Platform Final Documentation

December 4, 2019

CP317 Software Engineering

Group 7
He, Xiaohu
Lowe, Garret
Meng, Kaifeng
Wang, Shibo
Zhang, Weijie

Index

1. Introduction

1.1 General Description

This document is about the project of CP317 Software Engineering in Fall 2019 term. Our group has 5 people: ***He, Xiaohu; Lowe, Garret; Meng, Kaifeng; Wang, shibo; Zhang Weijie***. We plan to design a 2D RPG game named ***Platform*** with the help of the game engine ***Godot*** for map and character's animation. The art works, game mechanism, economic system and UI will be developed by our group.

1.2 Stakeholders

To entertain the players, which is the main purpose for making a game. And for our group members, we are able to get familiar with game development procedures.

1.3 Similar Product Information

The interaction mode and mechanism of the game will similar to ***Zelda II: The Adventure of Link***, however, the camera view will be from upside like the League of Legends.

1.4 Basic Requirements

We are going to use a game engine which name is Godot. This engine uses a programming language named GDScript, which is a python-like language with similar grammas and logics. The operation system for constructing the game is Windows 10.

2. Product Description

2.1 General Description About the Product

Since we are going to develop our game with the help of Godot, which has already integrated with built-in functions for the movement control and part of basic UI design. We only need to implement the Mechanism, Skill, Character (Player) and Item class (abstract) to run the game. These classes as well as their relationship will be showed in this part.

2.2 Purpose

This document outlines the necessary features planned for the project and should offer rudimentary insight for implementation and should serve as a sort of

progress monitor for stakeholders during development.

Let this document act as a resource for our project client, Professor Abdul-Rahman Mawlood- Yunis.

2.3 Definition, Acronyms and Abbreviation

All definitions in this section will be surrounded by quotation marks when used in other sections of this document.

Action Bar – A visual element of the User Interface which displays they "skills" and "consumables" equipped on the user character and the respective "key-binds" required for them to be activated.

AI – Artificial Intelligence. Pertaining to an "entity" scripted to act dynamically on different situations.

Consumable – An item type which can be used by the user character to provide them a "status effect" or effect one of the user characters "stats." Once used it is removed from the user characters "inventory."

Entity – A character with "stats" and/or the ability of movement. plural: entities.

Hit Point – A specific "stat" belonging to certain "entities," including the user character and enemies, pertaining to the amount of damage an "entity" can withstand before death. plural: hit points.

Inventory – A basic shop system belonging to the user character containing all items and equipment obtained by the user character.

Key-Bind – The specific key on a keyboard which must be pressed to perform an action.

Mana – A regenerating "stat" belonging to certain "entities," including the user character, pertaining to the amount of times "skills" can be used.

Skill – A procedure which may be activated anytime during gameplay at the cost of "mana." When activated, there are multiple effect which could take place depending on the exact skill, including: Increasing the user characters "stats", apply a "status effect" on the user character or an enemy, or perform a scripted action to move the user character or deal damage to an enemy. Stat - A numeric attribute belonging to a specific "entity" including movement speed, attack damage, "hit points", "mana", etc. plural: stats.

Status Effect – A special attribute belonging to specific "entities" (typically, not by default). Alters the scripted behavior of an "entity."

Character – the moveable part of the game, including monster, NPC and player

HP – The health point of character, implies whether character is alive or not

MP – mana point, skill may consume this to be activated

HUD – the character information that will always show on the screen

UI – the interaction parts that where the user can communicate with system

2.4 Product Characteristics and Basic Functions

In our game, players are able to:

a) Create their own character at the start of the game.

b) View their character's status.

c) Control their character. For example, move in multi directions, attack and defense.

d) Forging or collecting equipment such as weapons and consumables.

e) Have health (HP) and magic (MP)

f) Use different skills for battle or other goals.

g) Use trading system to get consumables or other stuffs from in-game shop or from NPCs

3. Specific Requirements

3.1 Function Requirements

See 2.4 and 4.5 for details

3.2 System Requirements

Designed to run under the environment of Windows 10. However, since the Godot engine is independent from platform, the only requirements is the hardware and software support certain version of OpenGL (see 3.3 for details). Therefore, it will also run under Linux/Unix, MacOS, Android, IOS system.

3.3 Hardware Requirements

Godot 3.0.

