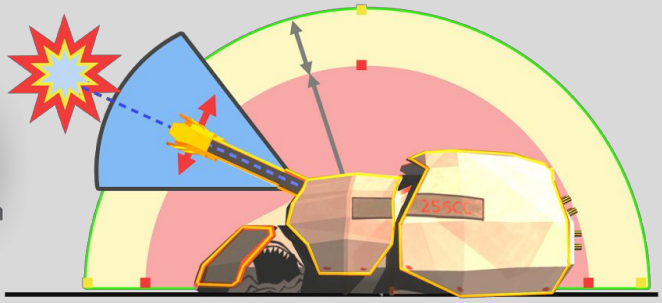
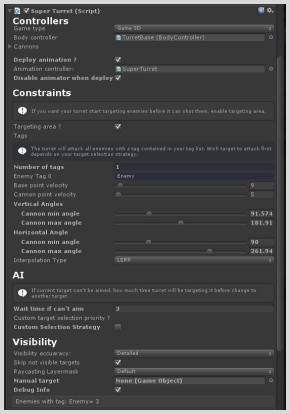


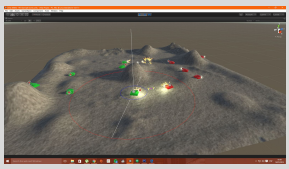
v2.0

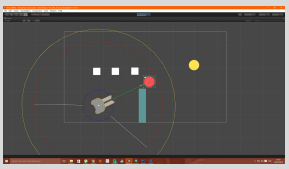
Description

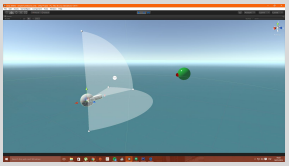
SuperTurrets is a component made for Unity3D like a complete aiming system. This component is not only limited to turret rotation, it's also includes AI with a FSM to select the proper objective. It has been developed with flexibility in mind so you can use it from 2D games to complex 3D games. Following the Unity philosophy SuperTurrets comes with a custom editor and custom handlers so you can configure it easily.

To take a better look of what SuperTurrets can do, see the features list, the video and images.









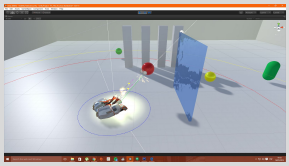


Table of contents

Features Overview

v1.0

- Two-parts rig working in any orientations with configurable speed and angles.
- Three optional areas: Targeting, attacking, too-near.
- Support from 0 to N cannons.
- Fold/Unfold animations before start attacking.
- Dynamic (non animation) recoil foreach cannon.
- Custom inspector and helpers that make your turrets configuration much more easy and fun.
- Enemy/friend system based on tags.
- Visibility tests so turrets will not shot through friends, walls, etc.
- Support for custom priority selection algorithms. Which one you are going to shot first, the strongest or the fastest?.
- Smart AI, turret will switch between enemies in an optimal way. There are public parameters adapt this behavior to your needs.
- Made with performance in mind, you can choose from different visibility tests and add layer masks. You can configure a very simple turret ready for mobile.
- Complete, clean, free of warnings and heavily commented source code included.
- Complete demo project included with an example of enemies, shoots, bullets, particles,scenes, etc.
- An awesome turret 3D model made by a programmer (Me).
- Works on iOS and Unity free version.

v2.0

- 2D Mode! Now you can use all the features in your 2D game with sprites.
- Added vertical angle to clamp turret movement.
- 3D handlers added to configure vertical and horizontal angle limits.
- Fold/Unfold animations now works with Mecanim. No more legacy animations.
- Updated custom inspector. Now you can edit multiple turrets at the same time and everything is better organized.
- Now a SimpleTurret is included if you want something to aim another thing without worrying about AI stuff.
- Improved debug info/gizmos.
- Improved coded architecture (ITurret interface added).
- Custom strategy policies now are scriptable objects.
- Added layer masks for visibility tests.

v2.0 Isn't compatible with previous SuperTurret versions. If you decide to update this component, you will have to configure all your turrets again and fix errors in code.

What It's not

SuperTurrets is designed to be easily integrated in your project so this component offers neither a solution for projectile or bullets logic nor an enemy system. Although SuperTurrets is related with bullets, enemies, explosions and that kind of awesome stuff included in the example project, SuperTurrets has little dependencies and need to know very little about your current project.

Contact

contact@optimizedguy.com

Online Video

If you want click in the following url to see a features overview.

Video

Before you start

Before you start you need to know some prerequisites that your turret model must accomplish.

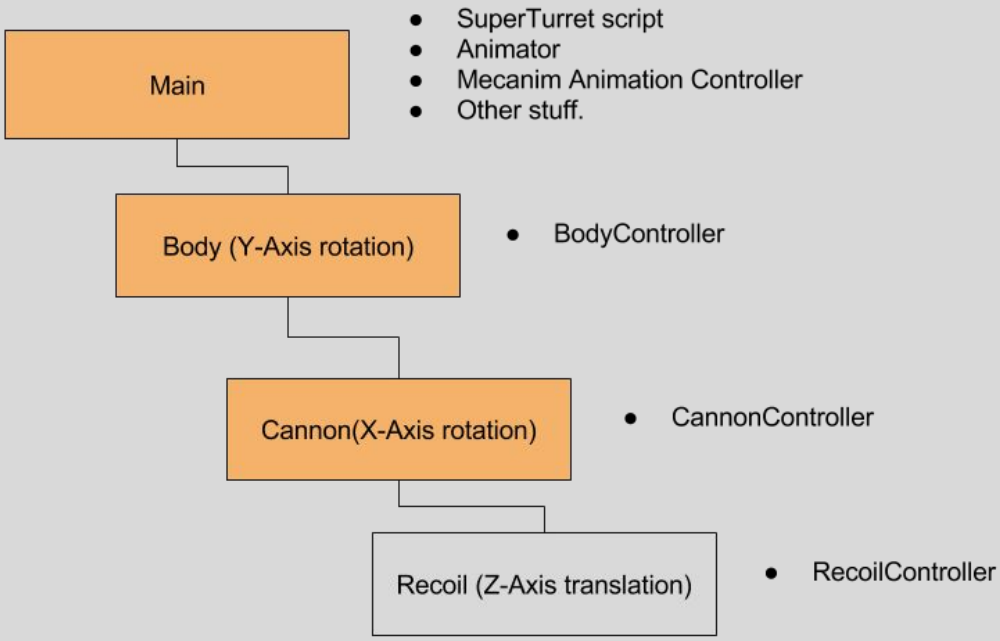
Z-Axis

The Z axis of your turret must be aligned with Unity Z axis in order the maths can work.

In other words, when you drag your turret model to the scene and the transform rotation is (0,0,0), if the Z axis of the gizmo (don't forget to be in local mode) is aligned with the world Z axis, everything would be fine.

