**Turn Based RPG with Scriptable Objects and Simple AI**

Ryan Tan

1800832F

P04

Table of Contents

**Cover Page1**

**Table Contents2**

**Main Menu 3**

**Camera Follow 5**

**PlayerController 7**

**PlayerData 11**

**UIManager 14**

**SceneLoader 18**

**Inventory 21**

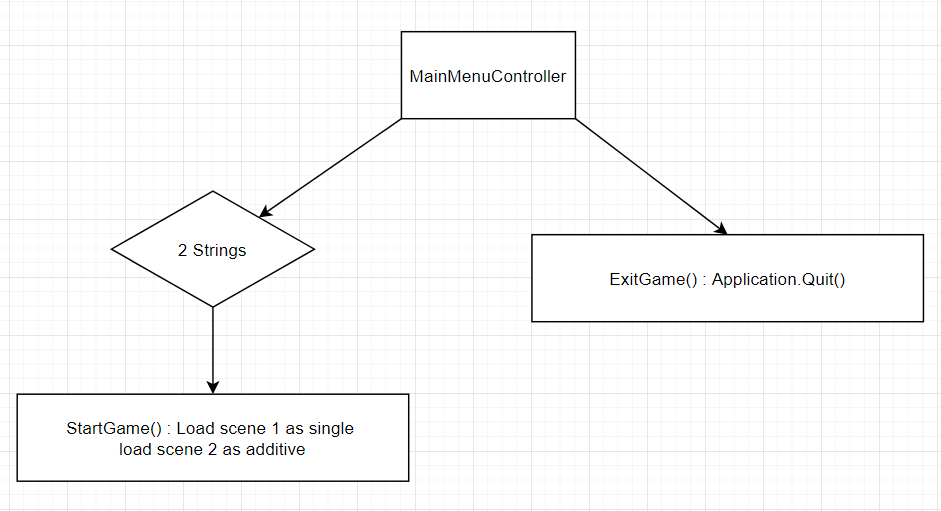
**Battle System 30**

MAIN MENU CONTROLLER

This script provides 2 methods for the 2 buttons on the start of the game. The Start Button and Exit Button. The Start button has a method attached the StartGame().

This method simple loads 2 scenes according the 2 strings that is assigned. 1 of the scene is responsible for the data, the second being a level. It loads the second scene using LoadSceneMode.Additive.

The Exit Button has a method called ExitGame(), which quits the application.



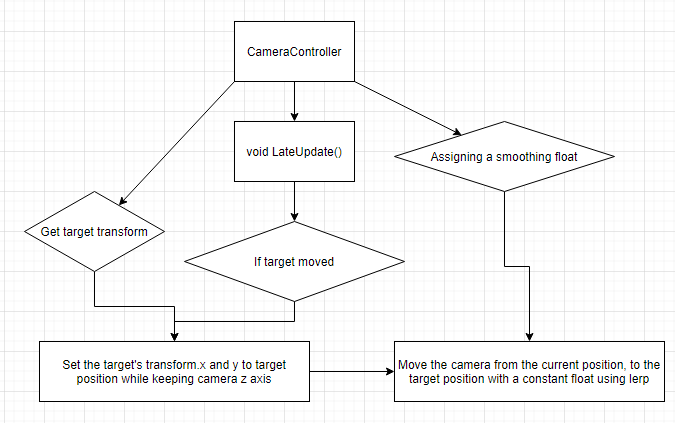
|  |  |
| --- | --- |
| Game object | Start Button |
| Script | MainMenuController() |
| Class | public class MainMenuController : MonoBehaviour |
| Method | public void StartGame() |
| Description | This method loads 2 scenes, the first scene is loaded using the LoadSceneMode.Single, the second is loaded as LoadSceneMode.Additive to add on top of the first scene. This is to help maintain a DataManagement scene that helps to maintain all the information for other scenes. |

|  |  |
| --- | --- |
| Game object | Exit Button |
| Script | MainMenuController() |
| Class | public class MainMenuController : MonoBehaviour |
| Method | public void ExitGame() |
| Description | This method is attached to the Exit Button. This method simple quits the application using Application.Quit(). |

CAMERA FOLLOWER

This method is in the CameraController.cs script, which is attached to the camera object. moves the camera to where the target referenced in the inspector moves, in this case the player. It gets the target position which is the XY axis of the target but for the Z axis, it uses the camera’s camera as moving it in the Z axis affects how the camera sees the game.