OpenGL ES 3.0 or OpenGL 3.0

Memory size at least 2 GB

Hard drive has more than 1 GB available

3.4 Non-functional Requirements

The product should be easy to maintain

Can add different characters (monster, player, NPC) easily

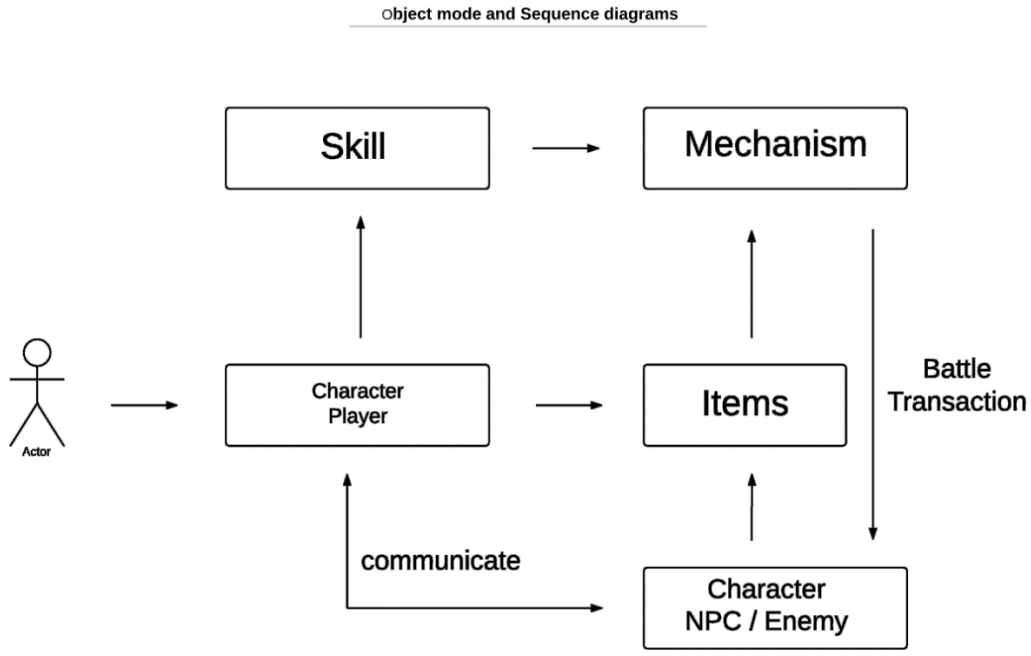
Can create new level easy and quick

Can add more skills easily

4. Product Details and Prototype

4.1 Object Mode and Sequence Diagrams

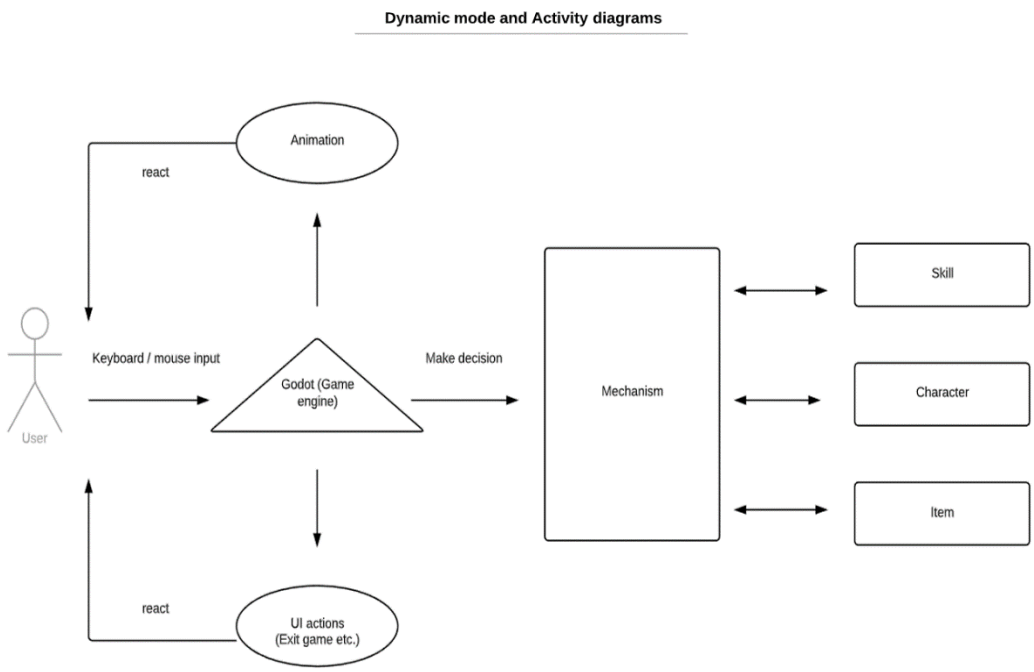
The below diagram shows how the player involves with the game and overall relationship between system components.



- Player will enter the game as a Character class instance which will have distinct skills and HP, MP and other attributes (see details on class diagram)
- Characters will able to communicate (story through text)
- Characters will able to battle (between enemy and player) or transact (between NPC and player) through the Mechanism class, which contains battle mechanism and shop setting.

4.2 Dynamic mode Diagram

This diagram is about the users' inputs and how system react.

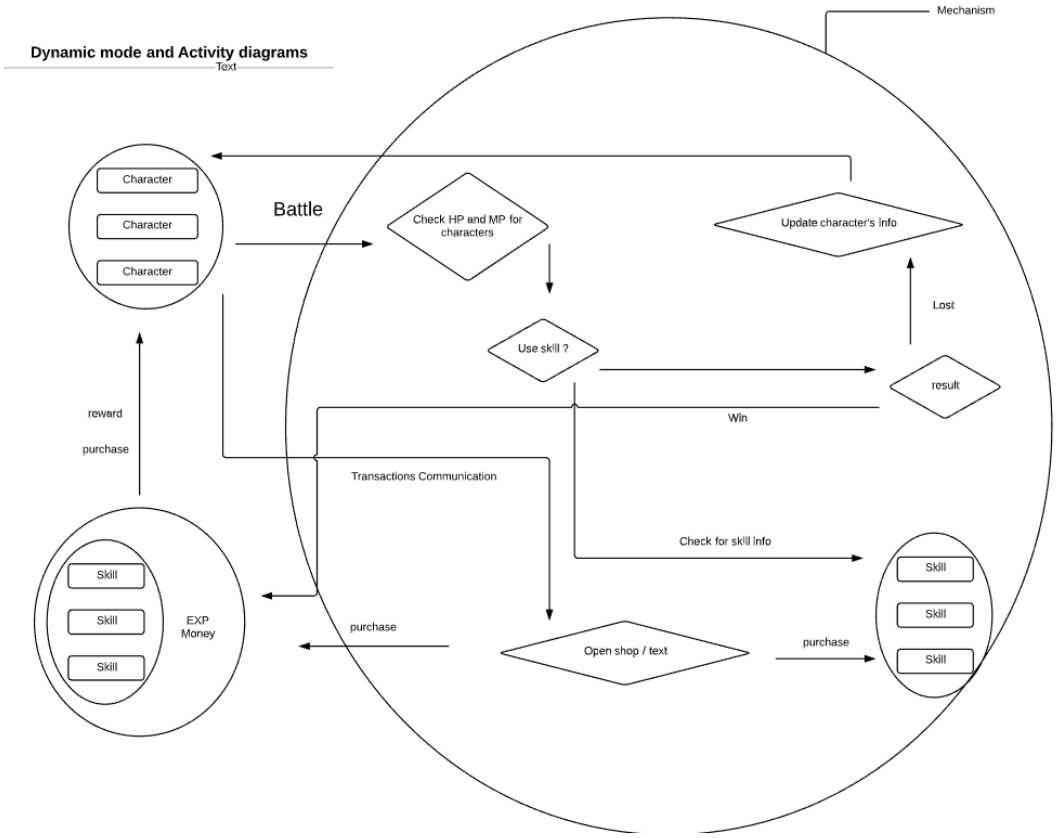


- After user make a keyboard or mouse input, the engine's built-in functions will catch the actions
- If the input is requesting for moving the character or saving and exiting game, the engine will respond it by updating the screen or create user's file
- If the input will lead to "decision", for example, whether the skill can be used or whether user has enough "money" to buy an item from shop. The engine will communicate will mechanism class and based on the user's requests, it may turn to check the information of characters or items.