Hierachy

The turrets have two moving parts plus one optional moving part if you want to configure the recoil for your cannons. To get this working you will need the following hierarchy:



- Recoil is optional.
- Don't forget you can have N cannons.
- You need an anchor at the end of each cannon to instantiate bullets and FXs. You can add it in your models or add it later in your prefab creating empty GameObjects.

This is the minimum hierarchy. You can have as many Transforms as you want always you have at least 3 Transforms for Main, Body and Cannon.

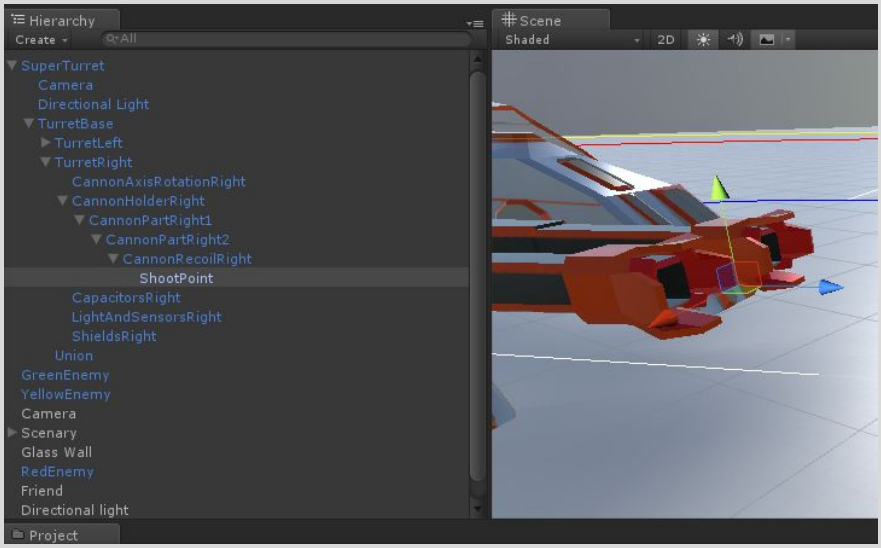
1 - Creating a turret from scratch (3D)

I strongly encourage you to see this video as tutorial to create your turret from scratch:

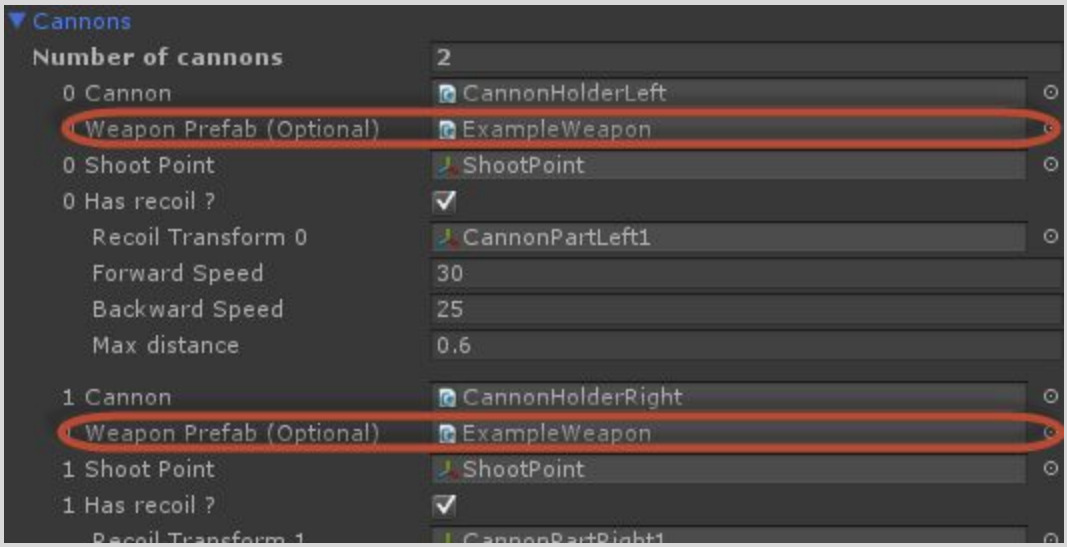
Step by Step Tutorial

1. Add **SuperTurretMainScript** to your turret. Some errors will be shown to guide through the next steps.
2. Now you have to add the **BodyController** to the turret Transform that you want to rotate through the Y-Axis.

3. You have to do the same with the **CannonController**, you have to add it to each cannon in your turret. This cannons will rotate in the X-Axis.
4. We need a place to instantiate the bullets and all the FXs related with shots, for this purpose, we will use a Transform anchor in the end of each cannon as **shoot point**. If you don't have any anchor in your original 3D model, add the anchors in your hierarchy and apply your turret as prefab.

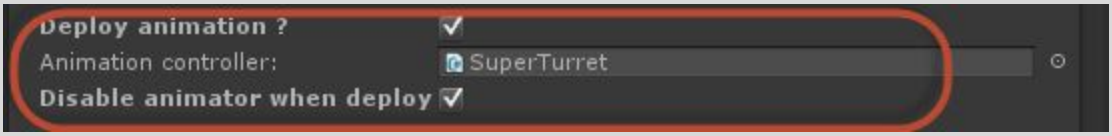


5. Return to your SuperTurret component and start to set the **BodyController** and the **CannonController**. In this point your turret should be able to aim to other targets.
6. To make things more interesting we are going to assign the **weapons** to our turret. You will need a prefab with your weapon script and set it in the Weapon Prefab field in each cannon. There is more info in this document about how to make your weapons compatible with SuperTurrets system.



7. Now set the shoot points in each cannon. At this point your turret should be able to shoot.
8. To configure your cannon **recoil**, you need to assign the Transform from your turret hierarchy that will move backwards and forward. You need to configure the recoil speed and the max recoil distance (a gizmo will help you to it).
9. If you want your turret to do a fold/unfold animation before it starts shooting things, we need to configure the animation. Add the **MecanimAnimationController** to your

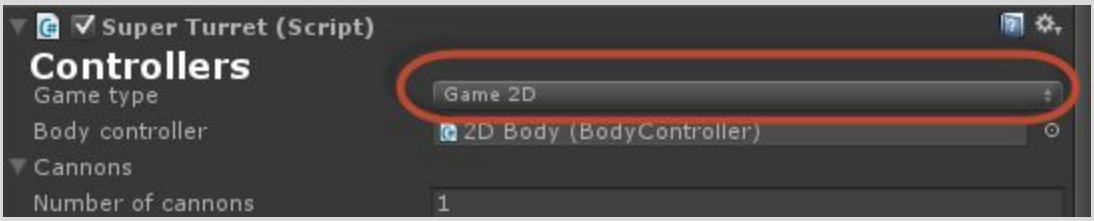
turret and configure each field. Don't forget to assign the MecanimAnimationController to your main SuperTurret script.



10. Configure the rest of fields to get the proper behaviour for your turret as aiming speed, angle clamps and AI parameters. Continue reading to know more about each configurable parameter.