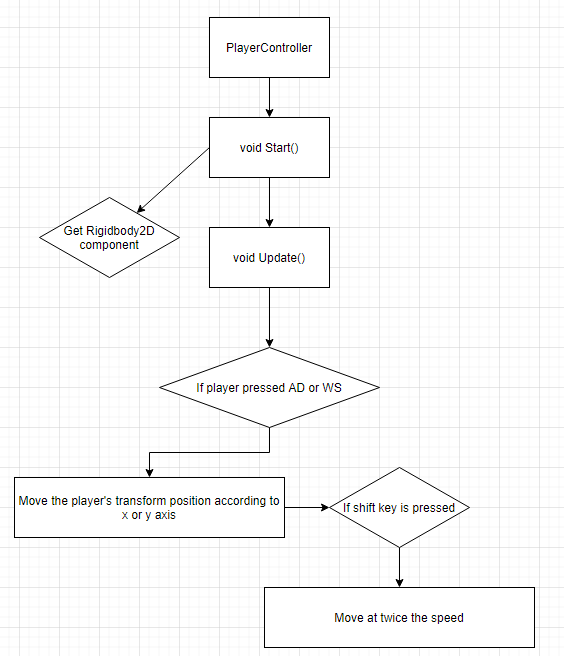
It then lerps the position of the camera and the target according to a smoothing float, to give it a smooth follow. Lerping adjusts the camera towards the target direction slowly, according to the float value in the parameter.



|  |  |
| --- | --- |
| Game object | Player and Camera |
| Script | CamerController.cs |
| Class | public class CameraController : MonoBehaviour |
| Method | void LateUpdate() |
| Description | The method gets the X and Y position of the target, in this case the player. It sets the target position to be the XY but the Z position will be the camera. It then lerps the position between the transform of the camera to the target’s position by a speed of the smoothing float. |

PLAYER CONTROLLER

The player movement is made a script called PlayerController.cs. This player controller is attached to the player object. It gets the Rigidbody2D of the player at the start. After which, it checks every frame if the player has pressed any horizontal and/or vertical keys (AS and/or WD). If it is, it moves the transform of the player accordingly in the XY axis. If further checks if the player held down the shift key. If it is, it moves the transform at twice the speed.



|  |  |
| --- | --- |
| Game object | Player |
| Script | PlayerController.cs |
| Class | public class PlayerController : MonoBehaviour |
| Method | void Start() |
| Description | This class is responsible for primarily moving the character around smoothly overtime via moving the rigidbody.  The speed, Rigidbody2D, a Vector3 called change and a private bool inventory is referenced before the start. The speed moves the rigidbody a set direction. The Vector3 change calls the movement whenever the player moves beyond the change Vector. The bool inventory is to give the player an inventory using the Inventory script later on in the documentation.  The Start method helps the script get the Rigidbody2D by getting the Rigidbody2D component attached to the player object. |

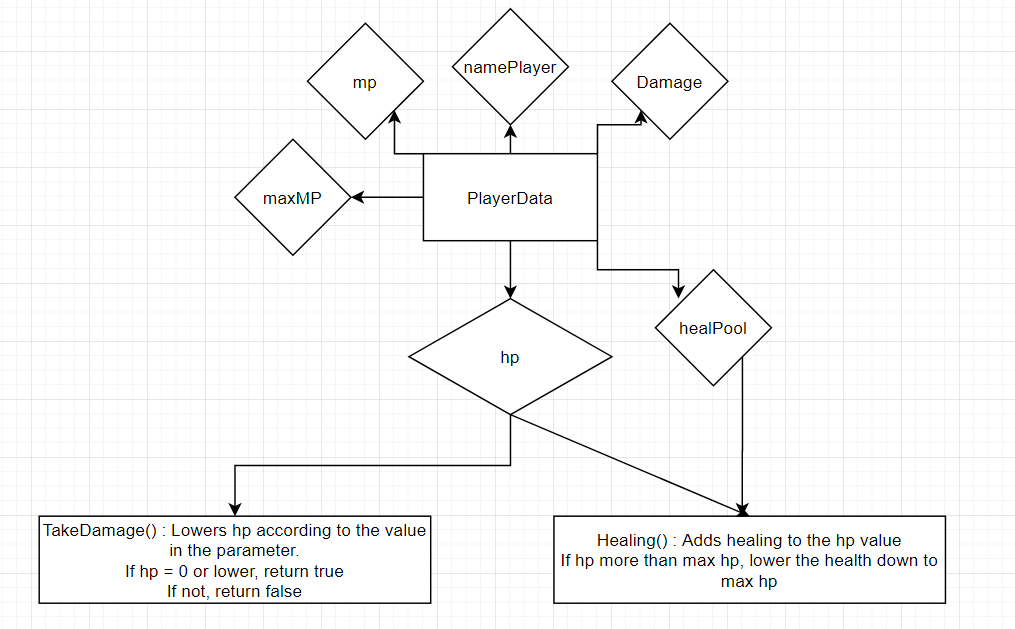
|  |  |
| --- | --- |
| Game object | Player |
| Script | PlayerController.cs |
| Class | public class PlayerController : MonoBehaviour |
| Method | void Update() |
| Description | This method helps to check if the player has pressed any input keys, horizontal (A and D) and/or vertical (W and D). It also checks if the change in position is zero or not. If it is not zero, it calls for the MoveCharacter method. |

|  |  |
| --- | --- |
| Game object | Player |
| Script | PlayerController.cs |
| Class | public class PlayerController : MonoBehaviour |
| Method | void MoveCharacter() |
| Description | This method moves the transform position of the player by any change, the change being the change in x or y axis depending on the input key, by a set distance and speed over time.  If also checks if the player is pressing shift. If the player is holding down the shift key, it moves at double the speed. |

PLAYER DATA

This script contains all the data the game needs to process for battle and Heads Up Display(HUD). It contains the maxHP, maxMP, hp, mp, namePlayer, damage and healPool.

