Welcome

The **Combat** module is now available for purchase on the Unity Asset store. Visit the asset's product page for details.

Combat Demos

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.
- Homing projectile.
- Targeting by visibility.
- Support for destructible targets.
- Weapon Stashes (weapon carrying/switching feature).
- Melee targeting integration (requires Melee module).

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

The **Melee** module is an optional dependency. Get it here:

- Melee

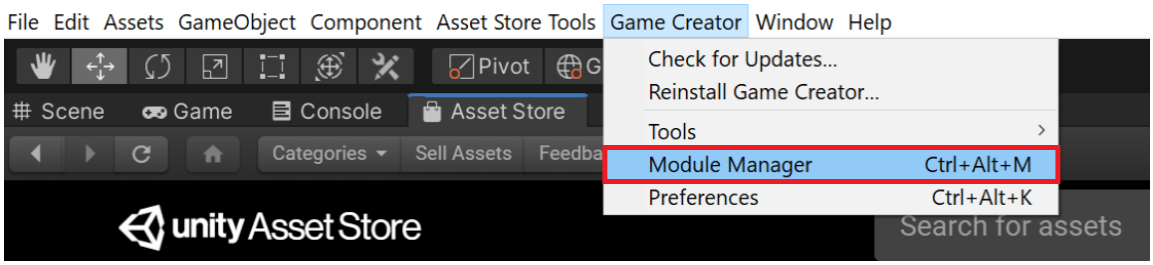
What's Included

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

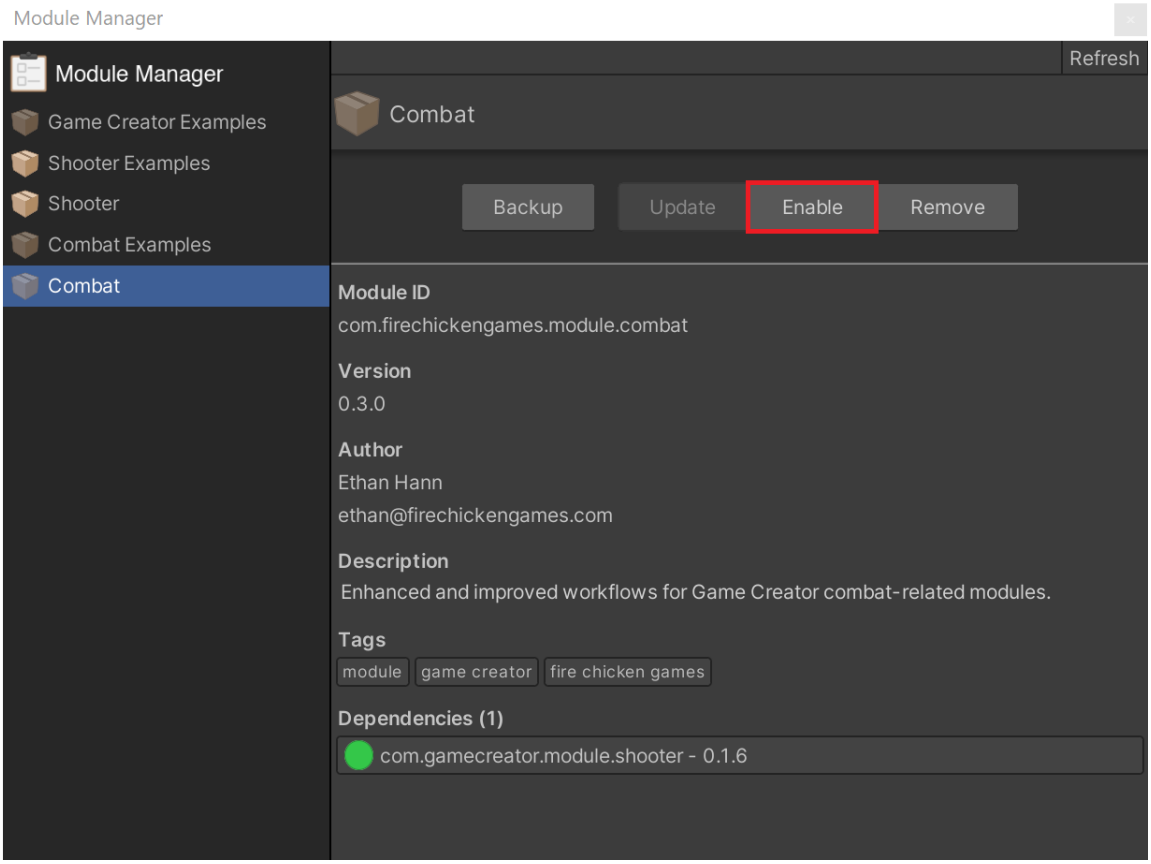
