

Project: Dimension Fantasia
Team No.: Team 9
Class: CSE 3310.003 - Fall 2020 Module: System Requirements Analysis (SRA) Deliverable: SRA Document & Test Plan

Version: [2.0]

Date: [11/30/2020]

Contributors:

Ganesh Kandel

Jorge Estrada

Bevan Philip

Zachary Trumbaturi

Ajaya Pyakurel

Revision History

<i>Version number</i>	<i>Date</i>	<i>Originator</i>	<i>Reason for change</i>	<i>High level description of changes</i>
1.0	10/15/2020	Ganesh Kandel Jorge Estrada Bevan Philip Zachary Trumbaturi Ajaya Pyakurel	Initial draft	
2.0	11/24/2020	Ganesh Kandel Jorge Estrada Bevan Philip Zachary Trumbaturi Ajaya Pyakurel	Updates	

TABLE OF CONTENTS

1. Introduction and Project Overview	4
2. Objectives	5
2.1 BUSINESS OBJECTIVES	5
2.2 SYSTEM OBJECTIVES	5
3. System Context Diagram	6
4. Systems Requirements	7
4.1 "MAIN MENU" REQUIREMENTS	7
4.2 "CHANGE SETTINGS" REQUIREMENTS	7
4.3 "LEVEL SELECT" REQUIREMENTS	9
4.4 "EQUIPMENT" REQUIREMENTS	10
4.5 "BATTLE" REQUIREMENTS	11
4.6 "PLAYER STATISTICS" REQUIREMENTS	14
5. Software Processes and Infrastructure	17
5.1 HARDWARE AND INFRASTRUCTURE	17
5.2 GAME ENTITY STRUCTURE UML	17
5.3 USER AND GAME SEQUENCE DIAGRAM UML	18
5.4 GAME PLAY SEQUENCE DIAGRAM UML	19
5.5 USER SYSTEM INTERACTION DIAGRAM UML	20
5.6 GAME PROCESS FLOW DIAGRAM UML	21
5.7 SETTINGS PAGE ACTIVITY DIAGRAM UML	22
5.8 COMBAT ACTIVITY DIAGRAM UML	23
5.9 GAME PLAY STATE MACHINE DIAGRAM UML	24
5.10 CONCEPTUAL DATA MODEL - DATABASE	24
5.11 Test Plan	26
A. Introduction and Plan of Approach	26
B. Test Cases: "MAIN MENU"	28
C. Test Cases: "Settings"	29
D. Test Cases: "Battle System "	30
E. Test Cases: "Select Level"	33
F. Test Cases: "Character Customization"	34
6. Assumptions and Constraints	36
6.1 ASSUMPTIONS	36
6.2 CONSTRAINTS	36
6.3 OUT OF SCOPE MATERIAL	36

7. Delivery and Schedule	37
8. Stakeholder Approval Form	38
9. User Manual	40
10. Source Code	49
Appendix:	58

1. Introduction and Project Overview

Group 9 has been employed by a small game development company to introduce a new RPG game to the mobile game industry. This RPG will be new in the sense that it will have a very simple interface and will also be very playable, which are traits RPG's in the industry are lacking. This RPG has the codename the Dimension Fantasia and it will be a game in which the player will embark on adventures and fight various types of enemies. As the player progresses and defeats enemies, the player will be rewarded with XP and the player's statistics will increase, but with statistics increasing, enemy difficulty will as well. The player will be able to face off against different kinds of enemies, and during the combat the player will have a menu that prompts the player to attack, block or run away from the fight. The interface of this game will be very simple and will allow the user to properly play the game with no complex features or functions. The simplicity will encourage many people to be willing to play the game and enjoy it.

2. Objectives

2.1 BUSINESS OBJECTIVES

The following is a list of business objectives:

Objective 1: “Main menu display” functionality

- Level Select
- Change settings
- Equipment Settings/Shop
- Exit game

Objective 2: “Change Settings” functionality

- Input Sensitivity
- Assist Mode (Difficulty)

Objective 3: “Game level selection” functionality

- Dungeon selection

Objective 4: “Equipment selection” functionality

- Weapons
- Consumables
- Level Upgrades
- Gold use as currency

Objective 5: “Battle” Functionality

- Player combat
- Enemy combat

Objective 6: “Player Statistics” functionality

- Health
- Ability Points
- Equipment Points
- XP

2.2 SYSTEM OBJECTIVES

The following is a list of system objectives:

Objective 1: System will be an Android application

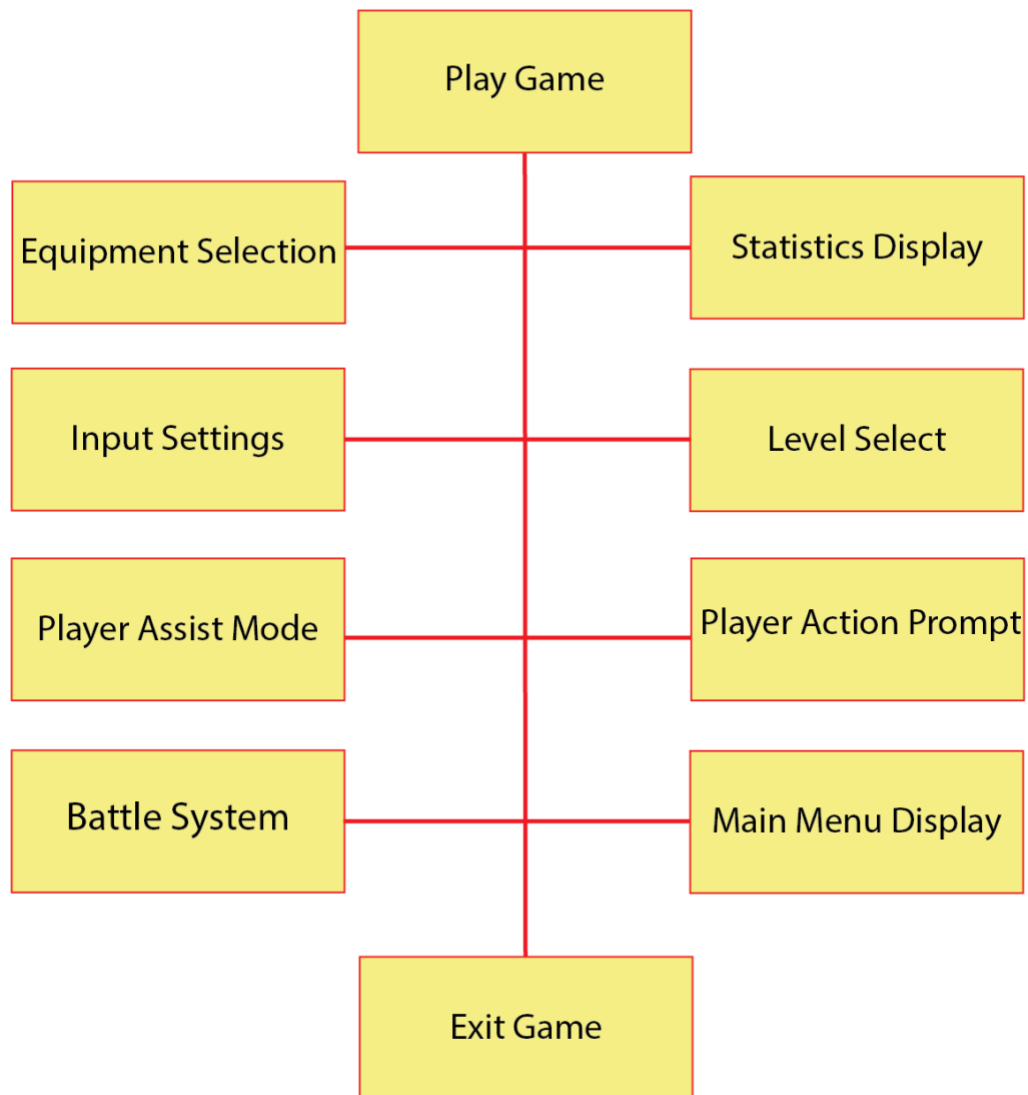
Objective 2: System will be built using Unity

Objective 3: C# will be used

Objective 4: User interface will be intuitive.

Objective 5: User interface will be touch sensitive

3. System Context Diagram



4. Systems Requirements

4.1 “MAIN MENU” REQUIREMENTS

Requirement Title:	Title Screen
Sequence No:	1
Short Description:	Shows title of the game, and different settings.
Description:	<p>On boot up, the title of the game will be presented to the player, along with a few choices for the player to make.</p> <ul style="list-style-type: none">-Select Level-Settings-Equipment Selection/Settings-Shop-Exit Game
Pre-Conditions:	Start up the application
Post Conditions:	N/A
Other Attributes:	N/A

4.2 “CHANGE SETTINGS” REQUIREMENTS

Requirement Title:	Options Screen
Sequence No:	1
Short Description:	Allows User to Change Settings
Description:	<p>This menu gives the user the opportunity to tailor their experience to their liking by adjusting some of the following:</p> <ul style="list-style-type: none">- Input Sensitivity- Assist Mode
Pre-Conditions:	Clicked on “Options” from title screen
Post Conditions:	N/A
Other Attributes:	N/A

Requirement Title:	Options Screen
Sequence No:	2
Short Description:	Allows user to adjust input sensitivity
Description:	<p>This menu allows the user to increase or decrease the input sensitivity according to their preference. This will change how far or how fast the user needs to swipe to get a swipe input.</p> <p>This menu will also have a real time diagram that allows the user to practice with their sensitive setting of choice.</p>
Pre-Conditions:	Clicked on "Input sensitivity" from title screen
Post Conditions:	The Sensitivity Option will be saved for other parts of the game to reference.
Other Attributes:	N/A

Requirement Title:	Options Screen
Sequence No:	3
Short Description:	Allows user to enable assist mode
Description:	<p>This setting is a checkbox that when checked will enable assist mode. Assist mode has to deal with the timing mechanism that is implemented into the battle system, which is when the player attacks he/she has to time the attack on the enemy. When assist mode is enabled then the game does it automatically for the player, which in turns makes it a bit easier to battle.</p>
Pre-Conditions:	Click on Settings button
Post Conditions:	Assist mode will be enabled and manual timing will be off
Other Attributes:	N/A

4.3 “LEVEL SELECT” REQUIREMENTS

Requirement Title:	Level Select
Sequence No:	1
Short Description:	Shows different dungeons for the user to pick.
Description:	This is the hubworld for where the player can choose different levels. These dungeons will unlock as the player conquers each one and claims their reward.
Pre-Conditions:	Clicked on “Start Game” from title screen
Post Conditions:	Every time the user leaves this menu the data is saved.
Other Attributes:	Will show completed levels based on save data.

Requirement Title:	Level Select
Sequence No:	2
Short Description:	Shows Dungeon Stats before Entering.
Description:	This sub-menu will show different attributes the dungeon has such as: <ul style="list-style-type: none">- Difficulty- Amount of Floors
Pre-Conditions:	Player chose the ‘n’t’h dungeon
Post Conditions:	N/A
Other Attributes:	Changes statistics depending on which level the player chooses.

