|  |
| --- |
| CMP302: Gameplay Mechanics Development |
| R-nader |
| Project Report |

|  |
| --- |
| LEWIS THOMSON  1601828 |

# Project Summary

The aim for the project was to create a gameplay mechanic which enabled the player(s) to interact with the world using their arsenal of specialised grenade types (R-nades) and their radar scan ability. The core of this design is based off of the systems used in 2009’s Manga Series ‘Btooom!’ (Junya Inoue, 1971). The core mechanics identified for this project were:

* Different R-nade types:
  + Impact Detonation.
  + Timed Detonation.
  + Pressure Detonation.
  + Tripwire Detonation. <Not Implemented>
  + Remote Detonation. <Not Implemented>
* Radar / Sonar Scan <Partial Implementation>
* R-nade swap system <Partial Implementation>

The main mechanics to be developed lies within the R-nades themselves. With each one having different, unique detonation conditions and different explosion sizes / effects.

# Requirements Specification

## Purpose

The gameplay I am aiming to produce is a first-person shooter style system where all firearms are replace with physics enabled grenade-type explosives (named, R-nades). The main objective for this project is to implement a variety of interesting weapon and tracking mechanics.

## Audience

In a real-world scenario, the target audience would be teens/adults for casual or competitive experiences. However for this being used as only a coursework submission, the audience I am targeting will be assessors and examiners who are familiar with the use of Unreal Engine and C-based coding.

## Scope

The project’s scope is already outlined in the brief provided as part of the CMP02 Coursework, thus I will not re-explain what is already written there.

However, my individual submission will focus primarily on combat mechanics with the use of explosive projectiles, as mentioned above in the summary. Aside from the mechanics and ‘weapons’ being implemented, there will also be a ‘testing range’ where it will be possible to demonstrate all of their functionality.

It is worth noting, that it is not a game that I am developing, but a foundation with functional mechanics that can be added to and built upon.

## Overview

The main submission will consist of the ~~5~~ 3 different types of functioning ‘R-nades’ (including basic models and visualisations), Radar / Scan system for gameplay mechanics regarding locating other players (though there will be no multiplayer functionality). There will also be NPCs with extremely basic pathfinding AI in the ‘test range’ to demonstrate weapon functionality. As previously stated, this is aiming to be a foundation for demonstrating the mechanics, rather than a full feature application.

All artistic-based objects in the application can be changed within engine without any prior knowledge of code. Each type of R-nade has been created in C++ through Unreal Engine, with a UE Generated Blueprint for designer use to change and tweak balancing values. All types of R-nades currently effect dynamic physics objects as well as effect enemy characters in terms of removing health.

## Development Environment

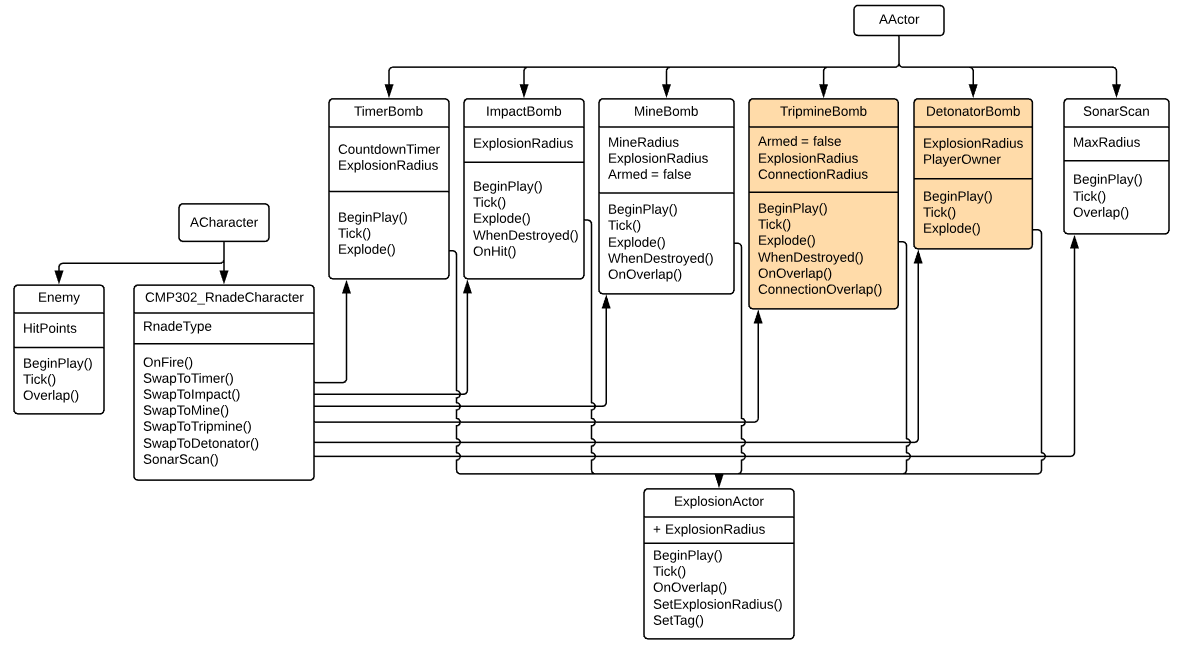
The application was created and intended to be used within Unreal Engine v4.22, however it can be opened in future generation, however it may jeopardise functionality. All code was created in Visual Studio 2017.