There are also 2 methods, TakeDamage and healing, they are primarily used in battle. TakeDamage reduces the hp value of the player. It further checks if the player’s hp is 0. If it is, it returns true for another bool in another script. If not, it returns false. For healing, it adds the hp according to the heal pool and checks if the hp is more than the maxHP. If it is more, it lowers the hp back down to the maxHP.

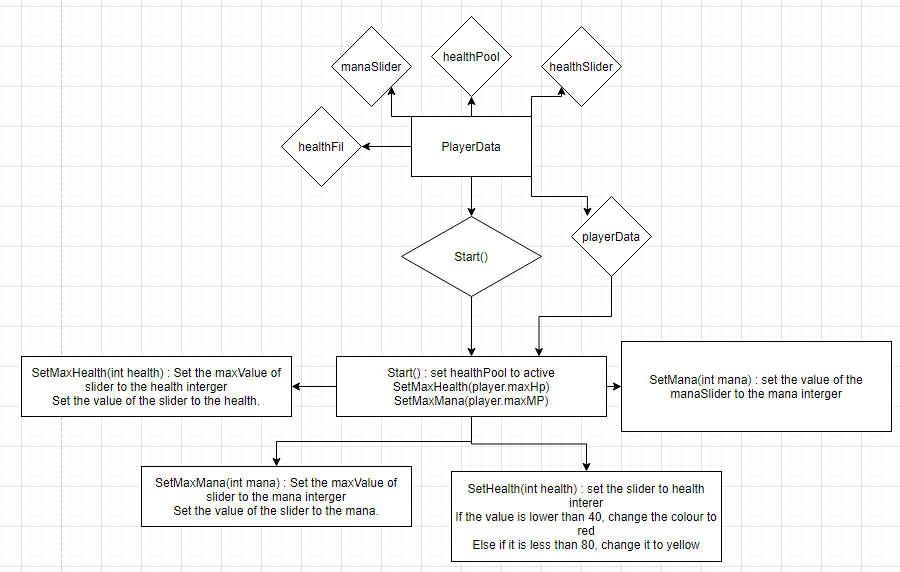


|  |  |
| --- | --- |
| Game object | Player |
| Script | PlayerData.cs |
| Class | public class PlayerData : MonoBehaviour |
| Method | public bool TakeDamage(int dmg) |
| Description | This method lowers the hp of the player according to the damage integer in the parameter. It then checks if the player’s hp is 0. If it is 0, it returns true. If not, it returns false  This bool will be used in another script for battle |

|  |  |
| --- | --- |
| Game object | Player |
| Script | PlayerData.cs |
| Class | public class PlayerData : MonoBehaviour |
| Method | public bool healing(int dmg) |
| Description | This method adds the healPool integer of the player to the hp. It then checks if the hp is more than the maxHP. If it is, it reduces the hp back down to maxHP. |

UI MANAGER

This script is in charge of managing the UI when the player is in the overworld. The UI comprises of several elements, the player’s name, hp and mp bars. The hp and mp bars are made with 2 images each, 1 for the border and 1 for filling up the border. The UI manager adjusts the fill image for the hp and mp, as well as the name, according to the player data. It adjusts all these data by referencing it in the scripts, it also references the player data to follow accordingly.



|  |  |
| --- | --- |
| Game object | UIM |
| Script | UIManager.cs |
| Class | public class UIManager : MonoBehaviour |
| Method | public void Start() |
| Description | This method activates the panel for the name, health and mana bar, followed by setting the value of the health and mana bar using a SetMaxHealth and SetMaxMana method. |

|  |  |
| --- | --- |
| Game object | UIM |
| Script | UIManager.cs |
| Class | public class UIManager : MonoBehaviour |
| Method | public void SetMaxHealth() |
| Description | This method sets the maximum value and current value of the slider of the health bar to be the maxHp of the player according to the player data |

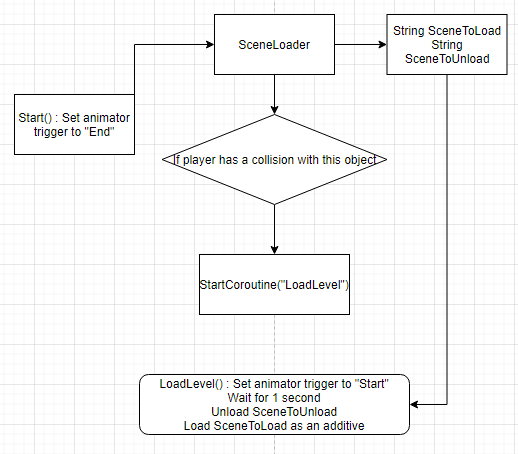
|  |  |
| --- | --- |
| Game object | UIM |
| Script | UIManager.cs |
| Class | public class UIManager : MonoBehaviour |
| Method | public void SetMaxMana() |
| Description | Similar to SetMaxHealth(), this method changes the maximum value and current value of the mana bar to be the maxMP of the player data. |