4.3 Activity Diagrams

The next diagram indicates the detail for the activities generated by internal system and how other components process it.

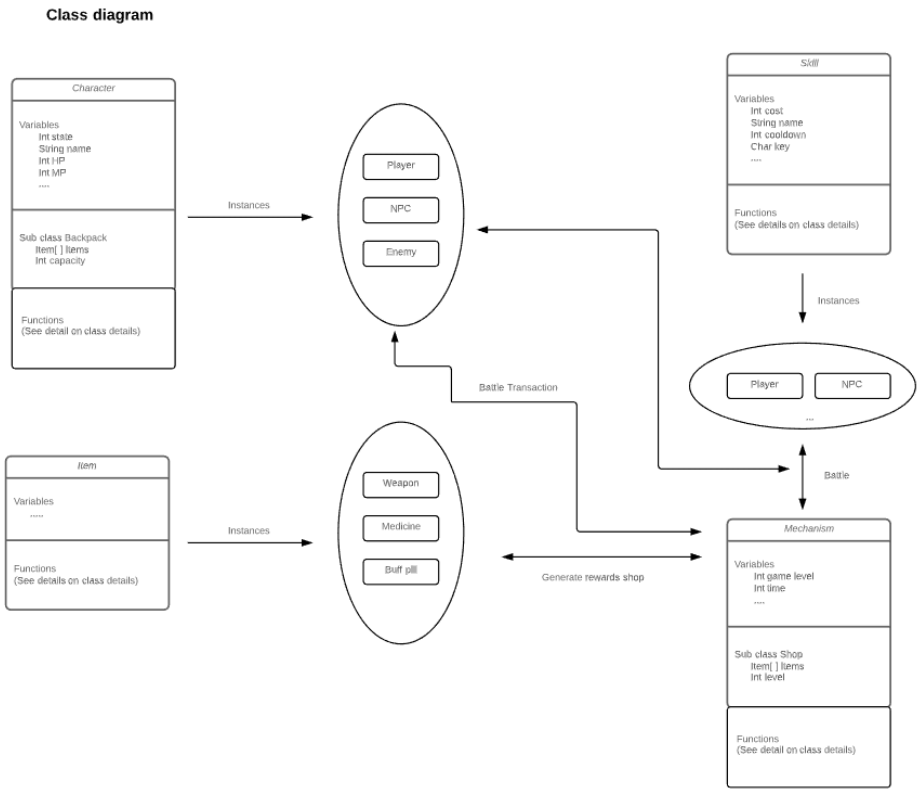
There will be other interactions and events happened within components, here we just show the actions that matter for users and developer to understand the structure of the system



The general purpose of “Mechanism” is to make the calculations and decisions. There might be some other functions in this class and not yet be showed in this graph such as how to calculate the damage cause by the characters to another character. This will be discussed with details later.

4.4 Class Diagram

The general structure of classes can be viewed here, for details please check the class list (later in this part)



- In this diagram, the elements in square block means the instances of a class.
- The Character, Item, Skill classes themselves should not be used directly
- Due to the way that Godot work, we will only use those basic class (or father class) as a template. Which means only the codes of instances will be found instead of standard way that implement the basic classed first and then create an instance through them. However, the instances belonged to same basic class will have exactly same structure.

4.5 Class List

More details of each class will be listed here. The details include for each class:

- the global variables
- inner (sub) classes
- basic functions

Here is the order of the classes:

1. Character
2. Mechanism
3. Skill
4. Item

Notice that the class might be abstract which should not be used directly, such as character, skill, and item. The mechanism class connects every class together and handle the user inputs, signals between components.

Due to the structure of Godot, the actual implementations of the classes may not be as exactly same as what they are listed here. For example, there will be interaction signals between components that are generated by engine automatically. The animation code will also not be included here. This document only provides the general description of the structure and how they interact. If you are looking for the detailed codes, please refer to the Godot scripts. The source codes will be available together with this document and exported product.

4.5.1 Character Class

#global variable

Type	Name	Description
Int	state	Constant, with valid value (PLAYER/NPC/ENEMY)
String	name	The name of this character
Int	hp	The HP amount
Int	mp	The MP amount
Int	level	The level of this character (have LEVEL_MAX limit)
Int	exp	The exp value, will have different upper bound to upgrade
Int	money	The currency this character hold
Int	damage	The damage can cause from the character
Int	move_speed	The moving speed of the character

#inner class

Type	Name	Description
Class	Skill	The skill(s) this character has
Item[]	Items	Items held by this character
Int	capacity	The max capacity of backpack

#basic functions

Type	Name	Variables (type)	Description
Void	__init__	Int state	Initialize the character (hp, mp, level, experience, state, name, money, backpacks value)
Void String	setName getName	Self Self, String name	Set/get the name to this character (able to change name)
Void	setHp setMp	self	Set the upper limit of hp/mp (usually when level up)
Int	getHp getMp	Self	Get the upper limit of hp/mp (usually when level up)
Void Int	setState getState	Self, int state Self	Change/return the state of the character
Void	addMoney	int amount	Set/get the amount of

	getMoney	Self	money
Boolean	isUpgrade	Self	Decide whether to upgrade
Boolean	isOutHp isOutMP	Self	check the hp/mp if it below 0
Void	upgrade	Self	Increase the level
Boolean	downHp downMp	Self	Decrease the hp/mp value. If < 0 return false, otherwise return true
Void	upHp upMp	Self	Increase the value of hp/mp
Boolean Item Item	addItem getItemN getItemI	Item String name Int index	Add/get item to/from the backpack
Void Int	addCapacity getCapacity	Self	Increase/return the capacity of backpack

#animation and physical related codes will not be listed here since they are Godot built-in functions, we do not need to explicit show them to either users or developers.