Module Installation

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

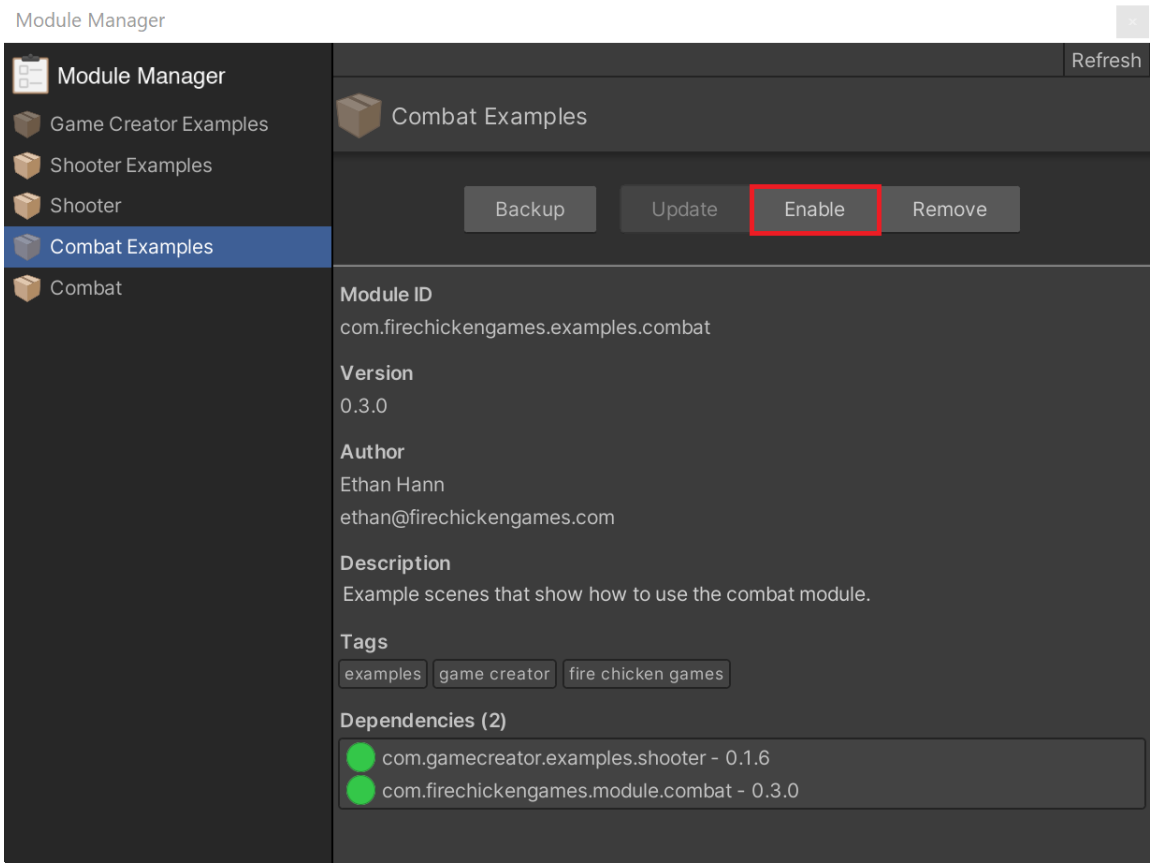
Step 1: Open the Module Manager



Step 2: Enable the Combat Module



Step 3 (optional): Install the Combat Examples Module



Melee Module Integration

The **Combat** module provides a lightweight integration to allow seamless Shooter/Melee targeting and weapon switching if the Melee module is present.

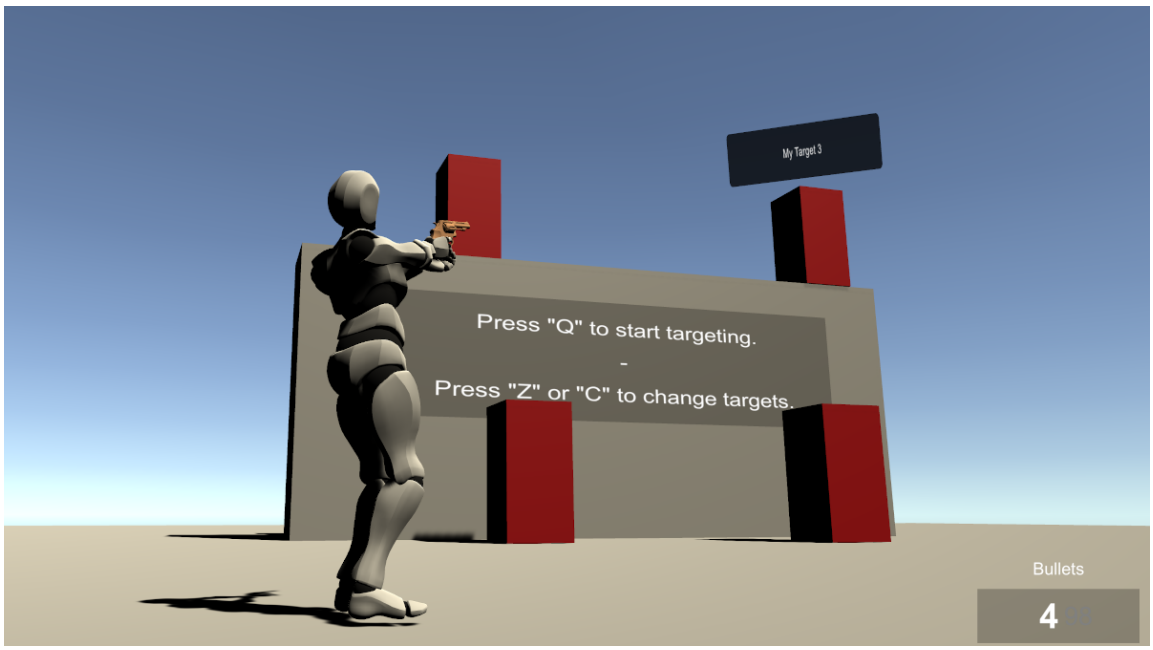
To enable the integration, simply enable the **Melee Combat** module included with the **Combat** module.

Note that there is also a **Melee Combat Examples** module that demonstrates Shooter/Melee targeting and weapon switching.