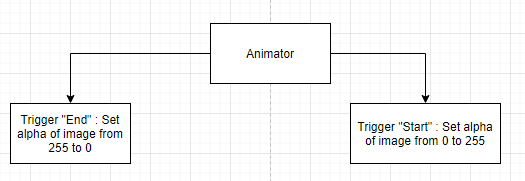
|  |  |
| --- | --- |
| Game object | UIM |
| Script | UIManager.cs |
| Class | public class UIManager : MonoBehaviour |
| Method | public void SetHealth(int health) |
| Description | This method is called whenever there is a change in health. It is not placed in Update() as it will take up some performance doing so. This method sets the value of the slider to the current hp value in the parameter. It then checks if the health is below 40. If it is, it changes the image of the fill image to red. Else it will check if the health is lesser than 80. If it is, change the image to orange. |

|  |  |
| --- | --- |
| Game object | UIM |
| Script | UIManager.cs |
| Class | public class UIManager : MonoBehaviour |
| Method | public void SetMana(int mana) |
| Description | This method changes the value of the mp slider to the mana integer in the parameter. |

UI MANAGER

This script manages an animatior. This animatior gives a fading effect when the scene’s change. There are 2 triggers of the animator, “End” and “Start”, the trigger that starts first is “End”. This animator controls a black image. In the Start trigger, the alpha of the image changes from 0 to 255. For the End trigger, the alpha changes from 255 to 0. These triggers are assigned in the animator under Parameters





|  |  |
| --- | --- |
| Game object | Crossfade image |
| Script | SceneLoader.cs |
| Class | public class SceneLoader : MonoBehaviour |
| Method | public void OnCollisionEnter2D(Collision2D collision) |
| Description | There are 2 strings and an Animator object to reference. The strings are to unload the current scene and load the next scene as an additive to maintain a DataManagement scene.  This method is for transition between levels. Upon detection, it checks if the object has a Player tag. If it detects a player, it starts a coroutine for a method called LoadLevel |

|  |  |
| --- | --- |
| Game object | Crossfade image |
| Script | SceneLoader.cs |
| Class | public class SceneLoader : MonoBehaviour |
| Method | private void Start() |
| Description | This method sets the state of the animator to “End”, running the animation that changes the alpha of the image from 255 to 0. |

|  |  |
| --- | --- |
| Game object | Crossfade image |
| Script | SceneLoader.cs |
| Class | public class SceneLoader : MonoBehaviour |
| Method | IEnumerator LoadLevel() |
| Description | This coroutine sets the trigger to Start, changing the alpha from 0 to 255. It then waits 1 second before it unloads the scene with the name that is the same as the SceneToLoad string. It also loads the scene the SceneToLoad string as its name, as an additive. |

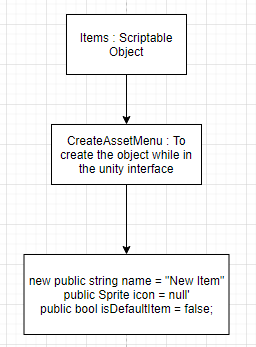
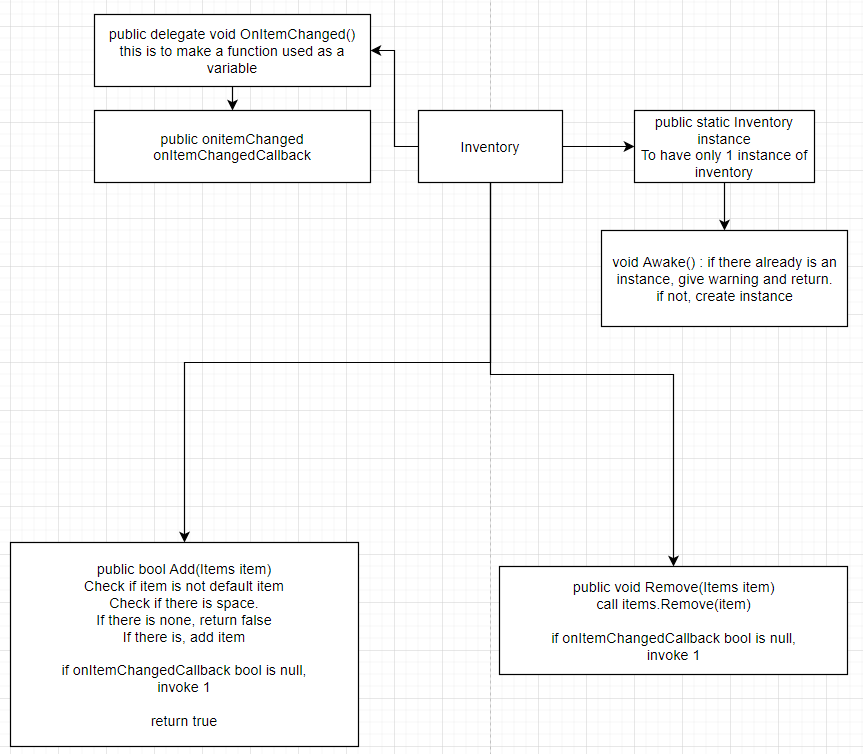
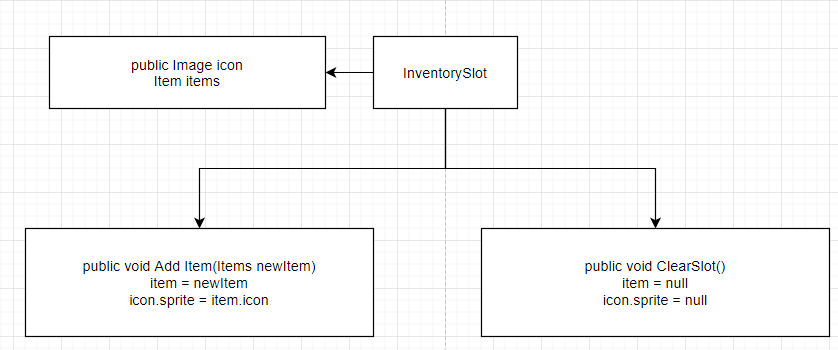
INVENTORY