All development is being performed by a sole developer, therefore there will be limited -if any- artistic creation for the project, and heavy use of Unreal Engine’s Sample Content packs.

# Design

## UML

Attached is a UML Diagram for the developed system that covers all the C++ classes created. The coloured classes were not implemented in time, but for proof of concept in design have been included.



In this section I will cover each class shown in the UML above, each representing a component of the gameplay system created.

## R-nade Mechanics

As previously stated, there are five different R-nades that have been designed, and only 3 fully implemented at the time of submission. The reason for two being omitted from the submission was due to time and knowledge constraints, however upon a resubmission, the further two would be implemented and expanded upon.

The current list has linked priorities. Some may have multiple priorities, this is for extended features. Priority scales from low to high. The lower the number, the higher the priority.

### Impact Bomb

The impact bomb was designed to do as it sounds. Be thrown, and upon impact with any surface, create an explosion that damages and will knock back. Currently, the impact bomb can be thrown at a standard speed/distance, and will create a constant blast radius of whatever is defined in the blueprint. The constant blast means that no matter where in the sphere you are, you will take damage and –if a physics object- get knocked back.

Priority - 2

### Timer Bomb

The timer bomb is created to be thrown and bounce / roll around for a pre-set amount of time. The default designed time is 10 seconds. This means that after the R-nade leaves the players hand, it will start counting down, upon detonation, it will create a blast radius that is pre-set in the blueprint, again creating a constant blast radius. The timer bomb is designed to withstand other explosives, only to be knocked around, creating a strategic element to their use, and counter-use. As they are cubes, they can be quite unpredictable in the way they interact with the surrounding environment.

Extended use would allow the player to activate the countdown timer while holding the device. Meaning they could throw the device with however much time remaining that they desire. This would also allow the player to sacrifice themselves in an attempt to close distance and secure an elimination.

Priority – 1

Priority – 6 (Extended use)

### Mine Bomb

Rather aptly named, the mine bomb is a puck-shaped explosive that when thrown, will travel and roll until it comes to a complete stop. Once the device lands stationary, it will arm itself, creating a sensor sphere that will only detect players. Note that this device CAN detect the player that has thrown it, so caution must be taken when traversing mined areas. The device itself can be blown and destroyed by any other explosive, meaning players can ‘defuse’ the explosive by alternative methods. The sensor-radius is defined in the blueprints, therefore can be fine-tuned by designers. Once detonated, it will create an explosion that is also pre-defined in the blueprint. It is worth noting, that if the mine is moved in any way, it will trigger its detonation. Meaning if you collide another non-player physics object into it, or move an object that it is placed on, the detonation will occur.

Priority - 3

### Tripmine Bomb

The tripmine bomb functions similarly to classic traps, in that you will need to throw two of them to create an actual case for detonation. The way this cylindrical trap works, is that one device will be thrown, and search for other devices in a blueprint-defined radius. If there Is another object in the area, it will create a link between the two objects, that if any object crosses, both traps will trigger. Ideally, the explosion should be large enough to cover any ground in between the two traps, this creating a well laid-trap. Multiple more devices can connect to create a mesh of traps, however only the latest trap will attempt to connect to ones in its radius.

Priority - 5

### Detonator Bomb

The detonator bomb is similar to a remote explosive, in that once thrown, can be detonated again by the press of a button. These pyramid-shaped bombs can only be detonated by the user, and function similarly to the timer bomb, in that they are effect by physics, thus allowing clever level manipulation. Once caveat with these, is that they can only be detonated if the user is within connection range, which will be pre-defined in the blueprints. Once detonated, will create an explosion like every other device.

Priority - 7

## Other Mechanics

There are other mechanics that are planned, however not fully implemented. These will be listed here, though some are subject to change in functionality.

### Sonar Scan

The sonar scan functions as a locating method for players to find enemies. Once the player activates their sonar, it will send out a pulse up to a maximum range (which again, is defined in the blueprint for editing). If an enemy is within the range, then the player will be alerted by an auditory tone, as well as gain a visible indication as to where the enemy is through the environment, in the form of a spherical dot. However, if the players scan is met with another players scan, both are cancelled out and will be placed on a cool down period. Both players will be alerted that they have been ‘counter-scanned’, but will receive no indication of where the scan originated. If a player is scanned, they will also be alerted they have BEEN scanned, but again, will not receive any information as to where from.

The scan also does not follow the player once used. The origin of the pulse will be the location the player was at when the ability was activated.

Priority - 5

### Weapon Swapping

It was designed that the player will start with only one type of Rnade, and must collect the other versions and their limited ammunition by eliminating other players, or by finding them laid about the map. Currently for demoing purposes, the player may swap weapons at any time and has an unlimited supply of them.

Extended use implies the rules of having limited Rnade types and spawning with only one type.

Priority – 4

Priority - 8 (Extended Use)

### Melee

There is also a melee function planned for when / if the player runs out of Rnades, meaning they can still attempt to take out other players, however less efficiently.

Priority – 10

# Development

Here I will discuss my development methods, discussing how each function works and how they could be further developed and carried forward.