little Red Riding Hood. The story is of a young girl who leaves to the forest to visit her grandmother while we have an enemy, the Big Bad Wolf reaching her grandma's house and disguising as her grandma.

While the game has been inspired by this story at first, it has changed to a story of a girl, called Ruby, who has to collect apples to visit and apologise to her grandma. She will need the help of a Creature to fight through the forest alongside her to reach her grandma's house. In the end, Grandma was so upset about having lost her prize apples to a thief that she challenges Ruby to a battle with her. The story is to end with a small celebration.

I chose this story because it was short and simple, leaving me to focus on the main game aspects; collecting items, catching Creatures and making a quest based system. The game will have three quests in total; befriending a Creature who lost an item, collecting the apples to make an apple pie, and using that apple pie to pass the guard, and then visit grandma's house in the deeper woods. Ruby will interact with all her villagers to help her apologise to her grandma. She will need the player's help to do this.

1.3 Problem Statement

The focus of the game is to make a working simulation of a game field of three levels, the beginning village, the first level, and the second level. The character will be introduced in the village with a main speech outlining what she must do on that day. The player will play as the main character and complete those tasks. To make the game interesting, I have added a small monster catching system.

The player must fight enemies with their monsters and just like Pokemon, they are able to catch the monsters and use them in the battlefield. The player will be able to fight through the forest and accept different quests in order to reach the next level. The boss will be at the end of the game with stats a little higher than the other monsters in the game. This project is meant to be an interactive game so that the player can play without the use of any outside instructions, just like how an actual game will work.

1.4 Approach

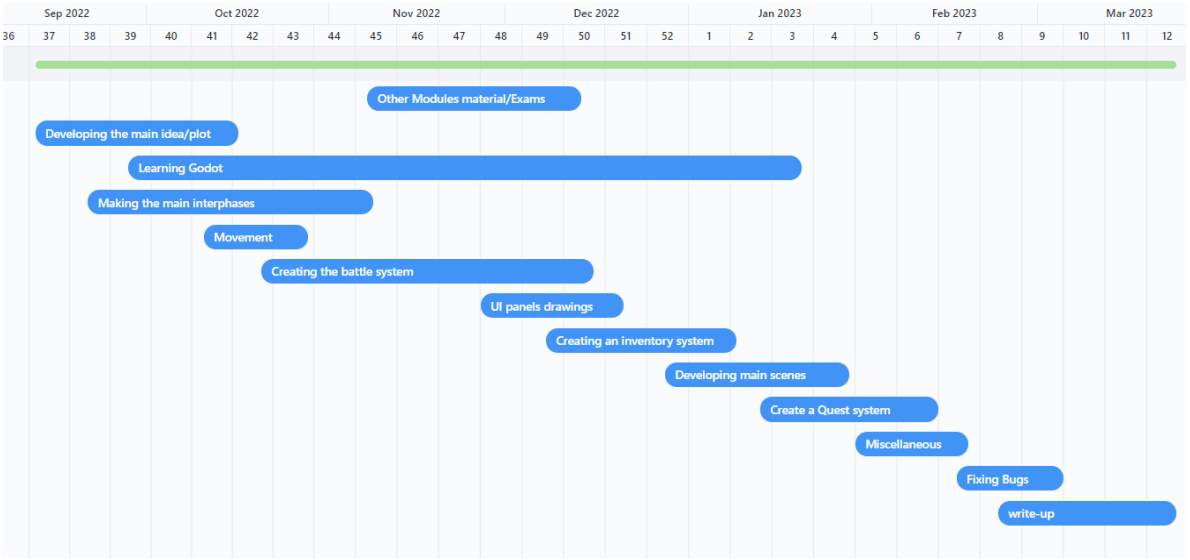
There is no formal approach to beginning this game. As this is the first game I will make, the first steps will be learning to make Godot scenes and scripts and how they interact with each other. This will involve going through YouTube videos, learning concepts and following the videos directly. I have made a Gantt chart taking into account all the tasks to get this project

working. Though each week, there will be a formal meeting with my supervisor to discuss any progress. The approach to all these tasks will be addressed further in this paper.

1.4.1 Fig. 1 - Text List of Tasks (made using ClickUp)

NAME	Start Date
Final Year Project	
Other Modules material/Exams	Nov 8 2022
Developing the main idea/plot	Sep 12 2022
Learning Godot	Sep 28 2022
Making the main interphases	Sep 21 2022
Movement	Oct 11 2022
Creating the battle system	Oct 21 2022
UI panels drawings	Nov 27 2022
Creating an inventory system	Dec 8 2022
Developing main scenes	Dec 28 2022
Create a Quest system	Jan 4
Miscellaneous	Jan 29
Fixing Bugs	Feb 9
write-up	Feb 22

1.4.2 Fig. 2 - Visual Representation of Tasks (made using ClickUp)



1.5 Metrics

During the development of this game, I will test it myself using the Godot exporting system and running the game on a browser so that I could check the runtime on different operating systems. I will prepare a questionnaire on the game about the visual aspects and the mechanisms and if they would like to play it again. With the results, I could improve the game in the future.

1.6 Project

The following achievements successfully implemented in the game were as follows:

1) Developed three worlds that the player could switch through achieving certain quests.

2) Provided a quest base system where the player would automatically receive on speaking with an NPC as well as a menu screen that updates upon quests received.

3) A turn-base system where the player could level up their Creatures, capture new creatures and the creatures saving system is automatically received in the gaming resources area.

4) An inventory system that saves on default when receiving an item.

5) A respawn area when the character is defeated midgame. And when the user quits the game, all resources are automatically saved.

Chapter two: Technical Background

Summary

This area of the paper will cover my research in my interested area in game development, specifically with the styles of Pixel and 2.5D games. This will cover examples in the gaming world on the types I resonated with the most, as well as how I will approach the development and what available resources I will use.

2.1 Topic Material

2.1.1 Pixel Art

Pixel art is a type of digital art that was made in the early 1970s due to necessity. Back in the day, the best computers were not capable of rendering the fine lined images we are used to today. The art in games were made on a small budget and were animated enough for the player to interpret movement and to act. In games today, the use of pixel art has decreased with the use of 3D objects. But even so, it remains popular today with the reason being simply because of nostalgia for the older game era.

2.1.2 Nintendo

Nintendo started off as a playing card manufacturer in 1889 and released ongoing franchises that existed today such as *Super Mario Bros.*, *The Legend of Zelda*, and *Metroid*. This company made some of my favourite games to date, *Pokemon* and *Octopath Traveller*.

2.1.3 Pokemon

The Nintendo game involves a fictional world that stemmed from the hobby of insect collecting. Players play as the Pokemon Trainer and have three goals; to complete catching all the monsters available in the game, to compete against other powerful Trainers and becoming the champion with raising a pokemon team throughout their journey. With the love for pokemon, many spins of games were made, including many other games made by fans.

Game Mechanics

The game was originally a pixelated game. The role-playing genre gives the game an interesting story to engage with other players. One of the main aspects of the game is the turn-based battle system that requires fighting an opposing challenger or wild monster. You are able to catch a wild Pokemon and level up as you move throughout each storyline. You are able to evolve them as they become stronger and fight alongside them with other players in the field. You travel to many other areas of the regions and learn about the Pokemon inhabiting those areas.

2.1.4 Octopath Traveller

Octopath Traveller is a Role-Playing video game developed by Square Enix for the Nintendo Switch, through the Unreal Engine. The main aspects of this game that interested me the most was the style of “HD-2D” that had the pixelated characters with pixel textures of the 3D objects in the background, along with the beautiful orchestra of music and lighting in the background. The goals did not have a main connection within the stories it holds. You will play any of the eight characters on the main screen and complete their first chapter, and then move on to another area and meet the other characters and learn and complete their adventure as you meet them.

Game Mechanics

Each character has a distinct job assigned to them on default. The game features a turn-based battle system where the player attacks with different kinds of weapons or elemental attacks as well as your items and abilities with different strategies used depending on your team level and team characters. Each party member receives a Boost Point at the end of each turn to boost a command and allow the character to attack multiple times, raise their defences or increase the power of an ability. The enemies are given a shield counter. When the player breaks it, then the enemy is stunted for a turn before the cooldown lifts.

2.2 Technical Material

The main resource technology and documents will revolve around Godot, Blender, Piskel, and YouTube videos.

2.2.1 Godot

The Godot Gaming Engine is a free, open-source game engine, released in 2014. It is a light platform that is capable of running on a small Graphics Card (GPU), web-browser or your mobile phones.

Nodes

Nodes are the basic building blocks of Godot. A Scene is a hierarchy of nodes and the scenes can be used as a custom template for making an object. Attaching scripts to a node can allow the node to hold export properties, have its own individual functions and make classes instances. A built-in Node can have its own unique properties that can be used in the scripts as a signal and *on_signal_called()* functions. You can attach a node to a child of another. Nodes can allow you to make the scenes which are the foundations of the game.