1 - Creating a turret (2D)

Configure a 2D turret is easier because you don't need the **CannonControllers** and you only have one angle clamp (horizontal angle).
You can follow the same steps as in the guide above and the video, the only difference is that you have to set the **Game Type** as 2D.

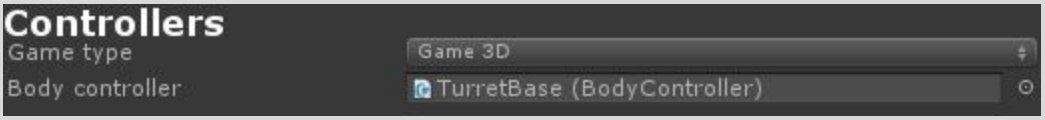


If you have fold/unfold animations for your 2d turret, you can reutilize the 3D animatior controller using an animation override component and assigning the animations clips for your 2d turret. Animator Override

2 - Advanced SuperTurrets setup

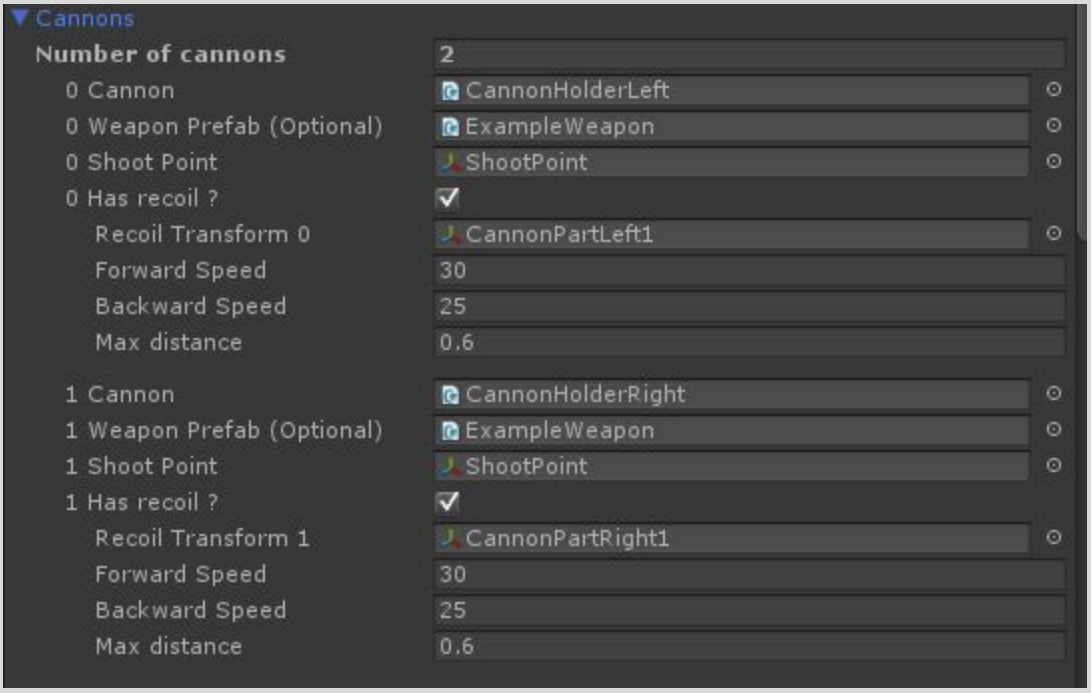
In the next section i will describe in detail SuperTurrets inspector so you can get your desired aiming behavior.

2.1 - Body Controller



This component is the base of your turret. This base only rotates in de local Y axis, like the superior part of a tank, or the torso of a mech. This base usually is the parent of the rest of the turret parts.

2.2 - Cannons



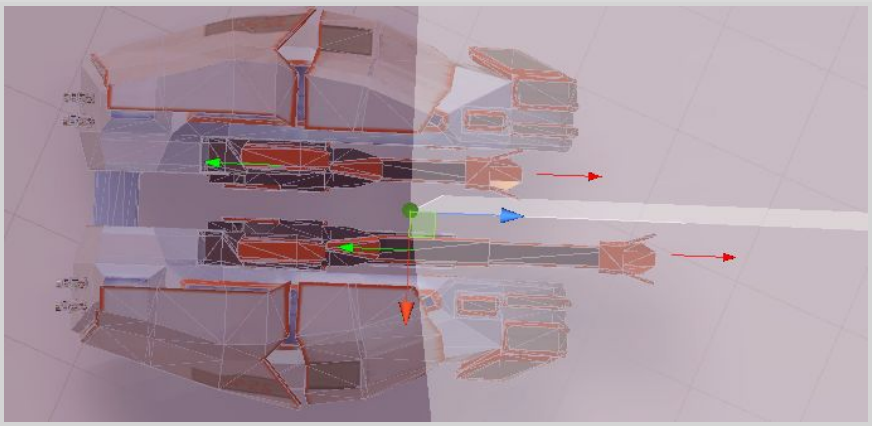
You can have from 1 to N cannons in your turrets. All cannons rotate in the local Z-Axis, combined with the rotation in local Y-Axis of the BodyController, you have a turret that aim 360º in Y and X axis.
All cannons need a CannonController component and it has the following fields:

1. The reference to the CannonController attached in your cannon.
2. The WeaponPrefab if you need weapons.
3. The shoot point. We use this point to launch rays for visibility tests and weapons use it to know where to instantiate bullets and fxs.

2.2 - Recoil Controller

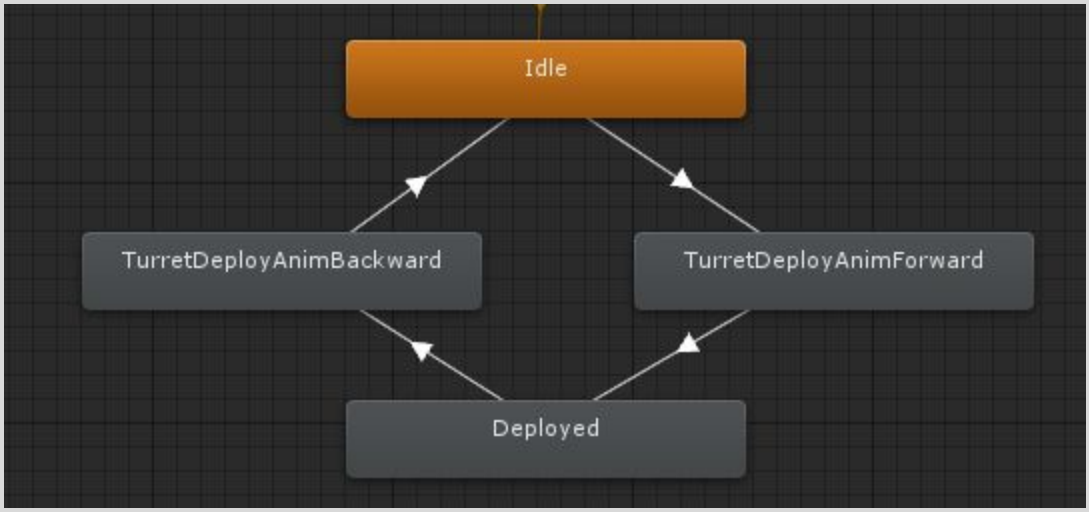
You need to assign the Transform you want to move as recoil when the turret shoots.

