Final Project Proposal

Fate/Game Maker

Team members: Ruichen, Chengxi, Eric Wei

Core Concept

This is a first-person shooter game where players use different guns to attack enemies appearing in the woods. There will be an NPC in the game who provides firearms and missions.

Mechanics

The player controls a first-person character using mouse input to aim and shoot enemies. The shooting mechanism is implemented using an invisible line segment from the gun. When the player fires, this line segment detects collisions with enemies. If a collision occurs, the enemy's health decreases. The length of the line segment determines the shooting range, and the amount by which the enemy's health is reduced defines the damage dealt. Enemies have a fixed amount of health that decreases when hit by the player's weapon. When an enemy's health reaches zero, the enemy is destroyed and the player earns points. Points are used both as a score indicator and as a currency for purchasing weapons from an NPC. The game ends when the player completes all available quests.

UI

The game includes several UI elements to support gameplay clarity. An inventory UI displays the weapons currently owned by the player and indicates which weapon is currently equipped. A score display shows the player's current point total in real time. When the player approaches an NPC within a certain distance, a text UI prompt will appear asking whether to start a conversation. After clicking to talk, a dialogue UI will show up with options to purchase guns and view quests.

Content

Environment

The game takes place in a forest environment with limited visibility created by trees and terrain. The environment is intentionally compact and designed to guide the player forward while still allowing exploration. Natural obstacles such as trees and rocks are used to control player movement and enemy encounters.

Items

Different gun

The game includes three weapons:

- G17
    1. This weapon spawns in the player's inventory at the start of the game,it is the player's starting weapon.
    2. It has a cooldown on the fire button. When the trigger is pulled, the weapon only detects objects colliding with an **invisible line segment** for one second.
    3. This is a semi-automatic pistol with the **lowest damage** of all weapons.
- AR-15
    1. This is a semi-automatic rifle.
    2. It uses the same gun Model of G17, but differs in range and damage, which means the length of the invisible line segment and the HP reduction applied to enemies upon collision are different.
    3. Players have to spend 30 points to get this rifle.
- AKM
    1. This is an automatic weapon and uses a separate Model.
    2. When the fire button is held down, the invisible line segment continuously detects colliding objects and applies HP reduction to them.
    3. Players have to spend 50 points to get this rifle.

Each weapon has unique damage and firing behavior to encourage player choice.

NPCs

The game features a single NPC who acts as a quest giver and weapon vendor. This NPC allows the player to purchase weapons using earned points and provides quests that guide player progression.When the player approaches an NPC within a certain distance, a text UI prompt if clicked starts dialogue with the NPC. After clicking to talk, a dialogue UI will show up with options to purchase guns and view quests.

2 choices ：talk and upgrade weapon

Audio

Audio feedback is used to reinforce player actions. Shooting a weapon plays a corresponding firing sound. Completing a quest triggers a short completion sound effect to reward the player and provide clear feedback.

Quest content

The game includes two quests:

1. Earn points by defeating enemies：
    a. When the player reduces an enemy's HP to 0 using a weapon, the player's Point UI increases by 1. At the same time, the counter (0/100) for this quest in the NPC dialogue UI also increases by 1.
    b. When the player's Point UI displays 100 or the counter in the NPC dialogue UI displays (100/100), a new UI prompt with the text **Submit** will appear in the NPC dialogue UI.
    c. Clicking **Submit**, the quest will show as **Completed**, and the player's Point UI will decrease by 100 points.

2. Purchase all three weapons from the NPC:
    a. The player can purchase guns in the NPC dialogue UI. The purchase UI displays **AKM** and **AR-15**.
    b. After the player clicks Purchase, the Point UI decreases by the corresponding points.

c. When the player obtains **3 guns** in their inventory, the quest will show as complete.

Completing these quests represents successful completion of the game.

System Architecture

Model-View-Controller pattern

The Model consists of data classes such as enemy health, weapon stats and mechanics of shooting, and score values.

The View includes UI elements, VFX like blood & hit effect, and audio feedback that respond to changes in the game state.

The Controller handles player input, such as shooting and interacting with the NPC, and updates the Model accordingly. This separation makes the system easier to maintain and extend.

Singleton pattern

We implement the GameController as a Singleton to serve as the single class of game for **event handling and dispatching**. This class can eliminate the extra need to handle interactions between different classes when developing other scripts, and making group coordination more simple by allowing each coder to focus only on their own script's internal logic. For instance, in the shooting mechanism, when the invisible line segment detects a collision with an enemy's rigidbody during firing, it dispatches the damaging event. The enemy class then subscribes to this event, and once the event is dispatched, the subscribed class will reduce the enemy HP.

Additionally, We choose GameController rather than the Player as the Singleton to ensure it can be purely dedicated to event management. By contrast, Player class is responsible for other gameplay logic including movement and shooting, making it unsuitable for a single global event handler.

Finite State Machine pattern

FSM pattern is implemented to manage **mutually exclusive states**, which means certain GameObjects can only activate one state at a time. We define different states using enum lists (enums). In our game, FSM is applied to manage the states of weapons and the inventory system. Each weapon has three states: **Unowned**, **Owned**, and **Equipped**, and the state will change when some statements reach. A weapon is marked as *Unowned* when the player has not purchased it yet; once the player completes the purchase, the weapon's state transitions to *Owned* and it becomes visible in the player's inventory. When a weapon is set to the *Equipped* state, it is actively held and usable by the player in-game.