Overview

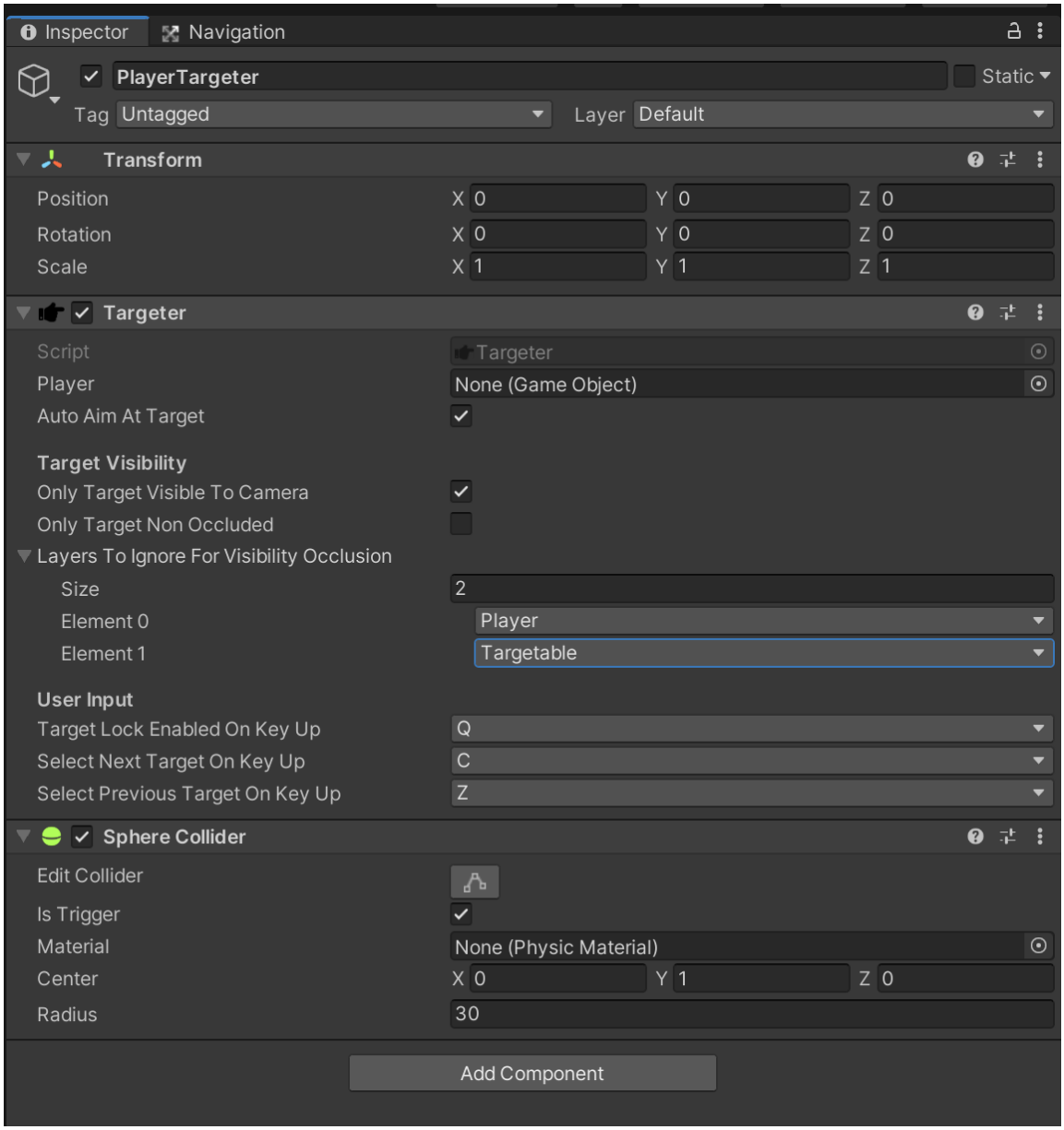
The **Combat** module provides a proximity-based targeting system. It allows a player to target characters (or other game objects) within a configurable range. This is achieved with two components: Targeter and Targetable.

When a player with a **Targeter** component approaches objects with the **Targetable** component attached, the player can press the `Q` key to lock onto the nearest available object. The `Z` and `C` keys are used to switch targets. Please note that these keys can be remapped.



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.



Auto Aim At Target

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.

Target Visibility

> **New in 0.4.0**

The component's target visibility options allow for target selection to be limited by what the camera can see.

Only Target Visible To Camera

If enabled, only targets possibly visible to the camera (i.e. in its view frustum) are targetable.

Only Target Non-Occluded

If enabled, targets hidden behind objects are not targetable.

Note that this option is turned off by default because the player and targetables need to be on dedicated layers which requires manual configuration.

Layers To Ignore For Visibility Occlusion

The layers to ignore when determining target visibility when Only Target Non-Occluded is enabled.

Typically, there should be two layers:

- A "Targetable" layer that contains all targetable objects.
- A "Player" layer that contains the player.