The RecoilController will be added automatically in Awake() so you don't need to worry about adding the script manually.
There is a Gizmo that will help you to configure the max distance your recoil can move backwards.

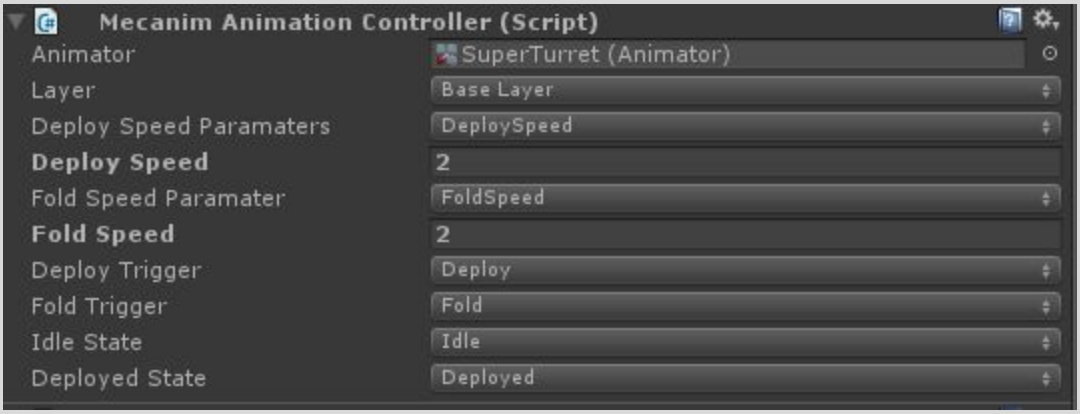


2.3 - Fold/Unfold animations

In order to have your fold/unfold animations working you will need this states in your Animator:



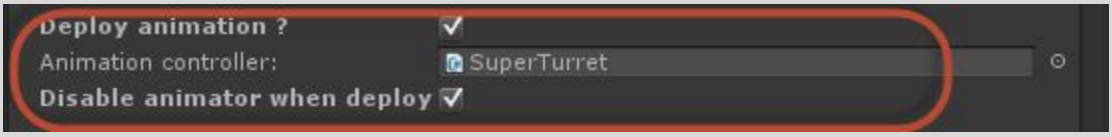
Then you will need to configure your MecanimAnimationController.



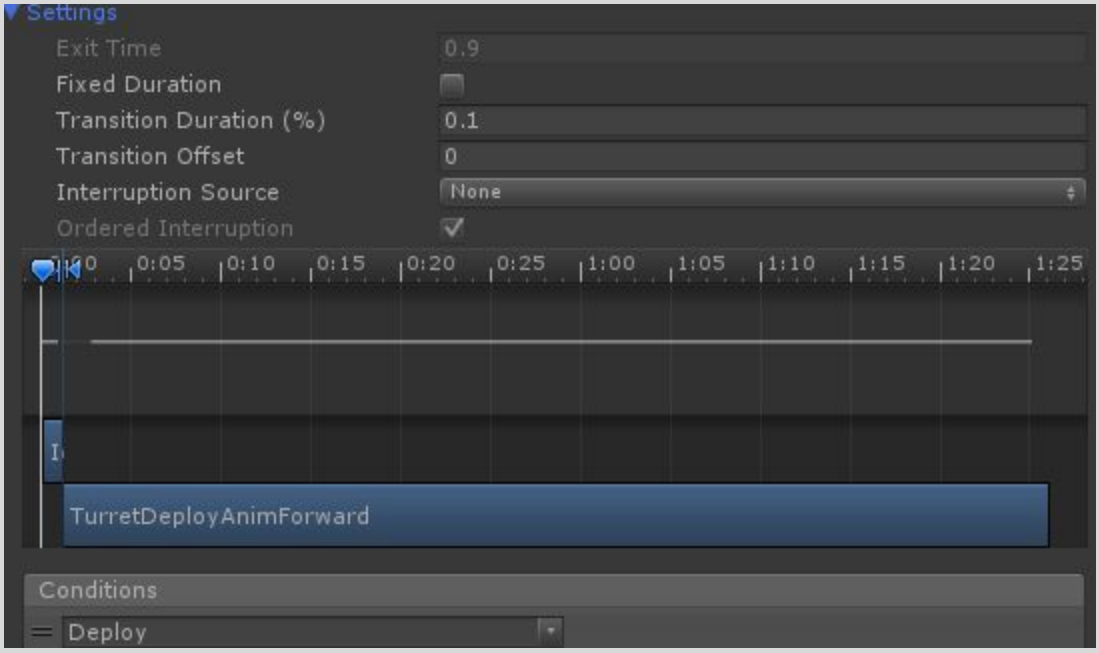
- Needed parameters in your animator: Deploy Speed, Fold Speed.
- Needed states in your animator: Deploy,Fold,Idle,Deployed Idle.
- Needed triggers in your animator: Deploy, Fold.

Because there is a custom inspector to guide you through the configuration, if you want you can change the name of states and parameters.

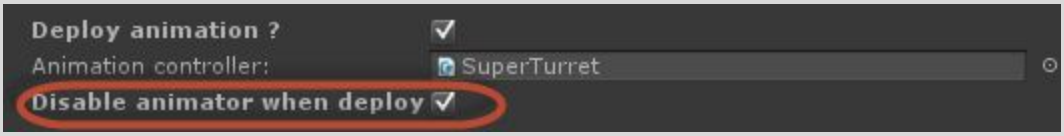
After setting up the mecanim animation controller component, don't forget to assign this component to the proper SuperTurret field.



Note that your turret won't start shooting until it has unfolded completely. So be careful with animations length. It is also advisable to remove blending times between states.