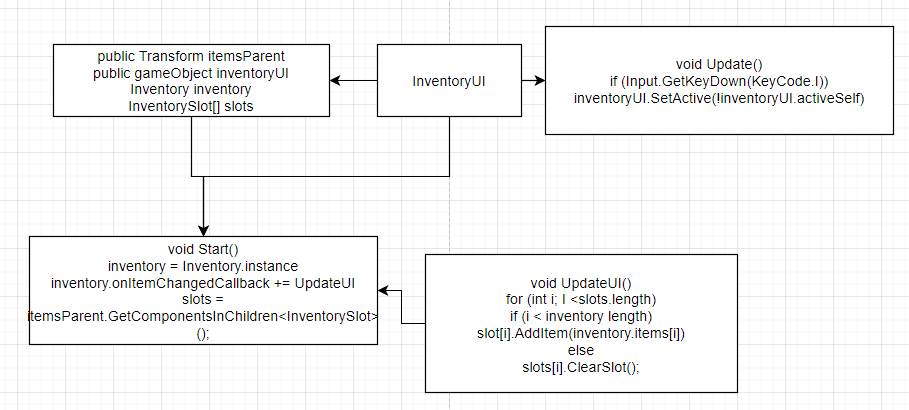
The inventory system is made using scriptable objects to create items and a list to store all of the items. This is done by making a class for the items that uses the ScriptableObject behaviour.

Next is the inventory. It uses an instance to store all of the items. With this instance, other script can add items to this instance. There is a space allocated to the inventory. A method called Add(Items item) checks if there is space to add items before doing anything else.

How other items add items is using this method as Inventory.instance.Add(item). How the inventory UI shows items in the list is using an InventorySlot script. It changes the button in the inventory UI canvas. There are 2 methods to this script. The first is AddItem(Items newItem), it changes the image of the button to be the sprite value of the scriptable object. The second is ClearSlot(), which sets the sprite to null.

All the items are manage under the InventoryUI script. This script references the InventorySlot and Inventory scripts. It creates an instance of the inventory if there is none. It checks if the player has pressed the I key, if it is pressed it sets the inventoryUI to be opposite of its active self by using InventoryUI.SetActive(!inventoryUI.activeSelf). Finally there is an UpdateUI() method.



|  |  |
| --- | --- |
| Game object |  |
| Script | Items.cs |
| Class | public class Items : ScriptableObject |
| Method |  |
| Description | This class helps to create objects without needing a game object or prefab containing a script. The object contains information like a string for its name and a sprite for the icon. An asset menu is also created so that we can add the scriptable object in any time. |

|  |  |
| --- | --- |
| Game object |  |
| Script | Inventory.cs |
| Class | public class Inventory : ScriptableObject |
| Method | Void Awake() |
| Description | This method is to check if there is an instance of the inventory. If there is an instance, it returns null. If there isn’t, it makes an instance |

|  |  |
| --- | --- |
| Game object |  |
| Script | Items.cs |
| Class | public class Items : ScriptableObject |
| Method | public bool Add(Items item) |
| Description | A method called OnItemChanged is made a delegate to be used as a variable. Next, an integer is named space for the inventory space. This method checks if there is space for an item. If there is none, it returns false. If there is, It adds an item. It check if a bool called onitemChangedCallback is null. If it isn’t, it invokes 1. |

|  |  |
| --- | --- |
| Game object |  |
| Script | Items.cs |
| Class | public class Items : ScriptableObject |
| Method | public void Remove(Items item) |
| Description | This method calls for the Remove method. It also checks if the onItemChangedCallback is null. If it isn’t, it invokes 1. |

|  |  |
| --- | --- |
| Game object | InventorySlots prefab |
| Script | InventorySlot.cs |
| Class | public class InventorySlot : ScriptableObject |
| Method | public void AddItem(Items item) |
| Description | This method changes the image of the buttons in the inventory UI to be the item’s sprite in the scriptable object. |

|  |  |
| --- | --- |
| Game object | InventorySlots prefab |
| Script | InventorySlot.cs |
| Class | public class InventorySlot : ScriptableObject |
| Method | public void ClearSlot(Items item) |
| Description | This method nullifies the item in the inventory and the image of the button on the slot prefab. |