4.4 “EQUIPMENT” REQUIREMENTS

Requirement Title:	Equipment
Sequence No:	1
Short Description:	Changes Character Equipment
Description:	<p>The character will have the opportunity to change their equipment anytime prior to entering a dungeon.</p> <p>These pieces of equipment include:</p> <ul style="list-style-type: none">- Weapons- Accessories
Pre-Conditions:	Choose “Character Edit” on the Level Select
Post Conditions:	Saves settings for the character.
Other Attributes:	N/A

Requirement Title:	Equipment
Sequence No:	2
Short Description:	Changes Character Weapons
Description:	<p>This submenu will allow characters to change their weapon. These weapons have different capabilities and permit different playstyles.</p> <p>As the player progresses, the player may hold more than one weapon in a fight.</p>
Pre-Conditions:	Chose “Weapons” from Character Edit Menu
Post Conditions:	Saves weapon loadout for dungeon treks
Other Attributes:	Weapon Variety improves the further into the game the player gets

Requirement Title:	Equipment
Sequence No:	3
Short Description:	Changes Character Accessories
Description:	<p>This submenu will allow characters to change their accessories. These augment weapons by giving them different attributes or by giving the player different capabilities in combat to allow for further flexibility in playstyles.</p> <p>As the character levels up, they may increase the total capacity of the accessories they carry.</p>
Pre-Conditions:	<p>Choose "Accessories" from Character Edit Menu</p> <p>Must own at least one accessory.</p>
Post Conditions:	Saves accessory loadout for dungeon treks
Other Attributes:	N/A

4.5 “BATTLE” REQUIREMENTS

Requirement Title:	Battle
Sequence No:	1
Short Description:	Player Combat when battle starts
Description:	<p>The player has the opportunity to attack.</p> <p>Options are defined based on the type of equipment and stats the player comes in with. The player chooses the attack they want to use, followed by the enemy they wish to attack.</p>
Pre-Conditions:	Must have entered a battle.
Post Conditions:	The enemy is prompted to take their turn.
Other Attributes:	N/A

Requirement Title:	Battle
Sequence No:	2
Short Description:	Player attack on the enemy
Description:	Once the attacks are chosen, the player partakes in a type of timing minigame that can give the opportunity to improve the amount of damage or other beneficial effects to make combat tilt into the player favor.
Pre-Conditions:	Must have prompted player to attack
Post Conditions:	The enemy will lose health based on attack done by the player
Other Attributes:	N/A

Requirement Title:	Battle
Sequence No:	3
Short Description:	Enemy combat with the player
Description:	<p>After the player has conducted the characters attack the enemy will be able to attack the player.</p> <p>The opponent has the opportunity to attack. Depending on the enemy, they can do either simple actions, or more complex and deceitful actions.</p>
Pre-Conditions:	Player attacks the enemy
Post Conditions:	The player will lose health based on the attack done by the enemy
Other Attributes:	N/A

Requirement Title:	Battle
Sequence No:	4
Short Description:	Player Reaction to enemy attack
Description:	<p>After the enemy attacks the player can sway off enemy attacks in different ways, which include Blocking, Dodging and Parry</p> <p>Blocking Done by holding a finger on the screen. This will reduce damage. If done closer to when the enemy is about to hit, the damage will be reduced further for perfect timing. Being early is OK, however being late can deal full damage to the player.</p> <p>Dodging Done by holding, then swiping a finger on the screen. This will initiate a block which is fine, but the swipe action does the dodge. Dodging will allow the player to dodge all damage taken if done with proper timing. More difficult than blocking, but has higher reward of no damage.</p> <p>Parry Done by holding, then releasing the finger from the screen without swiping. The timing on this tactic is the most strict of the three, however if performed correctly can lead to a possible counter attack with the enemy along with all damage being mitigated. Projectiles will also be reflected back at the enemy.</p>
Pre-Conditions:	Must have finished player combat turn and Enemy will have attacked the player
Post Conditions:	The player is prompted to take their turn.
Other Attributes:	N/A

Requirement Title:	Battle
Sequence No:	5
Short Description:	Player wins battle
Description:	<p>After the player wins the battle, the player will be congratulated.</p> <p>Along with that, the player will be rewarded with XP, Ability Points and Gold that the player can spend in the equipment shop</p>
Pre-Conditions:	Enemy health will have completely depleted
Post Conditions:	Player will be returned to the main menu
Other Attributes:	N/A

Requirement Title:	Battle
Sequence No:	6
Short Description:	Enemy wins the battle
Description:	Whenever the player's health completely depletes the player will be told that he/she has lost and will be able to replay the level.
Pre-Conditions:	Player health will have completely depleted
Post Conditions:	Player will be returned to the main menu and can replay the level
Other Attributes:	N/A

4.6 "PLAYER STATISTICS" REQUIREMENTS

Requirement Title:	Player Statistics
Sequence No:	1
Short Description:	The statistics held by the player character
Description:	<p>These statistics hold the values that the player character holds that can be interacted with prior to battle, or increased during battle.</p> <p>These statistics include:</p> <ul style="list-style-type: none"> - Health - Ability - Experience - Equipment
Pre-Conditions:	Must have started the application at least once.
Post Conditions:	N/A
Other Attributes:	N/A

Requirement Title:	Player Statistics
Sequence No:	2
Short Description:	Health Points for durability in battle
Description:	Health points are what determine the durability of the player character in battle. Changes to this are made either when an enemy successfully hits the player or when they heal. This value can increase by using experience points.
Pre-Conditions:	Must have been hit by the enemy or used an item to heal
Post Conditions:	Save changes to health points
Other Attributes:	N/A

Requirement Title:	Player Statistics
Sequence No:	3
Short Description:	How often the player can use abilities
Description:	These statistics hold the values of ability. These abilities allow for modification to standard attacks, to allow for different variance such as higher damage or other status effects at the cost of using ability points as a resource. These can increase by gaining enough experience.
Pre-Conditions:	Must have chosen to use an extra ability.
Post Conditions:	Changes to the value are saved after using the ability.
Other Attributes:	Not useful at the start because accessories are lacking, but as those increase, so will the ability points and what to spend them on.

Requirement Title:	Player Statistics
Sequence No:	4
Short Description:	How many accessories the player can use.
Description:	The player can equip many pieces of gear and accessories using equipment points. These equipment points can level up when experience points are granted and can allow for even more equipment to be worn.
Pre-Conditions:	Must have started the application at least once.
Post Conditions:	Saves values depending on the final equipment layout chosen.
Other Attributes:	N/A

Requirement Title:	Player Statistics
Sequence No:	5
Short Description:	Experience to improve player capability.
Description:	Experience Points allow for the player to increase their capabilities in combat after winning a battle. Once enough experience points have been achieved, any of three (Health, Ability, Equipment) may be increased to allow for more customization and growth.
Pre-Conditions:	Must have achieved the amount of experience necessary.
Post Conditions:	Saves values depending on the player choice.
Other Attributes:	N/A

5. Software Processes and Infrastructure

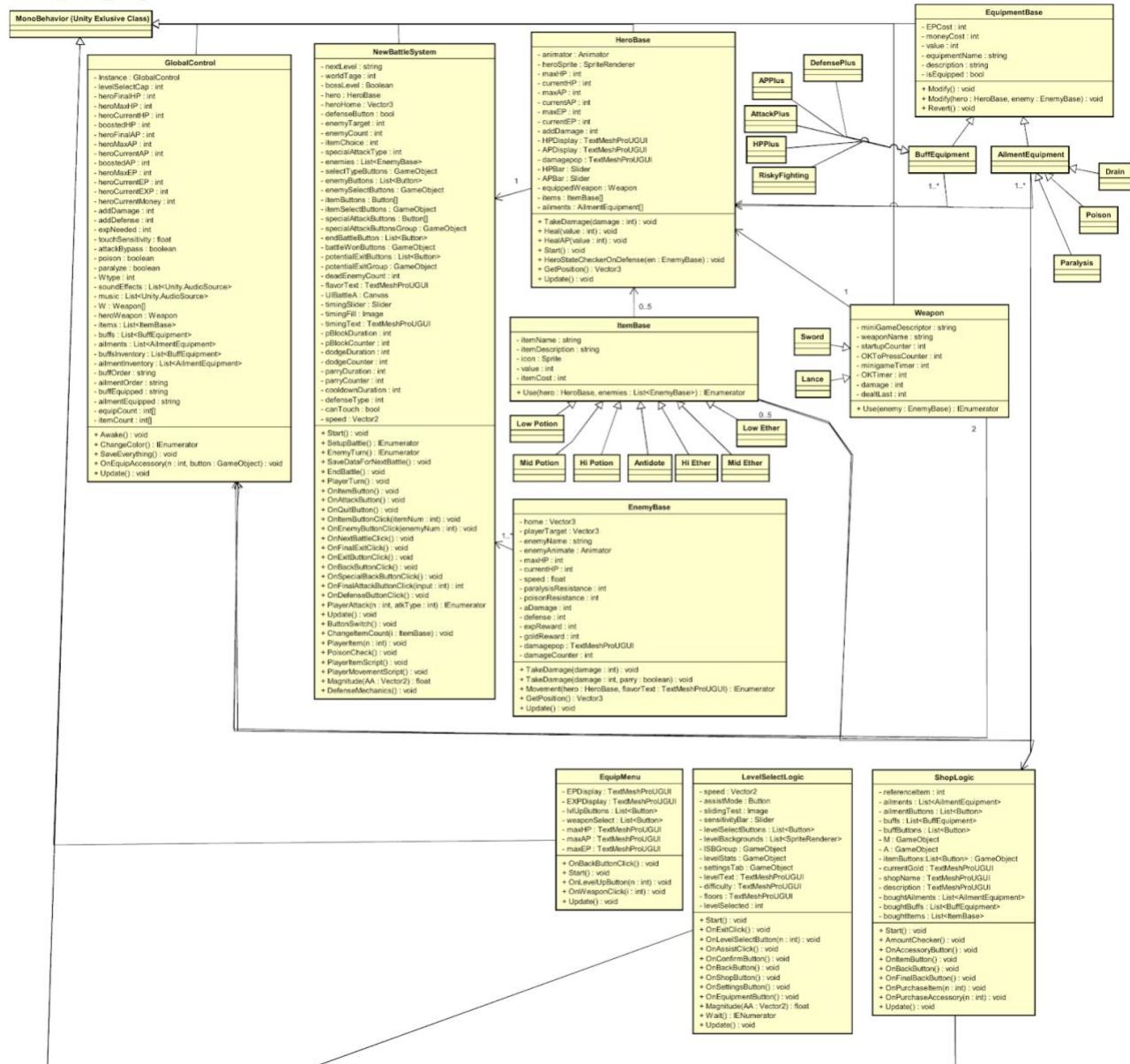
5.1 HARDWARE AND INFRASTRUCTURE

Program runs on an android platform on mobile devices only.