The main Nodes I used in my game as it is a 3D environment is the Spacial Node. The main difference between the Spatial Node and the 2DNode is that it has a third axis called the Z axis. There is one more node called the Control Node which I used to make my User Interfaces (UI) for the game screen. In the control node, I used the 2D node to make a small animation of the Creature captured and the food that is given to the Creature. Nodes have a

pre-build Sub-Node as part of the main class. I used *Kinematic3D*, *StaticBody*, *CollisionShape*, *Area3D*, *MeshInstance* and *Sprite3D* the most.

Scenes

Scenes have root nodes, and can be saved as a file and used later. You can make many instances of a scene. They are reusable. A Scene can be a character, house, items or world.

Signals

Nodes emit signals when a certain event occurs. This feature allows your nodes to communicate with each other without using the hardcode. A button node will have a signal function called *on_pressed().* This signal can be automatically connected to many scripts in the same scene so that the node can interact with all the nodes around them. You can define signals in the game by using the keywords *signal <signal_name>* and releasing the signal somewhere in the script using *emit "<signal_name>"*. Another script within the hierarchy can catch this signal using the:

<node_emitting_signal>.connect("<name_of_signal>", <node_connecting_to (usually 'self')>, "<function_called_when_signal_emits>")

Scripts

Games run in a loop. Every second, the game renders the game world and updates calculations and then redraws the world, around 60 times a second. These functions are available in every node. The script allows the player to edit the way a node will hold values when instantiated using *func _ready()* and a node can use *func _process(delta)* to keep listening for a variable or function. Your own function can be defined as *func <function_name> (...parameters).* This function can be used in any of the signals pressed or using the two main function above

Autoload / Global / Singleton variables

Godot has a feature in project settings where you can add a script that will have functions and variables available throughout the entire script. In React.js web development, this is like the *useContext()* hook. When a script is on the Autoload settings, it becomes a Global or Singleton variable. You can have as many files as you want in the Autoloads and all elements in the script can be accessed in other scripts by *<File_name>.<file_variable_or_function_name>.*

Animation Player

The animation player for Godot is useful for simple animations such as making the screen shake and making a node move. As an example for my project, I used this to make an animation for letting the user know to move on from a conversation sentence.

Resources and Classes

The extension Resources can be used when creating a lone script (A script independent of a node). While extending from resource, I used the keyword *class_name <name_of_class>*. With this, I could make a new resource script that inherits from that resource. I have access to all the various functions and variables in this script and can make new resources with this class to form other resources.

For example, I create a new resources script and give it *class_name Item.* I can then make other resources that extends from *Item.* I can now give this new resource script *class_name Food. Food* will inherit the functions and variables found in *Item*. This is very handy for making different item resources. I can separate a ‘Pen’ and an ‘Apple’, as a General Item class and a Food Item class respectively

Resources also has a built-in saving resource function keyword *ResourceSaver.save(<name_of_exisiting_directory/file_name_being_created/rewrite>, <resource_element_to_save>).* This will be seen in my examples of the solution.

Saved Styles

The Godot engine can easily save files from other Control nodes which makes it easy to style the UI.

Camera

The main camera I used was the Interpolated Camera which worked with the UI of giving a small screen delay when the character moves. It made the game scene flow more naturally.

2.2.2 Blender

Blender is a free, open source 3D graphics object platform capable of rendering animations, composition and modelling. I used it to make a 3D house for the game.

2.2.3 Piskel

I used Piskel to make my pixel art for the main character, some trees and the directional sign posts. Piskel helped me make my sprite sheet which I used in the Sprite Node that uses the Animation Player to animate the sprite frames.

2.2.4 YouTube Videos

YouTube videos are where I spend the bulk of my learnings; from building scenes, calling Signals, using Resources, using foreign pre-built tile Assets, Quest and Item Systems, making a Dialogue System, how to move the player, making the NPCs, to changing Scenes. Many of the YouTube videos I used will be outlined in the code.

2.2.5 StackOverflow and Git

For debugging and helping out with errors, I used the comment sections in these platforms to fix my issues and answer my questions.

2.2.6 UML Diagram

The use of UML Diagrams will help me understand how to work out the quests and give a good representation of my idea visually and how I will represent the player through each stage of the quest.

2.2.7 Gantt Chart

A gantt chart is a good way to help me plan out my course of action throughout my weeks of work. The feature of the gantt chart has already been addressed in the *Introduction* of this paper.

Chapter three: User Requirements

Summary

The purpose of this chapter is to identify what will make the project an interest to the user, such as; what will the user need to make this game worth their time and what will the user expect of this game.

3.1 Game Analysis

To further understand what the user will expect of the game, I have placed a User Requirements analysis to briefly understand what is needed in the game

3.1.1 User Requirements

The following criteria is needed to make the project a game of interest to the player:

1) A storyline that will give the user a purpose and a direction to the end.

   The storyline will lead to the final boss and the end of Forest 2, by then the player would have to raise a team of creatures. The storyline is dependent on the quests given to the player, which will be shown on the Menu Panel as a guide on what to do next.

2) An interesting game mechanics that will allow the user to engage in the game.

   The game will prompt you to have at least one creature in your party. You can catch more and have a better chance to defeat the boss at the end.

3) Interesting character designs and game environment to give the user a sense of adventure and art.

   The main reason why a player will choose a game is on the graphics. The game will have a colourful environment with interactive character sprites and dialogues. The item designs are a very important part as it adds another player collection in the game.

4) A calming soundtrack playing in the background.

   The soundtrack in the background shows the character of the scene and is important to keep the player energy and have them further engaged through the adventure aspect in this Role Playing game.

3.2 Project UML Documentation

This UML Diagram will show the overview of how the whole game will work. The Diagram will be based on how the Quest System works as this is the route to completing the game. The player will need to talk to NPCs' and unlock a quest to move to the next destination.

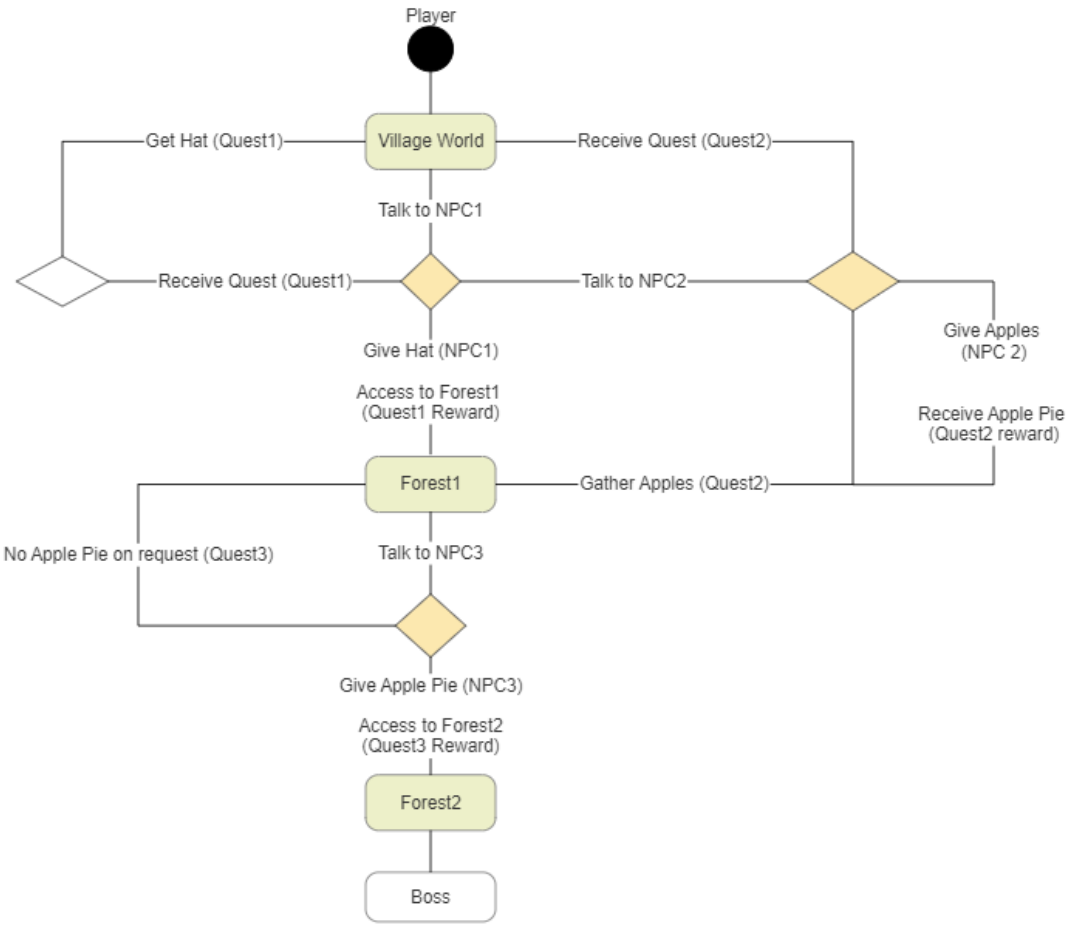
1) The **Quest 1** consists of talking to **NPC1** and retrieving the Hat item.