|  |  |
| --- | --- |
| Game object | Inventory |
| Script | InventoryUI.cs |
| Class | public class InventoryUI : ScriptableObject |
| Method | void Start() |
| Description | This method creates an instance of the Inventory, create a bool for the inventory called onitemChangedCallback and ask it to call a method called UpdateUI. Finally, it looks for slots in the Inventory game object by GetComponentsInChildren with the InventorySlot name |

|  |  |
| --- | --- |
| Game object | Inventory |
| Script | InventoryUI.cs |
| Class | public class InventoryUI : ScriptableObject |
| Method | void Update() |
| Description | This method detects the player pressing the I key. It sets the active state of the inventory UI to be different from the current active state |

|  |  |
| --- | --- |
| Game object | Inventory |
| Script | InventoryUI.cs |
| Class | public class InventoryUI : ScriptableObject |
| Method | void UpdateUI() |
| Description | This method is called whenever a item changed callback is invoked. It checks for all the slots using a forloop. If the index is lesser than the items count, it adds the item into the inventory. Else, it will remove the item. |

BATTLE SYSTEM

Similar to how the items are made, the enemies uses the scriptable object behaviour as it makes making enemies easier as all of them have the same statistics, health, max health, heal and damage. The script that holds all the information is called EnemyCreator.

The battle system enumeration to declare states as it is easier and requires less codes to change states as compared to bool. There are 5 states in total, START, PLAYERTURN, ENEMYTURN, LOSE and WON. Each state limits how the player can interact. START is when the scene is started. It sets the state to START and calls a coroutine to setup the battle.

This coroutine instantiates the player and enemy prefab that is referenced in the inspector. They are also linked to their respective scripts, PlayerData and for the enemy, EnemyCreator. It also calls a method that will set up the HUD for the player and the enemy which is in a script called BattleHUD. The coroutine then disables the buttons for the player so that the player cannot get first hits before the game starts. It also changes a text object to say “Enemy Approaching”. It waits for 2 seconds before setting it to the player’s turn using state PLAYERTURN.

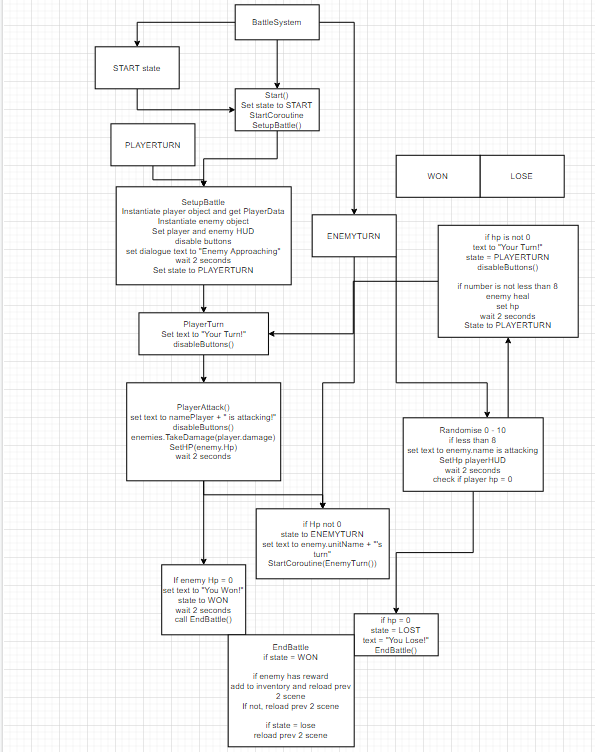
In the player’s turn, the text changes to “Your Turn!” and the buttons are enable. In this state, the player can choose to attack or heal. When the player attacks, it reduces health of the enemy using its EnemyCreator script and checks if the health is 0. If it is, the state changes to WON. Else, the state changes to ENEMYTURN. When the player chooses to heal, it increase the player’s health and sets the state to ENEMYTURN.

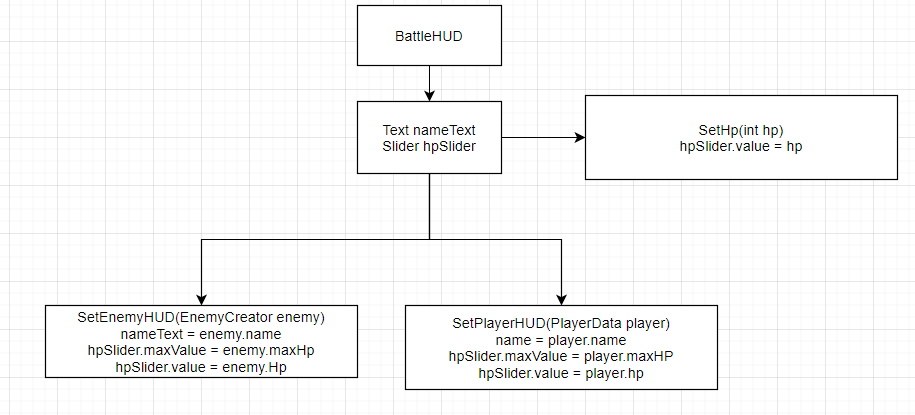
During ENEMYTURN, the enemy will randomise from 0 to 10. If the number is below 8, the enemy attacks, reducing the health of the player and check if it is 0. If it is, set the state to LOSE. If it is not, the state is set to PLAYERTURN. If the number is not below 8, the enemy heals and the state is set to PLAYERTURN.

