Joshua Caufaglione

Player Mechanic

Concept:

This game mechanic is the main player action character from top-down wave survival games like Nuclear Throne, Brotato, Enter the Gungeon, and Vampire Survivors.

Core Mechanic:

This Mechanic consists of a top down 2D player who can move at a constant speed in all directions using the following keys, “w” for north, “a” for west, “s” for south, and “d” for east. The player will only have two weapons, a sword and blaster. These weapons can be selected by either the mouse scroll wheel or the “1” and “2” keys. Based on the position of the user's mouse the sword will swing in that direction dealing damage to enemies. The sword will have a cool-down of 0.5 seconds. As for the blaster, it will shoot a projectile in the direction of the user’s mouse in a straight line with a constant speed. The projectile will have a small circular hitbox and need to be deleted from the scene after 1 second of flight time or when the projectile hits an enemy or a wall. The blaster will have limited ammo of 10 rounds in the magazine and a capacity of 50 rounds. The player will be able to reload the blaster by pressing “r” on the keyboard. The player will press “left-click” to attack, in this case, shooting the blaster or swinging the sword. Each sword and the blaster will utilize different layers. The blaster’s projectiles will be able to hit enemies on layer 1 (ground layer) and layer 2 (flying layer). Whereas the sword will only be able to hit enemies on layer 1. To keep track of the blaster ammo, a UI element will be placed in the bottom right corner displaying “ammo/capacity”. When the sword is equipped, this UI element will display the cool-down time with a sliding bar. Additionally, each weapon will start with a default damage of 1 but can be upgraded to do more damage as the game progresses.

Steering away from the weapons and movement, the player will need to have a health component. This health will start at 100 and be decreased by an integer value based on which enemy attacks the player. Additionally, the player will have a critical hit chance which starts at 1% and can be increased as the game progresses. The player will also have a debuff component which is initially empty but can be added too with debuffs such as slowness which will decrease movement speed or poisoned which will increase enemies damage to the player. Finally, the player will have a money component. This is what allows the player to upgrade weapons and buffs. 10 dollars is earned for each time an enemy is hit and 60 dollars is earned for each elimination.

Target Audience: The target audience would be for those who like survival games similar to Call of Duty Zombies or action games with simple but fun mechanics like Cuphead.

Visual Design:

In Figure 1 is the UI for the player. Ammo/Cool-down information is in the bottom right, Money is in the bottom left, and the round/wave number is in the top right.