2) In the **Quest 2** (accessible after **Quest 1**) you must gather Apples in **Forest 1.**

3) To get to **Forest 2**, you must complete **Quest 3**, which will require you to get the Apple Pie from **Quest 2.**

4) After all the three quests, you could move to the Boss in **Forest 2**.

3.2.1 Fig. 1 - UML Design of the Quest Based System using SmartDraw



Chapter four: Game Development and Solution

Summary

In this section I will explain each decision making, solutions to describe how I will make my in-game systems, and how those solutions work together to conclude the project. Some code will be presented to demonstrate my ideas..

4.1 Imports

4.1.1 Original Designs

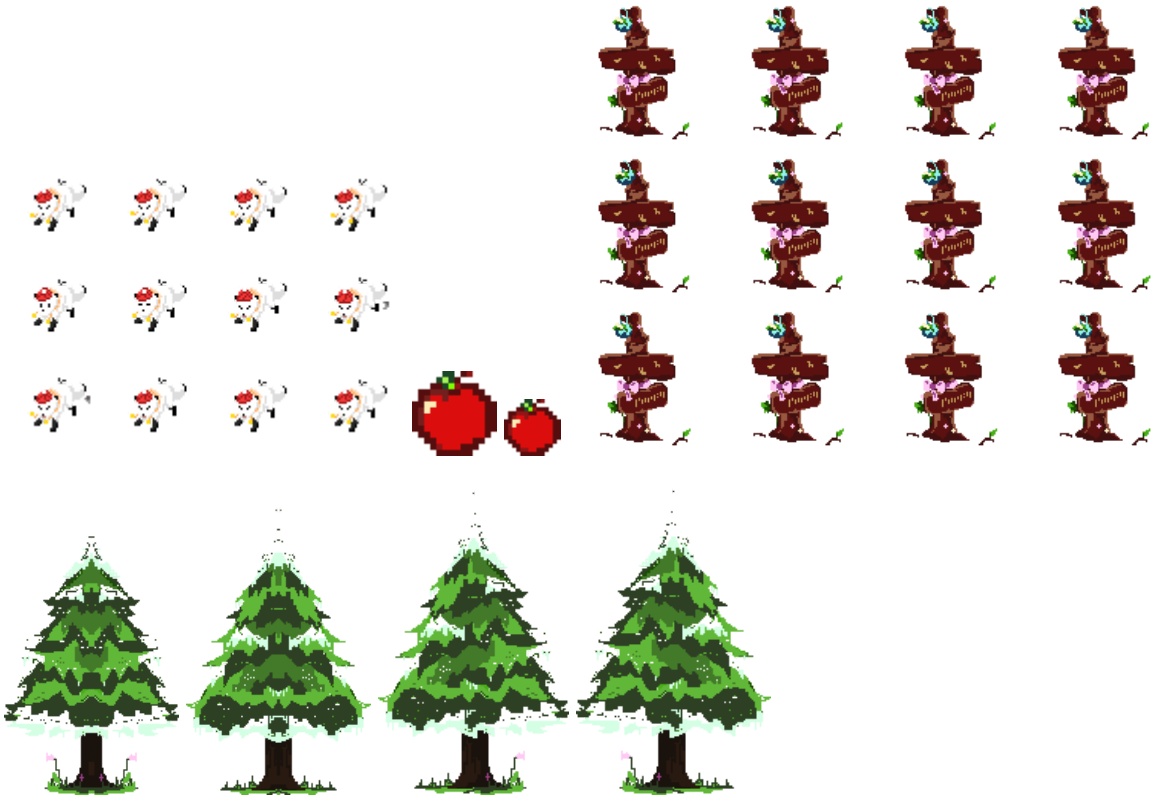
The character was designed through the app *Piskel.*

Player and NPCs' Sprite Sheets

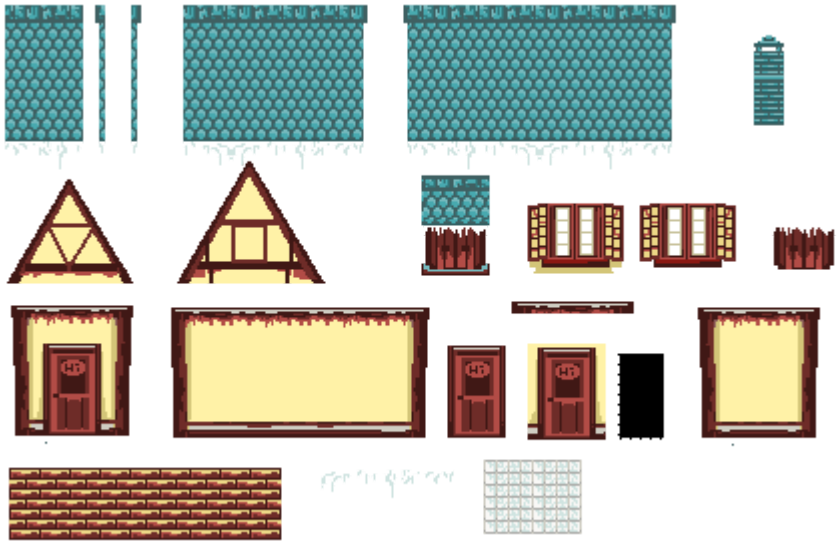
1) Idle Animations

2) Run Animations

Creature, Item, Trees and Sign



Blender House

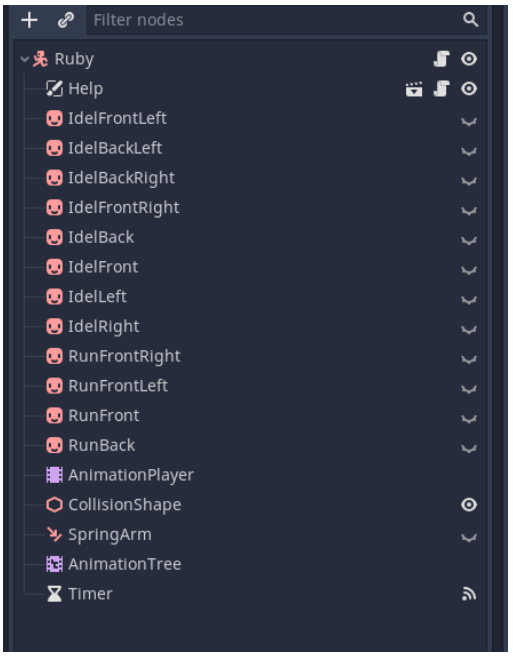


4.1.2 Music and Level Assets

All music I am using is taken from existing YouTube videos outlines in the Reference section
The assets I used is from a website linked in the Reference section

4.2 Step-by-step Tasks Completion

4.2.1 Movement



At first, I decided to make a scene with the player sprite moving. This was exceptionally difficult because I needed to find a way to switch through sprites.

The Player node consists of a *Kinematic3D* and other *Sprite3D* nodes containing individual sprites animation depending on the direction and the player movement keys.

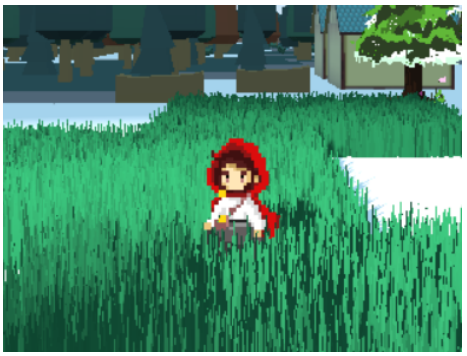
I ended up making my own functions; *showNode()*, *hideNode()* and *viewingSprite()* to switch the sprites being shown.

The functions are used in the *MOVE* enum in the *_physics_process(delta)* function.