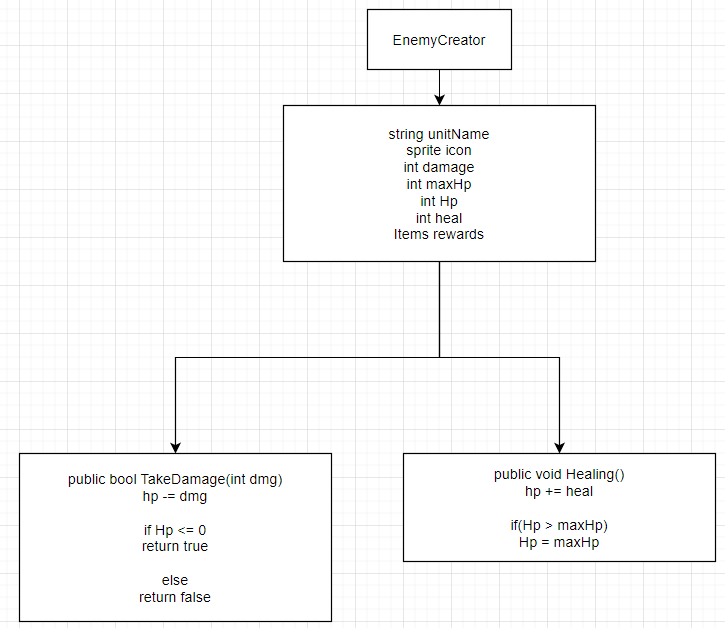
In either WON or LOSE state, it calls the EndBattle() method. This method check if the state is WON or LOSE. If it is WON, check if the enemy has anything in the Reward area. If there is none, disable combat scene and load the previous 2 scenes. If there is, add it to the inventory and re-enable last 2 scenes. If it is lose, re-enable last 2 scenes.

The HUD for the battle is managed by the BattleHUD script. The script references the text and hpSlider image for the health bar. It sets the enemy HUD by referencing the EnemyCreator script and setting the text to the enemy’s name, maxValue and value of the image to the enemy’s maxHP and Hp respectively.

Similarly, it sets the player HUD by referencing the PlayerData and assigning the name and hp value. Another method in the script is called SetHP(int hp), this sets the slider value to the hp in the parameter.







|  |  |
| --- | --- |
| Game object | EnemyHUD gameobject and PlayerHUD gameobject |
| Script | BattleSystem.cs |
| Class | public enum BattleState { INACTIVE, START, PLAYERTURN, ENEMYTURN, WON, LOSE } |
| Method |  |
| Description | This is to create the enums needed for the battle system |

|  |  |
| --- | --- |
| Game object | EnemyHUD gameobject and PlayerHUD gameobject |
| Script | BattleSystem.cs |
| Class | public class BattleSystem : MonoBehaviour |
| Method | void Start() |
| Description | This method is to change the state to START and call the SetupBattle() function |

|  |  |
| --- | --- |
| Game object | EnemyHUD gameobject and PlayerHUD gameobject |
| Script | BattleSystem.cs |
| Class | public class BattleSystem : MonoBehaviour |
| Method | IEnumerator SetupBattle() |
| Description | This method instantiates the player and enemy prefab, calls the SetHUD methods for the player and enemy that is found in the BattleHUD.cs script. This method also sets the dialogueText to “Enemy Approaching”, waits 2 seconds and set the state to PLAYERTURN |

|  |  |
| --- | --- |
| Game object | EnemyHUD gameobject and PlayerHUD gameobject |
| Script | BattleSystem.cs |
| Class | public class BattleSystem : MonoBehaviour |
| Method | void PlayerTurn() |
| Description | This method is to change the dialogue text to “Your Turn!” and calls a disableButtons() method. |

|  |  |
| --- | --- |
| Game object | EnemyHUD gameobject and PlayerHUD gameobject |
| Script | BattleSystem.cs |
| Class | public class BattleSystem : MonoBehaviour |
| Method | IEnumerator PlayerAttack() |
| Description | This coroutine is called whenever the player pressed the Attack button. It changes the text to “Player is attacking!” and disable the buttons. The enemy’s TakeDamage method is called and the Hp HUD is set up. The coroutine waits 2 seconds before checking if the enemy’s hp is 0.  If it is, it changes the text to “You Won!”, changing the state to WON before waiting 2 seconds and calling the EndBattle() method.  If it is not, it changes the state to ENEMYTURN, changing the text to “Enemy’s turn” and calling a coroutine called EnemyTurn() |