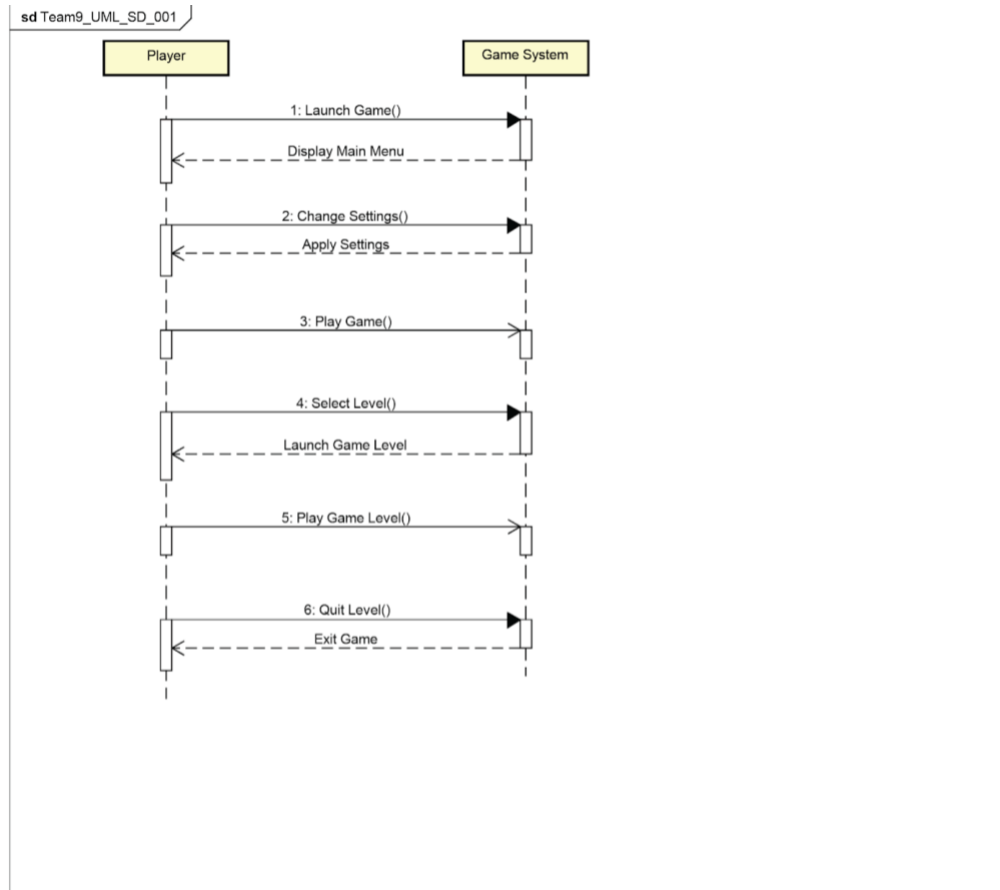
5.2 GAME ENTITY STRUCTURE UML

Team9_UML_CD_001

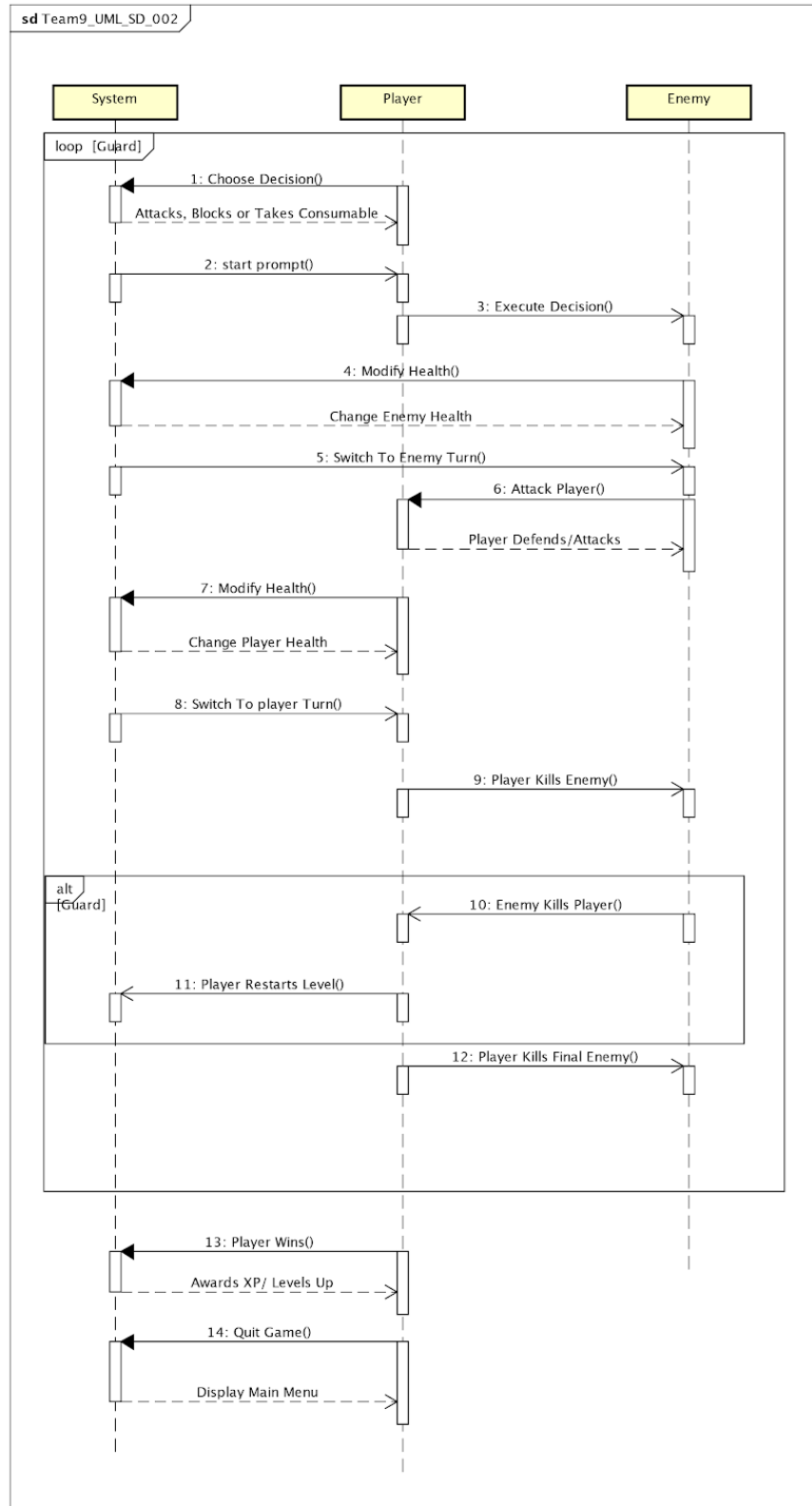
2020/12/01 astah* Evaluation



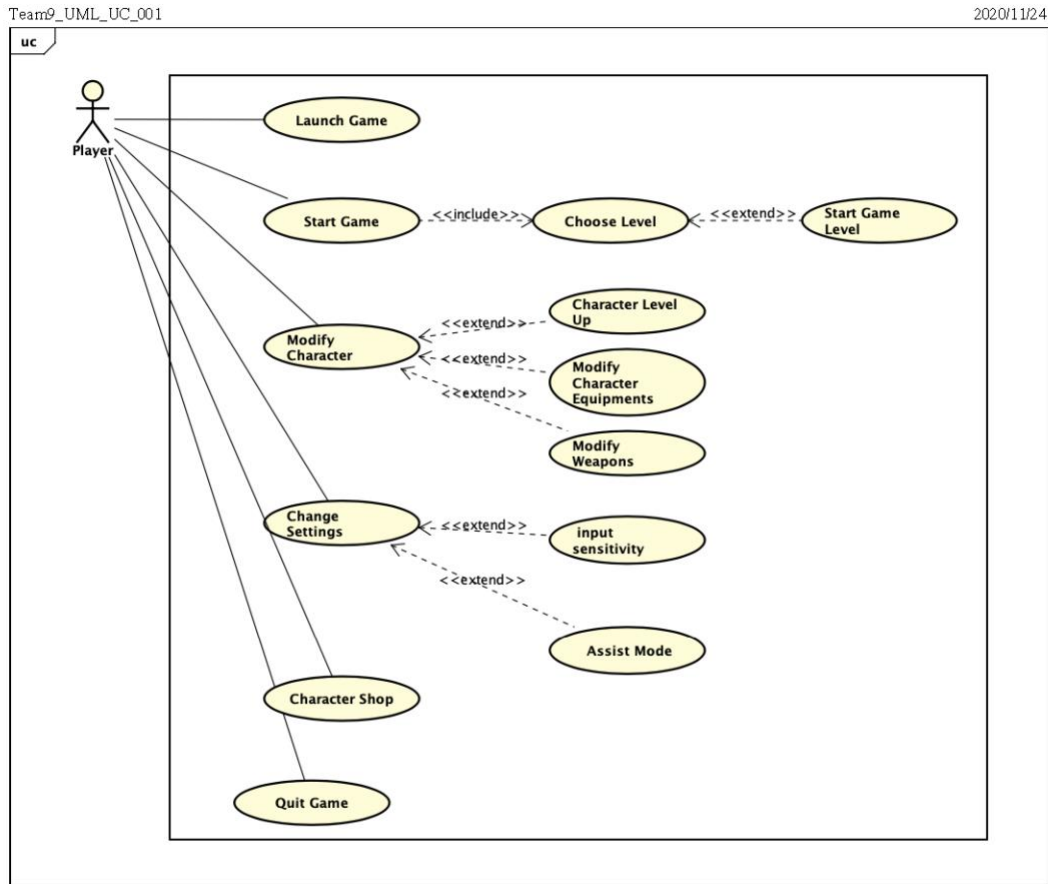
5.3 USER AND GAME SEQUENCE DIAGRAM UML



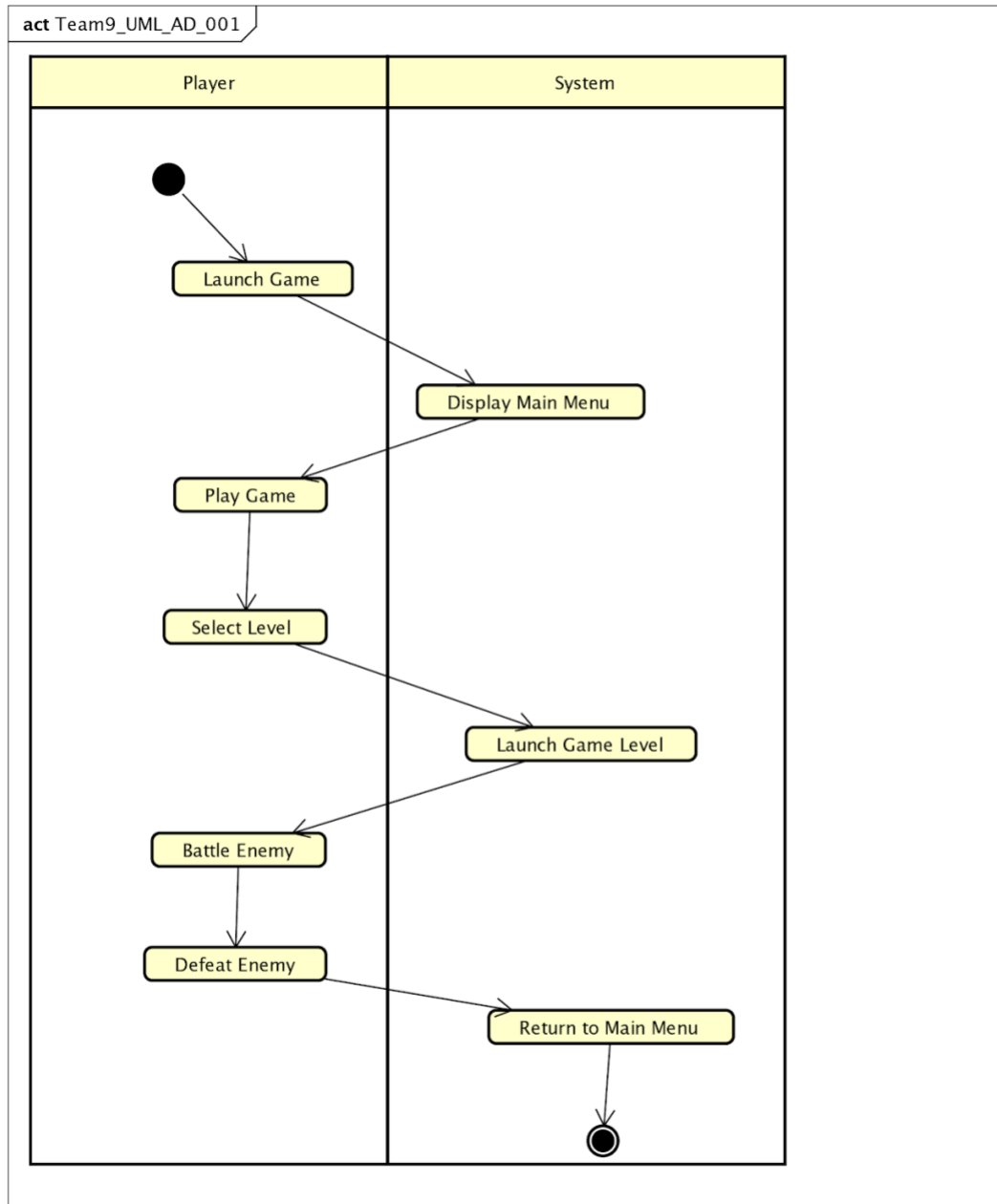
5.4 GAME PLAY SEQUENCE DIAGRAM UML



5.5 USER SYSTEM INTERACTION DIAGRAM UML



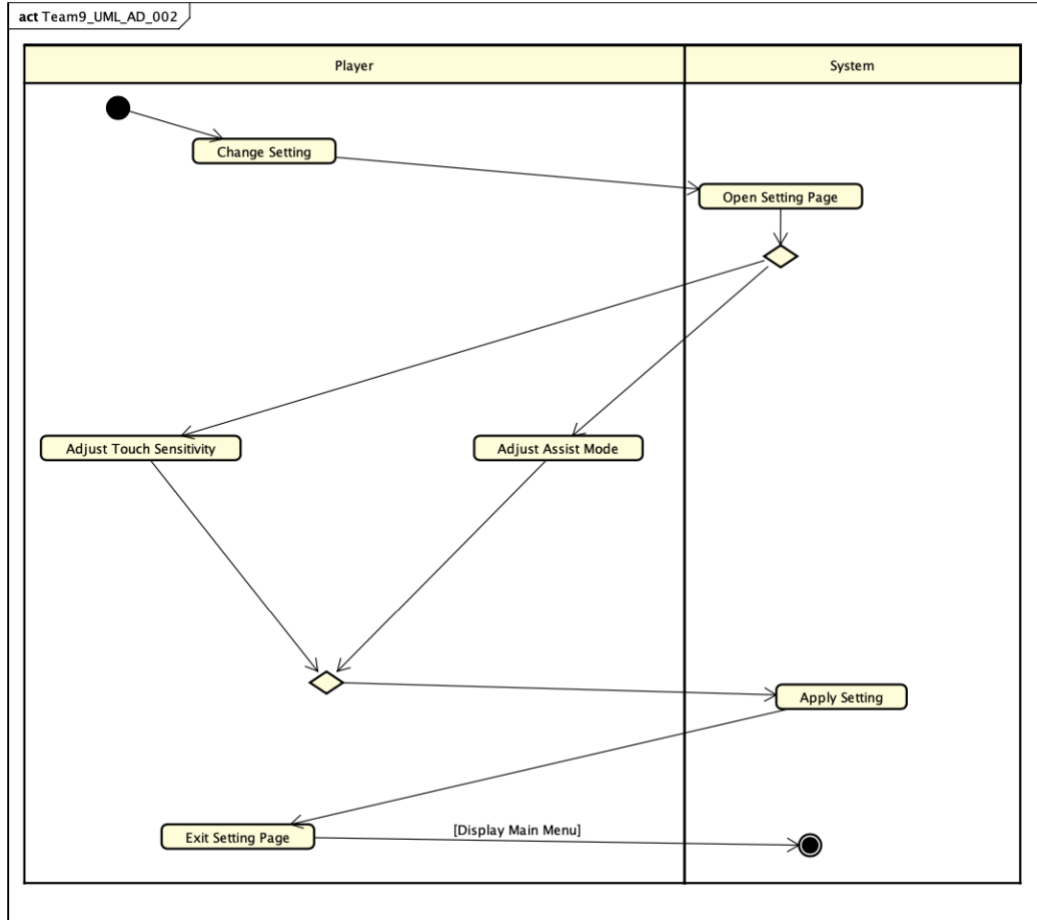
5.6 GAME PROCESS FLOW DIAGRAM UML



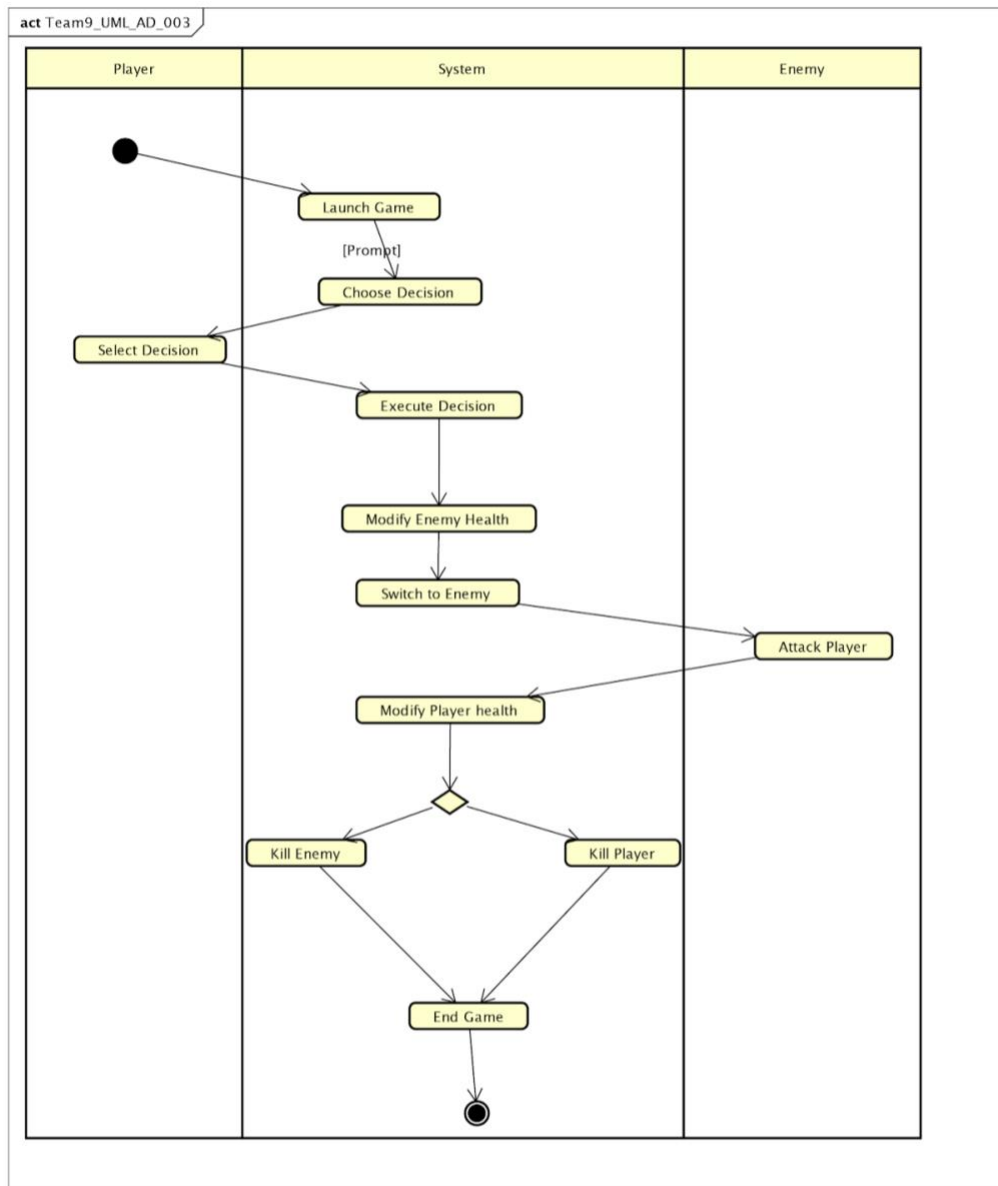
5.7 SETTINGS PAGE ACTIVITY DIAGRAM UML

Team9_UML_AD_002

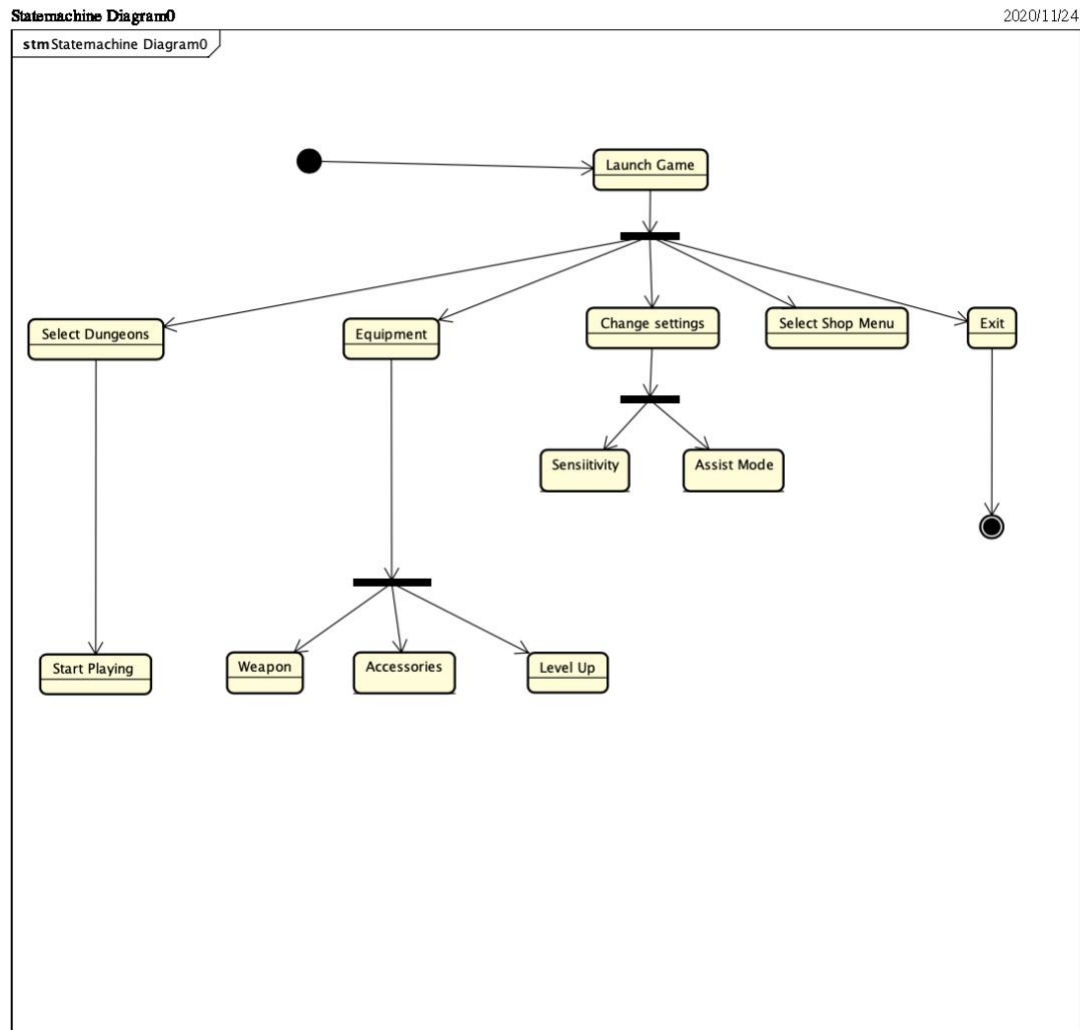
2020/11/24



5.8 COMBAT ACTIVITY DIAGRAM UML



5.9 GAME PLAY STATE MACHINE DIAGRAM UML



5.10 CONCEPTUAL DATA MODEL - DATABASE

Database is not required

5.11 Test Plan

A. Introduction and Plan of Approach

Provide a brief description for the following areas:

- Project overview:
Dimension Fantasia is a mobile RPG game in which the user can enter multiple dungeons and fight enemies. The user can embark on journeys and combat different types of enemies, which as a result creates a need for a sophisticated battle system and dungeon system. When the user launches the game he/she will be presented with the main menu, and consists of the select level option, settings option, equipment option, shop and quit game option. They are very simple menus, but are the beginning to the variety that this game offers, such as when you change the settings the user can mess with difficulty settings, which jumps into input sensitivity and an assist mode the game offers. On the other hand when the user decides to select a level, the user will be able to select a