Name: Bryce Hendrickson Mark \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/50

## Brief introduction \_\_/3

My feature for the Zombie Dog Outbreak 2 game is the character control, movement, design, animation, and weapon control and interactions.

My job is to create a character sprite equipped with responsive movement and attack(melee) functions that will interact with the enemies in the game. I will also be responsible for controlling player health when attacked. As well as the BC character mode (character subclass) that cannot die and will have different animations based off when they should be dead.

## Use case diagram with scenario \_\_14

### Use Case Diagram

Diagram

Description automatically generated

Scenario:

Name: Melee

Summary: The character activates the melee attack.

Actors: Character

Preconditions: player has been initialized.

Basic sequence:

Step 1: Input melee button

Step 2: Check for player motion and enemy detection

Exceptions:

Step 2.1: Character lunges with hit animation

Step 2.2: Character plays normal melee animation

Step 2.2.1: checks for collision

Step 2.2.2: Plays hit animation if collision

Post conditions: Character returns to previous animation / state.

Priority: 2\*

ID: BH1

\*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

## Data Flow diagram(s) from Level 0 to process description for your feature \_\_\_\_\_\_\_14

Diagram

Description automatically generated

Diagram

Description automatically generated

Process Descriptions

Reload weapon:

IF (“the equipped guns magazine” != full and there are reserve bullets)

THEN Play reload animation() & update gun inventory()

Else ignore input

Shoot:

IF (“the equipped guns magazine” > 0)

THEN spawn bullet in direction of gun

Else play empty gun click sound and call Reload weapon()

## Acceptance Tests \_\_\_\_\_\_\_\_9

Check movement inputs (WASD)

* Initialize the player in a test map that has four walls. Inputs each movement separately and checks each wall for its respective expected output (wall collision)
  + Check for collision up when W (top wall)
  + Check for collision left when A (left wall)
  + Check for collision down when S (bottom wall)
  + Check for collision right when D (right wall)
* Call melee() when enemy is close and directional input matching enemy direction is pushed.
  + - Check for lunge function to be called
    - Check for collision with enemy

## Timeline \_\_\_\_\_\_\_\_\_/10

[Figure out the tasks required to complete your feature]

Example:

### Work items

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Description | Duration (Hours) | Predecessor Task(s) |
| 1 | Sprite creation | 1 | - |
| 2 | Character artwork & animations | 20 | 1 |
| 3 | Movement | 3 | 1 |
| 4 | Melee | 5 | 1, 3 |
| 5 | Animation controller | 3 | 1,2 |
| 6 | Weapon sprite holder creation | 2 | 1 |
| 7 | bullet creation | 4 | 1,6 |
| 8 | Bullet interaction | 6 | 1,6,7 |
| 9 | Weapon inventory system | 5 | 6,7 |
| 10 | Reload animation and inventory implementation | 3 | 1,2,5,6,7,9 |

### Pert diagram

Diagram, engineering drawing

Description automatically generated

### Gantt timeline

