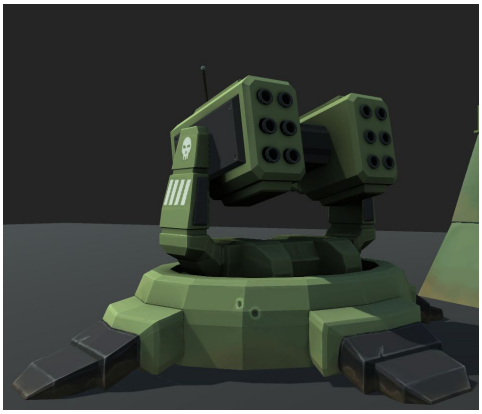
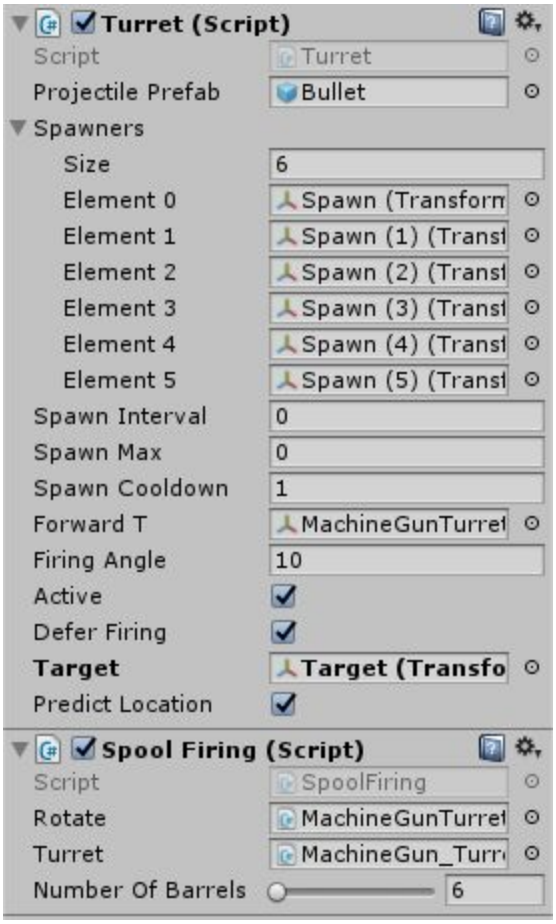
Auto Turret Pro - 2Ginge



Overview

Easy turret pack is a collection of assets and scripts designed to allow you to easily create your own automatic defence turrets for a variety of different game types. With the scripts provided, you will be able to create your own visual assets or make use of those included to have a two part turret working in no time. This turret pack focuses on turrets that have a split targeting system, with a segment that rotates around the y axis, and a separate segment that rotates around the x axis.



The Turret.cs Script:

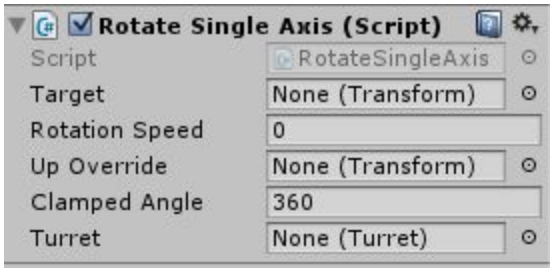
• The ‘Projectile Prefab’ slot determines which object (variable) will be used as the turrets projectile. In this example, the projectile provided is the Bullet prefab. Once this object is spawned, it is independent of the turret in both logic and hierarchy.
• ‘Spawners’ is an array of Transforms that specify the locations for the Projectile Prefab to spawn from. (The spawner forward vector should be pointing in the direction you wish the projectiles to spawn).
• The ‘Spawn Interval’ is the time between each projectile spawn. If there are 6 projectiles, and there is a spawn interval of 1 sec, the time between each projectile spawn will be 1 second.
• The ‘Spawn Max’ is the number of projectiles fired in one salvo or loop of firing. If it is set to 6, only 6 projectiles will spawn each salvo.
• The ‘Spawn Cooldown’ is the delay between salvos.
• ‘Forward T’ refers to the transform of the barrel. This transform’s forward vector should be facing

away from the turret directionally aligned to the way the barrel is facing. Position does not need to be centered on the barrel, but it is recommended that it is nearby for ease of reference.
• ‘Active’ determines if the turret script is active or not.
• Defer Firing means that Another Script will call the FireProjectile() function.
• ‘Target’ is simply the transform of the object that will be targeted by the turret when active. (though this is stored in a vector3 such that prediction can occur and override the ‘Target’ position).
• Predict Location uses a simple algorithm that predicts where the target will be, then overrides the corresponding scripts in order to rotate to the predicted location.

SpoolFiring.cs:

• Rotate is the Rotate.cs component of the Barrel.
• Turret variable is the Turrets… turret.cs script.
• Number of Barrels, such that the rotation can be matched up with the firing. This script is the deferred firing script for the corresponding Turret.cs script.

RotationSingle.cs:

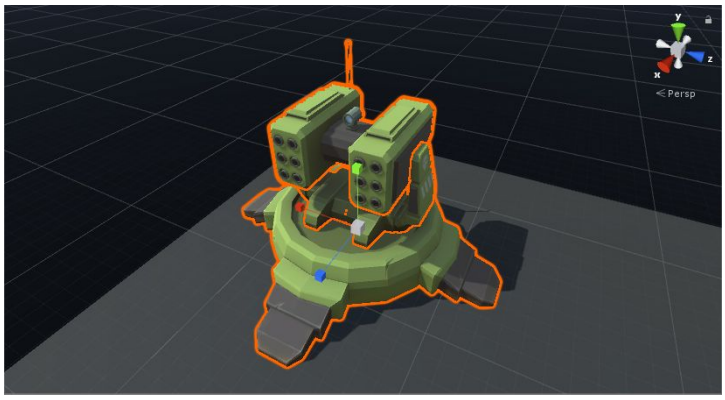


• Target is the Target to rotate towards.
• The rotation speed in degrees per second.
• The Up Override is very important, it is the Transform to use as an 'up' to provide a stable rotation platform. Rotation is always applied around the Y axis so in order to rotate on the X supply an 'Up Override' transform that is rotated by 270 degrees on the Z axis so in effect the X axis is pointing down, the Y axis horizontal and the Blue (Z) axis is pointing forward as usual.
• The Clamped Angle allows the ability to clamp the Hyperplane (from 3D rotation to 2D rotation). In effect this is applied from the FORWARD (blue) of the Up Override), Uniformly in each direction.

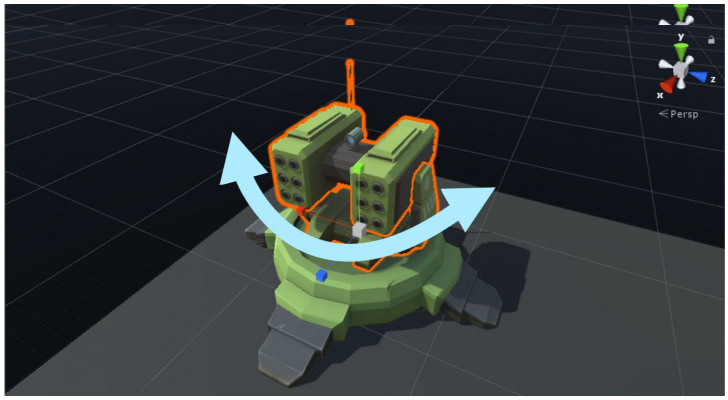
GuidedMissile.cs:

• Explosive Prefab contains the GameObject to spawn upon destruction of the missile.
• Explosion itself is a reference to the particle system. Note: this does not need to be a reference to the instantiated system.
• Trail is a container of trailing particle effects such that once the missile has ‘despawned’ the trails linger until they have completed their duration.

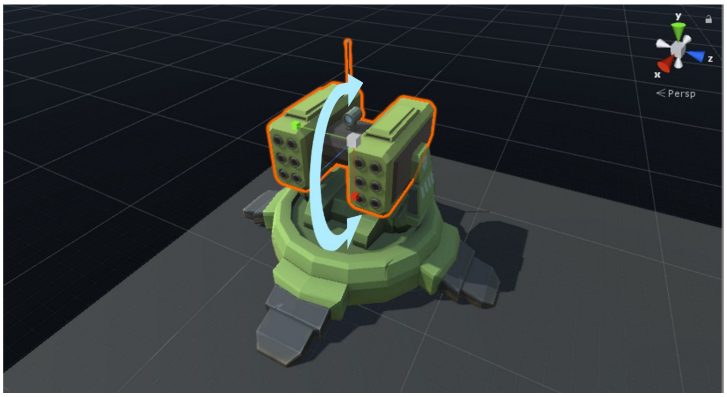
How to set up a turret using Auto Turret Pro



1. A parent object which can have the Turret.cs Script itself and any of its auxiliary scripts (such as deferred firing) attached. In this case the base can serve as the parent object, but an empty object titled ‘HellfireTurret’ will suffice.



2. As a Child the first part of the turret (in this case, the ‘base’) the arms holding the launchers are used to control the horizontal rotation of the turret. This object has the RotateSingleAxis.cs script attached. The Up Override is chosen from a parent. Itss own Transform can be used BUT the clamp will not work as the angle difference is always 0.



3. For the Vertical rotation, let's create two transforms (empty game objects) as a child of the first transform (see step 2). Both rotated 270 degrees about the Z axis, one is used as the UpOverride variable for the other transform with the RotateSingleAxis.cs script on it. This will clamp the turret rotation at that point.

4. Once the previous steps are complete create spawn points for the projectiles where you see fit (for an example, the red dots in the image below represent where the spawner objects are placed).

If there are any issues, refer to the two templates we have (Machine Gun, tracks and defers the firing and has multiple rotating parts and the Hellfire Turret which does not defer its firing).