dungeon to embark on, which leads to the battle system. The battle system consists of a prompt to the user that asks the user to make a decision, which is attack, block or run away. Based on the decision the player will execute he/she's decision then it will be the enemy's turn to attack or block. Underneath the enemy functionality the enemy will need to create its own attack and blocking patterns so the enemy can become unpredictable. Therefore, there are a lot of parts to the actual game play that will make playing the game fun for the user and make the user motivated to keep playing.
- Components of the Test Plan:
 - 1) Main Menu Screen: Making sure the main menu displays all basic and necessary options
 - 2) Settings Page: Ensuring the settings page properly adjusts the game to the user's liking
 - 3) Battle System: Player decisions against the enemy should properly be executed and should properly adjust enemy Health/XP. Enemy decisions should also properly adjust player Health/XP.
 - 4) Level Select Page: Properly displaying the different levels and displaying the level which the user wants to play

- Assumptions & Anomalies
 - 1) Testing will be conducted in implementation timeframes by implementers and locations.
 - 2) There will be a dedicated test environment accessible to the test team for conducting testing.
 - 3) The necessary testing tool and access to the tool are available and will be provided to the test team.
 - 4) All the requirements including business requirements, system requirements will be completed and subject to change
 - 5) Team won't be required to perform maintenance / evolution after the project.