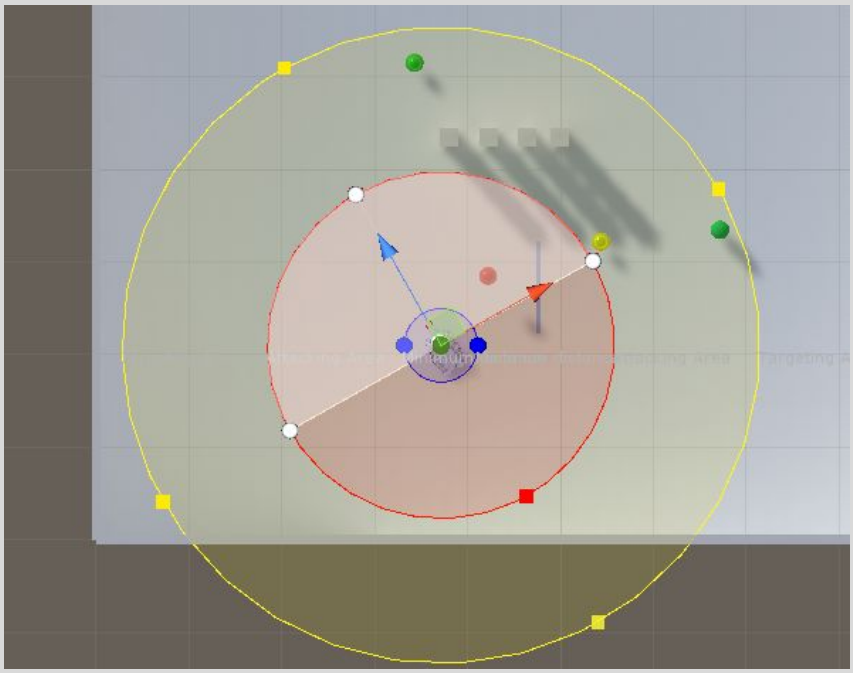


When the turret it's deployed, the Animator is disabled. This is needed to avoid conflicts between controllers Update() methods and animation rotation. If you don't want this behaviour because you have an idle animation when the turret is deployed, you can change it unchecking 'Disable animator when deployed'.



But be careful, because some problems may arise related with transform positions.

2.4 - Targeting Areas



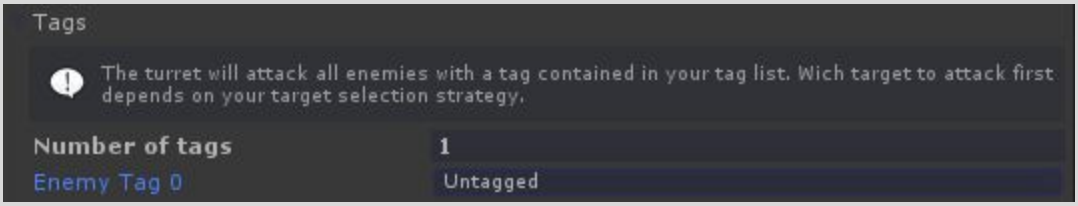
Your turret can have three targeting areas.

1. **Targeting Area:** This area is optional. If a target enter in this area, the turret will unfold and start aiming the target without shooting.
2. **Attacking Area:** If the target is in this area, the turret will shoot to it.
3. **Too-Near Area:** If the target enters in this area, the turret will stop aiming at it.

You can change the size of this areas using the scene view handlers.

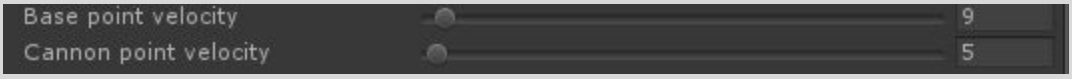
In order to a target to be detected by the turret, they need a Rigidbody component attached. If you aren't using physics in your enemies, set Is Kinematics toggle as true. Rigidbodies are needed because the turret uses default unity trigger events to detected if an enemy enters or exit from an area.

2.5 - Tags



You can set turret enemies adding enemy´s tag to this list. The rest of GameObjects are ignored by the turret. If the number of tags is 0, the turret will shoot to everything.
The turret will stop shooting if something with an invalid tag is between the turret and the current turret target. In other words, the turret will stop shooting to avoid friendly fire or shooting to walls.

2.6 - Aiming speeds and interpolation types



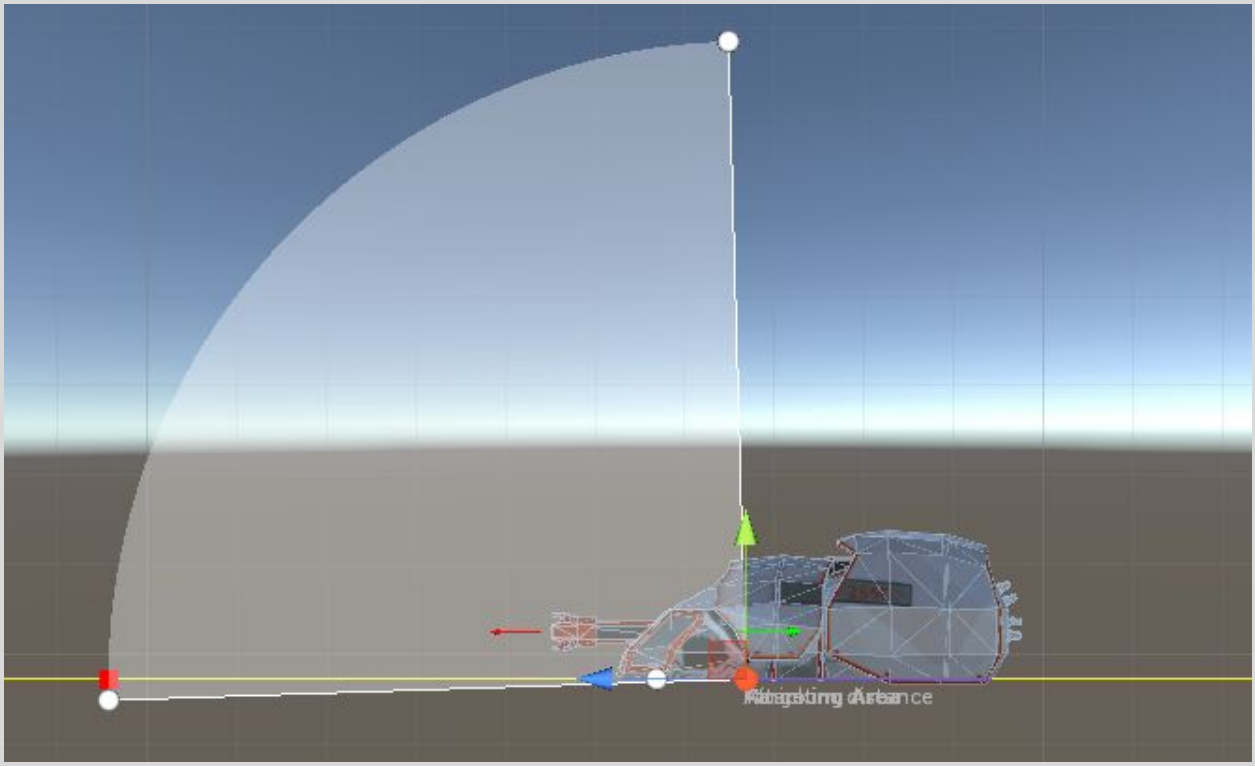
You can control how fast base and cannons aim to the current target.
Also you can control the type of interpolation when the turret rotates toward the target.