|  |  |
| --- | --- |
| Game object | EnemyHUD gameobject and PlayerHUD gameobject |
| Script | BattleSystem.cs |
| Class | public class BattleSystem : MonoBehaviour |
| Method | IEnumerator PlayerHeal() |
| Description | This coroutine is called whenever the player pressed the Heal button. It changes the text to “Player is healing!” and calling the healing function in PlayerData. It also sets up the HP HUD and disables the button before waiting 1 second to change the turn to the enemy’s turn, calling the EnemyTurn() coroutine. |

|  |  |
| --- | --- |
| Game object | EnemyHUD gameobject and PlayerHUD gameobject |
| Script | BattleSystem.cs |
| Class | public class BattleSystem : MonoBehaviour |
| Method | IEnumerator EnemyTurn |
| Description | Firstly, the coroutine randomises a number between 0 and 10 and waits 2 seconds.  If the number is lesser than 8, the text changes to “Enemy is attacking” and calls the TakeDamage() method in the PlayerData. It also calls for the SetHp method and waits 2 more seconds. After that, it checks if the player’s hp is 0. If it is, it sets the state to LOST, changing the text to “You Lost!” and calling the EndBattle() method. If the player’s hp is not 0, it changes the text to “Your Turn!”, setting the state to PLAYERTURN and calling the PlayerTurn() method.  If the number is not lesser than 8, the text changes to “Enemy is healing”, calling the healing method in the EnemyCreator and the SetHP() method. It waits 2 more seconds before setting it to the player’s turn. |

|  |  |
| --- | --- |
| Game object | EnemyHUD gameobject and PlayerHUD gameobject |
| Script | BattleSystem.cs |
| Class | public class BattleSystem : MonoBehaviour |
| Method | void EndBattle() |
| Description | This method checks if the state is WON or LOSE. If it is WON, the method checks if there is anything in the rewards area in the enemy scriptable object.  If there is none, reload previous 2 scenes. If there is a reward, add it to the inventory instance and reload previous 2 scenes.  If the state is LOST, reload previous 2 scenes |

|  |  |
| --- | --- |
| Game object | Attack and Heal Button |
| Script | BattleSystem.cs |
| Class | public class BattleSystem : MonoBehaviour |
| Method | public void OnAttackButton() & public void OnHealButton() |
| Description | For the OnAttackButton(), this button is attached to the attack button and it checks if it is the player’s turn. If its not, return nothing. If it is, call the PlayerAttack() coroutine.  For the OnHealButton(), this button is attached to the heal button and it checks if it is the player’s turn. If its not, return nothing. If it is, call the PlayerHeal() coroutine |

|  |  |
| --- | --- |
| Game object | Attack and Heal Button |
| Script | BattleSystem.cs |
| Class | public class BattleSystem : MonoBehaviour |
| Method | public void disableButtons() |
| Description | This method sets the active state of the heal and attack button to be different from its current active state |

|  |  |
| --- | --- |
| Game object |  |
| Script | BattleSystem.cs |
| Class | public class BattleSystem : MonoBehaviour |
| Method | public void enableScene() |
| Description | This method disable “Combat” scene and reloads the DataManagement and TutorialLevel scene |

|  |  |
| --- | --- |
| Game object | nameText.text and hpSlider image |
| Script | BattleHUD.cs |
| Class | public class BattleHUD : MonoBehaviour |
| Method | public void SetEnemyHUD(EnemyCreator enemy) |
| Description | This method sets the nameText to the enemy’s name, maxValue of the hpSlider to be the maxHp of the enemy and value to the enemy’s Hp. |

|  |  |
| --- | --- |
| Game object | nameText.text and hpSlider image |
| Script | BattleHUD.cs |
| Class | public class BattleHUD : MonoBehaviour |
| Method | public void SetPlayerHUD(PlayerData player) |
| Description | This method sets the nameText to the player’s name, maxValue of the hpSlider to be the maxHp of the player and value to the player’s Hp. |

|  |  |
| --- | --- |
| Game object | hpSlider image |
| Script | BattleHUD.cs |
| Class | public class BattleHUD : MonoBehaviour |
| Method | public void SetHP(int hp) |
| Description | This method updates the value of the hpSlider to the hp in the parameter. |

|  |  |
| --- | --- |
| Game object |  |
| Script | EnemyCreator.cs |
| Class | public class EnemyCreator : ScriptableObject |
| Method |  |
| Description | This script is for creating scriptable objects for the enemy without needing to create a prefab. It uses the CreateAssetMenu method so that we can add in enemies at any time. |

|  |  |
| --- | --- |
| Game object | Enemy ScriptableObject |
| Script | EnemyCreator.cs |
| Class | public class EnemyCreator : ScriptableObject |
| Method | public bool TakeDamage(int dmg) |
| Description | The reason why this method is a bool is to return true for the IsDead bool in BattleSystem. This method reduces the Hp of the enemy by the damage in the parameter. If the hp is equal or lesser than 0, return true. If not, return false. |

|  |  |
| --- | --- |
| Game object | Enemy ScriptableObject |
| Script | EnemyCreator.cs |
| Class | public class EnemyCreator : ScriptableObject |
| Method | public void healing() |
| Description | The method adds the heal integer in the script to the Hp. If the Hp is more than the maxHp, it brings the value of the Hp down to maxHp. |