The NPCs also use the same function as the player. They have an automove function to make the scene alive. When the player gets near them, then they stop and stay in the position they are currently facing.

```
57 func hideNode(name):
58     get_node(name).hide();
59
60 func showNode(name):
61     get_node(name).show();
62
63 func viewingSprite(hideSprite, showSprite) -> String:
64
65     hideNode(hideSprite);
66     showNode(showSprite);
67     return showSprite;
68
69
70 func _physics_process(delta):
71     match (state):
72         MOVE:
73             move_state(delta)
74         AUTOMOVE:
75             auto_move_state()
76     #       $AnimationPlayer.set("parameters/Idle/blend_position", velocity);
77
```

4.2.2 Creating the Battle System



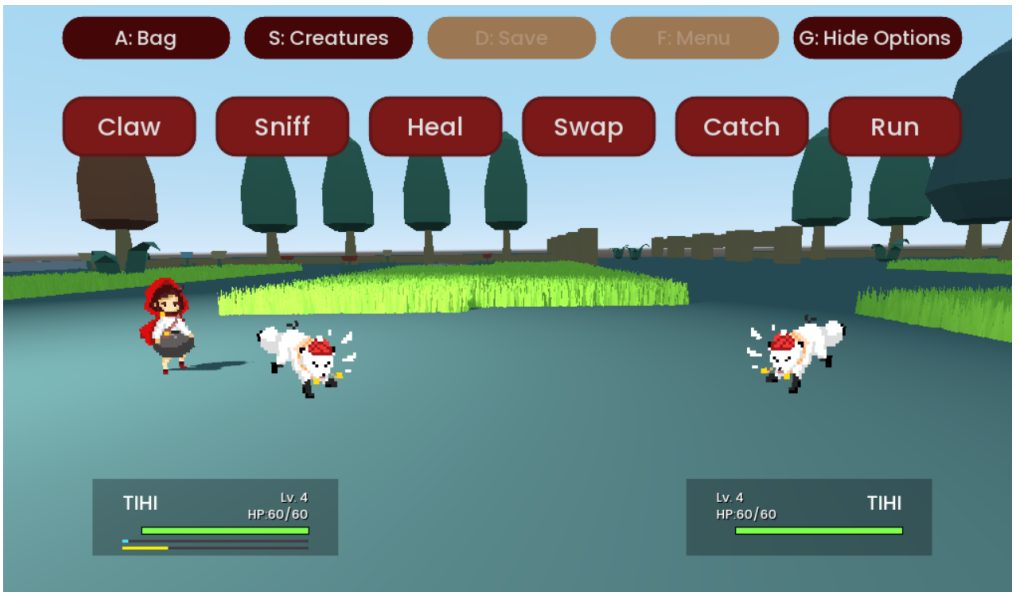
Grass Encountered Creature

In many RPG turn-based games, an enemy is encountered in an area. In Pokemon, the wild Pokemon are found through the tall grass. For this part of the game mechanics, I had to make grass objects.

The grass object consists of two *Area3D* nodes. These are to allow the player to find Creatures not just at one section of the grass. There are two boolean variables ready when the player enters a grass section, these update when the player gets back to the level. This makes sure that the player does not encounter a Creature after returning from a battle.

Only when you begin to walk on the grass, then you encounter a Creature again.

Battling a Wild Creature

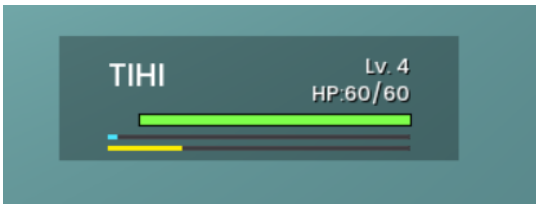


During a wild Creature battle, the player will be given three main options; Fight, Catch or Run. There are only two kinds of attacks a Creature currently has. When a Creature is instantiated, then two attacks from the eight below are randomly given to the Creature. The attacks are randomly selected depending on the chances.

```
var available_attacks_set = [
    { "name":  "Claw", "damage": 2 , "description": "A basic attack most monsters know!", "chance": 20 },
    { "name":  "CapHat", "damage": 5, "description": "Thats a Feet Special!", "chance": 5  },
    { "name":  "Bella", "damage": 4, "description": "Thats a Tihi Special!", "chance": 5  },
    { "name":  "Sniff", "damage": 2, "description": "Hey!", "chance": 20  },
    { "name":  "Paw", "damage": 1, "description": "soft feet", "chance": 20  },
    { "name":  "Hide", "damage": 3, "description": "Searching.... !", "chance": 10  },
    { "name":  "Aero", "damage": 3, "description": "Thats a Hasven Special!", "chance": 5  },
    { "name":  "Top Star :D", "damage": 100000, "description": "This is Tendo Maya!", "chance": 1  }
]
```

The wild creature also has two attacks of its own and will randomly take one out of its set each turn.

When the player wins the battle, the Creature used will gain experience points. The experience bar can be seen at battle as the ‘blue’ bar. The yellow bar is the dizziness level of the player. The Creature in the team will have stats updated each turn and the information will automatically be updated during battle.



Player Dizziness and Catching a Creature

When the dizziness level is too low, the player cannot catch a Creature. In order to increase it, the player will need to use a food item. In the Bag panel, give food to the Creature and the player’s dizziness level will also increase.

Another small feature of the game is that when the player uses an item (by dragging it onto the black screen of the creature panel), then the item sprite will appear. The dizziness section will fill up. The item stats cannot be currently viewed by the player, but the information on the item will show on the top section of the Bag panel.

Viewing a Creature

The player can view the Creature stats, information and attacks on the Creature panel. Here the player can rename and release any tamed Creature on the team. The player's first creature cannot be released since the game requires at least one creature in the team. The HP and the Experience bar is always updated in this section. The renamed creature will be saved and you could see the changed name during battle. This will give some uniqueness to the Creature.

Setting a Creature for battle

During battle, the healthiest Creature at the top of the party will always be first, this is the *Leader* Creature. However, the player can select any Creature to be the *Leader* in the Bag Panel. When the creature has the Leader tag, it will always be the first Creature to battle unless the HP is at zero, then the tag 'Fainted' will always show and the creature is unusable until the whole team is at zero. Here, only the first creature in your party has leadership.

1) Click on a Creature to select it.

2) Click on the Select button and then the creature will be first into battle

Fighting another Tamer and Swapping Creatures

During a fight with another Tamer, then the player cannot Run or Catch a Creature. The battle cannot be stopped until the player either wins or loses the battle. The player loses the battle if all Creatures from the player's Tamer Team are unable to fight.

When a Creature faints during a battle, the player will be forced to Switch Creature or Run, but in a Tamer battle, running is not permitted. The player could switch for a healthy Creature in the Bag Panel.

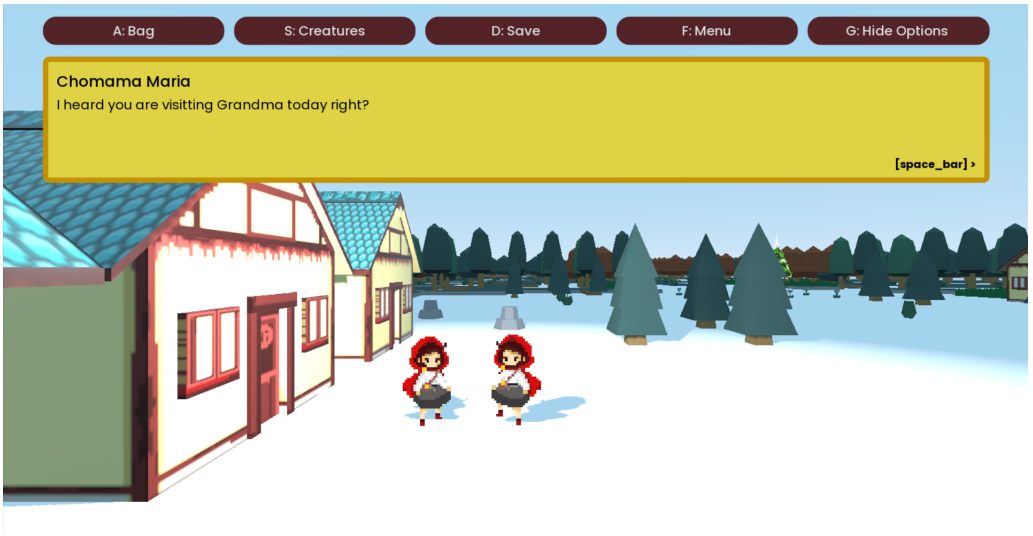
The stats change upon switching.

4.3 Implementation

These are all the interesting aspects of the game and further code and use of Godort features, such as Resources, Scenes and various Nodes.

4.3.1 Quests and Items

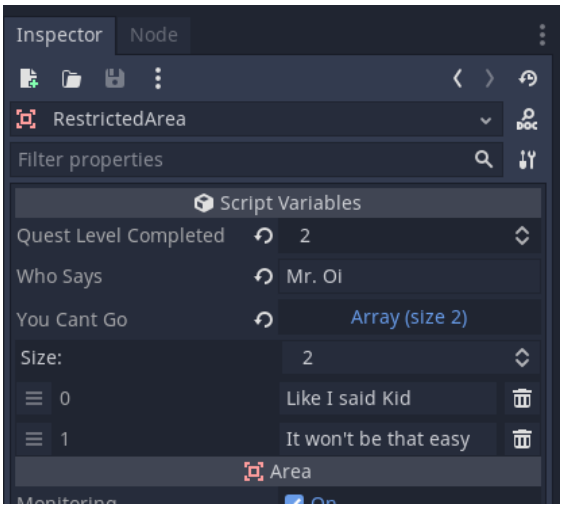
To get a quest, the player must speak to the NPCs (Non-Playable Characters). The player can only complete quests in a certain order.