User Input

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

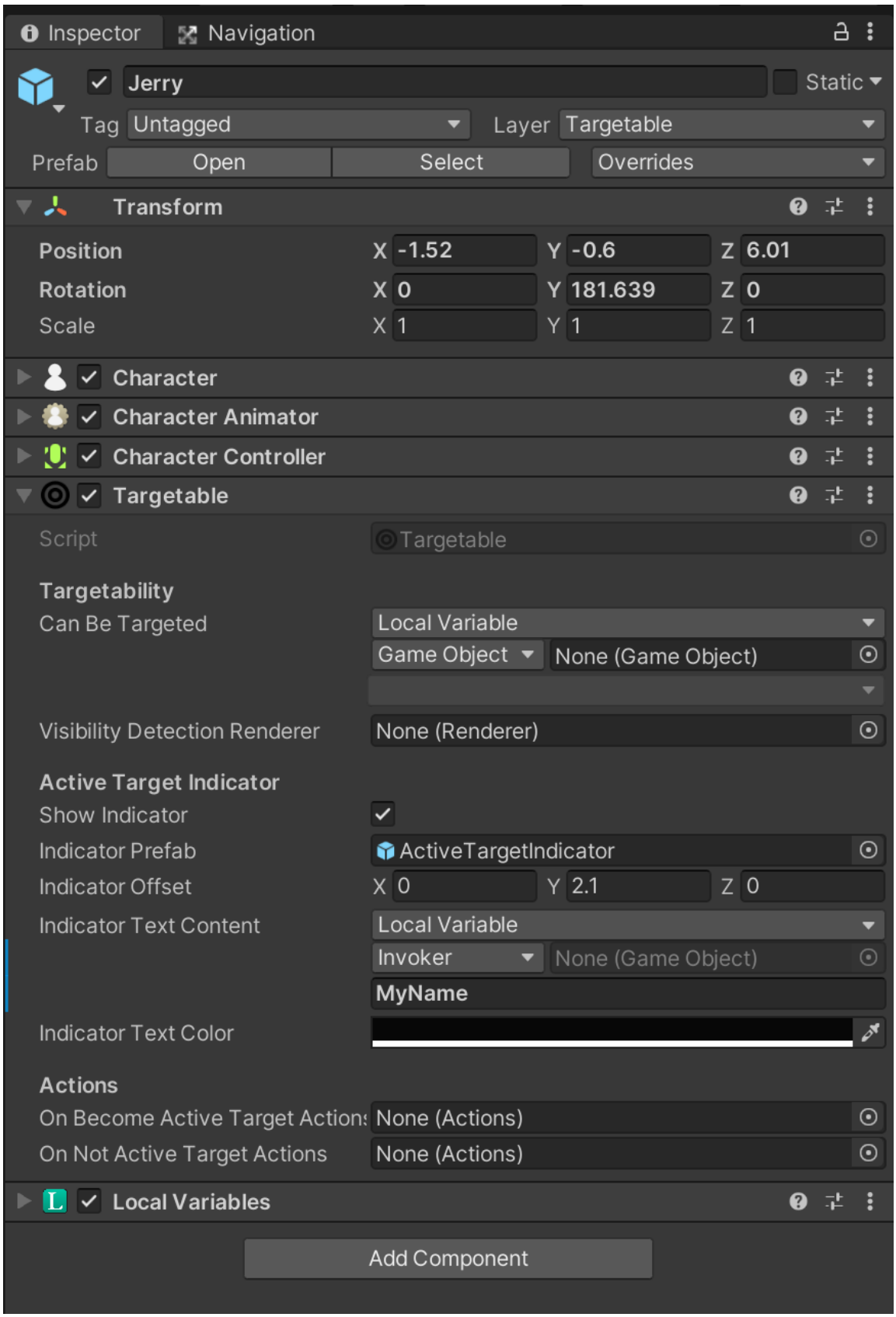
Targetable Component

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

Basic Setup

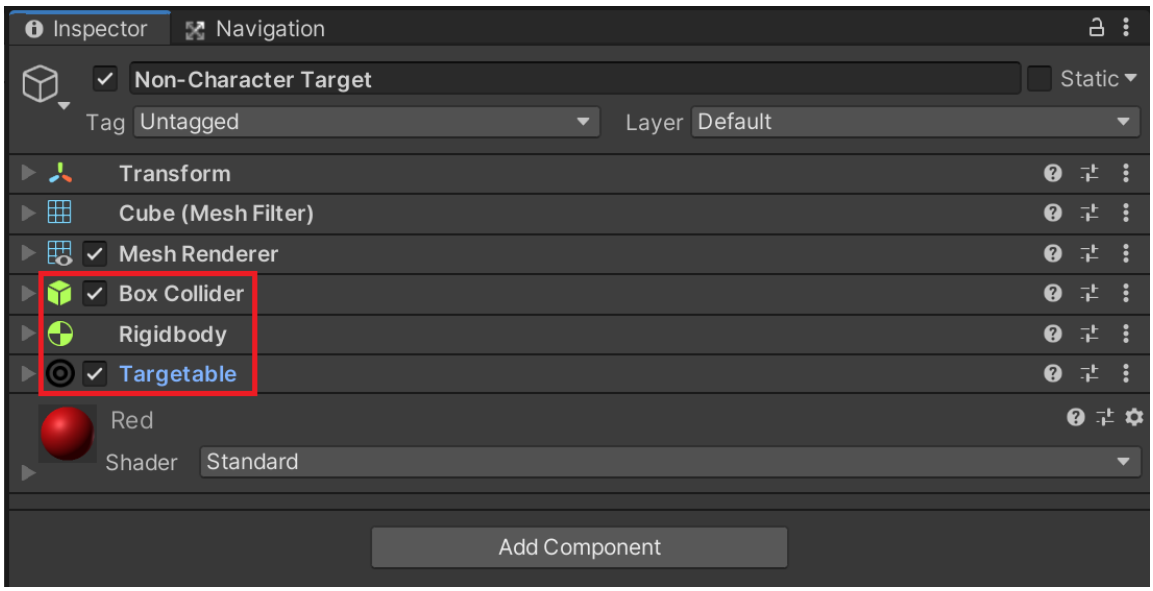
Characters

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


Inspector
Navigation
Jerry
Static
Tag
Untagged
Layer
Targetable
Prefab
Open
Select
Overrides
Transform
Position
X -1.52
Y -0.6
Z 6.01
Rotation
X 0
Y 181.639
Z 0
Scale
X 1
Y 1
Z 1
Character
Character Animator
Character Controller
Targetable
Script
Targetable
Targetability
Can Be Targeted
Local Variable
Game Object
None (Game Object)
Visibility Detection Renderer
None (Renderer)
Active Target Indicator
Show Indicator
Indicator Prefab
ActiveTargetIndicator
Indicator Offset
X 0
Y 2.1
Z 0
Indicator Text Content
Local Variable
Invoker
None (Game Object)
MyName
Indicator Text Color
Actions
On Become Active Target Actions
None (Actions)
On Not Active Target Actions
None (Actions)
Local Variables
Add Component

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:

1. Adding a boolean **Local Variable** to a character (e.g. "IsAlive").
2. Assigning the boolean variable to the **Can Be Targeted** property.
3. 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.

Visibility Detection Renderer

This property is used to determine if a **Targetable's** mesh is visible to a **Targeter** component. It does not normally need to be manually set (if not set, the component uses the first renderer found in the object's hierarchy). This property only needs to be manually set if the **Targetable** component's parent object does not have a mesh renderer in its hierarchy, or a mesh renderer other than the first one the `GetComponentInChildren<Renderer>()` returns by default is required.

If visibility features are not used, this property can be ignored completely.

"On Target Become Untargetable" Trigger

New in 0.4.0

There is also a trigger called "On Target Become Untargetable" that allows for actions to be executed when the target becomes untargetable. The **Combat Examples** module's **Example7-DestructibleTargets** demo scene contains a pre-configured **DestructibleTarget** prefab. It showcases how to use this trigger to implement destructible targets.

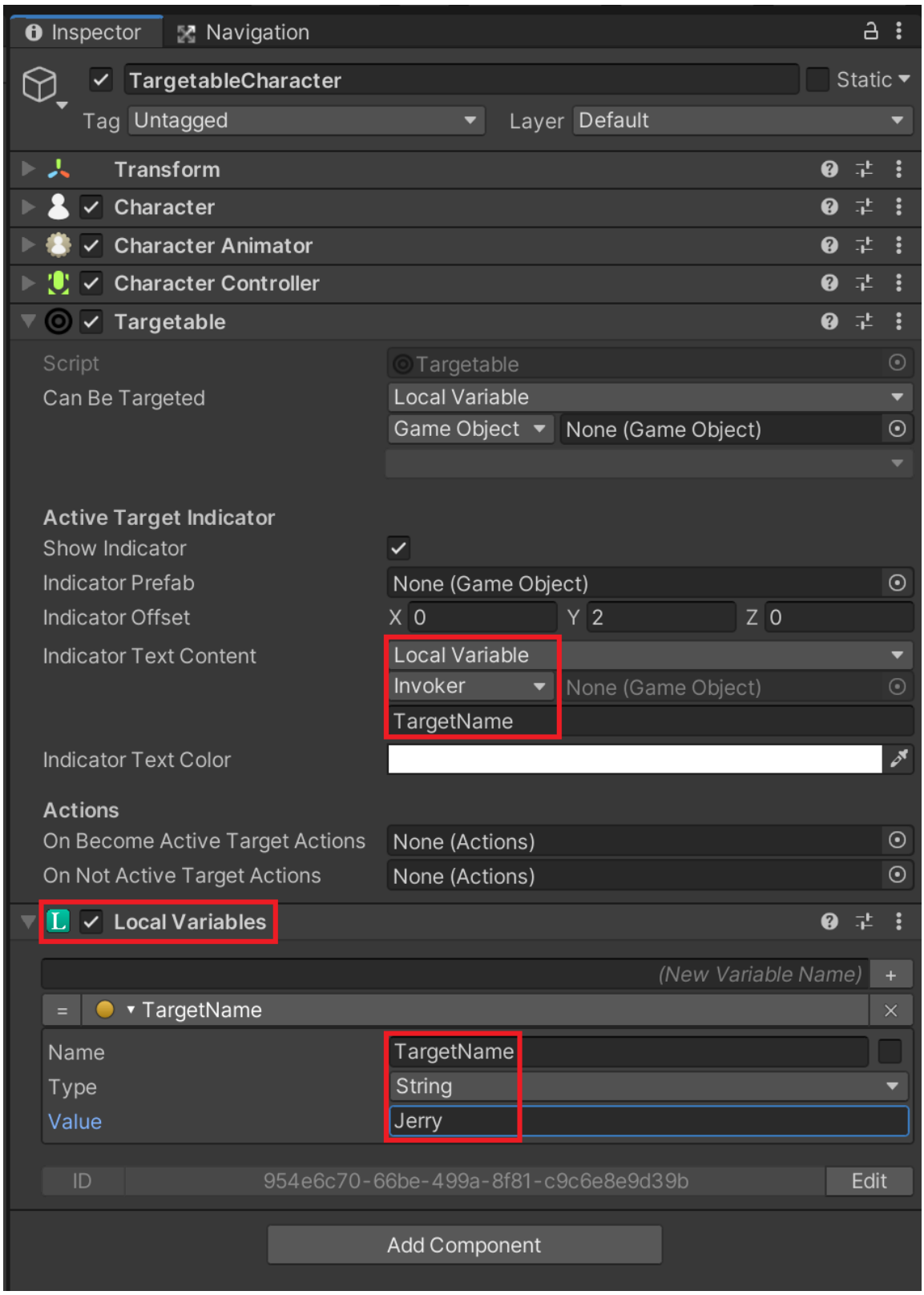
Advanced Options

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

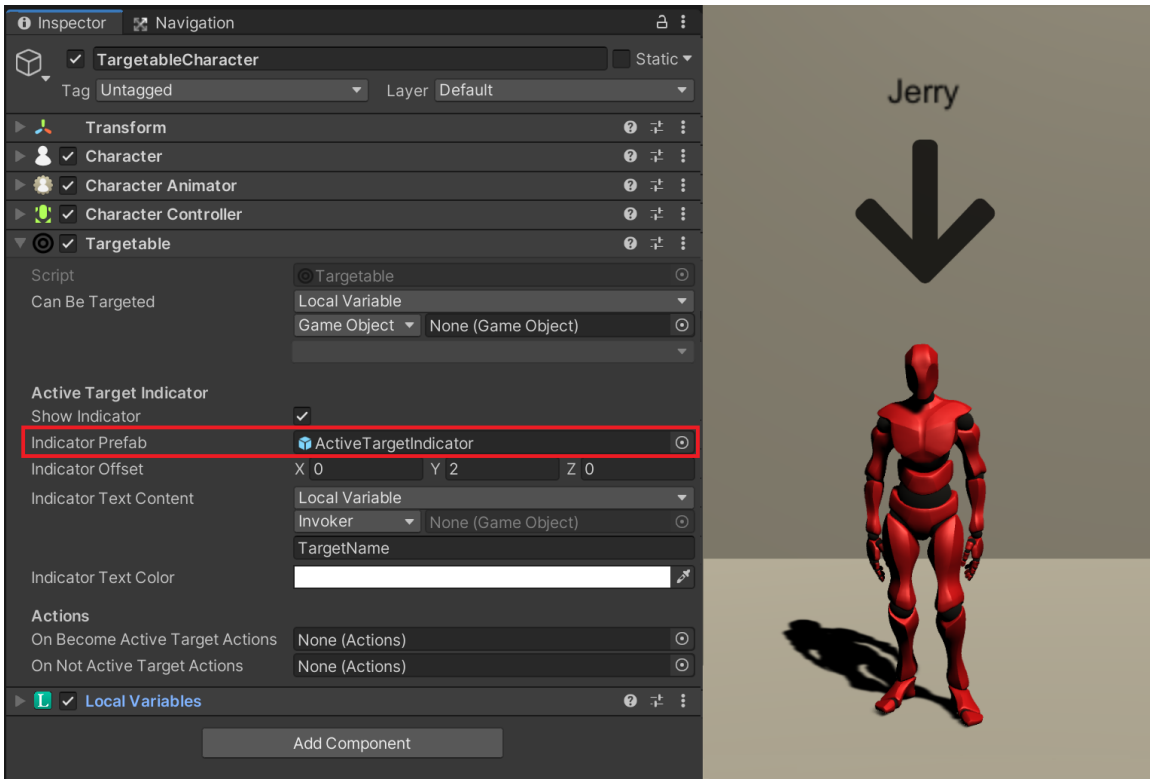
An indicator can have custom text, defined via a Game Creator **Global/Local Variable**. Practically speaking, it almost always makes sense to use a **Local Variable** packaged in the same prefab object that contains the **Targetable** component. The value of the Local Variable would then be configured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



Prefab

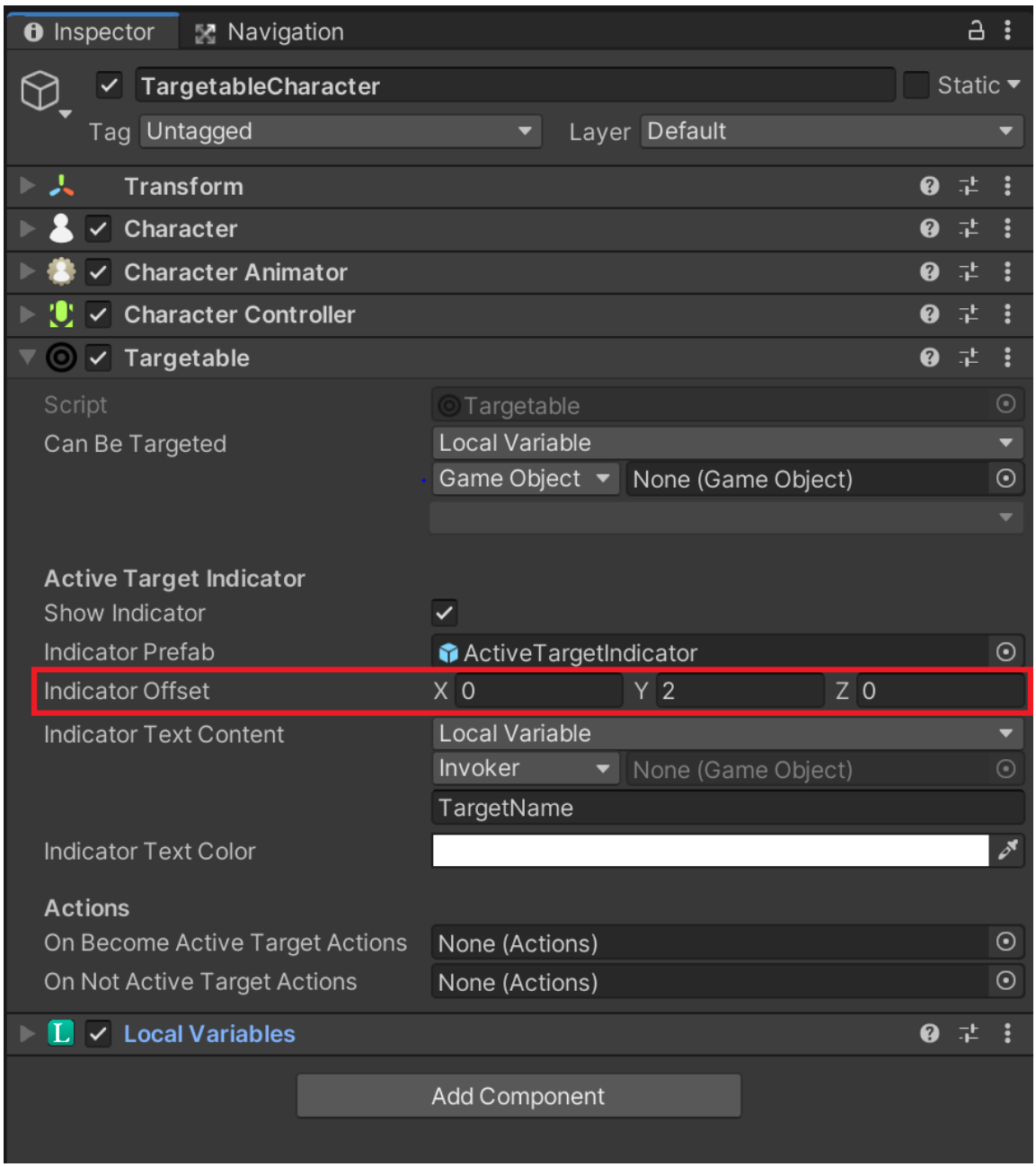
If not set, the Game Creator **Floating Message** prefab is automatically set as the target indicator prefab at runtime. The **Combat Examples** module includes

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



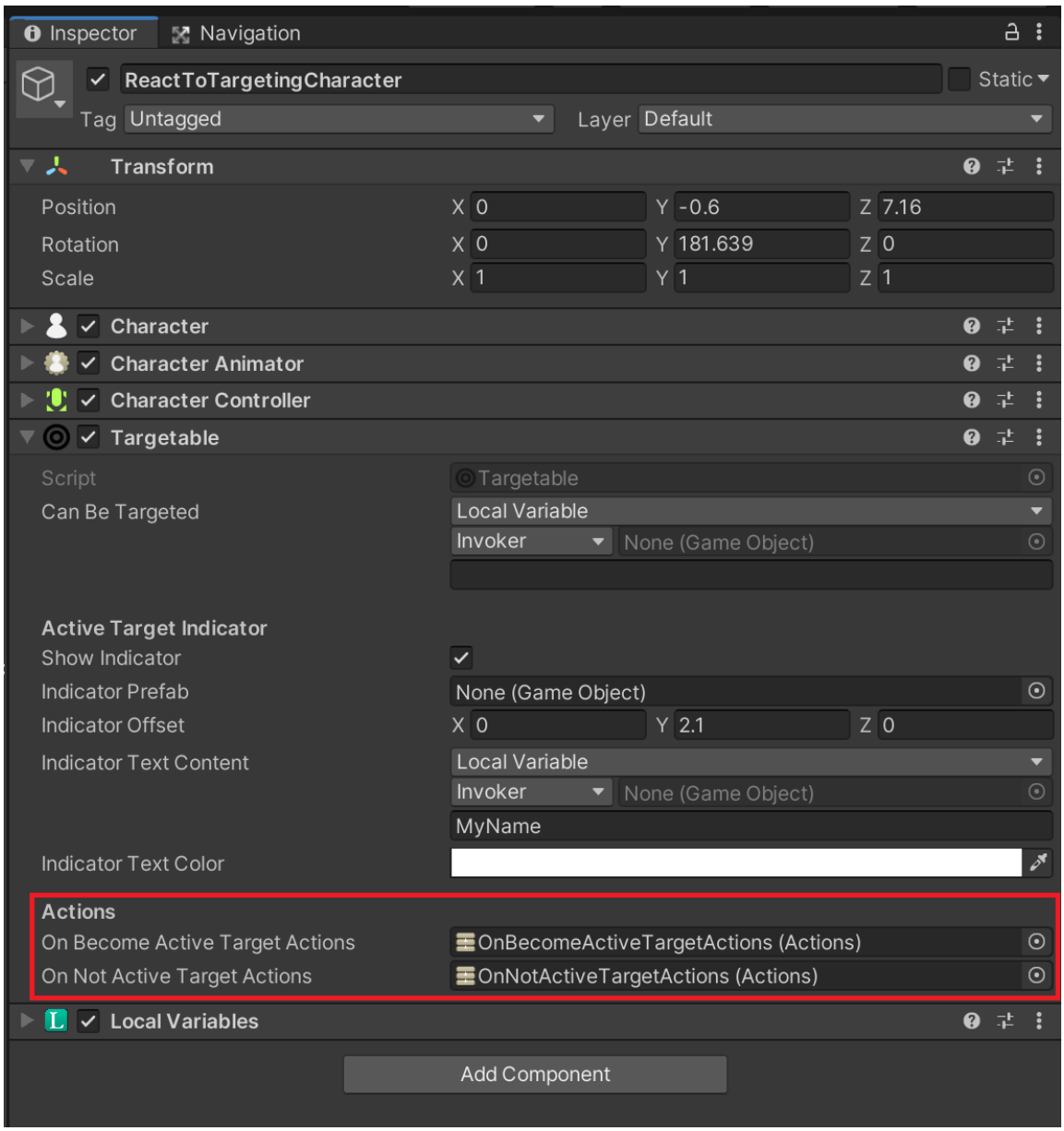
Positioning

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.



