Welcome

The Fire Chicken Games Combat module will be available for purchase on the Unity Asset Store in mid-April 2020.

Getting Started

The **Combat** module builds on the **Game Creator Shooter** module to add enhanced, game-ready, combat features:

- Proximity-based weapon targeting.
- Target indicator.
- Target changed event detection.
- Homing projectile.

Dependencies

Combat is an extension for **Game Creator** and the **Shooter** module. BOTH are required - **Combat** will not work without them.

They can be purchased from the Unity Asset Store:

- Game Creator

- Shooter

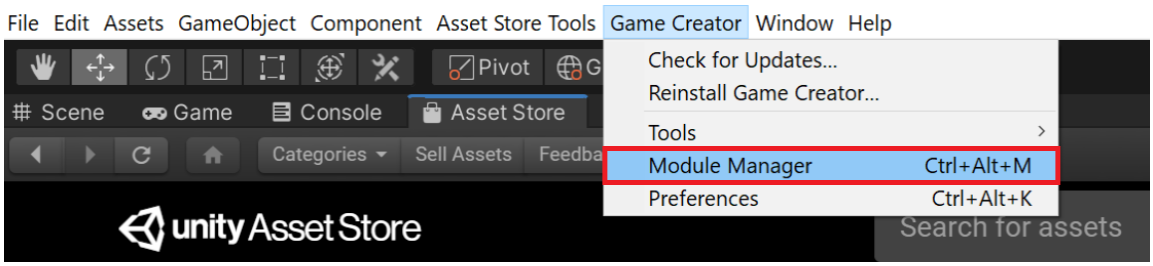
What's Included

- Full source code.
- An examples module that contains scenes that demonstrate the features listed above.

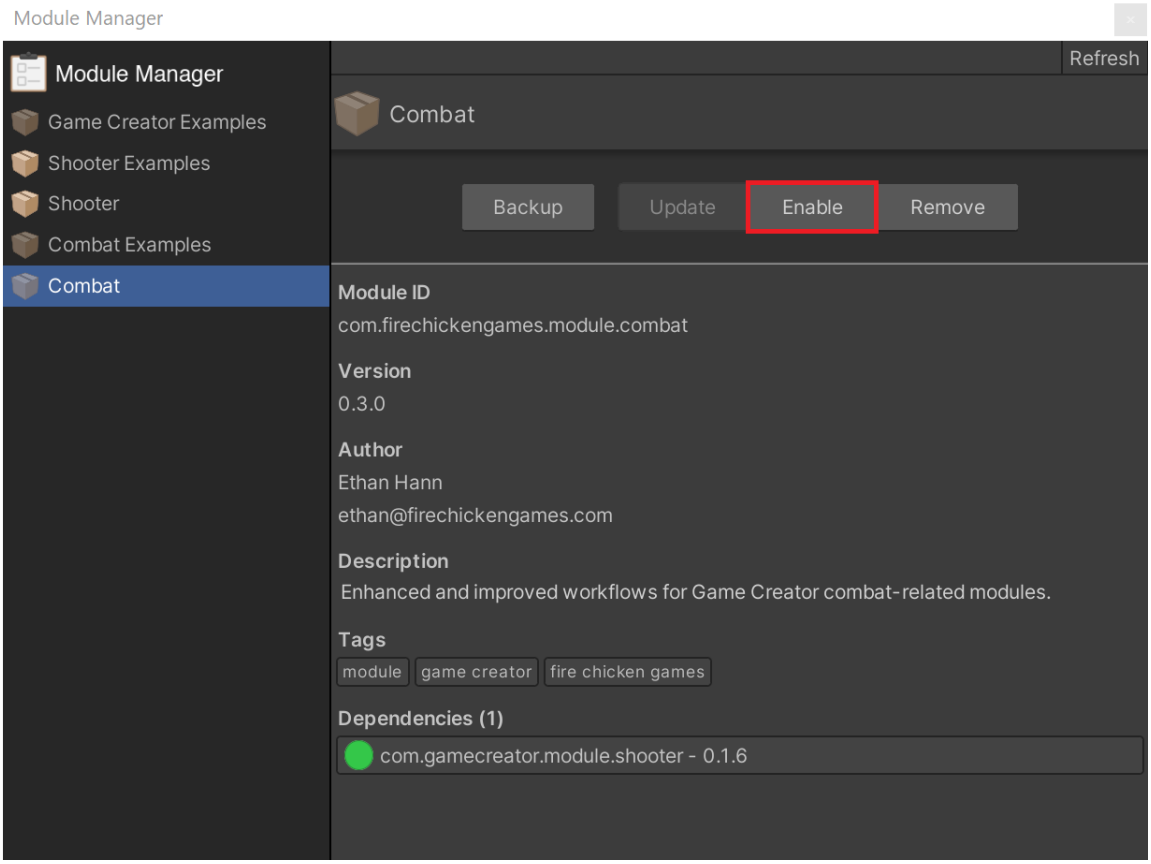
Module Installation

After purchasing and downloading the **Combat** module, it must be enabled with the Game Creator Module Manager.

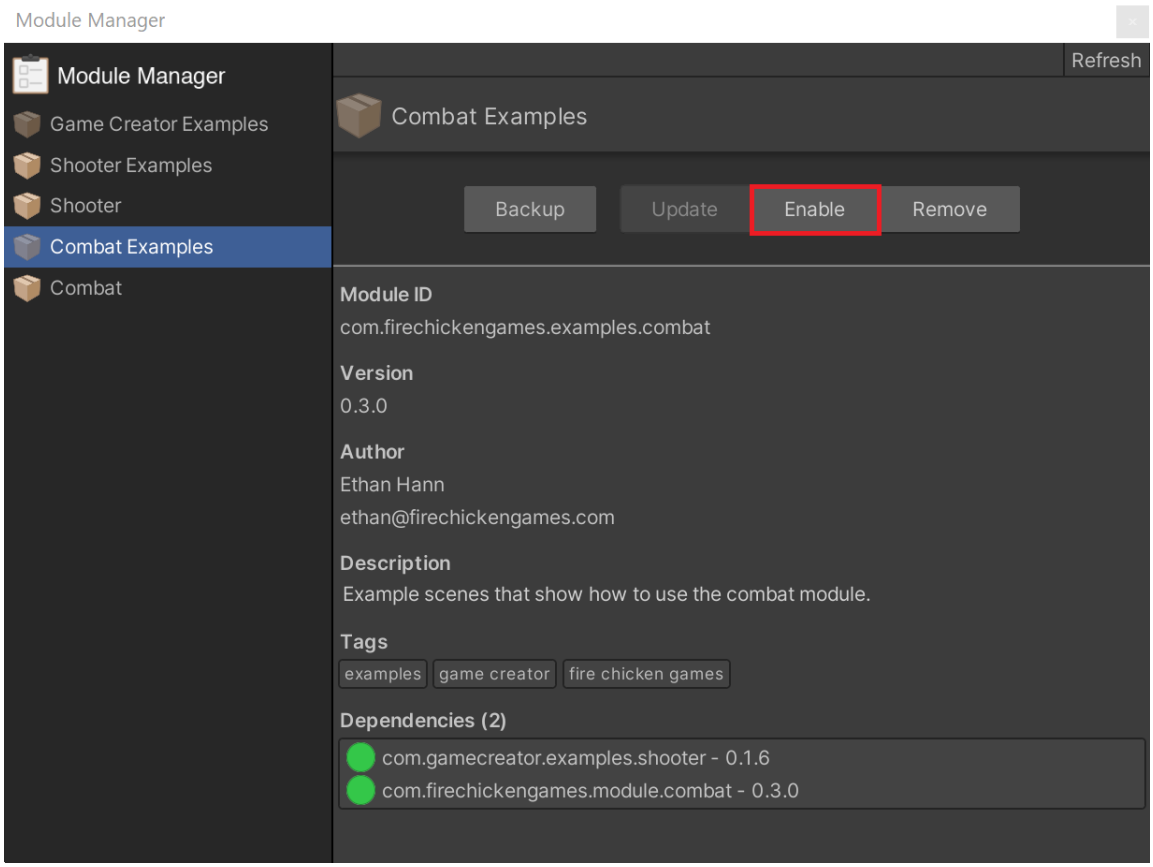
Step 1: Open the Module Manager



Step 2: Enable the Combat Module



Step 3 (optional): Install the Combat Examples Module



Targeting

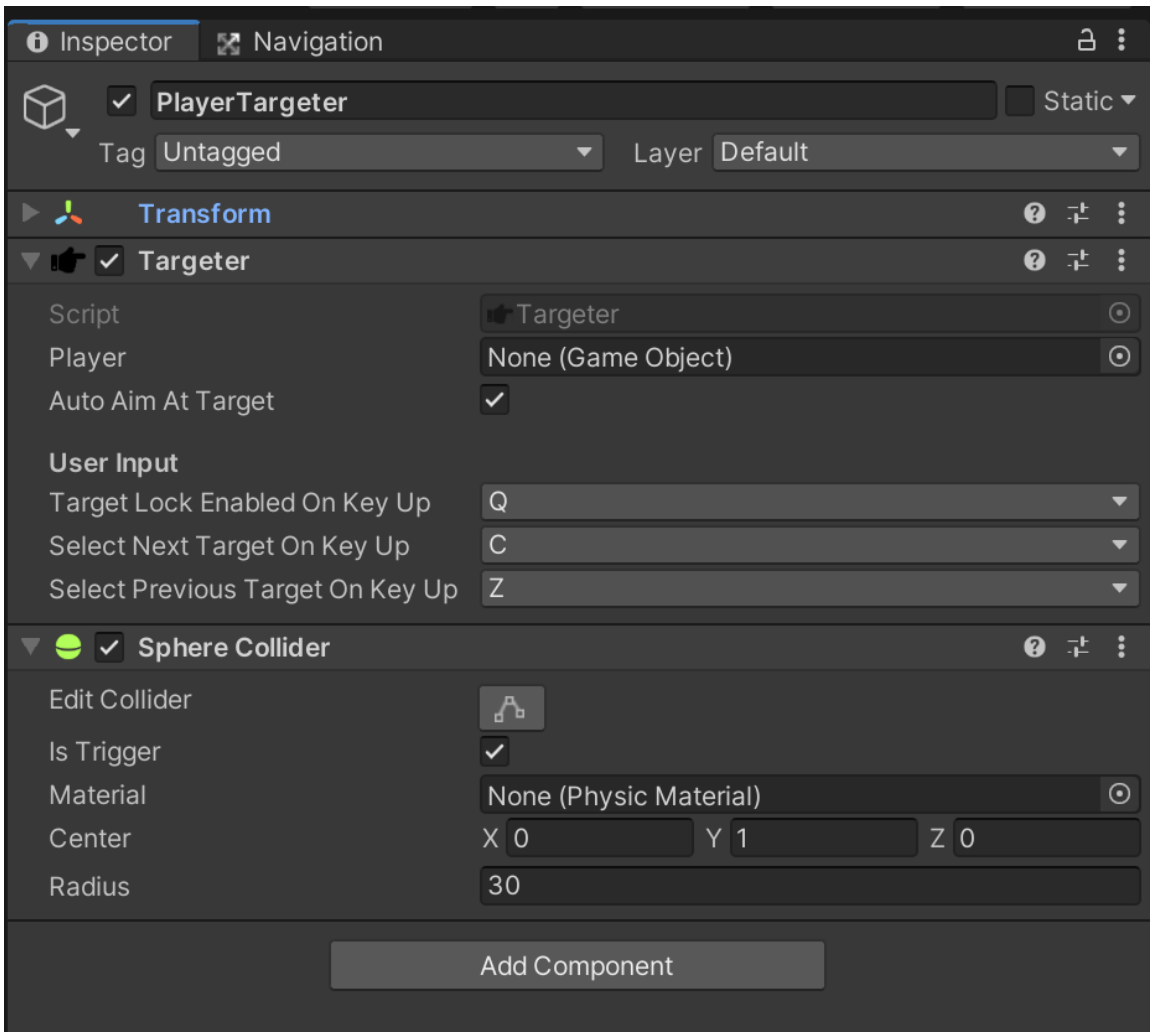
The **Combat** module provides a proximity-based targeting system. It allows a player to target characters (or other game objects) within a configurable range. This is achieved with two components: **Targeter** and **Targetable**.

Targeter Component

Included in the **Combat Examples** module is a prefab called **PlayerTargeter** that demonstrates how to use the **Targeter** component. The **Sphere Collider** attached to the same game object (as depicted in the inspector screenshot below) is required. Note that the range of the **Targeter** is dictated by the **Radius** property of this collider. The **PlayerTargeter** prefab should be nested under the Game Creator Player object.

Automatically aiming at a target can be disabled with the **Auto Aim At Target** option - the character will still be locked on to it, but will not fix their weapon on it while aiming. This may be desirable for some games, but should be left enabled for most.

The **User Input** section of the **Targeter** component allows the keys that control target locking and switching to be customized.



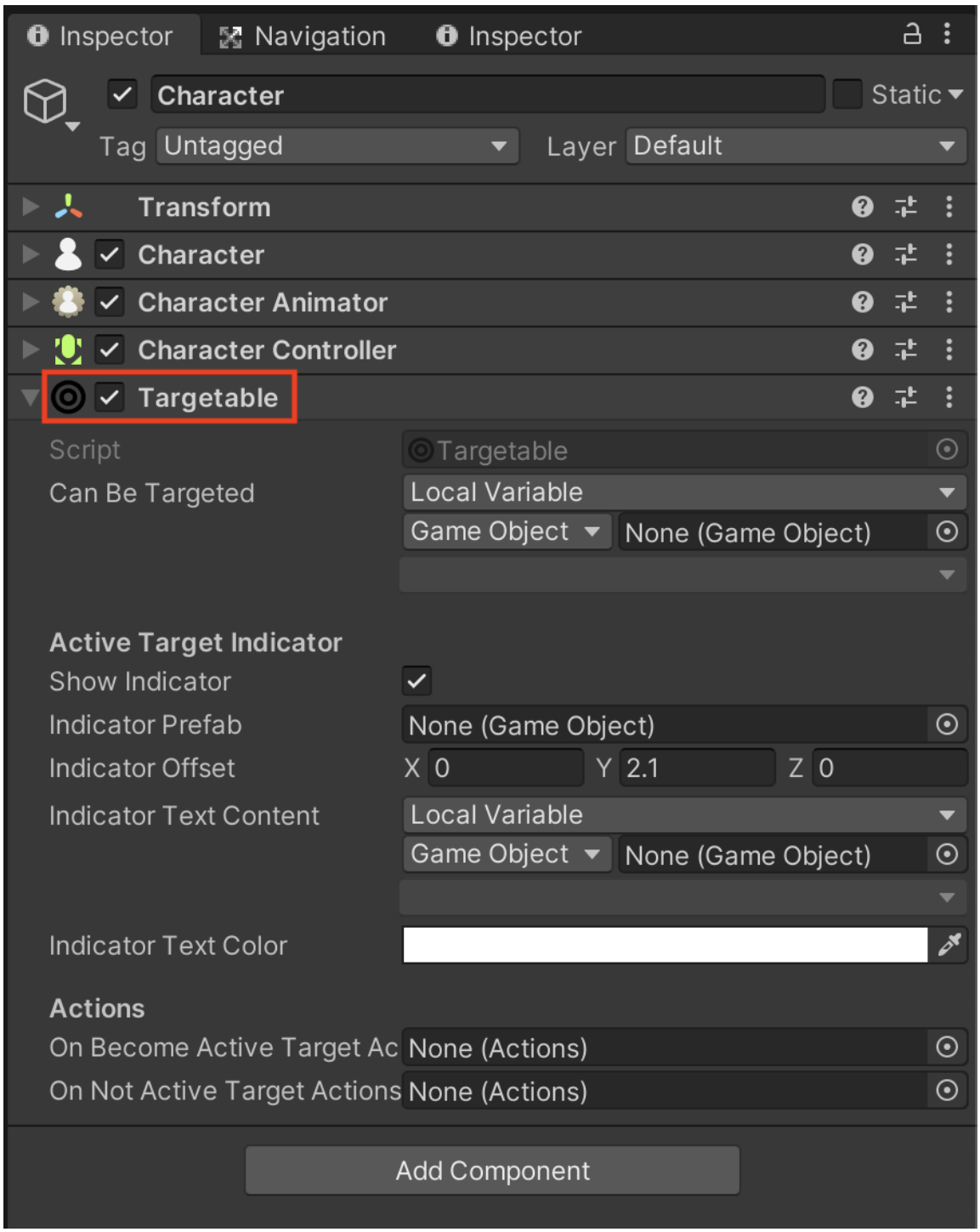
Targetable Component

Making game objects targetable using the **Combat** module's **Targetable** component is trivial for Game Creator Characters and other game objects.

Basics

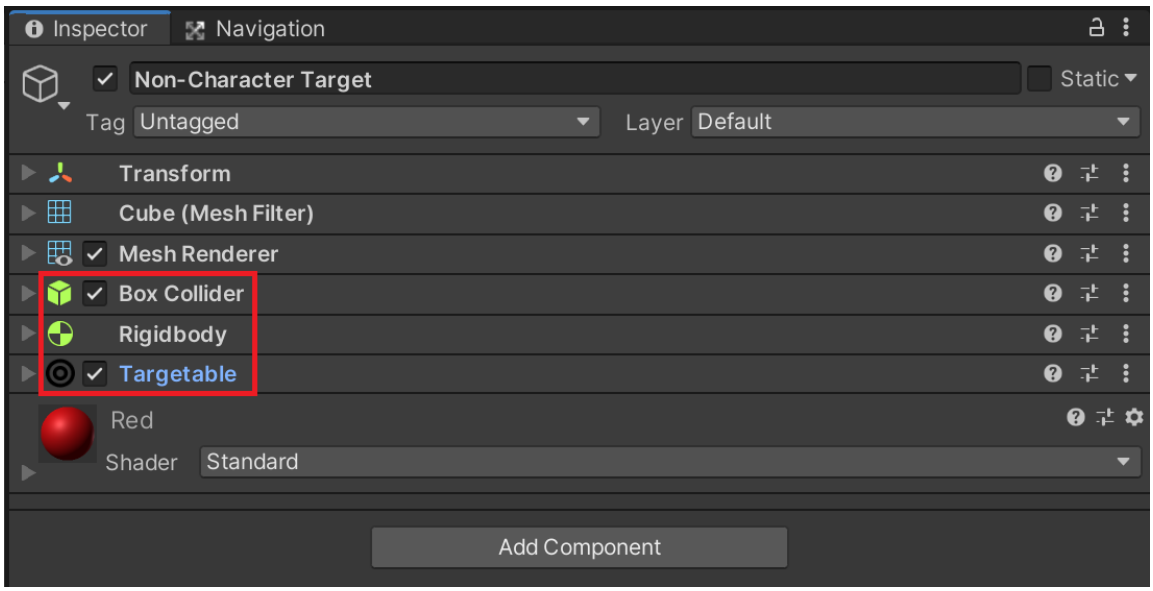
Characters

To make a character targetable, simply add the **Targetable** component to it.



Non-Character Game Objects

Any game object can be targetable if it has the **Targetable**, **Rigidbody**, and **Collider** components.



Making a Target Untargetable

The **Targetable** component contains a boolean Game Creator Variable property, called **Can Be Targeted**, that can make the target untargetable.

For example, once the target has been defeated, it is likely desirable for the player to automatically stop targeting it and not be able to target it again.

This is accomplished by adding a boolean **Local Variable√ to a character (e.g. "IsAlive"), then assigning it to the** Can Be Targeted** property.

In the character's **On Receive Shot Actions**, set the boolean value to false - this will deselect the target and make it no longer targetable.

The **Combat Examples** module's **Example4-KillableCharacters** demo scene contains a pre-configured **KillableCharacter** prefab. It demonstrates how to make a character killable/untargetable using the method described above.

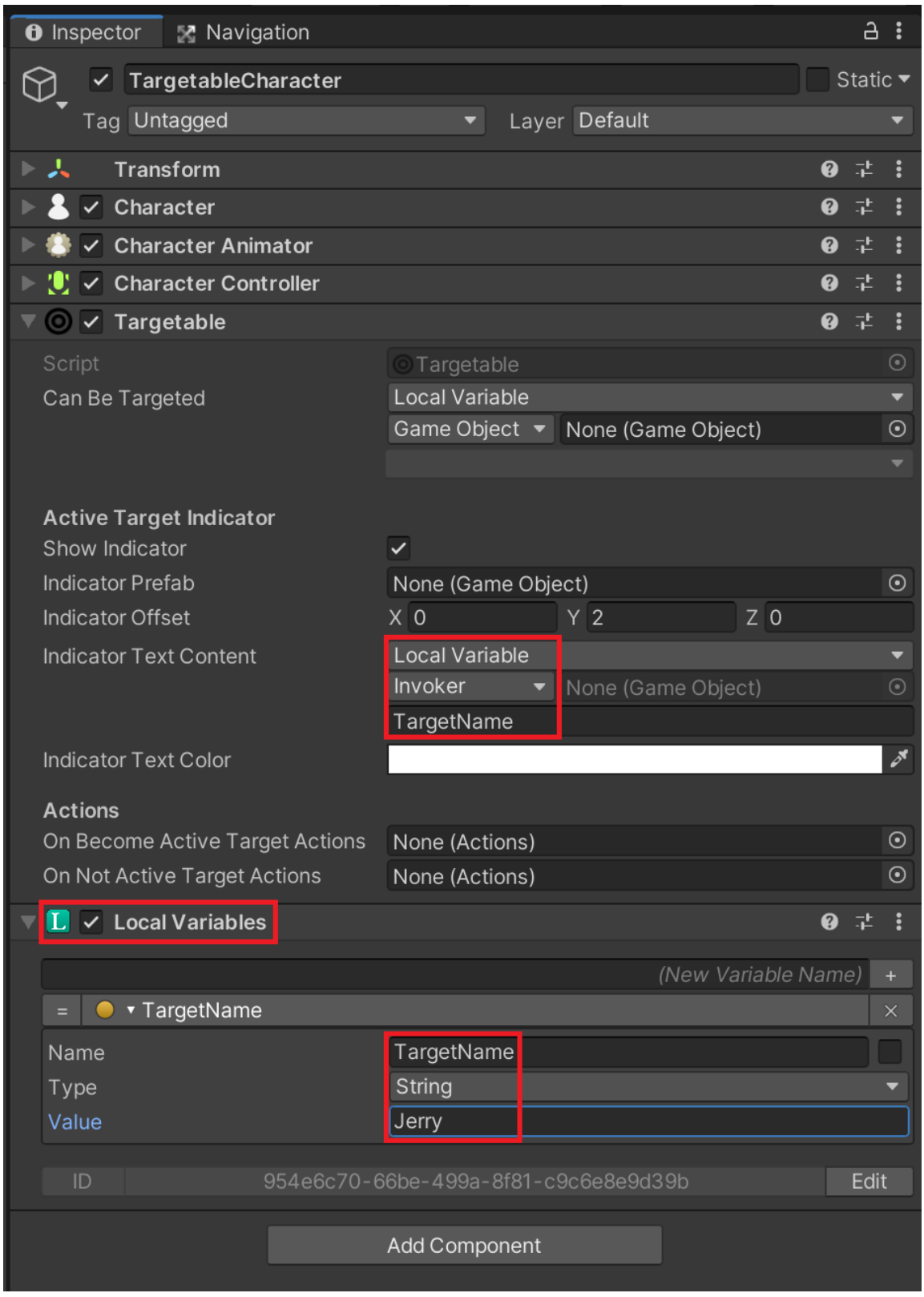
Advanced

Active Target Indicator

The **Targetable** component provides an "indicator" feature that highlights the currently targeted game object. The content and appearance of the indicator is configurable.

TEXT

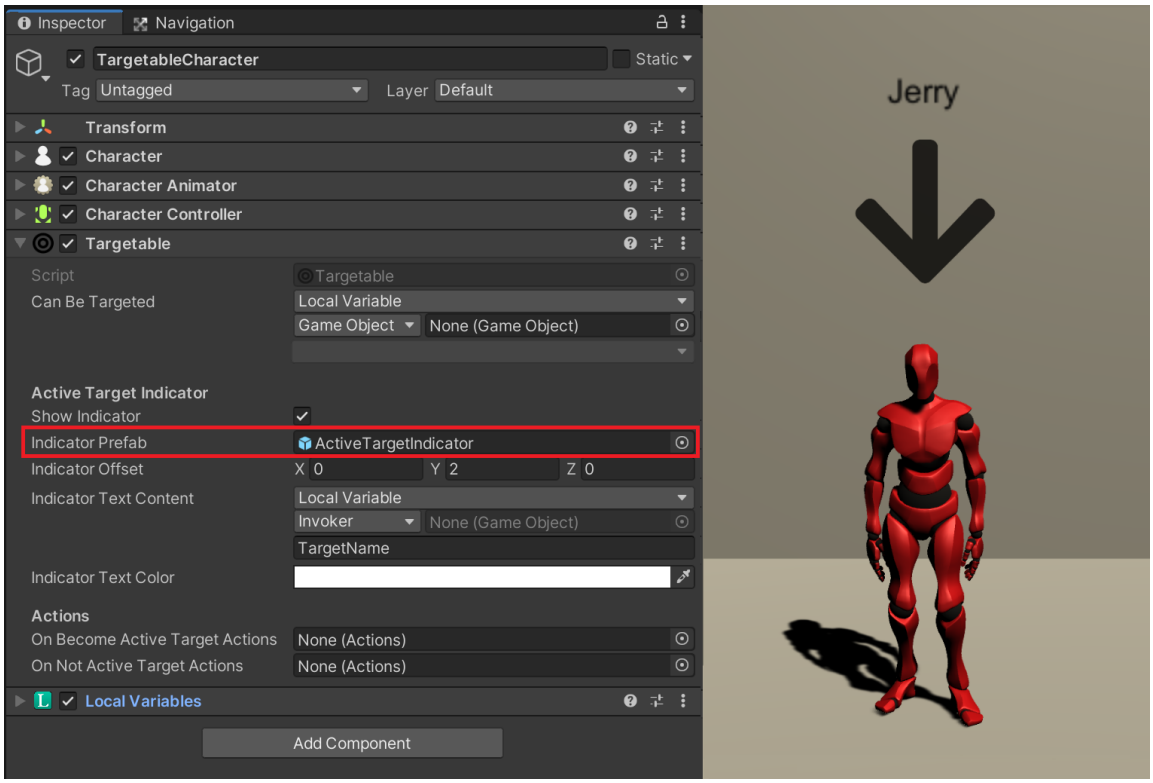
An indicator can have custom text, defined via a Game Creator **Global/Local Variable**. Practically speaking, it almost always makes sense to use a **Local Variable** packaged in the same prefab object that contains the **Targetable** component. The value of the Local Variable would then be configured on the instance of the prefab when used in a scene.



PREFAB

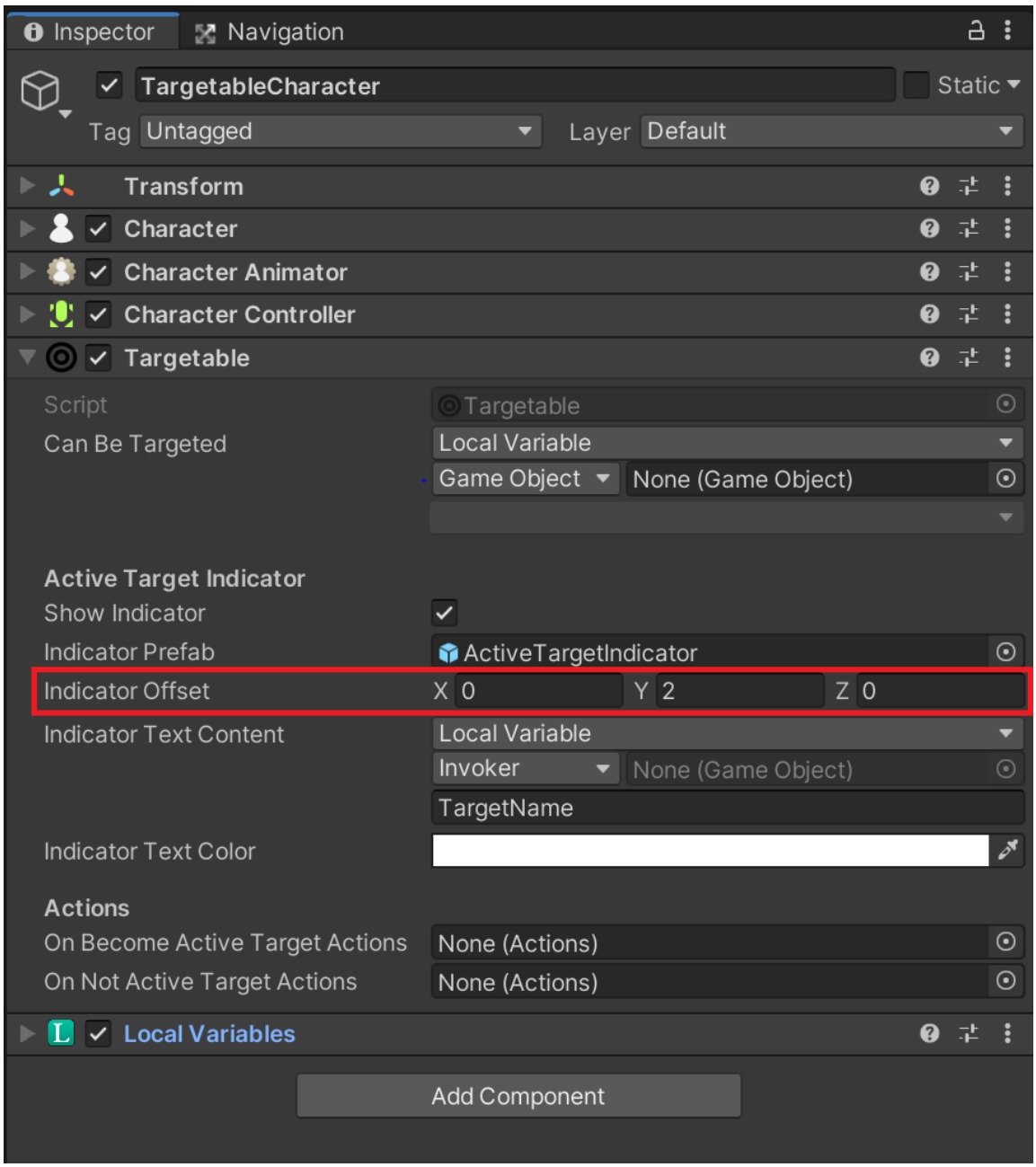
If not set, the Game Creator **Floating Message** prefab is automatically set as the target indicator prefab at runtime. The **Combat Examples** module includes an

example of a custom indicator. The example indicator has text above a downward pointing arrow.



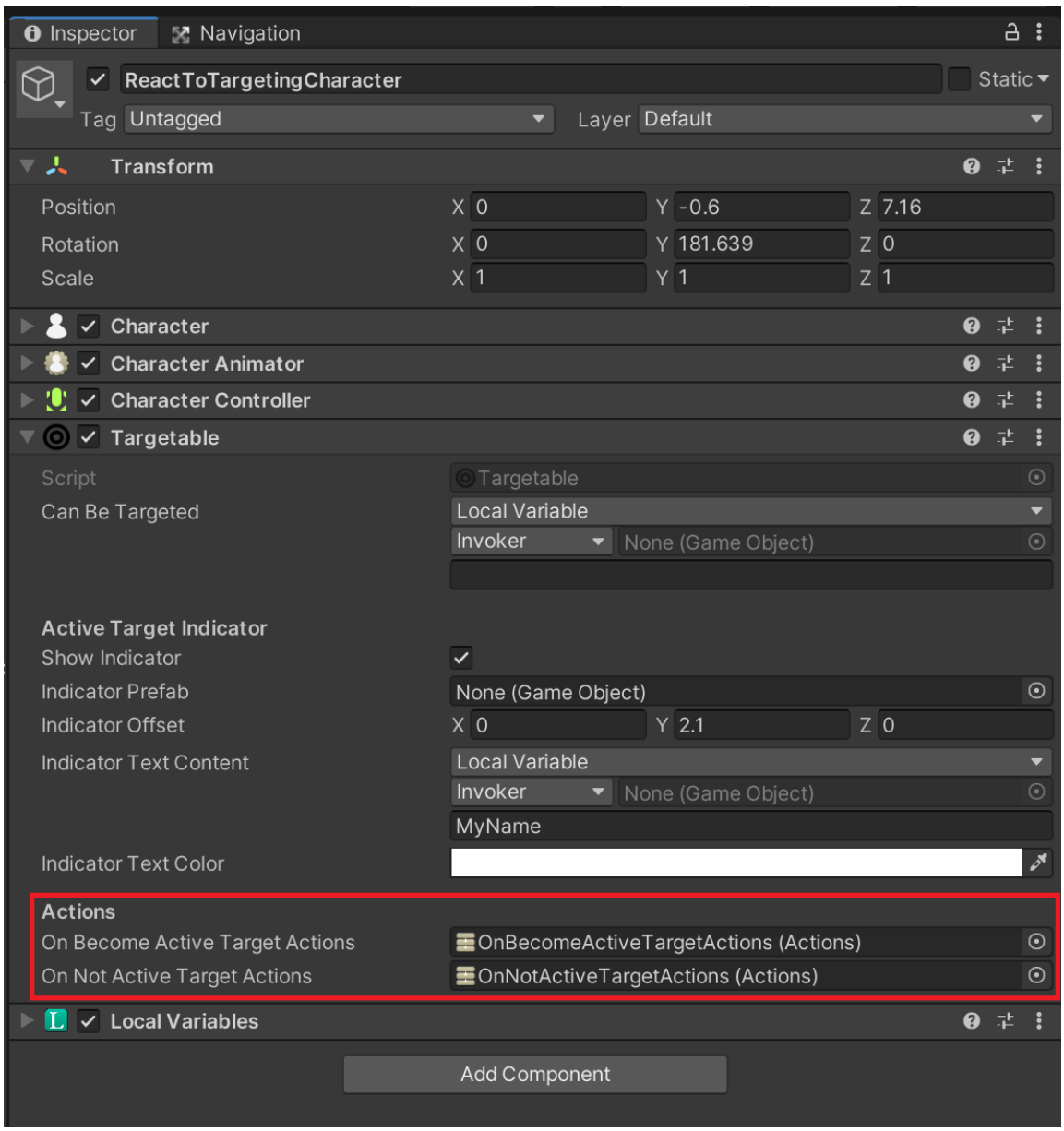
POSITIONING

The target indicator is positioned relative to the parent game object. By default, the **Indicator Offset** vector will position the indicator above the Game Creator example character, but may need to be adjusted for other characters and objects of with heights.



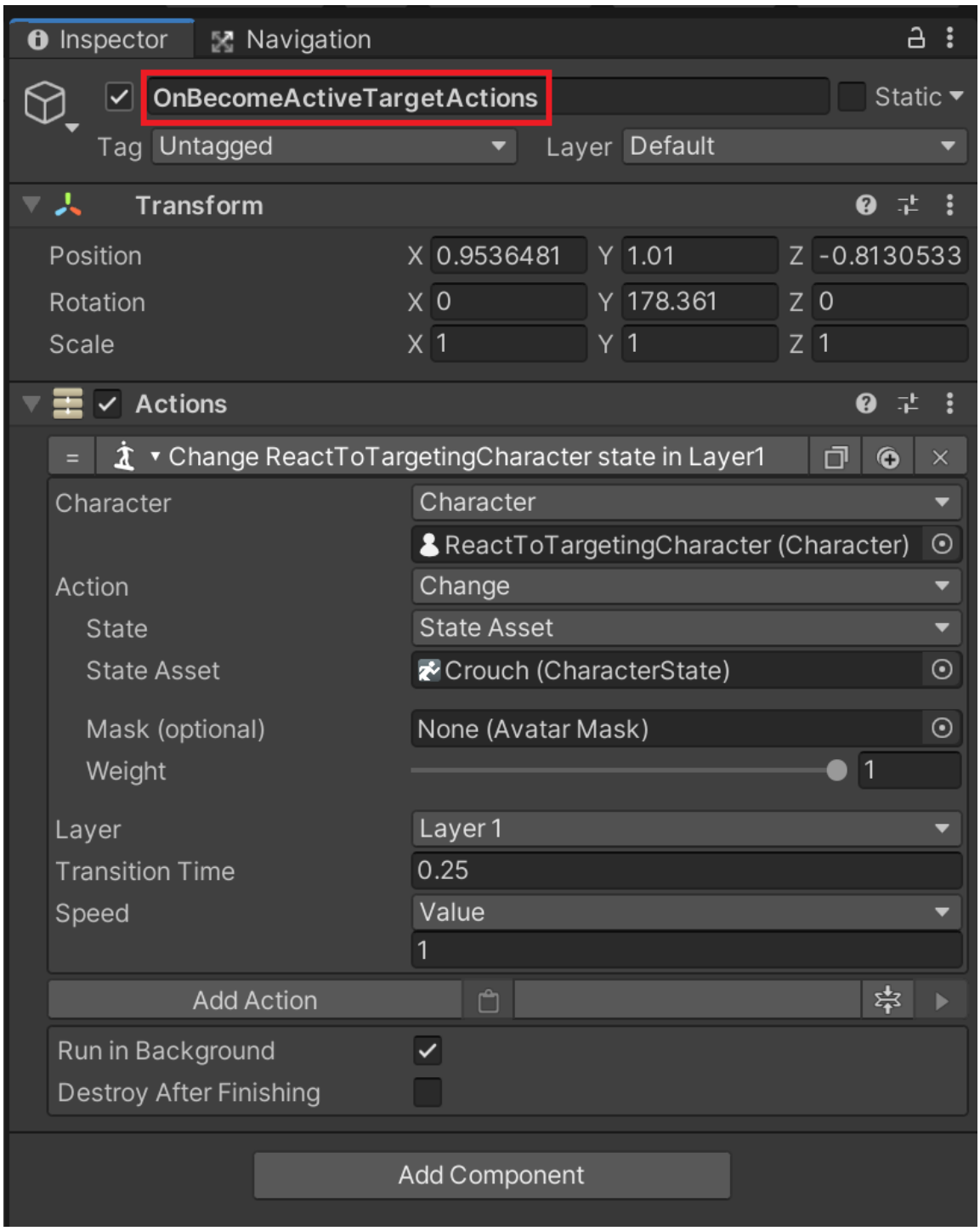
Targeting Actions

A **Targetable** game object can optionally execute actions when it becomes the active target, and another set of actions when some other object becomes the active target (or targeting is toggled off altogether).



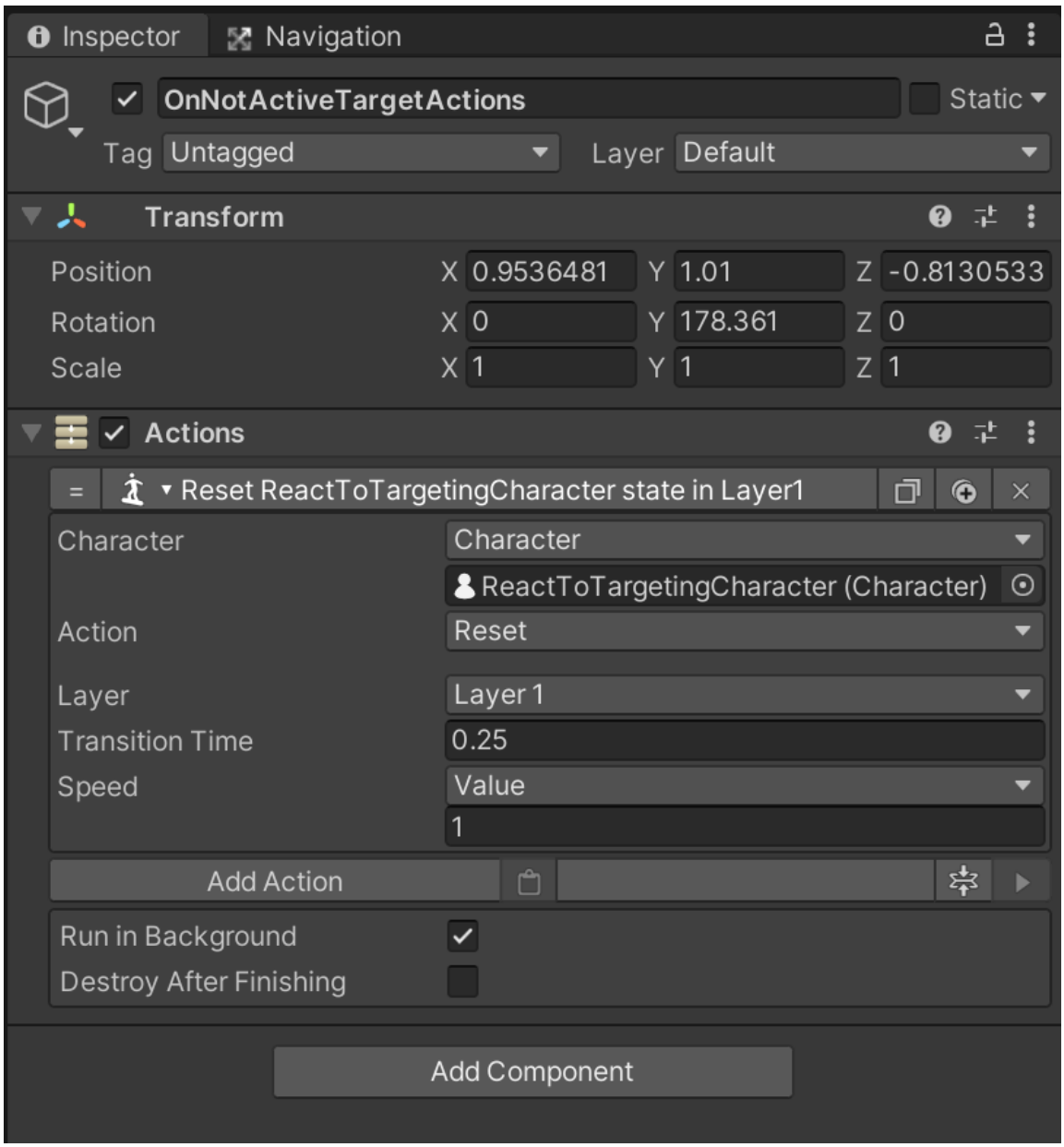
"ON BECOME ACTIVE TARGET"

When the target becomes the active target, these actions can (for example) make the target crouch. A more practical example might be adding an outline around the target or perhaps set a variable that triggers some behavior (e.g. make the target hostile or flee).



"ON NOT ACTIVE TARGET"

Related to the previous section, when the target is changed or targeting is disabled, this action will reset the target character's gesture state.


Inspector
Navigation
OnNotActiveTargetActions
Static
Tag
Untagged
Layer
Default
Transform
Position
X 0.9536481
Y 1.01
Z -0.8130533
Rotation
X 0
Y 178.361
Z 0
Scale
X 1
Y 1
Z 1
Actions
Reset ReactToTargetingCharacter state in Layer1
Character
Character
ReactToTargetingCharacter (Character)
Action
Reset
Layer
Layer 1
Transition Time
0.25
Speed
Value
1
Add Action
Run in Background
Destroy After Finishing
Add Component

Homing Projectiles

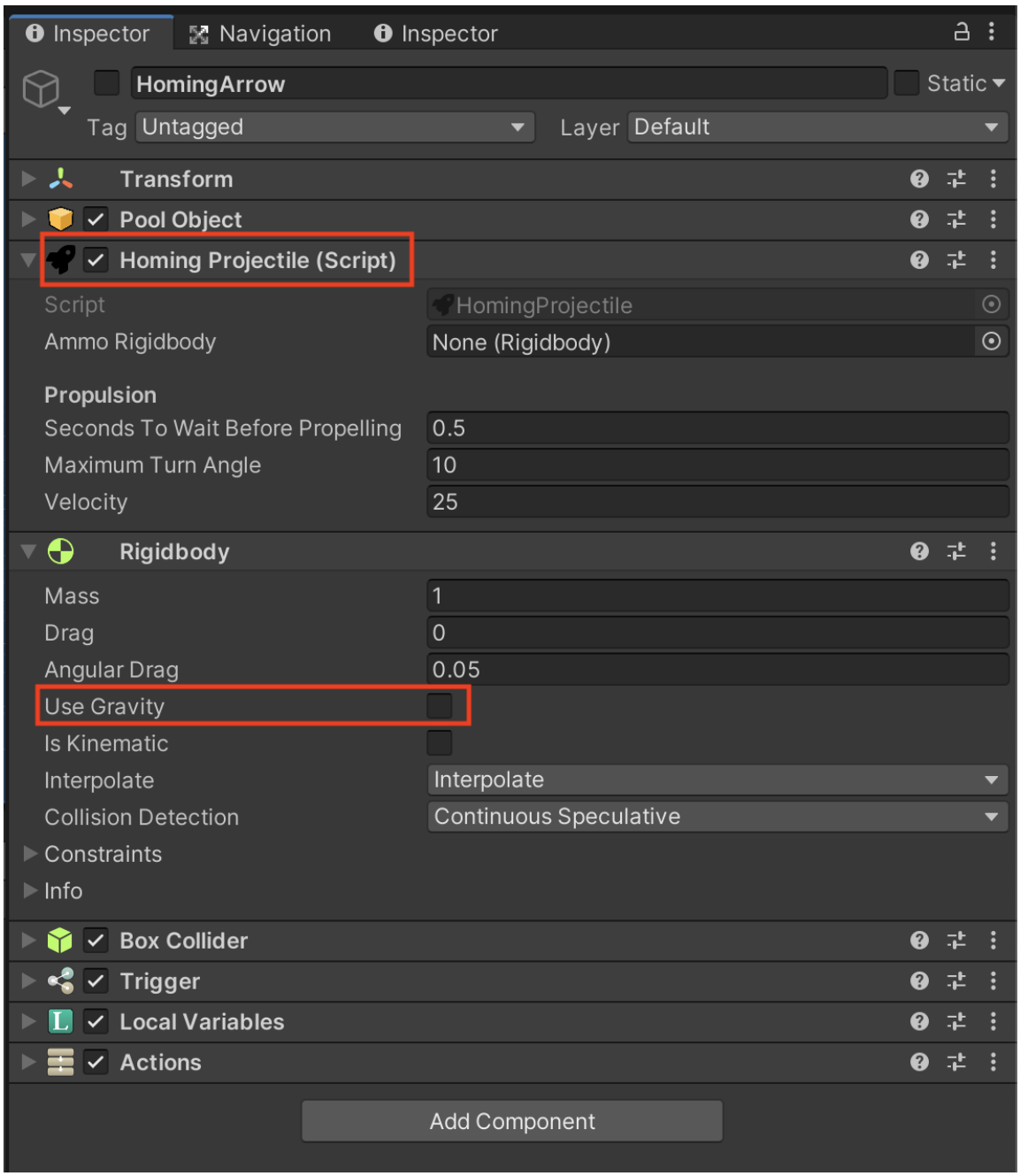
As the name suggests, a **Homing Projectile** seeks its target even if the weapon firing the projectile is not pointed directly at the intended target.

Setup is trivial. Simply attach the Combat module's **Homing Projectile** component to any projectile. The component's **Ammo Rigidbody** property will be automatically set if the game object contains a Rigidbody component.

The **Propulsion** settings control the movement behavior of the projectile:

- **Seconds To Wait Before Propelling**: Delays the propulsion of the projectile by a number of seconds. A value of 0 (or less) results in no delay.
- **Maximum Turn Angle**: The maximum angle, in degrees, that the projectile will turn while homing in on its target.
- **Velocity**: How fast the projectile moves toward its target - this should likely match or exceed the max velocity of the projectile ammo if propulsion is delayed. If there is no delay, this setting will effectively override the projectile ammo's min/max velocity.

> **Rigidybody Gravity**
>
> Turning off gravity on the Rigidbody is optional, but might be desired depending on the specific projectile.


Inspector
Navigation
Inspector
HomingArrow
Static
Tag
Untagged
Layer
Default
Transform
Pool Object
Homing Projectile (Script)
Script
HomingProjectile
Ammo Rigidbody
None (Rigidbody)
Propulsion
Seconds To Wait Before Propelling
0.5
Maximum Turn Angle
10
Velocity
25
Rigidbody
Mass
1
Drag
0
Angular Drag
0.05
Use Gravity
Is Kinematic
Interpolate
Interpolate
Collision Detection
Continuous Speculative
Constraints
Info
Box Collider
Trigger
Local Variables
Actions
Add Component