This is checked through the *Quest.CURRENT_QUEST_ID* and the quest of the NPC the player is talking to. If the player has the quest already, there is a check to prevent the player from having duplicated quests.

```
56 func get_quest ():
57
58     if Quest.CURRENT_QUEST_ID > quest.quest_id:
59         message = quest.post_quest_message
60         return
61
62     if Quest.CURRENT_QUEST_ID != quest.quest_id:
63         print("your current quest doesnt match the current id %d != %d  " % [Quest.CURRENT_QUEST_ID, quest.que
64         return
65
66     if Player.adventure_log.get_adventures().size() >= 0:
67         var new_quest = Player.adventure_log.get_adventure_by_name(quest.name)
68         if new_quest != null:
69             print('you already have this adventurue: ', new_quest)
70             check_quest_completed(new_quest)
71             return
72
```

If a certain quest is not finished in the area that is needed to get to the next level, there is a restriction area where a player cannot enter. The *RestrictedArea* also has a *quest_level_completed* variable that is checked depending on the player's *Quest.CURRENT_QUEST_ID*. There will be a dialogue shown and the player will automatically move back from the area.

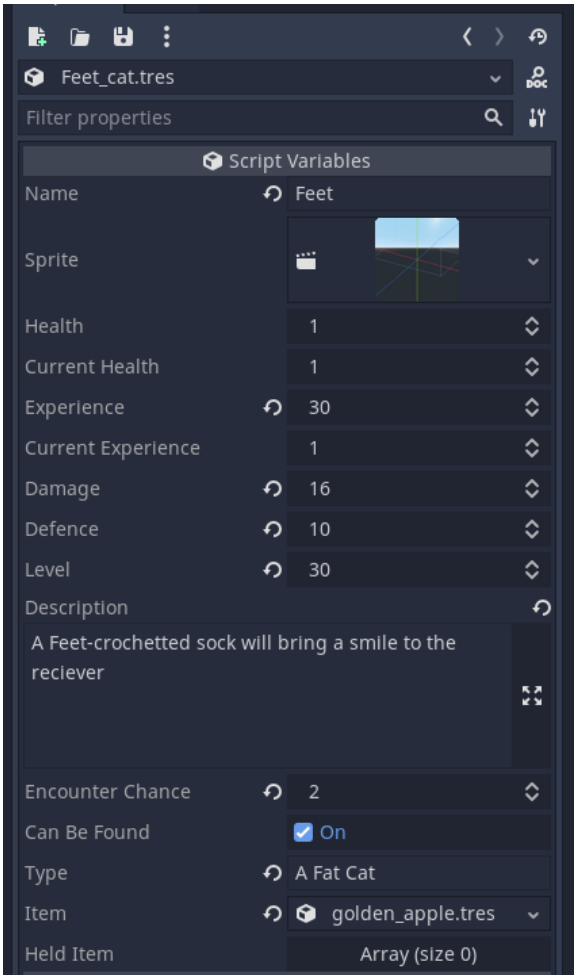



```gdscript
func _on_RestrictedArea_body_entered(body):
    if (body == Global.player):
        print("hi player :)))")
        if (Quest.CURRENT_QUEST_ID > quest_level_completed):
            print(" you are free to go")
            Global.auto_move_direction = body.direction
            Global.auto_move_run_sprite = body.automove_run_sprite
            body.auto_move()
        else:
            load_dialogue()
            print ("EHEHEHHEHHHEHH")
            Global.auto_move_direction = -body.direction
            Global.auto_move_run_sprite = body.OppositeDirections[body.automove_run_sprite]
            body.auto_move()

```

Each quest requires an item to move forward. Some items can be gathered through fighting a creature. But there are also pickable Items on the level.

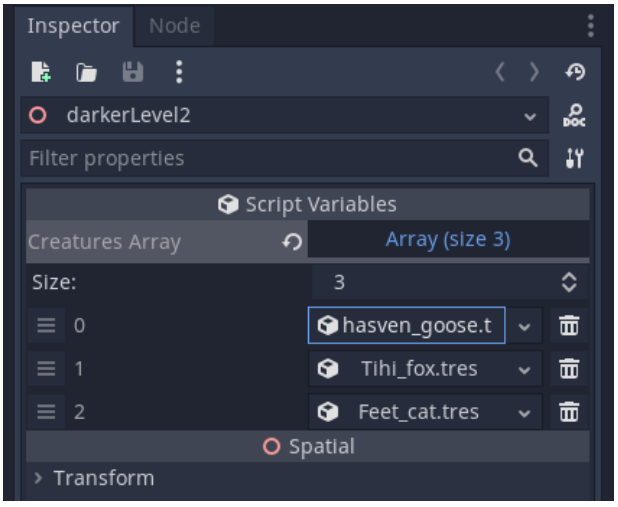
4.3.2 Levels and Random Creatures

In each Level, there is a list of Random Creatures available to catch. The *RandomCreature* is generated from the class *RandomFinder* and is resulted by the encounter chance of that Creature. Random Creatures also has an item they hold which the player could receive upon defeating or catching that Creature.




```gdscript
if creature_array.size() > 0:
    var overall_chance: int = 0
    for creature in creature_array:
        if creature.can_be_found:
            overall_chance += creature.ENCOUNTER_CHANCE
    print("overall_chance: " , overall_chance)

    var random_number = randi() % overall_chance
    var offset: int = 0

    for creature in creature_array:
        if creature.can_be_found:
            if random_number < (creature.ENCOUNTER_CHANCE
                chosen_value = creature
                break
            else:
                offset += creature.ENCOUNTER_CHANCE

print("chosen_value Final: " , chosen_value)

#genereates attacts
chosen_value.generate_creature_attacks()
chosen_value.creature_info()
```

The creature array is found in the Level's node the moment the player enters the scene. The overall chance of encountering a certain creature is all in the *ENCOUNTER_CHANCE* of all the creatures available added together as what is shown in the code above.



4.3.3 Player Initialisation, Resources, Saving and Databases

When the player first enters the scene, code from Global.gd (a Singleton variable in Autoload) will automatically initiate a player body that will connect the player to three important game files, functions and signals.

In Player.gd — line 44 - 51:

First, an instance of the resources are called from their respective scripts.

```
43  # load the save files from the directories and intialise a new file
44  var inventory_resource = load ("res://Resource/Inventory/Inventory.gd")
45  var inventory = inventory_resource.new()
46
47  var creaventory_resource = load ("res://Resource/Inventory/Creaventory.gd")
48  var creaventory = creaventory_resource.new()
49
50  var adventure_log_resource = load ("res://Resource/Inventory/AdventureLog.gd")
51  var adventure_log = adventure_log_resource.new()
52
```

In Global.gd — line 101-107:

The *initialise_player()* will use the *signal player_initialised* at the top of Global.gd and *emit_signal("player_initialised", player).*

```
 99  # when the scene changes emit a player initialisation to connect all the functions below
100  func initialise_player():
101      player = get_tree().get_root().get_node("/root/" + current_scene_name + "/Ruby")
102
103      emit_signal("player_initialised", player)
104
105      Player.inventory.connect("inventory_changed", self, "_on_player_inventory_changed")
106      Player.creaventory.connect("creaventory_changed", self, "_on_player_creaventory_changed")
107      Player.adventure_log.connect("adventure_log_changed", self, "_on_player_adventure_log_changed")
108
109
110      var directory = Directory.new();
111
112      var existing_inventory = directory.file_exists(inventory_save_path)
113      var existing_creaventory = directory.file_exists(creaventory_save_path)
114      var existing_adventure_log = directory.file_exists(adventure_log_save_path)
115
116
```

Global.gd then communicates with the Player.gd (a Singleton variable in Autoload).

In this example we will look at the *inventory* resource. When the player is initialised, the *Player.inventory* connects with a signal from that *Player.inventory* script and connects it with the function that exists in *Global.gd* called *_on_player_inventory_changed.* This will use *ResourceSaver.save()* to save the inventory in the path chosen.

```
151 func _on_player_inventory_changed(inventory):
152     var _not_in_use = ResourceSaver.save(inventory_save_path, inventory)
153     
154 func _on_player_creaventory_changed(creaventory):
155 #   print("creaventory saved: ", Player.creaventory.get_creatures())
156     var _not_in_use = ResourceSaver.save(creaventory_save_path, creaventory)
157 
158 func _on_player_adventure_log_changed(adventure_log):
159     var _not_in_use = ResourceSaver.save(adventure_log_save_path, adventure_log)
160     
161 
```

In Inventory.gd — line 5-10:

The following signal *inventory_changed* is used to connect the *Player.inventory* with the *Global._on_player_inventory_changed*

```
 3  class_name Inventory
 4
 5  signal inventory_changed
 6  export var _items = Array () setget set_items, get_items
 7
 8  func set_items (new_items):
 9      _items = new_items
10      emit_signal ("inventory_changed", self)
11
```

The game largely depends on the player's initialisation and *Global._on_player_inventory_changed* to function properly.