B. Test Cases: "MAIN MENU"

Project Name: Dimension Fantasia

Test Case Name: Main Menu

Test Case Id: CSE3310/Fall2020/Team9/ Main Menu

Test Case No.	Test Case Description	Expected results	Outcome Pass, Fail, Other (comments)
TC1	Game is launched	System should display the Main Menu (Levels, Settings Button, Equipment Settings Button, Shop button and Exit Button).	Pass
TC2	Levels can be chosen based on completion or progress. Designated Level is pressed by the user	System should display selected level(Number of Floors, Difficulty and Battle)	Pass
TC3	Settings Button is pressed by the user	System should bring up the settings page that displays all the settings the user can change.	Pass
TC4	Equipment Settings Button is pressed by the user	System should display weapon selection, equipment selection and level up chances.	Pass
TC5	Shop button in pressed by the user	System should display a Shop menu(accessories, items).	Pass
TC6	Exit Game Button is pressed by the user	System should successfully quit the game for the user.	Pass

C. Test Cases: "Settings"

Project Name: Dimension Fantasia
Test Case Name: Settings
Test Case Id: CSE3310/Fall2020/Team9/ Setting

Test Case No.	Test Case Description	Expected results	Outcome Pass, Fail, Other (comments)
TC1	Settings button is clicked from the main menu. Settings menu is launched	System should display the settings menu with an icon/slider for input sensitivity and a toggle for assist mode.	Pass
TC2	Input sensitivity is clicked within the settings menu.	System should display a slider labeled input sensitivity. Let you adjust how far or how fast the user needs to swipe for a swipe input. Moving the slider to the right will increase input sensitivity whereas moving it to the left decreases it.	Pass
TC3	Assist mode is off by default. Toggle switch is pressed by the user to turn on assist mode.	Allows the user to bypass the timing requirements for the offensive battle phase. Avoids gestures for attacks on the next game.	Pass
TC4	Back button is pressed.	System goes back to the main menu	Pass

D. Test Cases: "Battle System "

Project Name: Dimension Fantasia
Test Case Name: Battle System
Test Case Id: CSE3310/Fall2020/Team9/ Battle_System

Test Case No.	Test Case Description	Expected results	Outcome Pass, Fail, Other (comments)
TC1	(From Main Selection) Player push the attack button	Buttons based on the enemies will appear based on how many enemies are in battle (2 enemies in battle mean two enemy buttons appear) and a back button appears to go back to the main selection.	Pass
TC1.1	Player push the enemy button	A menu appears with the choice of using special attacks based on what was bought in the shop. As well as a back button to go back to the previous selection.	Pass
TC1.2	Player pushes the attack button	For the choice of attack (any of the special attacks or a standard attack), the player will then move to the previously selected enemy and perform the attack action.	Pass (For both standard attack and all possible attacking accessories)
TC1.3	Player is attacking enemy	With previous button presses done, the hero will now attack the enemy with the script attached to his current weapon (there are two different weapons, thus two different scripts). Afterwards the player will return back to its initial position.	Pass (Both weapons)
TC2	(From Main Selection) Player push the defense button	The player goes into a perfect block state that forgoes the player's attacking turn in exchange for taking half damage;	Pass

TC3	(From Main Selection) Player push the item button	The list of items that the player has purchased in the shop appears	Pass
TC3.1	Player pushes on the item from the item menu	The player (depending on the effect of each of the 7 items), will do the respective item's action (healing HP, AP, or healing a status ailment).	Pass (All 7 items work)
TC4	(From Main Selection) Player pushes the exit button	A prompt for the player to quit with a confirmation yes/no set of buttons appear.	Pass
TC4.1	Player pushes the confirm button	The player will return to the level selection screen with any gained experience points or money obtained from the fight.	Pass
TC5	From each of the previous menus, the player pushes the back button.	The back button will undo the player's menu choice to the previous option. (X.1 or X.2 will go back to the previous test case for example).	Pass (Back button works in all instances of its existence).
TC6	The enemy performs an attack action.	After any of the previous test cases (1 - 3) the enemies, one at a time, will attack the player using their individual script of their type.	Pass (Every unique enemy type has a working script.)
TC7	The player can perform defensive actions.	During an enemy attack action, the player can interact with the screen to alter the damage taken during battle.	Pass
TC7.1	The player taps the screen to perfect block	The player goes into a defensive stance. If timed right, damage will be halved. This is the same effect as simply pressing the defend button.	Pass
TC7.2	The player holds their finger on the screen for a standard block	After a perfect block, when the player holds onto the screen, the hero is still in a defensive stance. Damage will be reduced by 1, unless the attack itself already did 1 damage.	Pass

TC7.3	The user swipes the screen	The player will enter a dodge state, which will display “MISS” and the player will not take any damage.	Pass
TC7.4	The user releases their finger from the screen	The player will enter a parry state which will reflect some of the damage the enemy would have dealt to the player back at the enemy.	Pass

E. Test Cases: "Select Level"

Project Name: Dimension Fantasia
Test Case Name: Select Level
Test Case Id: CSE3310/Fall2020/Team9/ Select_Level

Test Case No.	Test Case Description	Expected results	Outcome Pass, Fail, Other (comments)
TC1	Player confirms level	System will place the user in selected level	Pass
TC2	System level button display	User will only be able to select levels that have been completed or is in the progress of completing	Pass
TC3	Player has not reached a certain level	System should not allow the user to play the level if the player has not reached that level.	Pass
TC4	Player selects back	The system should go from the select level screen to main menu screen	Pass

F. Test Cases: "Character Customization"

Project Name: Dimension Fantasia
Test Case Name: Character Customization
Test Case Id: CSE3310/Fall2020/Team9/ Character_Customization

Test Case No.	Test Case Description	Expected results	Outcome Pass, Fail, Other (comments)
TC1	Game remembers purchased accessories.	From accessories bought in the shop (there are 8 unique accessories), the game will remember what was purchased in what order.	Pass
TC2	Player can equip accessories	From purchased accessories, the game will keep track of what is equipped and if the player has enough points to equip it.	Pass
TC3	Equipped accessories effect stats	When the player equips any stat modifying accessory, the appropriate stat will be modified.	Pass (For all 5 stat modifying accessories)
TC4	Equipped accessories will give player access to attacks	When the player equips an attacking accessory, the player will be granted access to that attack in battle.	Pass (For all 3 attacking accessories)

6. Assumptions and Constraints

6.1 ASSUMPTIONS

The following is a list of assumptions:

- Assume all users are over the age of 18.
- There is an ample market to sell our app.
- Team members will have no vacations to take.
- Team won't be required to perform maintenance / evolution after the project.
- We can assume no wages have to be paid.
- There are no other company expenses that must be accounted for.

6.2 CONSTRAINTS

The following is a list of constraints:

- Team lacks android experience and unity
- Team has very limited time
- Learning new language
- Scheduling conflict
- Physical Meetups

6.3 OUT OF SCOPE MATERIAL

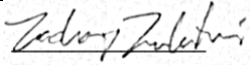
The following is a list of "out of scope" material:

- Team won't be responsible to perform maintenance / evolution after the project is done.

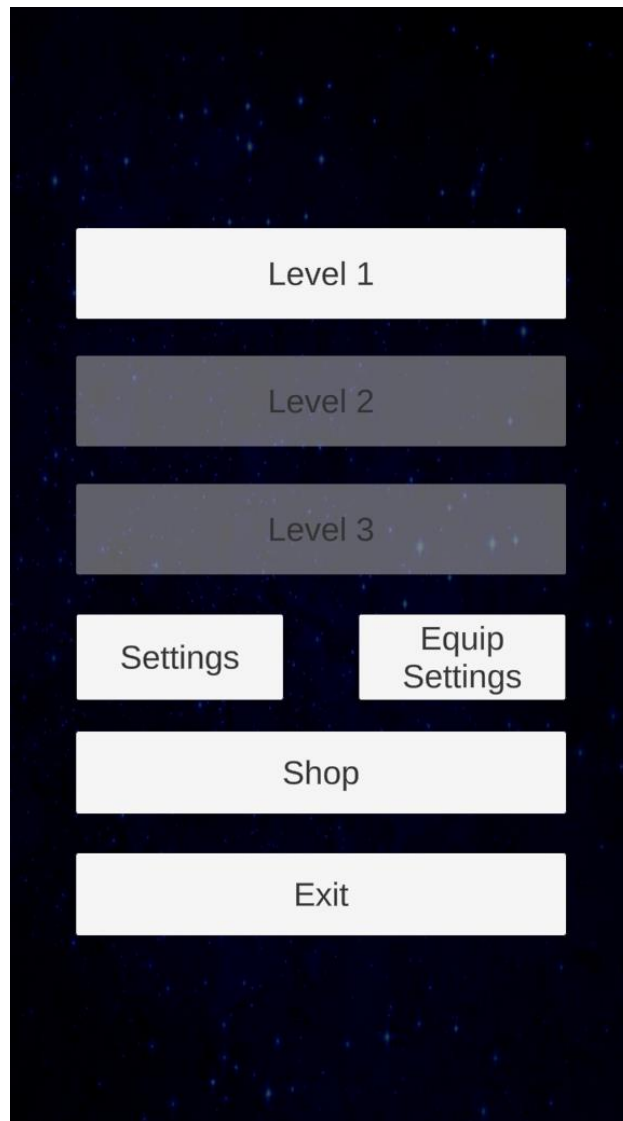
7. Delivery and Schedule

Task/Milestone Description	Anticipated Start Date	Anticipated End Date	Status	Comments
Project topic proposal	9/1/20	9/13/20	Completed	
UML Diagram	9/24/2020	10/01/2020	Completed	
SRA document (Includes project objectives, Requirements and UML diagrams)	10/12/2020	10/22/2020	Completed	Deliverable will be the SRA document. All stakeholders agree on the content of the SRA by signing in section 8.
Test plan doc + peer review	10/24/2020	11/11/2020	Completed	
Final product	11/20/2020	11/30/2020	Completed	

8. Stakeholder Approval Form

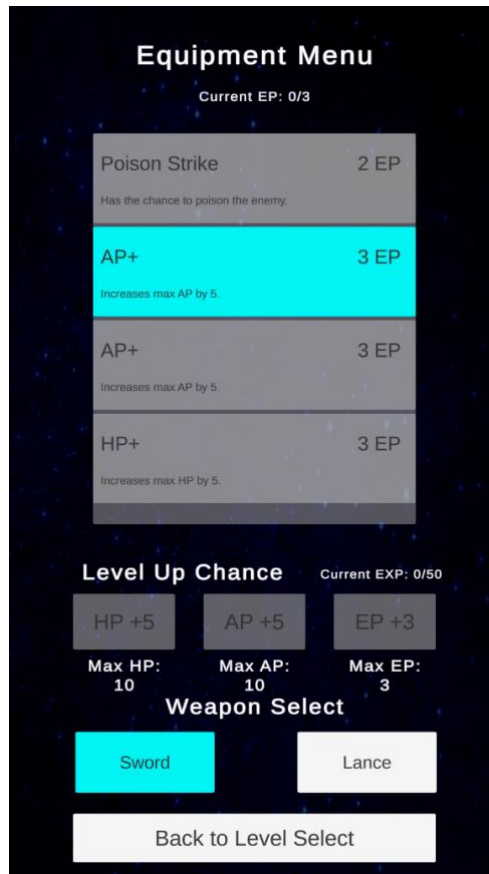
Stakeholder Name	Stakeholder Role	Stakeholder Comments	Stakeholder Approval Signature and Date
Rodrigo Augusto	Development Mgr		
Tianhao Li	Project Assistant		
Ganesh Kandel	Developer		Ganesh Kandel 12/1/2020
Jorge Estrada	Developer		Jorge Estrada 12/1/2020
Bevan Philip	Developer		Bevan Philip 12/1/2020
Zachary Trumbaturi	Developer		 12/1/2020
Ajaya Pyakurel	Developer		Ajaya Pyakurel 12/1/2020