4.5.2 Mechanism Class

Notice that all the functions in the tables below **do not** actually exist on the script! Or at least they are not explicitly shown here.

The structure of the Godot separates each stage of the game (for example the start stage, in-game stage, game over stage) as independent "scene". The mechanism of each scene is independent from others unless we require them to interact. In other words, the complexity and implementations of this class are highly dependent on what is currently shown on the screen.

Within the start scene, the mechanism will be as simple as responding to the click of the "start" button, however, once player enter the certain level of the game, the mechanism may come to totally different as it will be designed to judge on the lost or win of the game, apply physical oscillation, calculate damage from characters to characters, perform skill effect on the character. Also, this class takes the responsibility to update user information so that player can check for they status at any time.

Although the mechanism class in different scene will need to be implemented separately, there are still common parts through the scenes that have similar functions and purposes. We separate the scene into two main parts, one of which is the ***Operation*** scene and the other is ***Level*** scene.

In ***Operation*** scenes, they handle the basic initializations of the game, such as asking player's name, initialize the level scenes, restore the default values once the player dead. Since these scenes have simple and straight forward functions, the structure of them will be as simple as a button click or scene switch, therefore they will not be shown in this document. If you need to check for the details, you can just refer to the source codes.

In ***Level*** scene, the basic functions are similar: place the player – place the monster – initialize everything – wait for signals from characters and other components. Here in the following text, you will find the description of the battle mechanism and simple structure of the mechanism class in this case.

Pre-loaded class – there is a preloaded class in the program which is used for storing the current data of player and pass to the next level if the player chooses to continue the game. The interactions through different scenes normally should only about this class.

Encounter logic – each character will have its own physical oscillation shape, once two characters' (assume monster and player) oscillation shapes overlapped, then it will activate a signal to tell the monster to be ready to attack.

Monster AI – it depends on certain type of the monsters. In our test, the monster will move randomly within the map and attack the player if their shapes overlapped over one second.

Animation – the character should have animations for different actions, such as running, attacking, as well as stand unmoved or being attacked. Generally, different actions for a character will be controlled by the keyboard inputs. Mechanism should handle the inputs and give the right signals to the components.

Battle result – if the player kills all the monster in current scene, then the player wins the game or able to go to next level. If the player be killed by monster, then player lost.

Information buttons – there will be HUD shown on the edges of the screen, some of them will be in the form of buttons. By clicking those buttons, the program will open a map, status board or shop page.

#constant and global variables

Type	Name	Description
Int	game_level	The level of the game
Int	time	Constant (DAY/NIGHT)
Character[]	enemies	Enemy characters in this level
Character[]	NPCs	NPC characters in this level

#inner class

Type	Name	Description
Class	Shop	The shop of the game
Item[]	goods	Items in this shop
Character[]	NPCs	NPC characters in this level

#basic function

Type	Name	Variables (type)	Description
Boolean	getResult	Character player, Character boss	Return true if player.hp > 0 and enemy.hp <= 0 (draw counts as win), otherwise false
Void	items money exp	Character player, Item[] items Character player(enemy), int amount Character player, int exp	Give the target rewards to player/enemy
Void	toNextLevel	Void	To new level (new map) and place the

			characters from above lists on screen
Void	refresh	Character[] enemy, Character[] NPC	Set new characters. To show them, by calling toNextLevel
Void	showHUD	Void	Determine which items will be showed in current screen
Void	setLevel getLevel	int level void	Set/get level of current shop

As mentioned at the beginning of this part (4.5.2), the functions shown here are just to illustrate the structure of the mechanism class under in-game scenes. The implements of those functions might be a piece of code in main function or it can be just a signal passed from the components in the scene. It is highly recommended to ready the source codes for certain scene to understand the interactions between different parts of the program. For debugging or modifying purposes, the codes for each scene should be independent in principle.

4.5.3 Skill Class

#global variables and constant

Type	Name	Description
Int	Cost	Amount of mana required to execute
Int	Cooldown	Amount of time (seconds) until the skill can be used again
Int	mode	The skill will have effects on (self, other, AOE)
String	name	Name of the skill

#basic functions

Type	Name	Variables (type)	Description
Void	__init__	int cost, int cd, string name	Initialize a skill
Void	setCost getCost	int cost Self	Reset/return the cost of the skill
Void	setCD getCD	int time Self	Reset/return the cool down time of the skill
Void	coolDown	Void	Cool down the skill. Cannot be used until cool down time expire
Void	activate	-	Activate the skill and apply effects

For understand purpose, here is the source codes of one of the skills that exists in the game.

```
const MODE = { "SELF" : 0, "OTHER" : 1, "AOE" : 2, "AOE_SELF" : 3}
var skill_name = "skill"
var cost = 0
var cd = 0
var mode = MODE.AOE
var target
var can_use = true
signal cd
signal cd_done

func coolDown():
    $cd.start(cd)
    can_use = false
    emit_signal("cd")
    return
func _ready():
```

```
        skill_name = "speed up"
        cost = 0
        cd = 3
        mode = MODE.SELF
        return
    func activate(character):
        if can_use:
            can_use = false
            print("speed up")
            target = character
            target.move_speed *= 3
            $active.start(3)
        else:
            print("not ready")
        return
```

In the example, the skill is initialized with a new name and basic values in ***_ready()*** function. This skill will increase the moving speed of target character (player) for three seconds and then come into 3 seconds cooldown time before it can be used again.

4.5.4 Item Class

This class generally cover all the stuff that can be sold in the shop. In current version of the game, we only have skills to be sold in shop. In this case, those skills are all belonged to item class.