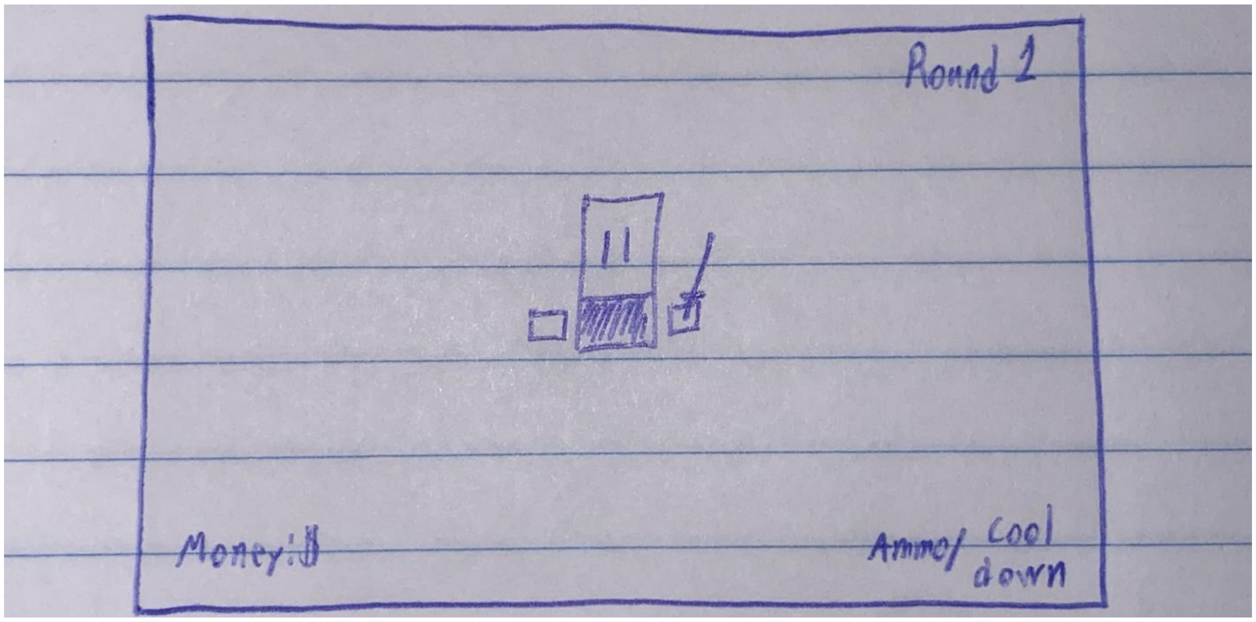

Figure 1.

Figure 2 is an example of how the blaster will be able to hit flying enemies.

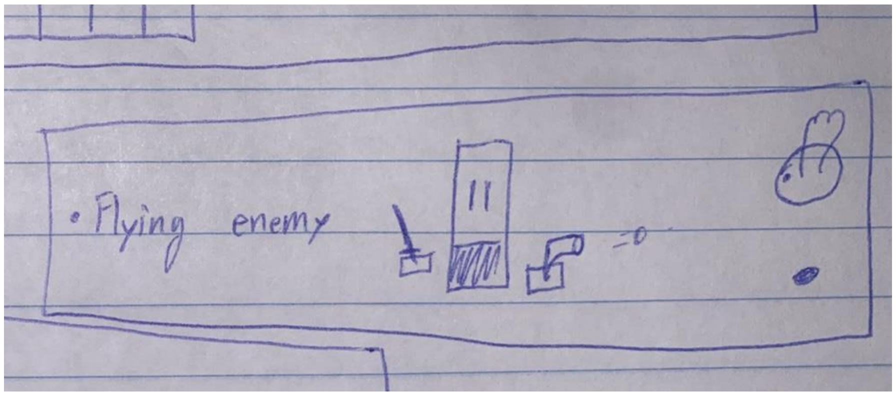

Figure 2.

Figure 3 is an example of how the sword will hit standing enemies.

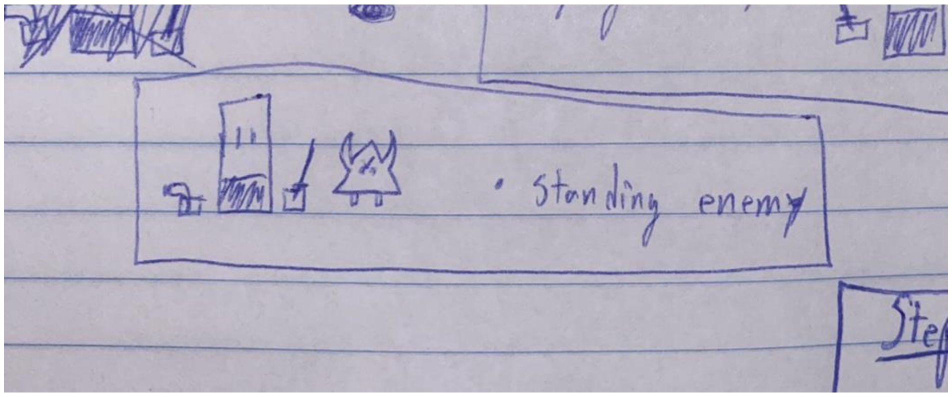

Figure 3.

Figure 4 shows how the blaster will follow the direction of the user's mouse.