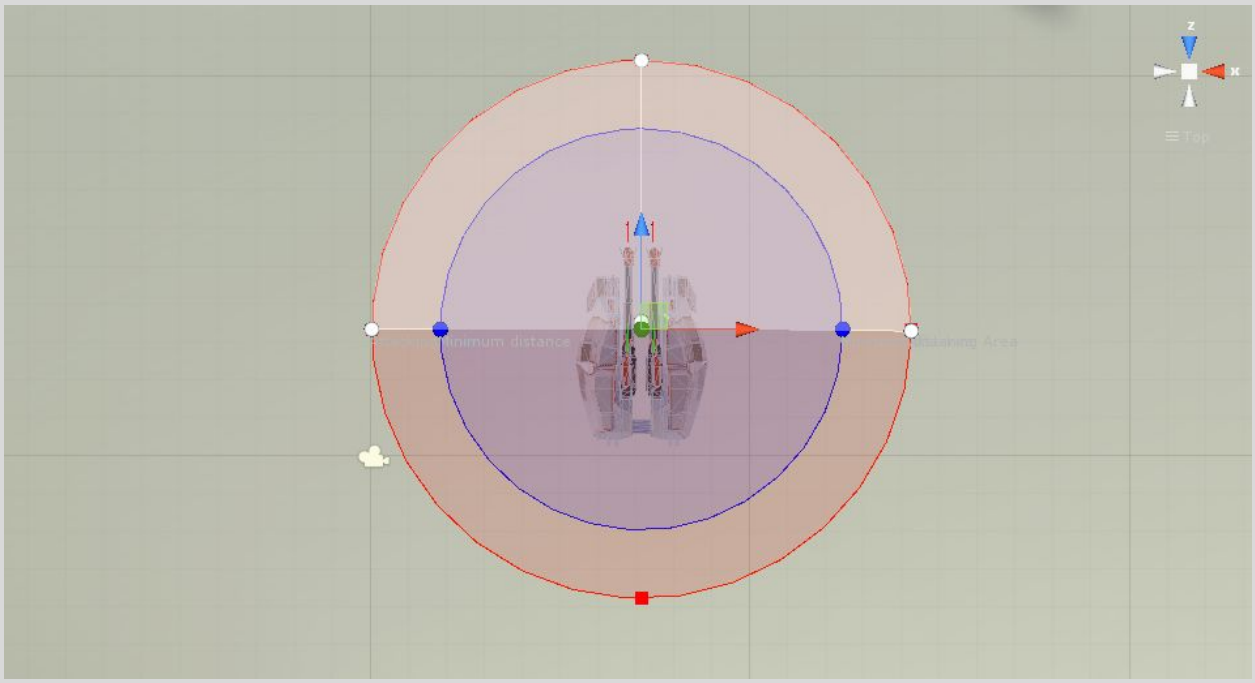
1. LINEAR
2. LERP
3. SLERP

2.7 - Angle clamps

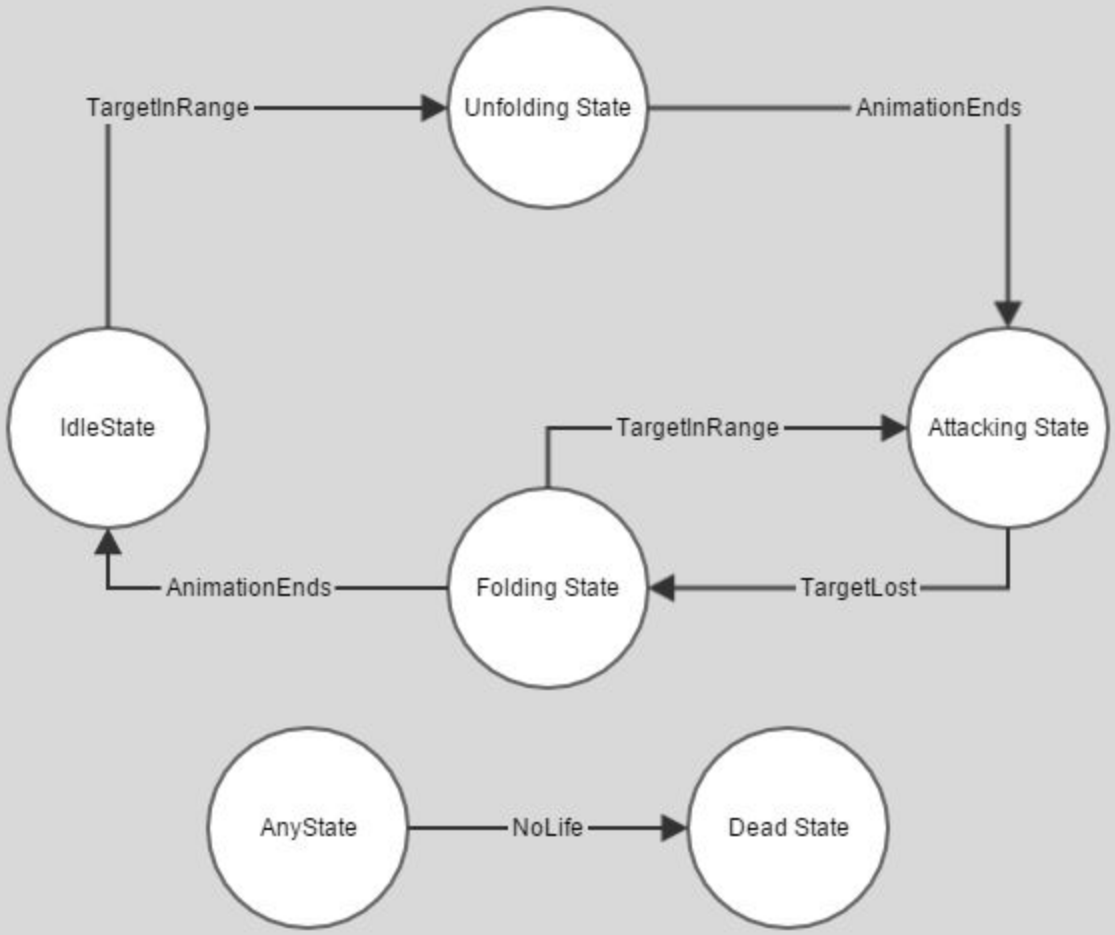
Angle clamps limit your turret rotation movement and also provides info to the AI to ignore enemies outside valid angles.

You can modify this angles directly in the inspector or using handlers in the scene view. Note that the min angle can't never be major than the max angle and vice versa.



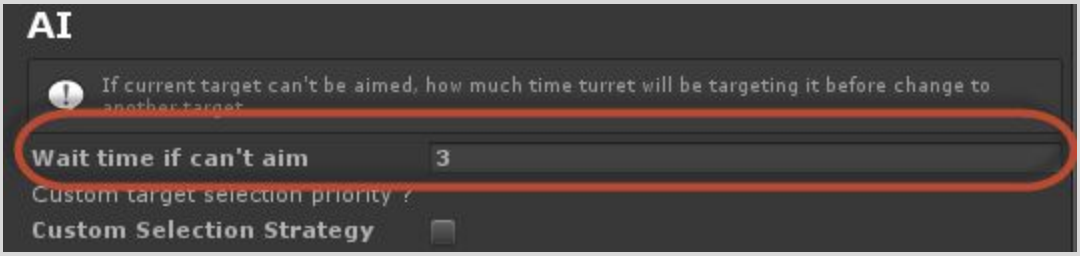


3 - AI



- **IdleState:** While turret is in this state, all controllers are disabled and is waiting to some enemy enters in one of the areas (Targeting or Attacking). The turret will accept this target only if it met all turret conditions as tags,valid angles, isn't occluded by an obstacle, etc.
- **Unfolding State:** When the turret acquires a target, if we have unfold/fold animation, we will remain in this state until the animation is finished.
- **Attacking State**: While the turret is in this state, it will ask to the weapons if they are ready to shoot. While we are in this state the turret will be continuously checking if the current target met all conditions. The turret can loses its target in case it was killed or gone out of range.
- **Folding State:** The turret will return to it's original orientation and then it will start the fold animation. If while it is returning to its original orientation, a new target is found, the turret will automatically return to AttackingState. In case there isn't any new targets available, we go to idle state again.

3.1 - Wait time

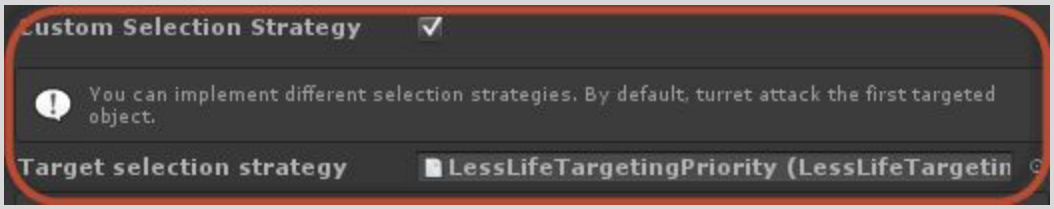


This time is used to know when the turret must change to another target or to idle if the current target becomes unavailable.

Imagine the following scenario where the turret is aiming to its current target and the target moves behind a wall.

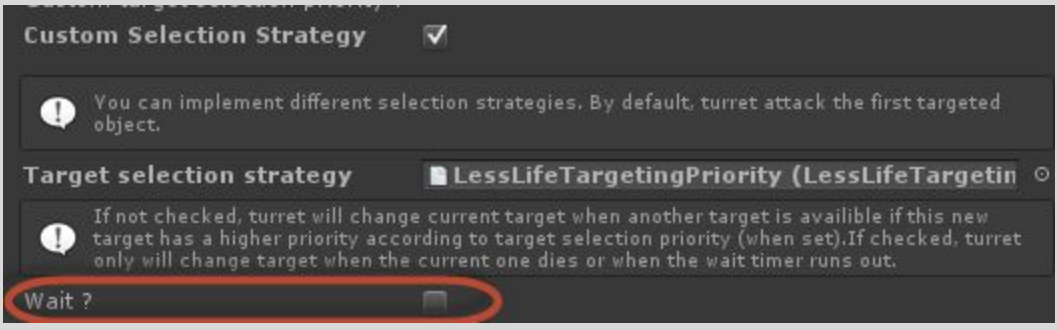
- If the time is set to 0, the turret will go immediately to Folding State.
- If the time is set to 3, the turret will continue aiming the target during 3 seconds. After the time runs out, the turret will change to Folding State.

3.2 - Custom selection strategies

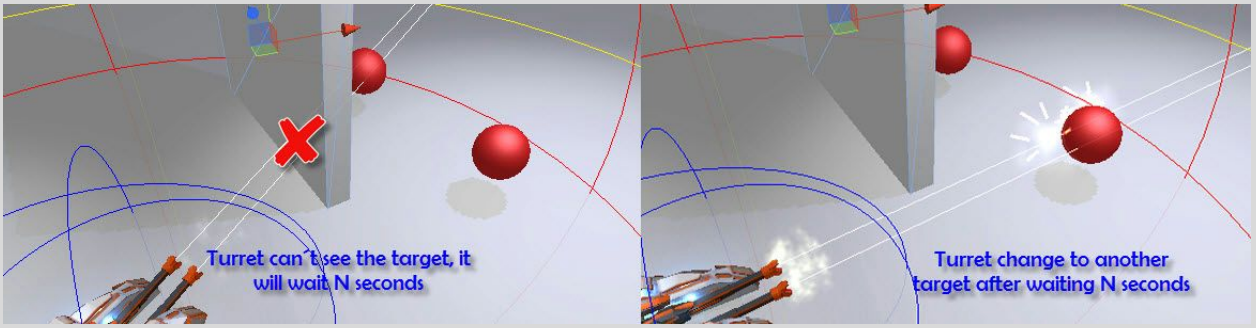


By default the turret will aim to the first target which enters in one of the areas (Targeting or Attacking), but this behaviour can be changed with custom selection strategies.
To know how to define your custom selection strategy, you must read Custom Selection Strategies in the scripting section of this document.

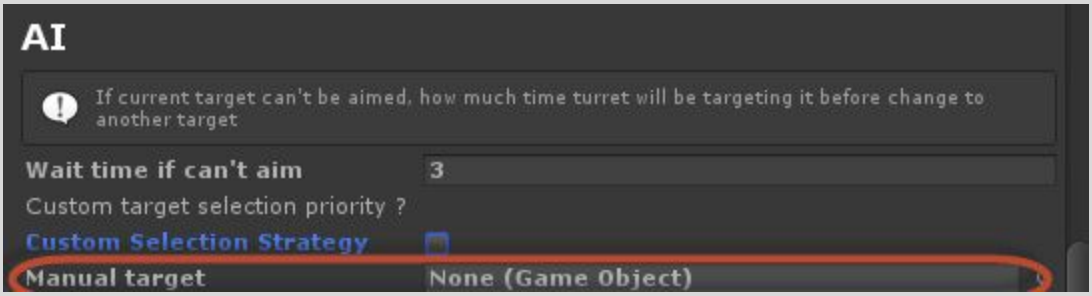
If you have assigned a selection strategy, you will have the additional toggle 'Wait?'.



- If Wait? is disabled, the turret will change to another target in the moment it is available if this new target has a higher priority in the assigned targeting strategy than the current target.
- If Wait? is enabled, the turret will continue shooting to the current target until it is unavailable.

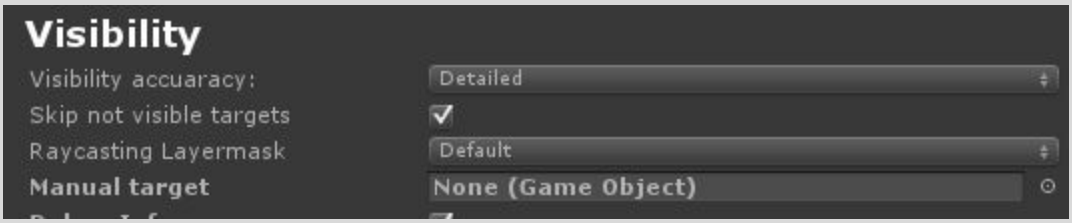


3.3 - Manual target



If you set a GameObject as manual target, the turret will aim and shoot to this target. You can set it by script. This can be useful in RTS games where the turret start attacking automatically at an enemy but then you click in another target.

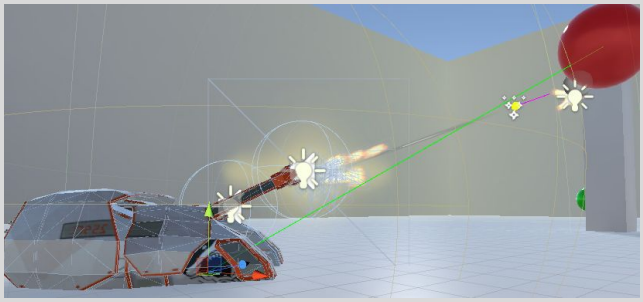
4 - Visibility



You can choose between three visibility modes:

1. **None**: No visibility test is done. It is the fastest mode because no raycasting it's done at all, on the other hand, the turrets will aim and shoot to the target no matter what is in the middle.

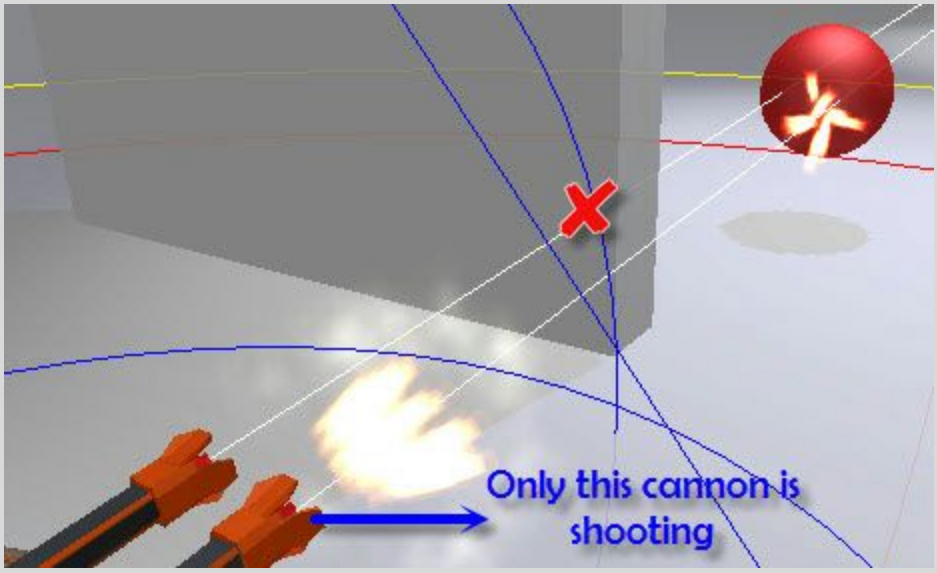
2. **Simple**: A single ray is launched to test if the target is visible.



The green line (only visible if 'Debug Info' is checked) represents the launched ray. If

the target becomes unavailable, the line will turn red.

3. **Detailed**: A ray is launched in each cannon. It is the most expensive method and only the cannons with clear view of the target will shoot.



Because we use Raycasting for visibility tests, all targets need colliders.

4.1 Skip Not Visible Toggle

- **True:** turret will ignore not visible targets when wake up.
- **False:** turret will point to a non visible enemy. This can be useful for turrets that see through glass like in Portal, the turret will aim the target but it won´t shoot until the target becomes visible. If another enemy enter in attack area, the turret will ignore to the current-non-visible one and will aim to the new target.

5 - SuperTurrets Integration in your project

Adding SuperTurrets to your project is very easy, you can use the example project like a start point but if not and you have some knowledges about programming, i will try to guide you with a few things you need to know.

5.1 Enemies

Enemies must have a collider to work with turrets raycasting and a rigidbody to be detected by turret´s radars. Do not forget to mark "Is Kinematic" as true if your enemies does not need physics behavior to avoid losing performance. In the project, you can see an script called **"AbstractEnemy.cs"**. You can inherit from this script and the only important method

here is **IsDead()** but inherting from this class is completely optional. Only if needed if your enemies aren't destroyed when they are killed.

5.2 Weapons

Your weapons must inherit from **"AbstractWeapon.cs"** to be used with SuperTurrets.

- The shootPoint var will be set automatically and is used to launch rays and if you want, you can use this anchor to instantiate bullets.
- WeaponReady property must be set to true when the weapon is ready to shoot. For example, if your weapons need to reload, while reloading, WeaponReady will be false. Always this property is true, the turret will shoot.
- public abstract void **Shoot(GameObject target, Vector3 _direction, Vector3 _localHitOffset = default(Vector3))** is the method called by the turret when the target is available and weapons are ready.
  The _localHitOffset var is the local offset in the target where the ray hit the target. If you have following bullets and you have more than one cannon, you don't want all your bullets going to the center of the target. Using this vector you can know the offset from target's center where bullets must go.
- public abstract void **Shoot(Vector3 _position, Vector3 _direction, Vector3 _localHitOffset = default(Vector3))** is used when you shoot to a point instead of a target. Useful if you want to control your turrets with your own AI.

If you have fast moving objects you can have problems with bullets hitting the enemies. There are multiple solutions for this, like instant shooting or like is done in the example project, using following bullets.

5.3 Custom target strategies

To make your own selection strategy you must create a new C# class that inherits from **"AbstractTargetingPriority.cs"** and implement the method **List OrderTargetsByPriority(List availibleTargets)**. You can take a look to the two strategies that comes with example project, but what you need to do in this method is a classic reordering based on the criteria you want, and order targets from high to low priorities. Turret will search for the higher priority target each time it becomes idle or a new target enters in the attack area.

6 - SuperTurrets Scripting.

6.1 - ITurret

This interface provides info to the controllers. All methods are related with turret movement and if you want to implement your own turret but reutilize BodyController, CannonController and RecoilController, you must implement this class.

The method SetShootingPointIgnoringAllConstraints is designed to have a functionality where everything is ignored and the turret will aim to the given point. This can be useful in scripted sequences where you don't want any logic guiding your turrets.
Remember that targeting areas, targets, tags, visibility tests, etc will all be ignored when this method was used.

If you want to aim to something but following your turret limits, use the Manual Target field.

6.2 - SimpleTurret

It’s the simplest implementation of ITurret. This turret doesn’t contains any AI and it doesn’t know nothing about weapons or enemies. It is designed to aim to a given point.

6.3 - SuperTurret

Much more complex than SimpleTurret it has all the parameters reviewed in this document.