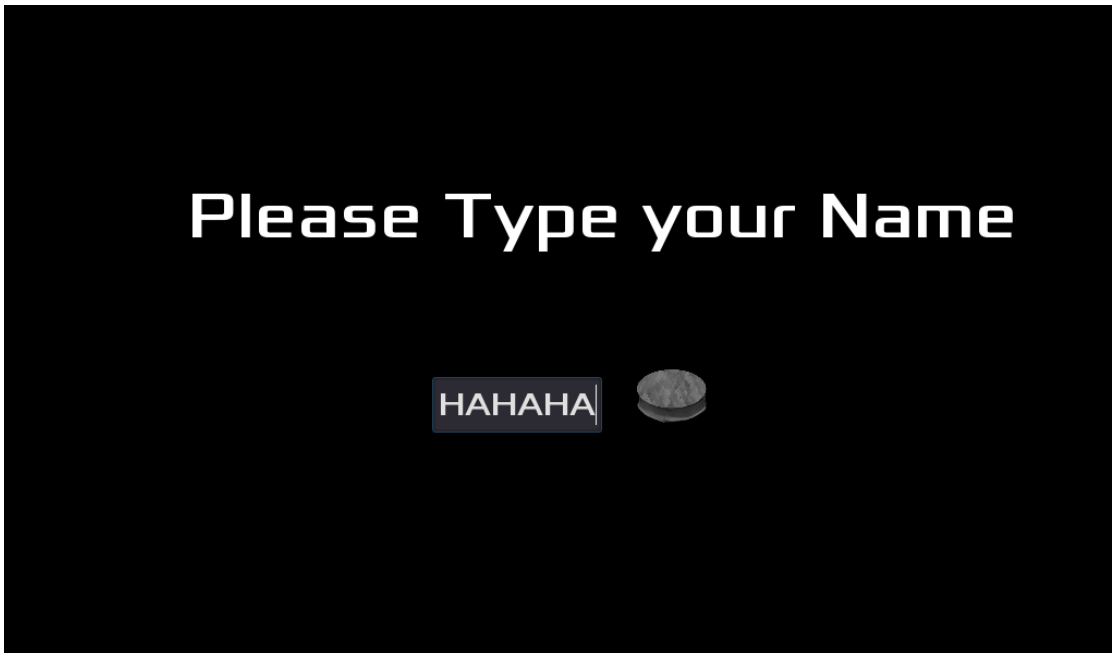
4.6 The Prototype of Platform RPG game

In this part you can find some real screens of Platform. For details about how to control the character, please refer to section 6.

Notice that the pictures are all from the test version of Platform, there will be difference in release version.