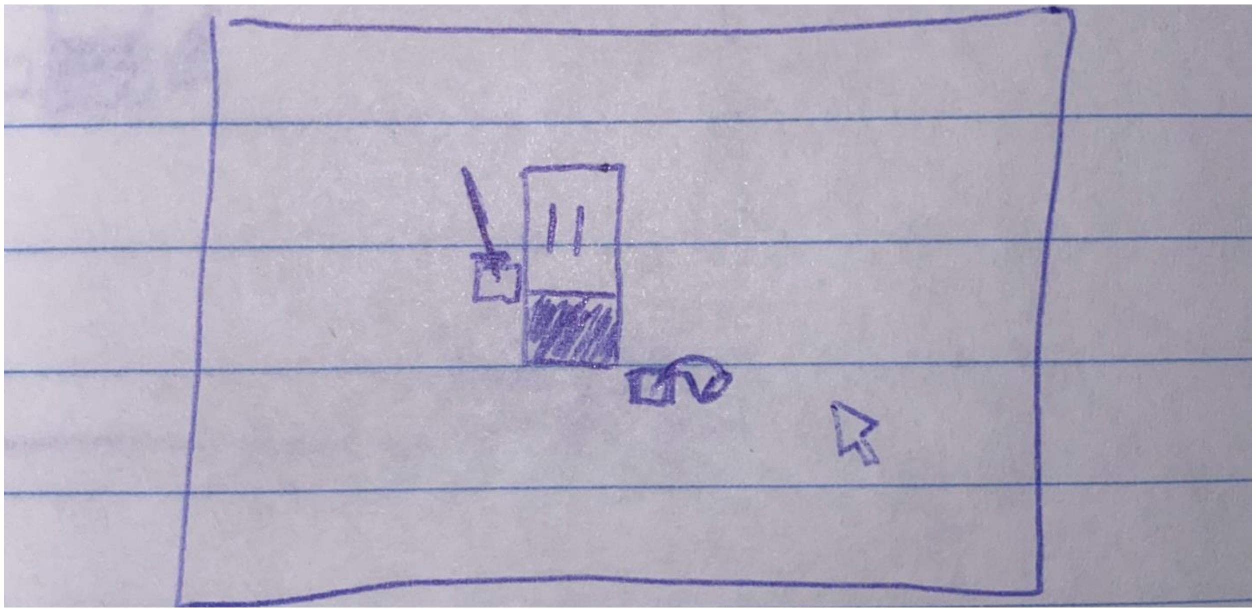

Figure 4.

Scope:

The demo will include a training map with two main sections. A ground enemy and a flying enemy. The enemies will be taken from publicly available assets and projects to simply showcase this player mechanic. The goals of these sections are to display the following:

1) Melee weapons dealing damage to ground enemies and not flying enemies.

2) Ranged weapons dealing damage to flying and ground enemies.

3) Character movement and UI elements.

Step-by-Step Instructions provide by ChatGPT:

1. **Implement Player Movement**

   - Code the player movement using the “w”, “a”, “s”, “d” keys for 2D movement.

2. **Create Basic Player Character**

   - Design a simple player sprite and add it to the scene.

3. **Weapon System**

   - Implement the switching mechanism for the sword and blaster using mouse scroll or number keys.

4. **Sword Mechanics**

   - Code the sword swing based on mouse position, ensuring it has a 0.5-second cooldown.

5. **Blaster Mechanics**

   - Implement the blaster shooting mechanic, including projectile movement and hit detection.

6. **UI Elements**

   - Create a basic UI to display ammo/capacity and cooldown timers.

7. **Health and Damage System**

   - Implement the health system for the player, including damage taken from enemies.

8. **Critical Hit System**

   - Add a critical hit chance mechanic and code its effects.

9. **Debuff System**

   - Create a debuff system to apply effects like slowness and poison.

10. **Enemy Implementation**

   - Design and code enemy behaviors, including how they attack and respond to player actions.

11. **Scoring and Currency System**

   - Implement a currency system that tracks player earnings from hits and eliminations.

12. **Upgrades System**

   - Create a system for upgrading weapons and buffs using in-game currency.