Number Shooting Game

Contents

[Analysis 3](#_Toc149834396)

[Introduction 3](#_Toc149834397)

[Features 3](#_Toc149834398)

[Calculations required 3](#_Toc149834399)

[Design 4](#_Toc149834400)

[Top-down diagram 4](#_Toc149834401)

[Class Diagram 5](#_Toc149834402)

[Methods 6](#_Toc149834403)

[Mock up screen 7](#_Toc149834404)

# Analysis

## Introduction

The game I am creating is a very small game where the player will shoot down circles with varying movement patterns by firing projectile bullets.

## Features

The player will have a crosshair that represents where the player is aiming along with a dotted line from the crosshair to a gun that will be at the bottom middle of the screen.

When the player clicks the mouse, a bullet will travel from the gun, in the direction of the cross hair until it leaves the screen or hits a circle.

Each circle will have a number in it which represents the movement pattern of the circles.

Some circles will move straight, some will move in an arch, some will move in circles, and some will have an irregular pattern.

When a circle is hit, it will disappear and be added to the score.

Each circle will have to spawn in within a suitable range and travel in a suitable direction to give the player time to react.

Circles must spawn continuously but not too frequently so as to swamp the player.

The player will lose a life when a circle spawns in and leaves the screen.

The game will end when the player loses all lives.

The player will have a limited number of bullets that they can fire before having to reload.

Reloading is done manually and takes time before the player can fire again.

## Calculations required

The location that each circle can spawn must be calculated with the random function with the x and y coordinates being within a certain range.

The direction of travel must also be calculated in a similar way and will be given as a gradient for a cartesian graph.

The bullets and circles that move in a straight line will be modelled with a cartesian equation for a straight line.

The circles that move in an arc will be modelled using the cartesian equation for an anti-clockwise rotation.

The circles that move in circles will combine both circles’ patterns and move in a straight line and make small rotations simultaneously.

The irregular circles will move in a straight line but change direction randomly.

The score must be continuously added to whenever the player shoots a circle.

The number of lives must be subtracted from whenever a circle leaves the screen.

# Design

### Top-down diagram

### Class Diagram

A black screen with white text

Description automatically generated

### Methods

***Class Game***

*Method game\_end:*

When lives are zero, the game will quit.

***Class Sprites***

*Method check\_collision:*

Bullets and circles must constantly be checked to see if they have collided.

*Method check\_inside\_window:*

Bullets and circles must constantly be checked to see if they are within the screen.

*Method die:*

If the conditions have been met where a sprite must disappear, they will be hidden.

*Method move\_in\_circle:*

By multiplying x and y coordinates by the anticlockwise rotation matrices, circles can be made to move in circles.

*Method move\_in\_line:*

By modelling a sprite’s trajectory as a straight line graph, a sprite can be made to move in a straight line.

***Class Circles***

*Method subtract\_life:*

When a circle leaves the screen, a life of the player must be subtracted.

*Method add\_score:*

When a circle collides with a bullet, the player’s score will increase.

***Class Ammo***

*Method reload:*

When the amount of ammo that the player has is less than 3, the player can choose to reload for a delay before they can fire again.

***Class Crosshair***

*Method check\_mouse\_pos:*

The mouse’s position will constantly be checked so that the crosshair’s position can be updated to match it.

### Mock up screen

1

1

1

Gun

Bullet

Bullet

Ammo:

Score: 9

Lives: