## 1 Introduction

I am in charge of creating the weapons for D.R.E.A.D. This includes different weapons for damage, their projectiles, and a system to keep track of how much ammo a player has.

## Diagram Description automatically generated2 Use case diagram and scenario

### Scenario

**Name:** Kill Enemy

**Summary:** The player is trying to kill the enemy by shooting them with a weapon that must be picked up.

**Actors:** Player Character

**Preconditions:** Game is created and player has started playing the game.

**Basic sequence:**

**Step 1:** Pick up weapon

**Step 2:** Pick up ammunition

**Step 3:** Fire weapon towards enemy

**Step 4:** Ammunition tracker keeps track of how much is ammunition is left

**Exceptions:**

**Step 3:** Health reaches zero before player fires the weapon: player loses the game.

**Post Conditions:** Player has fired a weapon at (and hopefully hit) the enemy.

**Priority:** 1\*

**ID:** JR01

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

## 3 Data Flow Diagrams

In the diagrams below, I will describe the weapons feature I will implement. I will describe the “fire” method in pseudocode below the diagrams.

Diagram

Description automatically generatedDiagram, schematic

Description automatically generatedDiagram

Description automatically generatedDiagram

Description automatically generated

**Process Description**

The fire function will call multiple methods from a different class in order to put all the information together to create a bullet to shoot towards a target. The following is pseudocode showing this.

Fire()

{

bulletDamage <- ammoManager.getBulletDamage()

if(bulletDamage == 0)

do nothing

else

totalDamage <- weaponModifier + bulletDamage

ammoManager.updateTotalBullets(-1)

spawnBullet(totalDamage)

}

## 4 Acceptance Tests

One test I will run will be to test the random weapon generator. A way I can test this would be to run the generator 30 times (or more) and evaluate the results. I would be looking for an even distribution of the weapon types, as well as no collisions when the weapons are placed on a randomly generated map. This can be modeled by sending output to either the console or a file to represent what weapon type was generated and if a collision was encountered.

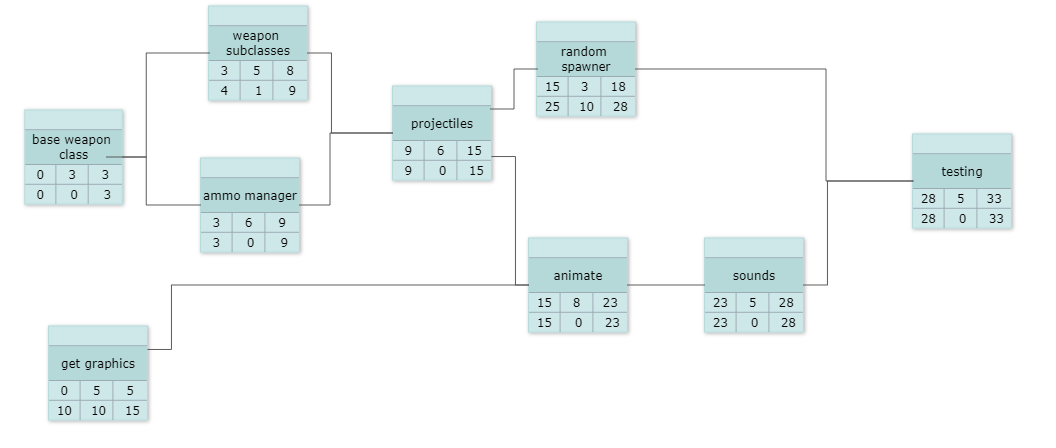
Another test I can run would be to test the projectiles and their interactions. I can generate a hundred bullets sequentially and make sure they are destroyed on impact, or at least moved out of the gameplay. I can also check to make sure the bullets are doing damage appropriately when they collide. At the same time, I will make sure the ammo total on the ammo manager class is counting bullets correctly. This can also be modeled by sending output to a file for the damage dealt when colliding with an object.

## 5 Timeline

### Work Items

|  |  |  |  |
| --- | --- | --- | --- |
| task | | time (hours) | predecessor |
| 1 | base weapon class | 3 | - |
| 2 | weapon subclasses | 5 | 1 |
| 3 | ammo manager class | 6 | 1 |
| 4 | projectile class | 6 | 2,3 |
| 5 | get graphics | 5 | - |
| 6 | animation | 8 | 4,5 |
| 7 | random item spawner | 3 | 4 |
| 8 | integrate sounds | 5 | 6 |
| 9 | testing & integration | 5 | 7,8 |

### Pert Diagram



### 

### Gantt Chart

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 |
| 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  | 2, 3 | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 4, 5 | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 6 |  |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 7, 8 | |  |  |  |