# Technical design document

## <NameOfGame>

### Game particulars

#### Platform:

The prototype will be built to be and tested on Pc for the purpose of releasing to PC through Stream.

With no intentions of portioning to other platforms.

While the Games systems will be designed around the use of both the mouse and keyboard.

#### Genre:

Shooter tower defence.

Playing as a tank, defending off waves of other enemy tanks from your position.

#### Target audience:

### Document particulars

#### Scripting language:

The use of C# for this prototype is to work within the Unity game engine environment, with the intention of having lower-level scripts in visual studios using Unity’s functions and classes, with the process of making a functional prototype faster, and less worry of it bringing the most optimal program.

#### Naming conventions:

Variables - camelCase

Functions - camelCase

Classes - PascalCase

Assets - PascalCase

### Evaluation of Systems

#### Similar systems:

Player controller script versus enemy controller script. While both scripts relate to the control of the tanks within the game, the player will only be able to play as and control one tank of their own, this script for the player would need to include mouse and keyboard input where as the enemy control script would work off of the player’s movement, while not needing everything that the player controller script as in it, such as drawing a line to where the shell will land while zooming in.

#### Mechanics:

Player controller mechanics include:

Forward and backward movement, rotating on the spot for a change of direction, moving camera with turret using a mouse allowing for aiming at targets, zooming in on targets allowing for an understanding of shell trajectory, change of barrel elevation, shooting.

#### Code requirements:

Code requirements for the player controller include:

Applying a change in position alone one axis for the tank’s forward and backward movement, moving faster when going forward and slower when backward.

Applying a change in rotation along the vertical axis going left and right for the tank’s direction.

The camera and the turret move in sync to allow for aiming at targets using the mouse inputs, only along the horizontal movement and no vertical movement.

While zooming in, draw a line where the shell will flow, changing the line as the barrel elevation changes, and the direction of the turret.

On input, change the barrel’s height by rotating at the base of the barrel.

On input, clone a shell asset and apply and speed modifier to its position and shoot from the end of the barrel where the barrel was aiming as of input.

#### Pseudocode:

**Task: Shell coalition.**

**Problems:**

1. Moving shell once in action
2. Detecting if the shell has hit something
   1. If it does remove it from the game

**Solution:**

1. once the shell is spawned in the game, set its direction of the shell to the barrel’s direction.
   1. Set the shells position to the end of the barrel
   2. Set the shells speed to chosen number
   3. With the set speed number apply it with a change in time to gain a change in forward position
2. Setting a true or false statement to see if the shell has hit something or not
   1. IF the shell has hit something check to see if it’s gone though or not
      1. IF the shell has gone though an object remove the shell from the game

#### Flowchart:

A diagram of a game

Description automatically generated with low confidence

#### start