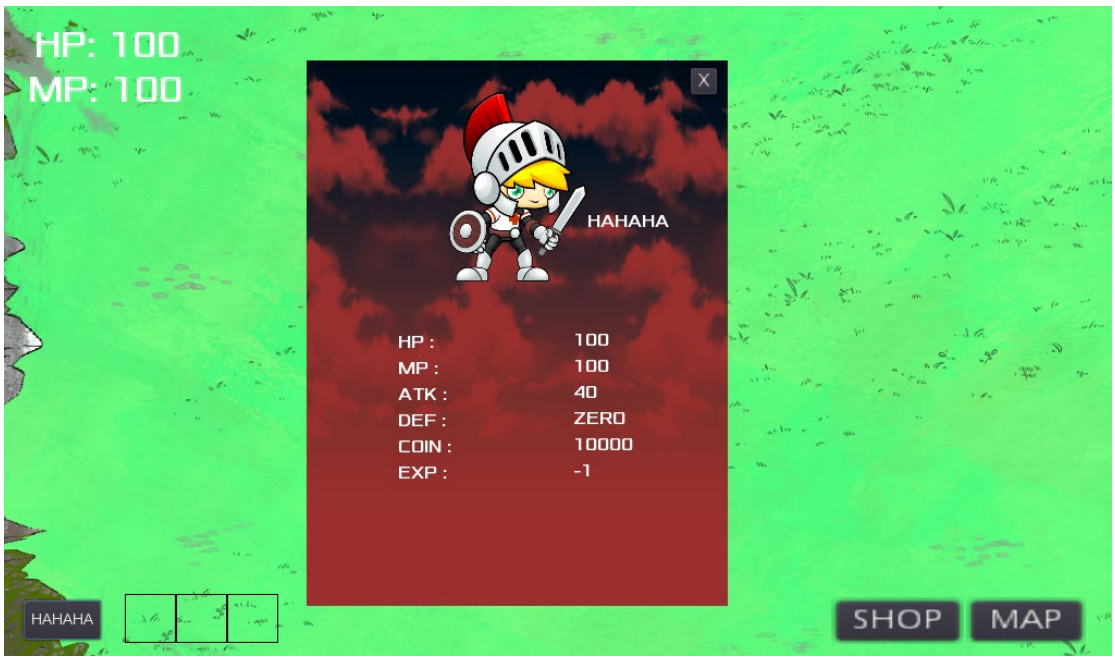
4.6.1 Start Scene

4.6.2 Name Scene

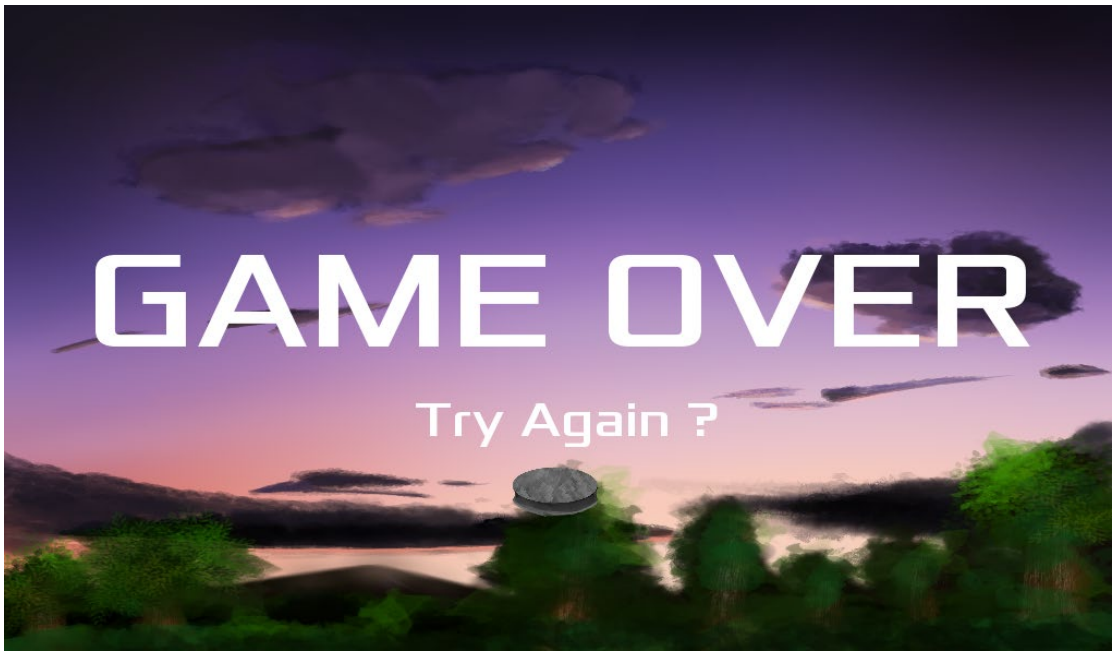


4.6.3 In-game Scene

4.6.4 Player Info Page



4.6.5 Lost Scene



4.6.6 Win Scene

5. Components and System Tests Result

5.1 Definition of Marks

	Not Applicable / Title or subtitle
	Pending
	Failed
	Pass

5.2 Test Results

The attached excel file will contain exactly same tables.

5.2.1 UI Tests

GUI Test						
UI	***Test for GUI location and layout***					
UI-1	Test for the layout of log in (start) window	Open the game will automatically enter the start window.	-	The buttons should stay in the center of the window within one column		2019/11/12
UI-3	Test for the in-game UI layout	During the in-game activities such as in village, in battle, user can check the basic information in HUD	-	1. the HP and MP bar will be shown in the up-left corner of the window 2. the minimized character appearance and the character name, level, money exp bar will be shown at the left bottom of the window		2019/11/12
UI-4	Test for the backpack window's layout	User can open their own backpack in certain area by pressing pre-defined key	-	1. The character will be shown at left side of the window together with the statistic of the character (damage, MP, HP) 2. The current equipment (weapon, armor) will locate at the right of the character in column 3. On the right side of the window, there are all available slots to store other items		-
UI-5	Test for in-battle specific information	When encounter with monster, there should have specific information to indicate the actions between player and monster	-	The amount of the damage or HP/MP consumption will be shown above the character who will apply the effects to.		2019/11/22
UI-6	Test for in-game shop layout	User can open the shop page in certain area by pressing pre-defined key	-	1. The items will be shown on the left half side of the window 2. The right have side will show the cost and simple description about the selected item with "Purchase" button		2019/12/2
UI-7	Test for reward window layout	After each boss battle, whether player win or lose, there will be a window show the reward to the player	-	The layout will almost same as that of the shop, instead there is no "Purchase" button but a "Confirm" button		-
UI-8	Test for layout after battle	Show the Win or lose information	-	1. If win, a big message says "WIN" and congratulations will be shown at the center of the window 2. If lose, a big message says "You Died" will be shown in the center instead.		2019/11/12
UI-9	Test for layout of communication	There are some communications between player and NPCs	-	The message box will show at the bottom of the window		-
UI-10	Test for the layout of map	User can open the completed map by pressing pre-defined key	-	1. The map will take the place of the whole window 2. A go back button will be available on the map corner		2019/12/3

The UI-4, UI-7, UI-9 are unavailable due to the change of functions for the program,

5.2.2 Animation Tests

Animation Test						
AN	***Test for the animation and user-controlled interactivities***					
AN-1	Test for the movement of the character	Press the pre-defined key to move the character to left	Pre-defined key	1. The user character will move to certain direction after press the key		2019/11/15
AN-2		Press the pre-defined key to move the character to right		2. Character will continue moving if the key does not be released		2019/11/15
AN-3		Press the pre-defined key to move the character to jump up		1. User character will jump up and then fall down to ground after pressing the key 2. Character will jump up again after one completed jump action if the key does not be released		-
AN-4	Test for the gravity	Jump the character Go through the map to where has the different in heights		The character should fall to the (lower) ground after jumping or just going through the gap.		-
AN-5	Test for attack action	Press the key to ask character attack but nothing to hit		The character will perform pre-designed actions after pressing the key		2019/11/15
AN-6		Press the key to ask character attack and hit the monster		The monster will react to the character's attack		2019/11/15
AN-7	Test for being attacked	Design a monster to attack the character	-	The character will react to the attack		2019/11/16
AN-8	Test for character self-moving	To see if the NPC and monster are able to move randomly without the control of user	-	The NPC and monster should able to move or attack or react to the communication from the player.		2019/11/17
AN-9	Test for integration action	Control the player to have a battle with the monster or communicate with NPC	-	Characters should react properly as described above.		2019/11/16

The AN-3, AN-4 are unavailable due to the camera view change of the program

5.2.3 Mechanism Tests

Mechanism Test						
MC	***Test for the interaction between the system classes***					
MC-1	Test for start game	Open the executable file	-	The initialization window will show up and ask user to start or exit		2019/11/12
MC-2	Test for create a character	Click "start" on start window and fill all the information to create a character	-	User now will enter the game with the certain name shown on the specific place on the window and all other related information with default value set up		2019/12/2
MC-3	Test for open backpack	Click button	Pre-defined key	The empty backpack will be opened		-
MC-4	Test for communicate with NPCs	Interact with NPCs	-	Will open a text box with pre-defined strings		-
MC-5	Test for attack (by) monster	Attack (by) the monster	Pre-defined key	Character should lose HP which equals to the damage from the other character		2019/11/22
MC-6	Test for skill	Attack (by) the monster with MP skill		1. Character should lose HP and the other character will lose MP 2. If the remaining MP is not enough to take the skill, then nothing will happen or change		
MC-7	Test for skill act on player himself	Apply the effect of the skill to player himself		Player will be healed by consuming MP (+HP, -MP)		2019/12/2
MC-8	Test for wining	Kill the monster (boss)	-	Corresponding GUI change will happen and check the rewards from battle		2019/11/22
MC-9	Test for losing	Kill the player	-			2019/11/22
MC-10	Test for get item from reward	Get the reward	-	1. The backpack will be filled with certain items. 2. If the items number exceed the capacity of backpack, the exceed items will be discard (delete from backpack)		-
MC-11	Test for get item from shop	Purchase from shop	-	1. The backpack will be filled with certain items. 2. If the items number exceed the capacity of backpack, a message box will be shown to notify the player and the latest transaction will be denied without changing on user's data		-
MC-12	Test for (un)equip with new item	(un)Equip weapon or armor	-	The equipment will show at specific place on backpack window and the data of character will act to the change from (un)equipping the item		-
MC-13	Test for using an item (consumables)	Use an item	-	The effect (in this case, healing HP) will be applied to the character (+HP)		2019/12/2
MC-14	Test for transaction	Purchase from the shop	-	The amount of the money kept by character will change due to the transaction with the shop		2019/12/3
MC-15	Test for upgrading	Level up	-	The level and exp amount will either increase or decrease due to the upgrading process		2019/11/22