The item resources are checked by *ItemDatabase.gd.* Each resource has a database.

```
 1  extends Node
 2
 3  # this is the code to get all the adventure files
 4  # from "res://Resource/Adventures/Adventure/"
 5
 6  var items = Array ()
 7
 8  func _ready():
 9      var directory = Directory.new()
10      directory.open("res://Resource/Items/itemRepository/")
11      directory.list_dir_begin ()
12      
13      var filename = directory.get_next()
14      while (filename):
15          if not directory.current_is_dir():
16              items.append(load("res://Resource/Items/itemRepository/%s" % filename))
17          
18          filename = directory.get_next()
19      
20  # check if the item exists
21
22  func get_item (item_name):
23      for i in items:
24          if i.name == item_name:
25              return i
26      
27      return null
28
```

Chapter five: Evaluation

Summary

This part of the paper will show the results and feedback of the players who tested my game on this computer, as well as what could be implemented better. I will explain how the project met with the original user specifications and how the test methods have been carried out.

5.1 Solution Verification

The project is very similar to the original User Specification. The game had a storyline, a working battle and quest mechanic, and a good game environment. The only requirements not met was that it was supposed to have different character sprites to interact from. This means that, visually with the character sprites, the game was not very appealing or very exciting. As a monster catching game where one possible goal of the player is to collect the monsters, any monsters with the same sprites does not make the game attractive.

There was also a small issue with the quest system. When an item in a quest is found in the player inventory, the quest will complete no matter how much of it is required. This means that the quest could be bypassed. This is something I hope to work out in the future.

5.2 Software Design Verification

I was unable to run the game on other computers since I had no access to them. My test players had a windows computer only. In this case, the application worked well for the windows operating system for this draft. The players only tested the game through the application export (Windows Desktop) from the Godot IDE.

5.3 Software Verification

The application for the game worked as it should but with a small error. The resource files would not save upon user quitting. The game worked as it should in the entire playthrough, but the game did not save the same NPC dialogue moments as it did in the development process. And so, the game cannot be playable as a customer product currently. There was no exact way I could test my game on the browser. I used Python to set up a fake server in this case and watched how my game ran.

5.3.1 Test Approach

I used Python to put up a fake http-server on the background of my project files to test the browser side of the game. The game was exported as a HTML5 (runnable). There was a huge delay in between each render frame. For the application, the game was exported as a Windows Desktop (runnable). It worked perfectly with no delays during playthrough.

5.3.2 Results

On the fake server, the game lagged and slowed down. The music was bugging in between and so it was a failure on the test server. The exported application worked well. In this case, the application would be used for User Testing.

5.3.3 Results Interpretation

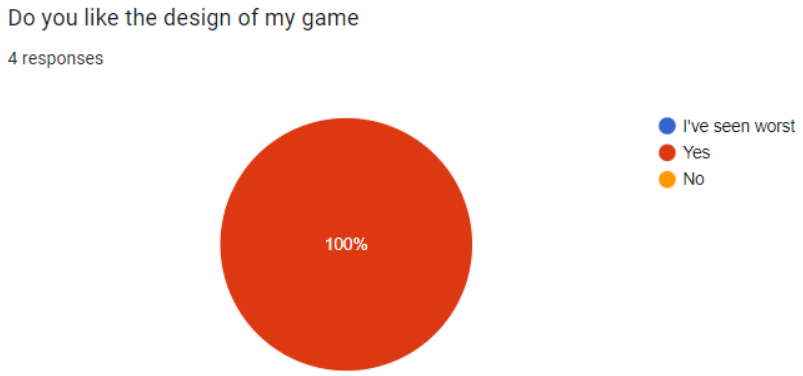
The fake server may have not as much render power or the files might have been too big for the game to work. The delta function in the game might have caused the huge delays and if I had added more NPCs, then the game would have further lagged on the fake server.

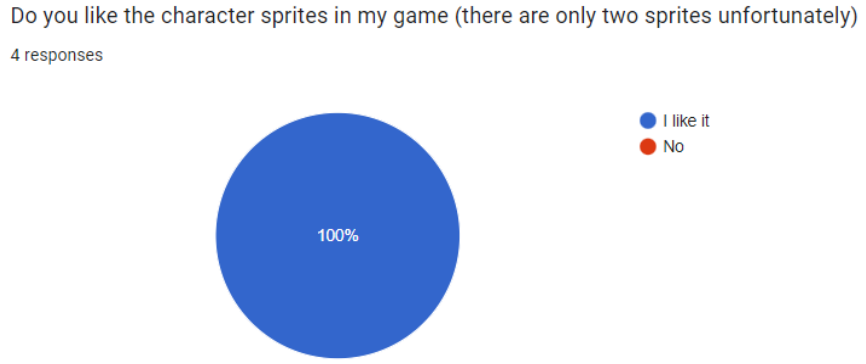
On the application side, the game had its own folder and did not require anything else but to run the application. The app had no limit or need of an internet connection to run, which may have been the reason for a good working render in each frame and NPC movements.

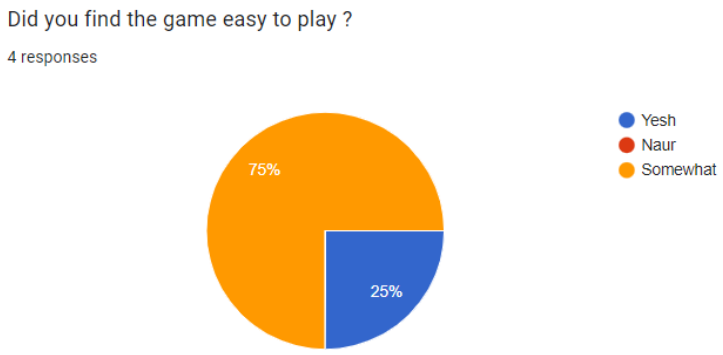
5.4 User Validation

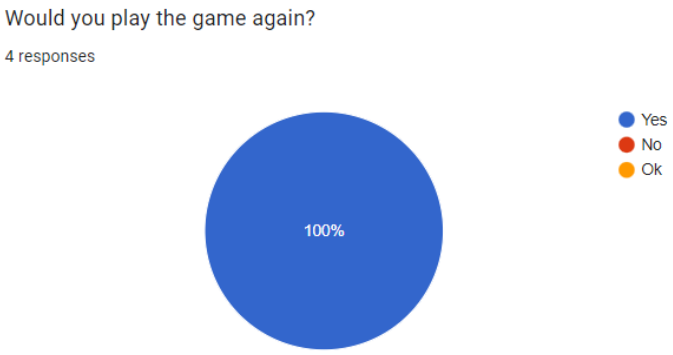
The following are the results of four test players. These results were by a Google Form that was answered after playing at least 5 minutes of the game (The players played on either the game application export or Godot engine terminal). .

5.4.1 Results









Are there any comments or any further improvement would you like to see in the game?

4 responses

More monsters.
Maybe have a pop-up when there is a new chapter.
its good but hard to complete. make 20 apples 5 or smth. granny is hard to beat
All good!

5.4.2 Results Explanation

From these results, I found that the game was relatively harder to play than I thought, partially because the players did not know where to see the quest information at the start. It was also difficult when the monsters encountered were hard to defeat and the food item used will mostly be needed to keep playing. Therefore, the second quest of 20 apples should be reduced. To three of the four players, I needed to help them complete the game.

The game runtime for each player was around 10 minutes on average. The game completion for the player who played without much instructions took 15 minutes. One of the players who reached the end game commented that the boss was too difficult to defeat, and she felt lost during the switching phase of the creature.

All the players needed help on switching their creature during battle.This shows that I needed to perhaps show a short tutorial of the game mechanics at the start of the game especially to the players who have never played a monster catching game before.

5.4.3 Results Analysis

Upon the results of the form, the players liked the graphics and the gameplay, but the game was too difficult for a beginner. They are still willing to play the game again and I would say that this is a good progress for my project and gives me confidence to try again. This also showed some interest in the game tested.

The comments at the end give me good feedback on how the game interaction with the user could improve. The whole game is about catching monsters to defeat the boss at the end. The monster needs variety in the sprite sheets and the stats of all the Creatures in the game needs to be more balanced.

Chapter six: Conclusions

Summary

The conclusion will offer insights to what could have been done better in the project; testing, learning and presenting.

5.1 Results Discussion

My test results are very limited as I finished my main project game late, many students were focusing on their project deadlines. I have had one player not from university play as part of my tests. There was trouble exporting the game as the game did not work as it should. This was fixed only until a day before the deadline. By then, my test players were even more limited.