MAIN MENU



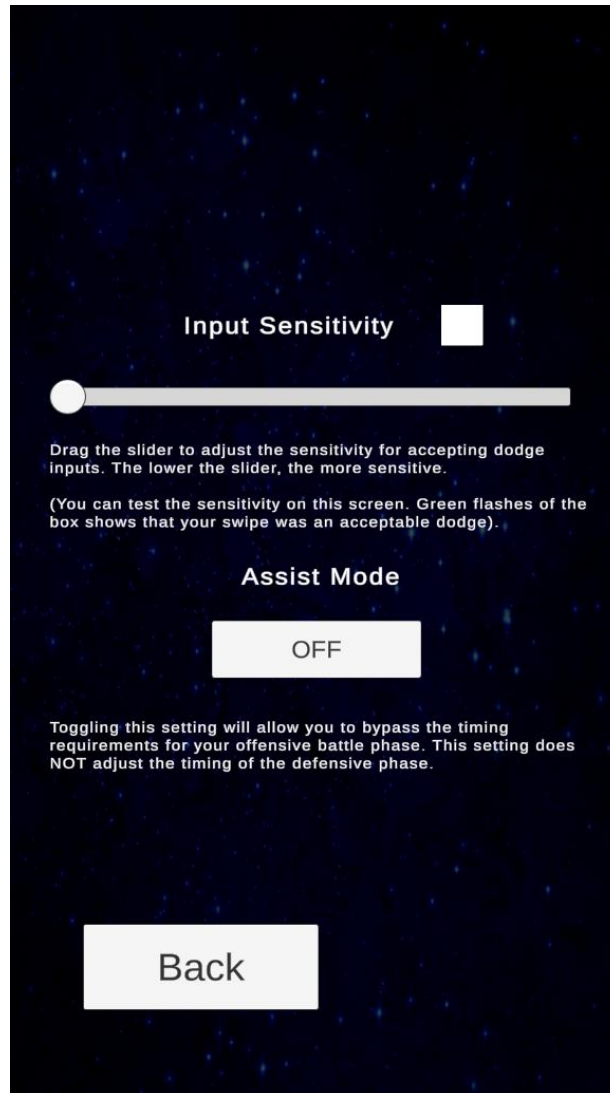
- Press “Level 1” to play the first level of the game. (Greyed out levels are inaccessible until previous level is completed)
- Press “Settings” to access the game settings
- Press “Equip Settings” to access character equipment settings and change the equipment your character has
- Press “Shop” to access the item shop
- Press “Exit” to quit the game

EQUIPMENT SETTINGS



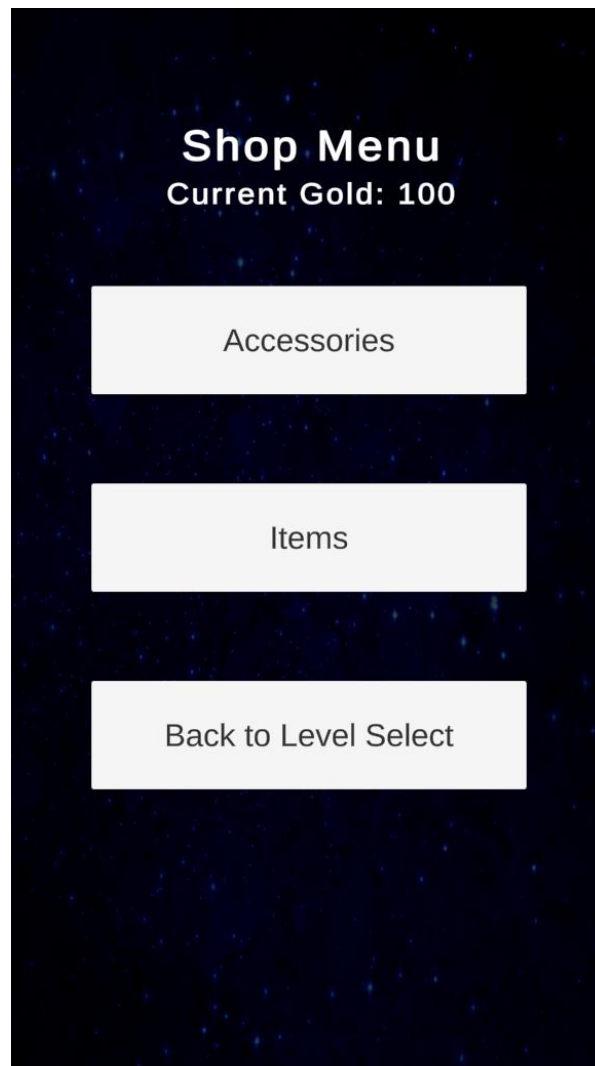
- Current player EP (Equipment Points) are displayed at the top, which allow the player to equip certain upgrades and abilities.
- Scroll through the window in the middle to view all of your available accessories you can equip.
- Press the accessory to equip it.
- accessories are greyed out if you can no longer equip them due to not having enough EP.
- Abilities that are in blue are abilities that have been equipped.
- Current EXP is displayed, which allows the player to make permanent HP, AP and EP upgrades.
- Level up chance displays permanent upgrades you can make, they are greyed out due to insufficient EXP.
- Press an upgrade in level up chance and your character will be upgraded (ex: If player selects HP +5 , then the character will have a permanent increase of +5 HP).
- Press "Sword" to equip a sword.
- Press "Lance" to equip a lance.
- Press " Back to Level Select" to go back to the Main Menu.

SETTINGS



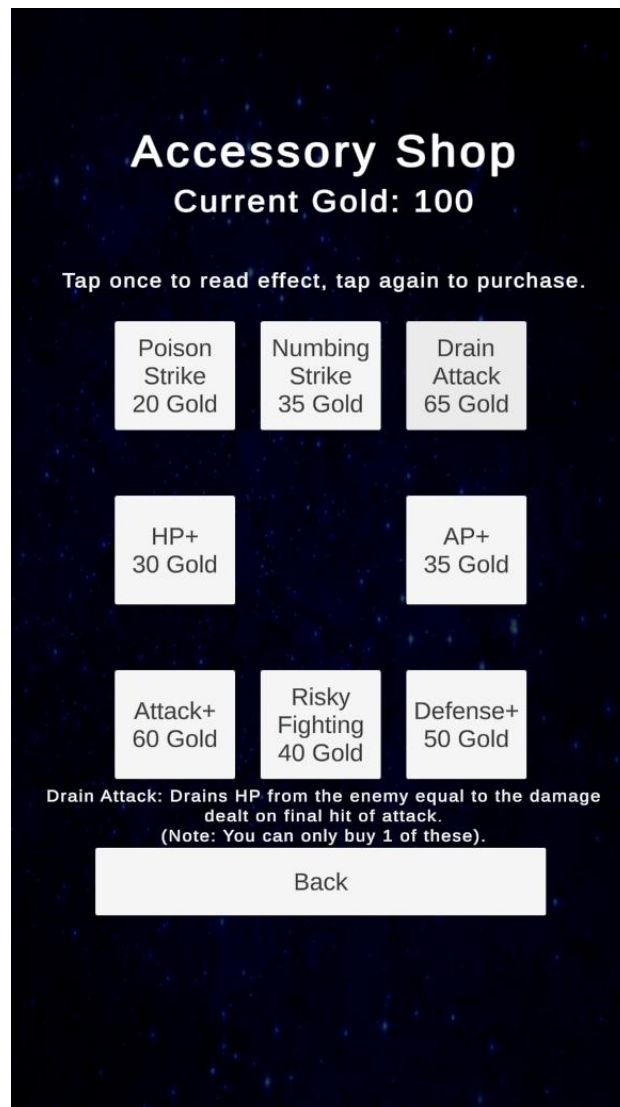
- Slide the bar to adjust input sensitivity, which will adjust the sensitivity at which dodge inputs are accepted. (Effects of sensitivity are given by the settings page)
- Press the "Assist Mode" box to toggle whether or not it will take effect during gameplay.
- Press "Back" to go back to the Main Menu screen.

SHOP



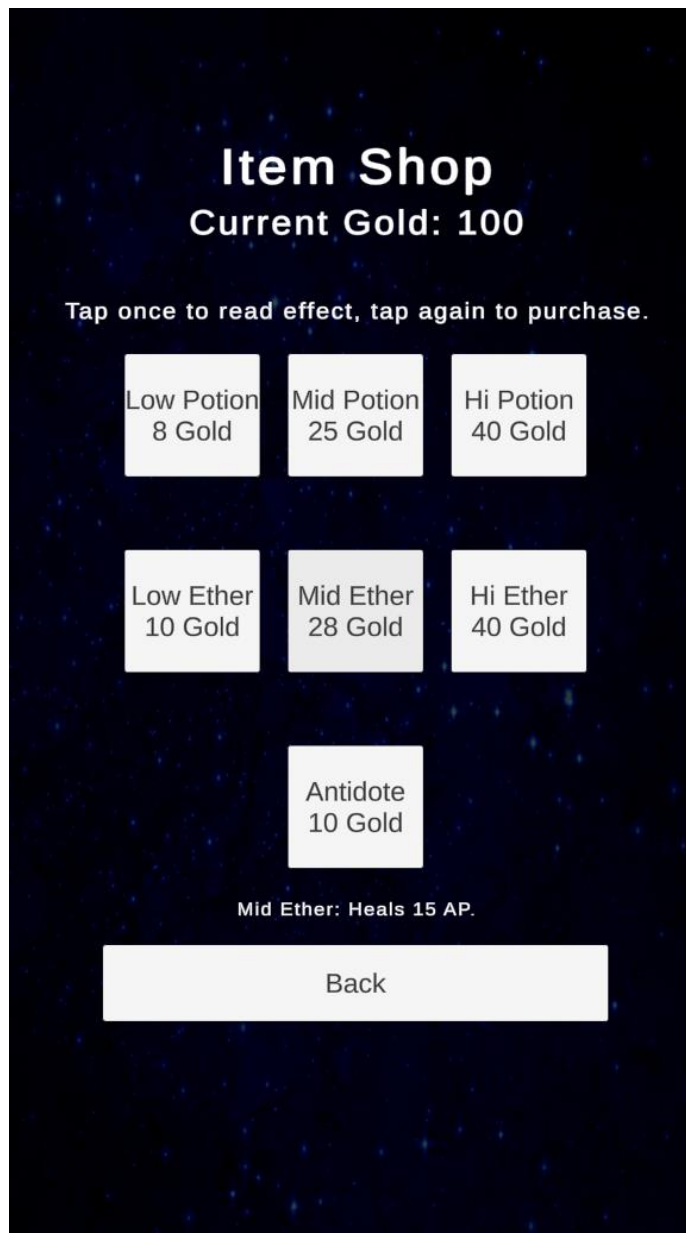
- Current Gold displayed at the top of the shop.
- Press “Accessories” to go to the accessories shop.
- Press “Items” to access the item shop.
- Press “Back to Level Select” to go back to the main menu.

ACCESSORY SHOP



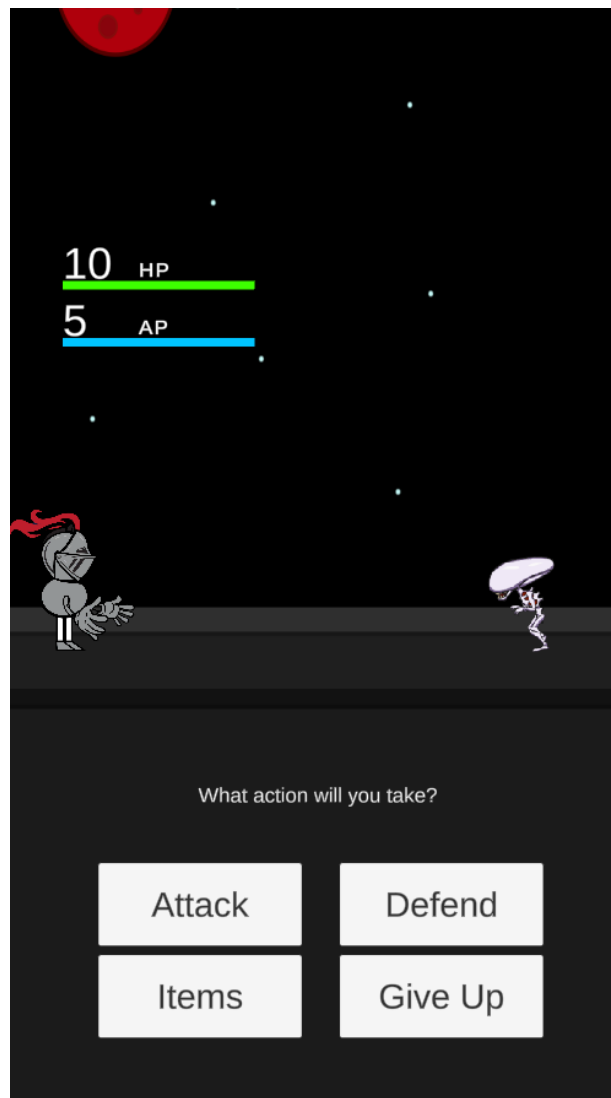
- Current gold is displayed to inform the user of how much gold they currently have.
- Press any accessory once to read details about the accessory the player is considering to purchase.
- Press the accessory twice to fully purchase the accessory.
- Press “Back” to go back to the shop menu.