The MC-3, MC-4, MC-10, MC-11, MC-12 are unavailable due to requirements changes and MC-6 is pending due to art works delayed.

6. User Manual

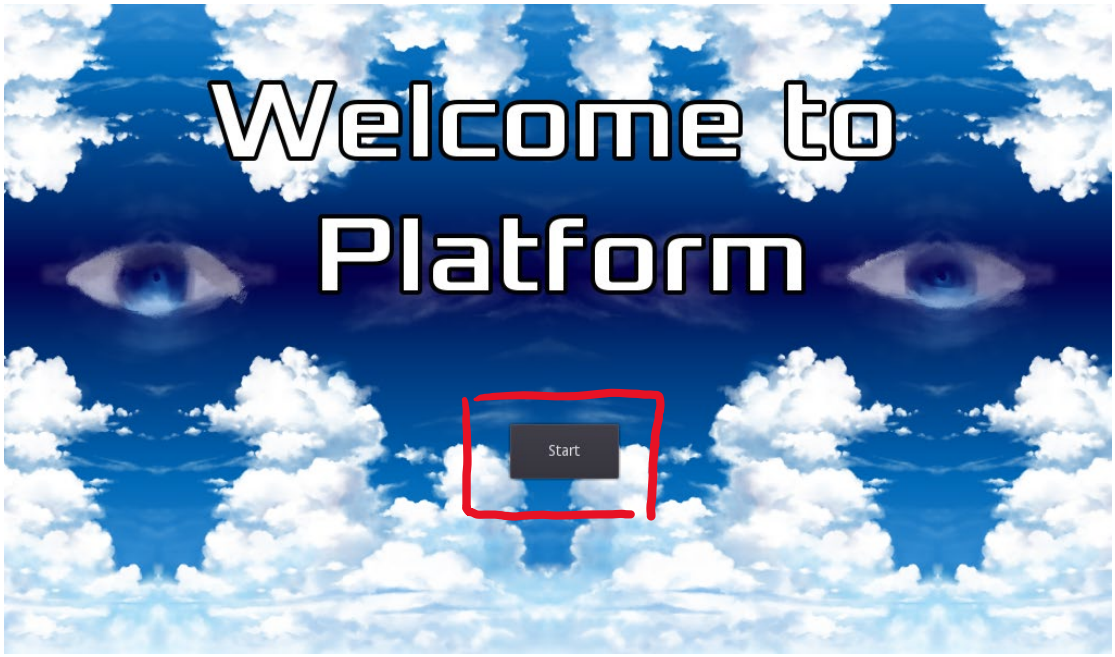
For players or other users. In this section, you will find the descriptions about UI functions and how to control your character.

6.1 Induction to UI

First is about how to start the game (6.1.1) and where to check your character status in game (6.1.3).

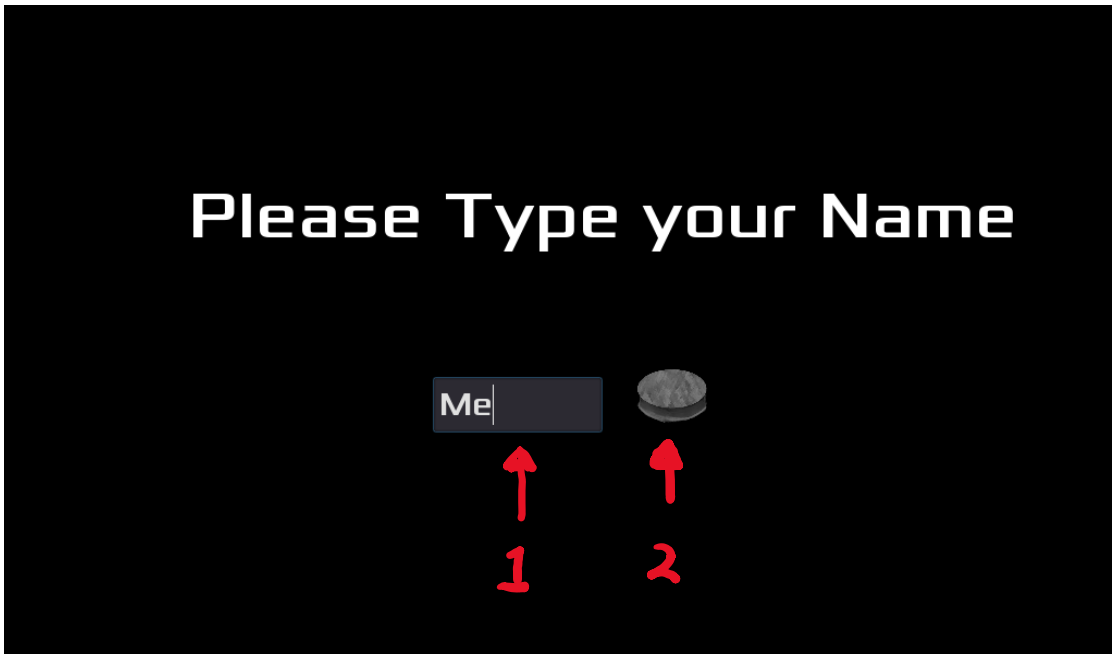
6.1.1 Start the Game

After running the game, you will directly come to the home page of Platform. By clicking the button below the welcome message to go to the user information screen.