I was also not able to find a way to run the game on a public website, so therefore the tests on checking the performance of the game are only limited to my computer, Windows Operating Systems. One other limitation is the computer I am using. I do not have access to a better system that is stronger in graphics and rendering, which is partially why one of the results said that my world needs more added NPC characters. With a better graphics system, I would be able to render more grass strands in the mesh instance and the project would not slightly delay as it does now in my computer.

There was also the fact that each player played at around different playing time depending on their schedules. Some can only test out for 10 minutes of their time while one of them took 20 minutes on the first gameplay and then 15 minutes after beating the game the first time. This shows that the game was relatively easy to understand, but with the pressure around the other assignment deadlines of other modules and projects, it was hard to put all focus on the game storytelling or the mechanics.

5.2 Project Approach

My motivation for the project was very big at the start, and I was able to spend hours working on the player sprites. I was not aware of how my project was supposed to look at the start. The sprites I made were mostly suitable for an isometric 2D game style. This means that when I had a certain idea of how the game would look, I had to redraw the pixel sprites.

My interest in the project began to fall down as I realised that the engine I chose did not have the sufficient information on the type of game I wanted to make. This means that I would need to mostly learn how to make the game I wanted without previous knowledge of another person to guide me. Godot does have the features to make the game as I wanted, but as a beginner, I was intimidated and demotivated by the engine’s lack of tutorials for my preferred game.

My interest in the game came back after spending two days of watching a playthrough of Octopath Traveller. I decided that I would make the simple and fun features first. For

example, the world designs, the grass and the changing scenes. I found myself working on graphics even though I was told to look further into how to make the code to make my ideas real. I found for myself that I needed to keep myself satisfied with the graphics first so that I could enjoy developing the inner more interesting parts of the game. After the graphics, I improved other areas such as the battles, and then moved on to making the items inventory and so on. I was able to keep up with all the elements I wanted to implement at the start of the project, except the saving system. All the remaining tasks in the Gantt Chart were either somewhat completed, or fully completed.

I conclude that for this project or any project to be completed in the time given, you will need some small tasks at the start with the estimated amount of time to complete beside them. You will need to know what you want your project to look like so that you will not waste any time on elements that will become redundant. Look at the aspects of the tasks that interest you the most and work on them. For me, this was the graphics, and then scene transition. The remaining tasks that are more demotivating will work out as you have learnt from the tasks that have interested you the most.

On the technical side, I have attended most weekly meetings with my supervisor. In some of those weeks, I have made slight improvements in my code. The information on the progress can be seen in the Appendix section. A large amount of this project is based on implementation and learning from other videos. These will be referred to in the References.

5.3 Future Works

Most of the goals for this project have been met. There are more functionalities that have been overlooked, like the saving system and a weather landscape that will be reflected off the sky in the world. There could be a specific plot that could lead to the next areas of the game. I could have increased the chances of a creature in a certain world level. I could make the game work on a phone by adding a touch pad to the screen. Perhaps I could have more test players in this case. If I went with the original storyline of Little Red Riding Hood, then there could be more potential to make the story more interesting.

Here are the summary of what should be improved, added or fixed in the game overall:

a) Player guides in the battle system.
b) Quest system that requires multiple items to pass.
c) Quest accepted notifications.
d) Adding additional Creature sprites and adding more NPCs
e) Balance the game stats and levelling up systems.
f) Make the scene transitions smoother

With the time ahead in my career, I would like to learn another engine that has more tutorials and learning videos on the type of game I want to make. Octopath Traveller is made through the Unreal Engine. Now that I have learned aspects of game development in Godot, it would be easier for me to push myself to make other games. Further in my career I plan to pursue Game Development. This game has only been the beginning.

References

Research websites:

J. (2023b, January 13). *Pixel Art Games and the Reasons Behind its Wide Popularity*. Juego Studio. https://www.juegostudio.com/blog/pixel-art-games-and-the-reasons-behind-its-wide-popularity

History.com Editors & A&E Television Networks. (2017, September 1). *Video Game History*. HISTORY. Retrieved March 29, 2023, from https://www.history.com/topics/inventions/history-of-video-games

Wikipedia contributors. (2023, March 24). *Octopath Traveler*. Wikipedia. https://en.wikipedia.org/wiki/Octopath_Traveler

Cohen, D. (2020, February 13). *The History of Nintendo Video Games*. Lifewire. https://www.lifewire.com/history-of-nintendo-729734

Why Pixel Art Games Have Become Widely Popular. (n.d.). https://rocketbrush.com/blog/pixel-art-games-popular

Barnes, S. (2022, March 23). *How the Humble Pixel Became a Building Block To Groundbreaking Art*. My Modern Met. https://mymodernmet.com/what-is-pixel-art/

Griffiths, D. (2018, September 27). *The History Of Pixel Art — The Factory Times*. The Factory Times. http://www.thefactorytimes.com/factory-times/2018/9/27/the-history-of-pixel-art

Statista. (2023b, January 31). *Global gaming penetration 2022, by country*. https://www.statista.com/statistics/195768/global-gaming-reach-by-country/

TheDevDoc. (2020, October 19). *How To Connect Signals Via Code In Godot In Less Than 3 Minutes* [Video]. YouTube. https://www.youtube.com/watch?v=WWXHurInRcU

TheDevDoc. (2020b, October 19). *How To Connect Signals Via Code In Godot In Less Than 3 Minutes* [Video]. YouTube. https://www.youtube.com/watch?v=WWXHurInRcU

Wikipedia contributors. (2023c, March 27). *Pokémon*. Wikipedia. https://en.wikipedia.org/wiki/Pok%C3%A9mon

Forest assets used in the game:

Rainware. (2019c, July 12). *Create a Mesh Library for Gridmaps in Godot 3 to design your level* [Video]. YouTube. https://www.youtube.com/watch?v=UGltqKZFxrs

Music used in the game:

Melonate1219. (2017b, July 29). *“Freesia” - Sakura Quest* [Video]. YouTube. https://www.youtube.com/watch?v=68xqKUSzB-w

7GamerMinutes: Video Game Music. (2013, November 26). *Snowbelle City [Pokémon: X & Y]* [Video]. YouTube. https://www.youtube.com/watch?v=6bB3EILH_7o

marasy8. (2015, March 10).【ピアノ】「吹雪」を弾いてみた【艦これ*ED*】[Video]. YouTube. https://www.youtube.com/watch?v=NOiMt5ip-4s

JooshyFruit. (2021c, March 2). *Octopath Traveler Main Menu Theme* [Video]. YouTube. https://www.youtube.com/watch?v=SXRdCnSnOO8

Learning Godot and help websites (these are also code references)

How to set and get position of an Object in Godot (3D). (n.d.). Stack Overflow. https://stackoverflow.com/questions/72459978/how-to-set-and-get-position-of-an-object-in-godot-3d

How to detect which key is pressed? (2018, July 12). [Video]. Godot Engine - Q&A. https://godotengine.org/qa/30582/how-to-detect-which-key-is-pressed

Define the limits of a 3D camera. (2017b, October 26). [Video]. Godot Engine - Q&A. https://godotengine.org/qa/19177/define-the-limits-of-a-3d-camera#:~:text=0%20votes%20In%20Godot%2C%20it%20seems%20that%20only,and%20does%20not%20stop%20following%20the%20Player%20permanently.

Jacob Foxe. (2022b, June 6). *RPG Items in Godot Engine!* [Video]. YouTube. https://www.youtube.com/watch?v=nR0nCFJ8-qM

John Ivess. (2021b, May 5). *Chance-Based Random Picker in Godot!* [Video]. YouTube. https://www.youtube.com/watch?v=IAhBZLlz5wY

Resources. (n.d.-e). Godot Engine Documentation. https://docs.godotengine.org/en/stable/tutorials/scripting/resources.html

Gamefromscratch. (2020b, January 14). *Publishing Your Godot Game* [Video]. YouTube. https://www.youtube.com/watch?v=89q1XeYYeXY

Emi. (2020b, February 14). *RPG or Visual Novel dialog box for Godot 3!* [Video]. YouTube. https://www.youtube.com/watch?v=kkLqW8WhCgg

ExploreGameDev. (2022, January 11). *Drag and drop in Godot part 1, UI Control nodes* [Video]. YouTube. https://www.youtube.com/watch?v=cNvzGKCkNXg

Godot Tutorials. (2020c, June 12). *SceneTree & Root Viewport | Godot Basics Tutorial | Ep 09* [Video]. YouTube. https://www.youtube.com/watch?v=U3GbeR5CYEI

Code with Tom. (2020b, June 19). *INVENTORY & ITEM SYSTEM in Godot* [Video]. YouTube. https://www.youtube.com/watch?v=hYRN0eYttLc

HeartBeast. (2020b, September 24). *Resource Based Inventory - Godot 3.2 Intermediate Tutorial* [Video]. YouTube. https://www.youtube.com/watch?v=rdUgf6r7w2Q

NULL Game Dev. (2022d, September 30). *Level Up System + Finishing Player Dash! || Part 14 - Godot Engine Tutorial 2D* [Video]. YouTube. https://www.youtube.com/watch?v=Hc7W9AVZ2Hw

DELTA12 STUDIO. (2021c, November 20). *The easiest way to Change Scenes in Godot Engine 3.4* [Video]. YouTube. https://www.youtube.com/watch?v=qaznUllK7Hg

Kasper Frandsen. (2021b, February 4). *Simple Interactive Grass in Godot* [Video]. YouTube. https://www.youtube.com/watch?v=qcScJ_vgsGU

DELTA12 STUDIO. (2021e, November 26). *How to make the Character Move Automatically in Gogot 3.4* [Video]. YouTube. https://www.youtube.com/watch?v=N5xkEZzPK1M

DevWorm. (2022b, October 8). *How to MAKE an NPC for a RPG in Godot* [Video]. YouTube. https://www.youtube.com/watch?v=u0r1lVhQO8U

Jon Topielski. (2022b, February 9). *How to create a Turn-based Combat System in Godot* [Video]. YouTube. https://www.youtube.com/watch?v=ifXGvlAn0bY

NULL Game Dev. (2022f, October 1). *RPG Quest System + Pausing The Game! || Part 15 - Godot Engine Tutorial 2D* [Video]. YouTube. https://www.youtube.com/watch?v=pSb36q-wM3I

Ralf Bierig. (2023, February 20). *Godot to HTML5* [Video]. YouTube. https://www.youtube.com/watch?v=rmMWQAisItE

Citation Website:

Scribbr. (2022b, December 21). *Free Citation Generator | APA, MLA, Chicago | Scribbr*. https://www.scribbr.com/citation/generator

Diagrams and Planning Website:

ClickUp. (n.d.-d). clickup.com. https://app.clickup.com/

Documents — SmartDraw. (n.d.-b). https://cloud.smartdraw.com/

Appendices

Appendix 1: Download the game, Source Code and User Manual

1) You can play the game by downloading this zip in this google drive link: Ruby RPG. The zip must be extracted. The files extracted must be in the same directory.

2) The Source Code can be downloaded here in the same google drive. The manual on how to run it is in the link. Code can be seen in **FYP Support Section in Moodle.**

Appendix 2: Project Management and Meetings

My original Gantt Chart is at the Introduction. There are parts that took longer than it should such as the battle system working out and the quest system implementation.

The meeting information is as shown below. My supervisor may be contacted for verification. Some meetings later have no progress, I will only record the specific week intervals if so.

Week 1: Friday, 14/10/2022

Work accomplished:
I decided the player character sprites

Discussions:
Talked briefly about the game outlines?
What will the character do? What is needed to happen to achieve what she wants to do.

What kind of game will it be?
A 2D game or a 3D game. I wanted to do a game in between both, which is called a 2.5D game

Week 2: Friday, 21/10/2022

Work accomplished:
I designed a Tree, and Sign Sprite. The player is now able to walk on the platforms

Discussions:
My supervisor and I talked about the camera workings in the game. He sent me some videos to watch and learn from on YouTube.

We talked more about the graphics of the game. Perhaps the player will need a map of some sort to reach the final boss

Week 3: Friday, 28/10/2022

Work accomplished:
I made the camera work as of last week. The camera has some sort of a delay as the player is walking. I made the plot of the game to be a story of Little Red Riding Hood. Perhaps there could be a twist in the game.

Discussions:
My supervisor mentioned that I would need to make boundaries in the game to make sure she does not fall

Week 4: Friday, 4/11/2022

Work accomplished:
I was able to make a battle system of following a YouTube video, and using my sprites.

Discussions:
I was told by my supervisor that I need to implement more, like a level system or maybe the creature can change form at some stage.

Week 5 - 8: Friday, 11/11/2022 - 2/12/2022

Work accomplished:
I was able to make a dialogue system from a json. But this is really hard to understand. The code was not used later in the final project. But there was still not much progress. The enemy bar in the project is not updating as it should and I would need to fix it.

Discussions:
My supervisor said that I should focus on adding more features as opposed to working with the graphics aspect. The features are to do with the battle mechanics. What would make the game more interesting?

Week 8 - 17: Friday, 2/12/2022 - 3/2/2023

Work accomplished:
There is not much progress, but I am able to change scenes of the game.

Discussions:
My supervisor mentioned that a quest/mission system would be interesting. I might need to add a saving function, but I must first focus on some features.

Week 17 - 21: Friday, 3/2/2023 - 3/3/2023

Work accomplished:
I am making progress with the world's features. I am still working on the graphics aspect but now the player can travel across three different worlds and encounter battles from the grass. But the battle system will need to change with different encounters of creatures with different stats. Right now, there is only one creature available to fight.

I had trouble with the savings system, so I asked my supervisor for help. I was given some videos to learn from, and some Ideas, such as perhaps using Godot's custom Resources features

The json saving is too complicated and I found a way to save eventually using ResourceSaver.

Discussions:
There were no meetings formally discussed online or in office for this month

Project Ended: Week 21 - 24 : Friday, 3/3/2023 - 24/3/2023

Work accomplished:
I have learnt Godot mechanics and made a working game with all the different features as discussed with my Supervisor earlier in the inactive weeks.

I wrote up a list of tasks I must complete before a certain date given to myself and everything except the saving from the menu and the title screen creation is completed. Since the game can work independently of the saving screen, I decided to submit the work as it is.

The game has working level mechanics, levelling up and switching and talking to NPCS that can move on their own. The quest system somewhat works and the inventory system is up to date. There is also a creature saving systema and you can see changes in the UI panels as you work through the game.

The rest is now doing the write ups and the code commenting.

Discussions:
I showed the current game to my supervisor who said that I should fix the quest system but also focus on writing the solution of my project because I am caught for time.Any leftover ideas must be recorded in the Future Works section of the portfolio. I was given tips on portfolio submission and ideas to submit near the end of the portfolio deadline

Portfolio write-up: Week 25, 24/3/2023 - 29/3/2023

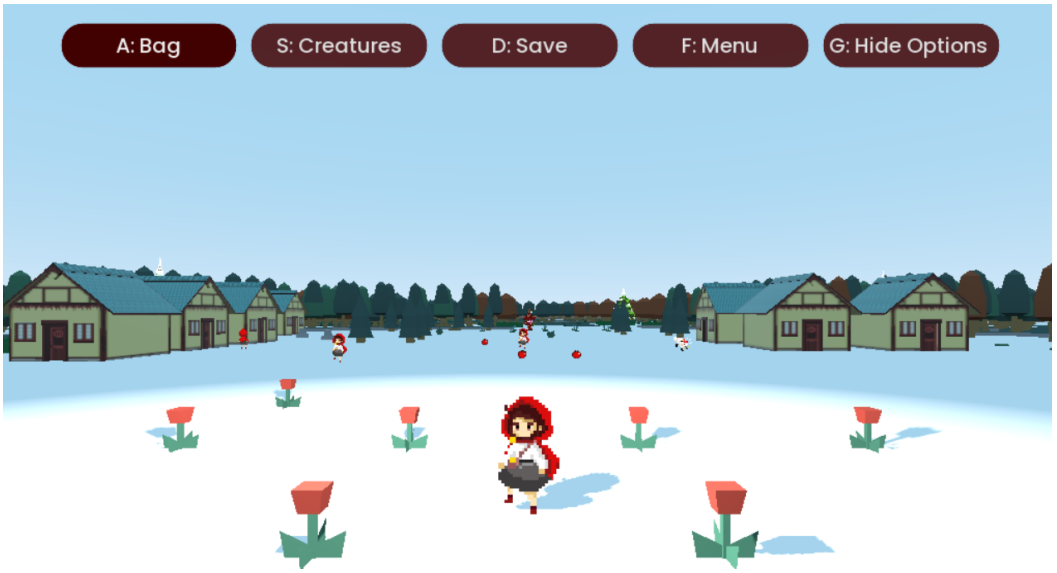
Discussions: All related to portfolio and research.

Appendix 3: What I would do differently in the Project

Perhap I would have used another engine since the videos on this particular engine are very limited. I only found the solutions to my problems on new videos that are just a month or two old. Though Godot is easy to learn, the community is more focused on 2D objects and is most suitable for teaching 2D games. I found it difficult to get started as my game was ultimately a 3D game at its core. There were some functions I had to do by myself. The code I often wrote for the functions is messy and may not be understandable. If I were to do this project again, I would need to find a way to make all my main functions global, instead of repeatedly not fully understanding what one is doing after a few weeks of implementation.

Appendix 4: Project Implementation

Starting World:



End Level:

NPC Conversations:

Menu Panel showing quest log: