WELL PLAYED

Troll Turret

Remote Gameplay Programming Test

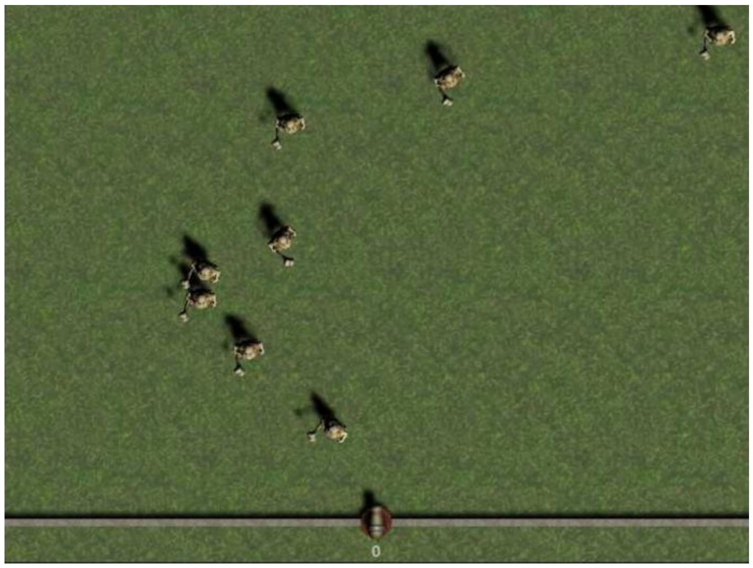
Introduction

In this Unity/C# programming test, you will be implementing the behaviour of a stationary turret with the aim of protecting a village for as long as possible from a horde of attacking trolls.

Once you're set up and ready, we would ask you to spend around three hours on the test.

You are free to use the internet to look up general maths or programming questions, but not for anything specific to the test.

The Problem



As shown in the illustration, the trolls will be moving towards the village wall - each with a different speed and direction.

The turret can rotate and fire cannonballs, which will kill a troll if they hit it.

There is no player controlling the turret - so its behaviour is in your hands.

The aim is to kill as many as possible before a troll gets past the turret.

The Project and Code

You have been provided with a Unity project containing one scene, `Game`. The scene is set up to be viewed in a 4:3 aspect ratio.

The project was created in Unity 2019.3.6f1. If you can't quickly get the project running for any reason, you should attempt to write the code you think will solve the problem anyway. Tell us when you return it, and we will take into account that you weren't able to test it.

Most likely you will only need to edit `Assets/Scripts/Cannon.cs`, and in particular the `Cannon::Update()` function. That function is called once a frame and has access to the list of currently active trolls, so it's where you can implement your chosen algorithm for rotating the turret and firing. A very poor stub implementation is there, with comments marked `TODO`.

The troll list may change between frames, as dead trolls are removed and new trolls are added. However, the Troll component includes an id which you can rely on to be unique and consistent for a given troll.

Your sole responsibility is the behaviour of the turret - you do not need to implement other areas of the hypothetical game.

When rotating the turret, please ensure you respect the turret's maximum rotation speed `Turret::maxRotationSpeed`.

You should feel free to use the Unity/C# libraries if you wish. `Mathf`, `Quaternion` and `Vector3` have useful maths functions. And you have access to useful containers such as `List` and `Dictionary`. All of these can be easily Google'd for documentation.

Good luck!

```
Asset Store assets used:
FREE Stylized PBR Textures Pack - Lumo-Art 3D
Battle Cannon - Zugzug Art
Troll Cannibal - Waytan
```