ITEM SHOP



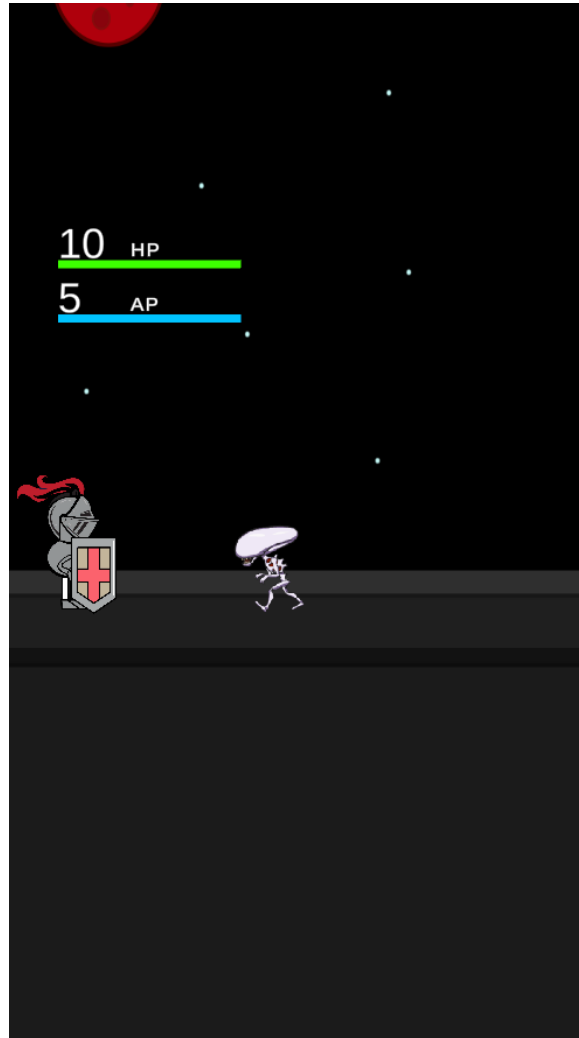
- Current gold is displayed to inform the user of how much gold they have available.
- Press any item once to read the effects of the item.
- Press the same item again to purchase the item.
- Once an item is purchased you can use the item as a consumable in battle.
- Press back to go back to the Shop menu

BATTLE SYSTEM – OFFENSE



- The hero will have HP and AP values available during battle.
- If the hero loses all HP during battle, they will lose all their progress and return to the level select screen.
- The player may choose to “Attack” an active enemy, go into a perfect block state using “Defend”, or use “Item” to use a purchased item from the shop. You may also “Give Up” if you need to end the fight early.
- The player, if they equipped any attack-based accessories, may use the appropriate amount of AP to use the equipped attack. (Note: You may only inflict one status ailment at a time. Some enemies or bosses are more resistant to status ailments than others or even be immune).
- Once an attack is finalized, the player will participate in a timing minigame based on their weapon choice. Follow the on-screen instructions to increase damage dealt.
- A note on status ailments: Poison will deal 1 point of damage after an action is performed. Paralysis has a chance of skipping the enemy’s attack turn as long as the status ailment is applied.

BATTLE SYSTEM – DEFENSE



- After the hero's attacking phase, the enemy or enemies will have their turn to attack.
- During this phase, the player may tap or swipe the screen to potentially mitigate the damage dealt to them during combat.
- When tapping the screen initially, the player will go into a perfect block state which halves the damage taken from the enemy (or if the enemy deals less than 3 damage, no damage will be applied).
- After the perfect block state, the player will still hold up their shield and will reduce damage by 1. The enemy however will deal a minimum of 1 damage even during this state unless you have stat modifying equipment.
- If the player swipes the screen, the player will dodge the enemy attack to take no damage. This will be notified by a "MISS" message above the player's head.
- Without swiping the screen, if the player lets go of the screen instead, there is a small window where the player will parry the enemy's attack. This is a very tough maneuver to pull off, however if timed correctly all the damage the player would have taken will be dealt to the enemy instead. A successful parry will have an "!" next to the damage number.

10. Source Code

The following code is an excerpt from the battle system from Dimension Fantasia

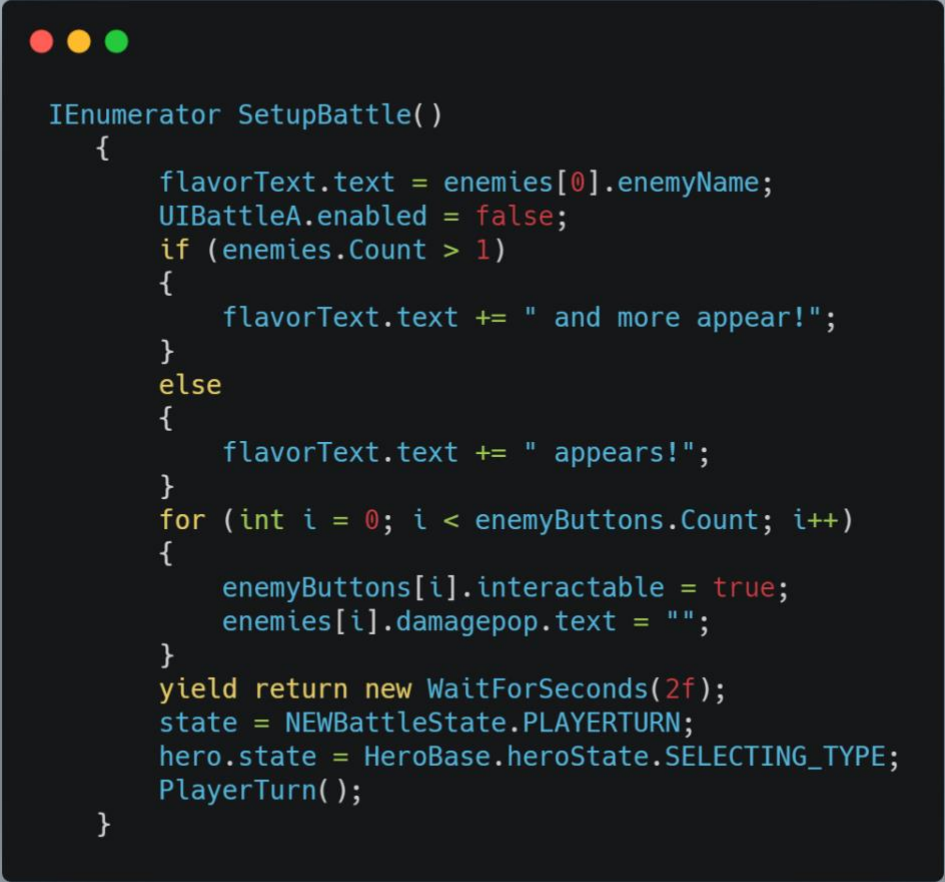
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public enum NEWBattleState { START, PLAYERTURN, PLAYERITEM, ENEMYTURN, WON, LOST }
public class NewBattleSystem : MonoBehaviour
{
    public HeroBase hero;
    Vector3 heroHome;
    bool defenseButton;
    public int nextScene;
    public int enemyTarget;
    public int enemyCount = -1;
    public int itemChoice = -1;
    public int specialAttackType = -1;
    public List<EnemyBase> enemies;
    public NEWBattleState state;
    public GameObject selectTypeButtons;
    public List<Button> enemyButtons;
    public GameObject enemySelectButtons;
    public Button[] itemButtons = new Button[5];
    public GameObject itemSelectButtons;
    public Button[] specialAttackButtons = new Button[3];
    public GameObject specialAttackButtonsGroup;
    public List<Button> endBattleButton;
    public GameObject battleWonButtons;
    public List<Button> potentialExitButtons;
    public GameObject potentialExitGroup;
    public int deadEnemyCount;
    public TextMeshProUGUI flavorText;
    public Canvas UIBattleA;
    public Slider timingSlider;
    public Image timingFill;
    public TextMeshProUGUI timingText;
    public int pBlockDuration, dodgeDuration, parryDuration, cooldownDuration;
    int pBlockCounter = 0, dodgeCounter = 0, parryCounter = 0, cooldownCounter = 0;
    int defenseType;
    bool canTouch;
    private Vector2 Speed;
```

```

void Start()
{
    hero.equippedWeapon = GlobalControl.Instance.heroWeapon;
    hero.currentHP = GlobalControl.Instance.heroCurrentHP;
    hero.maxHP = GlobalControl.Instance.heroMaxHP;
    hero.currentAP = GlobalControl.Instance.heroCurrentAP;
    hero.maxAP = GlobalControl.Instance.heroMaxAP;
    hero.damagepop.text = "";
    hero.items = new ItemBase[5];
    hero.ailments = new AilmentEquipment[3];
    for (int i = 0; i < 5; i++) {
        itemButtons[i].interactable = false;
        itemButtons[i].GetComponentInChildren<TextMeshProUGUI>().text = "";
    }
    for (int i = 0; i < GlobalControl.Instance.items.Count; i++) {
        hero.items[i] = Instantiate(GlobalControl.Instance.items[i]);
        itemButtons[i].GetComponentInChildren<TextMeshProUGUI>().text =
hero.items[i].itemName;
        itemButtons[i].interactable = true;
    }
    for(int i = 0; i < enemies.Count; i++) {
        enemyButtons[i].GetComponentInChildren<TextMeshProUGUI>().text =
enemies[i].name;
    }
    for(int i = 0; i < 3; i++) {
        specialAttackButtons[i].interactable = false;
        specialAttackButtons[i].GetComponentInChildren<TextMeshProUGUI>().text = "";
    }
    for (int i = 0; i < GlobalControl.Instance.ailments.Count; i++) {
        hero.ailments[i] = Instantiate(GlobalControl.Instance.ailments[i]);
        specialAttackButtons[i].GetComponentInChildren<TextMeshProUGUI>().text =
hero.ailments[i].equipmentName + "\n" + hero.ailments[i].APCost + " AP";
        specialAttackButtons[i].interactable = true;
    }
    foreach (Transform child in GlobalControl.Instance.transform) {
        if (child.tag == "Item") {
            GameObject.Destroy(child.gameObject);
        }
    }
    hero.HPBar.value = hero.currentHP;
    timingSlider.maxValue = hero.equippedWeapon.minigameTimer;
    timingSlider.value = 0;
    hero.equippedWeapon.startupCounter = 0;
    hero.equippedWeapon.OKToPressCounter = 0;
    hero.equippedWeapon.miniGameState = Weapon.attackStates.FREE;
    timingFill.color = new Color(1, 0, 0, 1);
    defenseButton = false;
    state = NEWBattleState.START;
    potentialExitGroup.SetActive(false);
    heroHome = hero.GetPosition();
    StartCoroutine(SetupBattle());
}

```



```
IEnumerator SetupBattle()  
{  
    flavorText.text = enemies[0].enemyName;  
    UIBattleA.enabled = false;  
    if (enemies.Count > 1)  
    {  
        flavorText.text += " and more appear!";  
    }  
    else  
    {  
        flavorText.text += " appears!";  
    }  
    for (int i = 0; i < enemyButtons.Count; i++)  
    {  
        enemyButtons[i].interactable = true;  
        enemies[i].damagepop.text = "";  
    }  
    yield return new WaitForSeconds(2f);  
    state = NEWBattleState.PLAYERTURN;  
    hero.state = HeroBase.heroState.SELECTING_TYPE;  
    PlayerTurn();  
}
```

```

IEnumerator EnemyTurn()
{
    yield return new WaitForSeconds(0.5f);
    flavorText.text = "";
    hero.damagepop.text = "";
    enemyCount = -1;
    for (int i = 0; i < enemies.Count; i++) {
        Debug.Log(enemies[i].name + " attack pattern start");
        if (enemies[i].state != EnemyBase.enemyState.DEAD)
        {
            if (enemies[i].statusAilment == EnemyBase.ailmentState.Paralyzed &&
UnityEngine.Random.Range(0,100) <= 30) {
                flavorText.text = enemies[i].name + " is paralyzed and can't attack
this turn.";
                yield return new WaitForSeconds(1.5f);
            }
            else {
                StartCoroutine(enemies[i].Movement(hero, flavorText));
                yield return new WaitForSeconds(1.5f);
                EnemyBase.moveState.FINISHED);
                hero.damagepop.text = "";
                if(enemies[i].statusAilment == EnemyBase.ailmentState.Poisoned) {
                    enemies[i].TakeDamage(1);
                    yield return new WaitForSeconds(0.5f);
                    enemies[i].damagepop.text = "";
                }
                if (enemies[i].state != EnemyBase.enemyState.DEAD) {
                    enemies[i].statusAilment = EnemyBase.ailmentState.NORMAL;
                    enemies[i].state = EnemyBase.enemyState.IDLE;
                    yield return new WaitForSeconds(0.7f);
                }
                if (enemies[i].state == EnemyBase.enemyState.DEAD) {
                    Debug.Log(enemies[i].name + " is Dead");
                    yield return new WaitForSeconds(0.5f);
                    enemyButtons[i].interactable = false;
                    deadEnemyCount++;
                    continue;
                }
                if (hero.currentHP == 0) {
                    state = NEWBattleState.LOST;
                    break;
                }
                enemies[i].movingState = EnemyBase.moveState.STATIONARY;
            }
            flavorText.text = "";
        }
    }
    deadEnemyCount = 0;
    enemyCount = -1;
    Debug.Log("Final enemy loop");
    for (int i = 0; i < enemies.Count; i++)
    {
        if (enemies[i].state == EnemyBase.enemyState.DEAD)
        {
            deadEnemyCount += 1;
        }
    }
    if (hero.currentHP == 0)
    {
        state = NEWBattleState.LOST;
        EndBattle();
    }
    else if (deadEnemyCount == enemies.Count)
    {
        state = NEWBattleState.WON;
        EndBattle();
    }
    else
    {
        defenseButton = false;
        hero.defending = false;
        PlayerTurn();
    }
}

```

```

void SaveDataForNextBattle() {
    int expGained = 0, goldGained = 0;
    GlobalControl.Instance.heroMaxHP = hero.maxHP;
    GlobalControl.Instance.heroCurrentHP = hero.currentHP;
    GlobalControl.Instance.heroMaxAP = hero.maxAP;
    GlobalControl.Instance.heroCurrentAP = hero.currentAP;
    for(int i = 0; i < enemies.Count; i++) {
        if (enemies[i].state == EnemyBase.enemyState.DEAD) {
            GlobalControl.Instance.heroCurrentEXP += enemies[i].expReward;
            expGained += enemies[i].expReward;
            GlobalControl.Instance.heroCurrentMoney += enemies[i].goldReward;
            goldGained += enemies[i].goldReward;
        }
    }
    if (state != NEWBattleState.START) {
        flavorText.text += "\nYou gained " + expGained + " EXP & " + goldGained + "
gold.";
    }
    GlobalControl.Instance.items.Clear();
    for (int i = 0; i < 5; i++) {
        if(hero.items[i] != null) {
            Debug.Log("Loop" + i);
            hero.items[i].transform.parent = GlobalControl.Instance.transform;
            GlobalControl.Instance.items.Add(hero.items[i]);
        }
    }
}

void EndBattle()
{
    if (state == NEWBattleState.WON)
    {
        hero.state = HeroBase.heroState.WON;
        flavorText.text = "You won the battle!";
    }
    else if (state == NEWBattleState.LOST)
    {
        hero.damagepop.text = "";
        flavorText.text = "You were defeated...";
    }
    SaveDataForNextBattle();
}

void PlayerTurn()
{
    state = NEWBattleState.PLAYERTURN;
    flavorText.text = "What action will you take?";
    hero.state = HeroBase.heroState.SELECTING_TYPE;
}

```



```

IEnumerator PlayerAttack(int n, int atkType) {;

    hero.equippedWeapon.miniGameState = Weapon.attackStates.FREE;
    yield return new WaitForSeconds(15 / 60f);
    StartCoroutine(hero.equippedWeapon.Use(enemies[n]));
    yield return new WaitUntil(() => hero.equippedWeapon.miniGameState ==
Weapon.attackStates.FREE);

    if (atkType != -1 && enemies[n].state != EnemyBase.enemyState.DEAD) {
        hero.ailments[atkType].Modify(hero, enemies[n]);
    }
    if (enemies[n].state == EnemyBase.enemyState.DEAD) {
        enemies[n].statusAilment = EnemyBase.ailmentState.NORMAL;
        deadEnemyCount++;
    }
    if (enemies[n].statusAilment != EnemyBase.ailmentState.NORMAL) {
        flavorText.text = enemies[n].name + " has been " +
enemies[n].statusAilment.ToString();
    }
    yield return new WaitForSeconds(1);
    enemies[n].damagepop.text = "";
    canTouch = true;
    hero.state = HeroBase.heroState.RETREATING;
    yield return new WaitUntil(() => hero.state != HeroBase.heroState.RETREATING);
    if (enemies[n].currentHP <= 0) {
        enemies[n].state = EnemyBase.enemyState.DEAD;
        enemyButtons[n].interactable = false;
        deadEnemyCount++;
    }
}
}

```

```

IEnumerator PlayerItem(int n) {
    hero.items[n].state = ItemBase.itemState.FREE;
    yield return new WaitForSeconds(15 / 60f);
    StartCoroutine(hero.items[n].Use(hero, enemies));
    yield return new WaitUntil(() => hero.items[n].state ==
ItemBase.itemState.FINISHED);
    ChangeItemCount(hero.items[n]);
    hero.items[n] = null;
    itemButtons[n].GetComponentInChildren<TextMeshProUGUI>().text = "";
    yield return new WaitForSeconds(1);
    deadEnemyCount = 0;
    for (int i = 0; i < enemies.Count; i++) {
        if (enemies[i].state == EnemyBase.enemyState.DEAD) {
            deadEnemyCount++;
        }
    }
    Destroy(hero.items[n]);
    canTouch = true;
    itemChoice = -1;
    hero.state = HeroBase.heroState.RETREATING;
    yield return new WaitUntil(() => hero.state != HeroBase.heroState.RETREATING);
    if (enemies[n].currentHP <= 0) {
        enemies[n].state = EnemyBase.enemyState.DEAD;
        enemyButtons[n].interactable = false;
        deadEnemyCount++;
    }
}
}

```

```

void PoisonCheck() {
    if (hero.statusAilments == HeroBase.heroAilments.POISONED) {
        hero.TakeDamage(-255);
    }
}

void PlayerItemScript() {
    switch (hero.state) {

        case HeroBase.heroState.MOVING:
            hero.transform.position = Vector3.MoveTowards(hero.GetPosition(), (new
Vector3(-0.5f, -0.21f, 0)), .05f);

            if (Vector2.Distance(hero.GetPosition(), (new Vector3(-0.5f, -0.21f, 0)))
== 0) {
                hero.state = HeroBase.heroState.USINGITEM;
            }
            break;
        case HeroBase.heroState.USINGITEM:
            StartCoroutine(PlayerItem(itemChoice));
            hero.state = HeroBase.heroState.BUSY;
            break;
        case HeroBase.heroState.RETREATING:

            hero.transform.position = Vector3.MoveTowards(hero.GetPosition(),
heroHome, .07f);

            if (Vector2.Distance(hero.GetPosition(), heroHome) == 0) {
                hero.state = HeroBase.heroState.IDLE;
                timingSlider.value = 0;
                flavorText.text = "";
                UIBattleA.enabled = false;
                timingText.text = "";
                hero.equippedWeapon.startupCounter = 0;
                PoisonCheck();
                if (hero.currentHP == 0) {
                    state = NEWBattleState.LOST;
                    EndBattle();
                }
                else {
                    state = NEWBattleState.ENEMYTURN;
                    StartCoroutine(EnemyTurn());
                }
            }
            break;
    }
}

```



```

void PlayerMovementScript() {
    switch (hero.state) {

        case HeroBase.heroState.MOVING:
            hero.transform.position = Vector3.MoveTowards(hero.GetPosition(),
(enemies[enemyTarget].playerTarget), .061f);

            if (Vector2.Distance(hero.GetPosition(),
enemies[enemyTarget].playerTarget) == 0) {
                hero.state = HeroBase.heroState.ATTACKING;
            }
            break;

        case HeroBase.heroState.ATTACKING:
            if (specialAttackType != -1) {
                hero.currentAP -= hero.ailments[specialAttackType].APCost;
            }
            StartCoroutine(PlayerAttack(enemyTarget, specialAttackType));
            hero.state = HeroBase.heroState.BUSY;
            break;

        case HeroBase.heroState.RETREATING:

            hero.transform.position = Vector3.MoveTowards(hero.GetPosition(),
heroHome, .07f);

            if (Vector2.Distance(hero.GetPosition(), heroHome) == 0) {
                hero.state = HeroBase.heroState.IDLE;
                timingSlider.value = 0;
                UIBattleA.enabled = false;
                timingText.text = "";
                hero.equippedWeapon.startupCounter = 0;
                PoisonCheck();
                if (hero.currentHP == 0) {
                    state = NEWBattleState.LOST;
                    EndBattle();
                }
                else if (deadEnemyCount == enemies.Count) {
                    hero.state = HeroBase.heroState.WON;
                    state = NEWBattleState.WON;
                    EndBattle();
                }
                else {
                    state = NEWBattleState.ENEMYTURN;
                    StartCoroutine EnemyTurn();
                }
            }
            break;
    }
}

```

```

float Magnitude(Vector2 AA)
{
    float result = (AA.x * AA.x) + (AA.y * AA.y);
    return (float)Mathf.Sqrt(result);
}

void DefenseMechanics(){
    if (Input.touchCount > 0 && canTouch)
    {
        parryCounter = 0;
        Touch touch = Input.GetTouch(0);
        // Change the 'color' property of the 'Sprite Renderer'
        Speed = touch.deltaPosition / touch.deltaTime;
        if (touch.phase == TouchPhase.Began)
        {
            hero.state = HeroBase.heroState.PERFECT_BLOCK;
        }
        else if (touch.phase == TouchPhase.Ended)
        {
            pBlockCounter = 0;
            canTouch = false;
            if (Magnitude(Speed) >= GlobalControl.Instance.touchSensitivity) // THIS
CAN BE CHANGED ON TITLE MENU
            {
                hero.state = HeroBase.heroState.DODGE;
                defenseType = 1;
            }
            else
            {
                hero.state = HeroBase.heroState.PARRY;
                defenseType = 2;
            }
        }
        else if (touch.phase == TouchPhase.Stationary || Magnitude(Speed) <
GlobalControl.Instance.touchSensitivity)
        {
            if (pBlockCounter != pBlockDuration)
            {
                pBlockCounter++;
            }
            else
            {
                hero.state = HeroBase.heroState.BLOCK;
            }
        }
    }
    else if (defenseType > 0 && !canTouch)
    {
        switch (defenseType)
        {
            case 1:
                if (dodgeCounter != dodgeDuration)
                {
                    dodgeCounter++;
                }
                else
                {
                    hero.state = HeroBase.heroState.COOLDOWN;
                    defenseType = 0;
                    dodgeCounter = 0;
                    parryCounter = 0;
                }
                break;
            case 2:
                if (parryCounter != parryDuration)
                {
                    parryCounter++;
                }
                else
                {
                    hero.state = HeroBase.heroState.COOLDOWN;
                    defenseType = 0;
                    dodgeCounter = 0;
                    parryCounter = 0;
                }
                break;
        }
    }
    else
    {
        if (cooldownCounter != cooldownDuration)
        {
            cooldownCounter++;
        }
        else
        {
            canTouch = true;
            cooldownCounter = 0;
            hero.state = HeroBase.heroState.IDLE;
        }
    }
}

```

Appendix:

None