6.1.2 Enter the Name

On this screen, you can enter a string as the name for your character, then click the round button on the right to start the game.

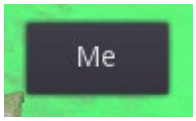


6.1.3 In-game UI

After entering the name, you will now come to the first level of Platform. There are some buttons or square blocks on the side of screen, now we are going to introduce the functions for those buttons and panels.

6.1.3.1 Character Status

The button on the left bottom corner is the character status button. You can easily find that your name on it.

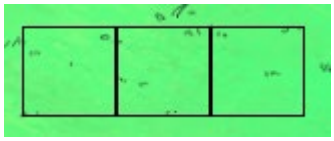


By clicking this button, your full character data will be show in the middle of the screen like this

From here, you can check everything about your character. By clicking either the cross on the top right of this small panel or the character status button again to back to game.

6.1.3.2 Skill Block

On the right side of the character status button is the skill block.



A character can load three skills at most, currently we don't have any skills since we are new to this game. Skills can be bought from shop.

You can also check for the availability of the skill. If there is a red cross on the skill, that means this skill now is in cooldown time, you have to wait until the cross disappears to use it again.

There are three skills available. From left to right:

Speed up: Increase the moving speed for certain amount of time (Lv.1 → x2)

Attack up: Increase the damage of the player for certain amount of time (Lv. 1 → x2)

HP up: increase the HP of the player for certain amount (Lv.1 → +30 hp)

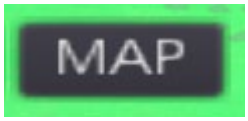
6.1.3.3 Shop

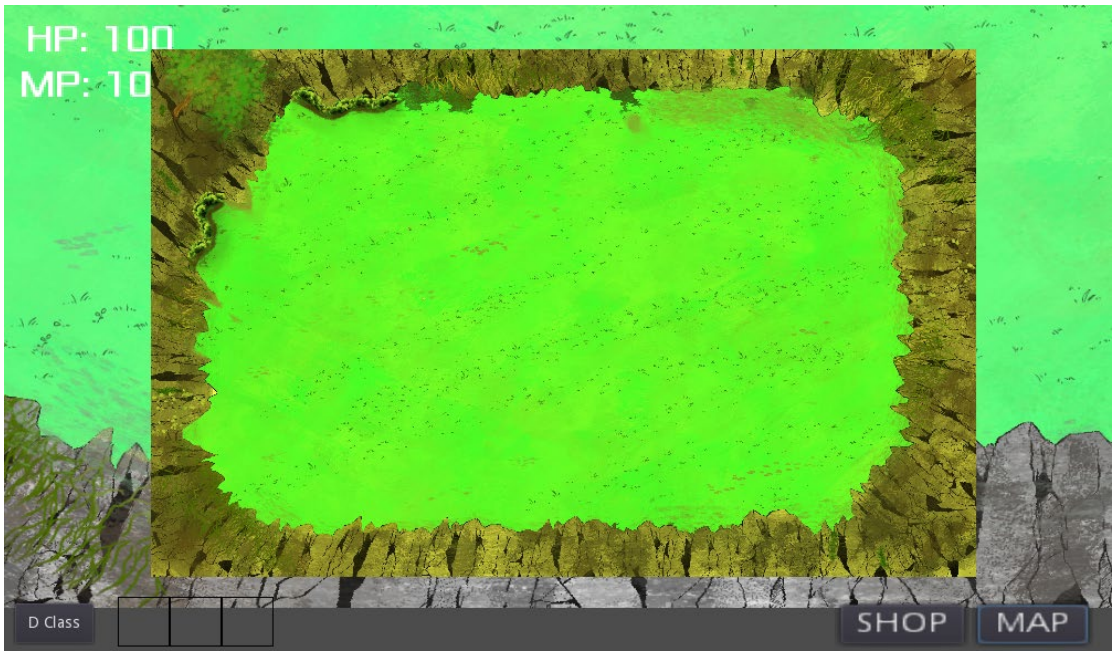
On the right bottom of the screen, we have two buttons. One for shop and the other one for map. Click the buttons will open the corresponding pages.

In the shop page, you can use money to unlock or upgrade the skill.

6.1.3.4 Map

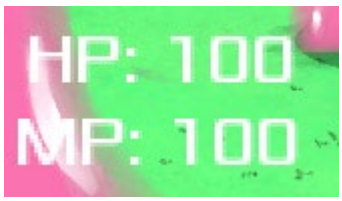
Click the MAP button or press M will open a small map that shows the borders and characteristic of the map for current level.





6.1.3.5 HP and MP

Although the HP and MP information can be found in status panel, it will be more friendly and easier for players if we list the information that need to be frequently checked on the HUD.



6.2 Default Values

The default (initial) values of player related data

Variable Name	Value
Name	D Class
HP	100
MP	100
Money	10000000000
Skill 1	Unavailable
Skill 2	Unavailable
Skill 3	Unavailable
Damage	40

The name of the character is the only variable that can be changed at beginning by player.

6.3 Control

The table below contains the keyboard mapping for controlling the character.

Action	Key
Move up	W
Move down	S
Move right	D
Move left	A
Attack	[Space]
Map	M
Skill 1	1
Skill 2	2
Skill 3	3

6.4 Win or Lose

To finish or win the game, player need to kill all the monsters in the level(s). If player dead, then the game ends.

7. Developer Team Information

Information about the developer involved in the developing of Platform.

7.1 Declaration

All the classes, component structures and mechanism are designed by members of Group 7 of CP317 in Laurier 2019 Fall term.

The art works shown in Platform except for the texture of in-game buttons, shop backgrounds are all produced by our own artist.

7.2 Members

He, Xiaohu (Artist)

hexx2400@mylaurier.ca

Lowe, Garret (Developer)

lowe7070@mylaurier.ca

Meng, Kaifeng (Developer & Leader)

meng0880@mylaurier.ca

Wang, Shibo (Developer)

wang9820@mylaurier.ca

Zhang, Weijie (Developer)

wang9820@mylaurier.ca