# PROJECT-ACCEPTANCE-STANDARDS

Zhu Jiahao
Dong Shuhe
Yi Haixin

March 2025

## Constraint Objectives

### Map

1. The map will be generated around the player, featuring land and water.

2. The map will randomly generate resources, such as rocks and trees.

3. Enemies will be generated periodically on the map.

4. The game should feature a sky with day and night cycles.

5. Map generation will consume relatively few resources, and the frame rate can remain stable at more than 100 fps when only the map, resources on the map, and players are rendered.

### Player

1.The class **Player** must be inherited from **Entity**. The script can also get access to **PlayerStats**, which is designed to record and compute the player's current attribute data. The **PlayerStats** is inherited form **EntityStats**, and all attribute information must be declined in this script.

2. Players' movement can be controlled by using the keys up, down,

left, and right (or W A S D) to move in those respective directions, and can also move diagonally. Consequently, animations are played during the movement.

3. Players can move by rolling, and they can roll by pressing the space key.

4. Players have health points, and they die when their health points reach zero.

5. Players can attack using melee weapons, while the left mouse button serves as the attack key.

6. Players can mount and dismount vehicles, and ride them. Vehicles, as independent entities, can be parked on the map.

7. Players can pick up guns and ammunition and use them to carry out ranged attacks.

8. When the camera rotates, the player can still play the animations normally.

## Enemy

1. The class **Enemy** must be inherited from **Entity**, and different enemy has its own class inherited from **Enemy**. The subclasses of **Enemy** can get access to their corresponding attribution classes. All the necessary attribute information are declined in these scripts. e.g.Class **DustWarrior** is able to access to **DustWarriorStats**, whose parent class is **EnemyStats**. And the **EnemyStats** is inherited from **EntityStats**.

2. Enemies can move around freely when they are in their idle state.

. Enemies can detect the player, track the player, or be eliminated by the player.

3. Enemies can attack the player, and their attack types are diverse.

4. Enemies can take damage, and they die when their health reaches zero.

5. Enemies can play corresponding animations based on their current actions, and the animations continue to play normally even when the camera rotates.

6. After the enemy dies and their skill execution ends, the entity can be correctly removed.

## UI

1. A main menu is required when entering the game.

2. In the game, pressing the ESC key brings up the pause menu, where players need to choose whether to resume the game or exit.

3. In the game, players can find the inventory menu, which has slots to store the items.

4. The game should have a UI panel that displays the player's health, items, and other necessary statuses.

5. The game should include a mini-map that shows the player's location and the surrounding environment.

## Camera and Shader

1. The camera can be rotated 360 degrees to the left and right using the Q and E keys. There are a total of eight angles, with each angle being 45 degrees apart.

2. The player's view of the map is limited, with the edge of the view being a curved horizon and the sky above it.

3. Between the sky and the ground, there is a fog effect. Other appropriate places in the game (such as when a motorcycle is being driven by the player) should also have suitable smoke effects.

4. The display of all entities should be affected by lighting, with suitable shading and shadowing.

## Task

1. The Task game objects must run individually. Reduce its independence as much as possible.

2. When a Task is finished, the Task game object will destroy itself.

3. When a Task loop is over, player will get rewards.

## Game

1. The game needs to support saving and loading.

2. The game must run smoothly for extended periods without frequent lagging, which means its operation cannot consume excessive resources. Additionally, the graphical frame rate is expected to remain stable at more than 60 fps while running the software.