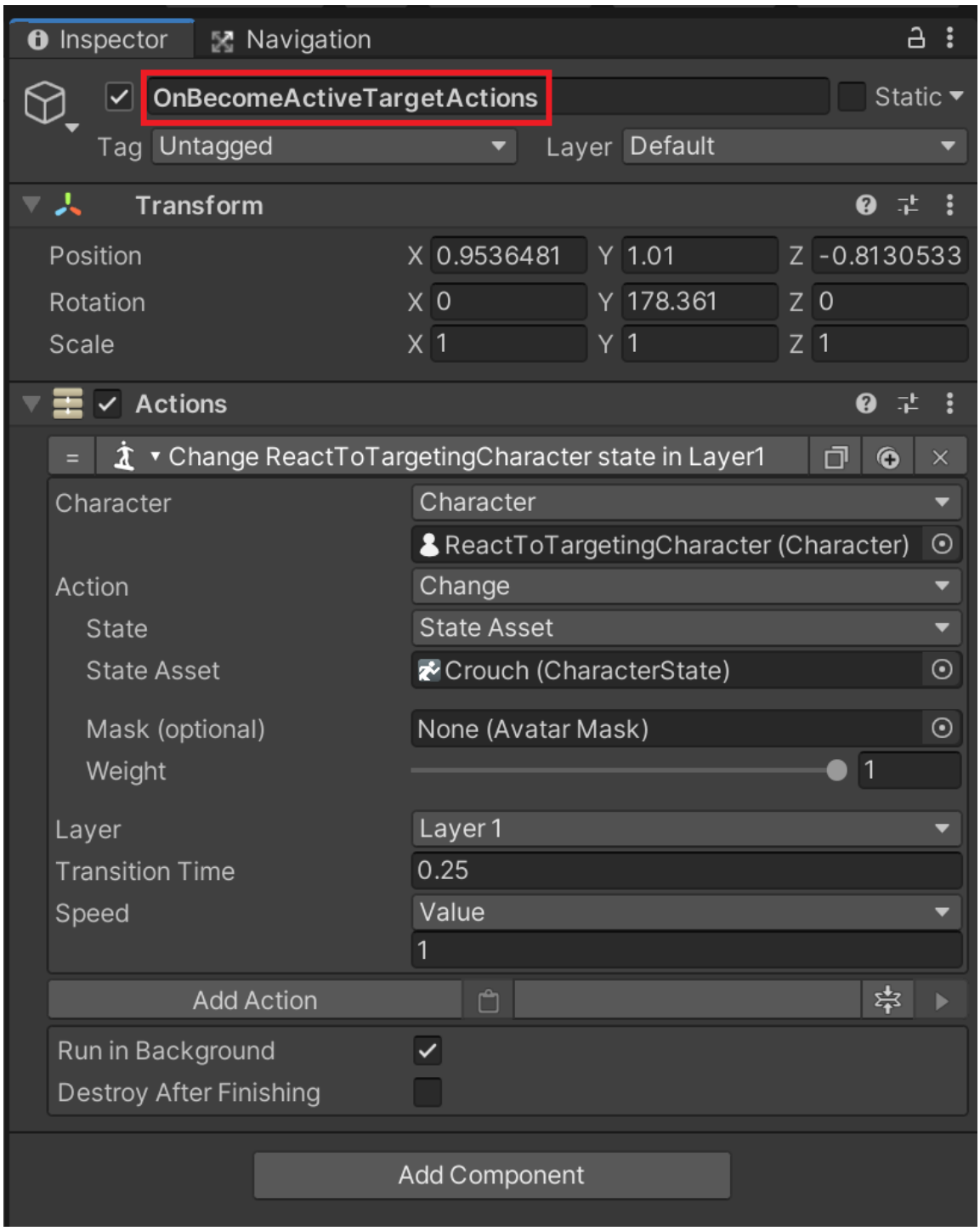
Targeting Actions

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).



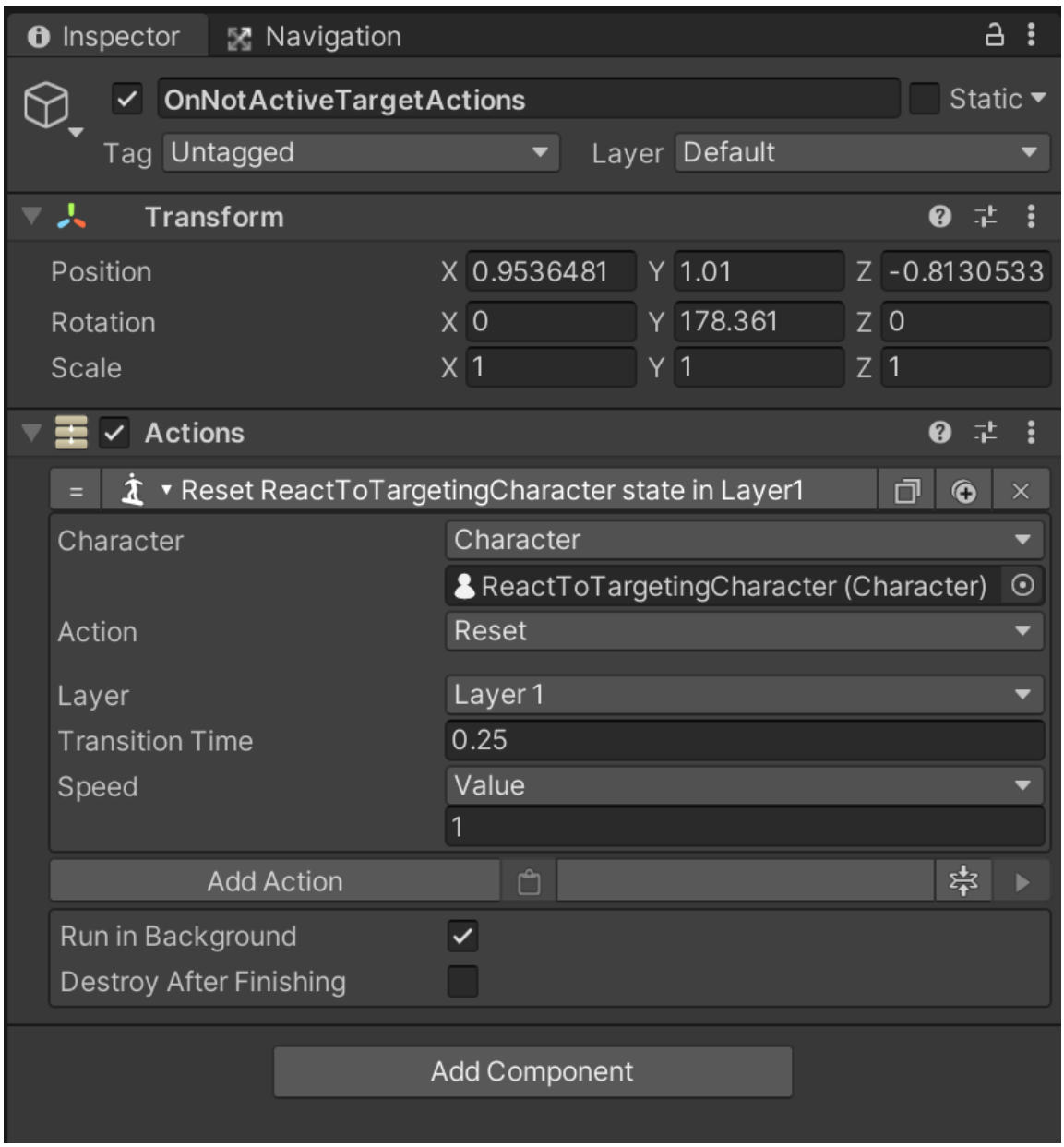
"On Become Active Target"

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.

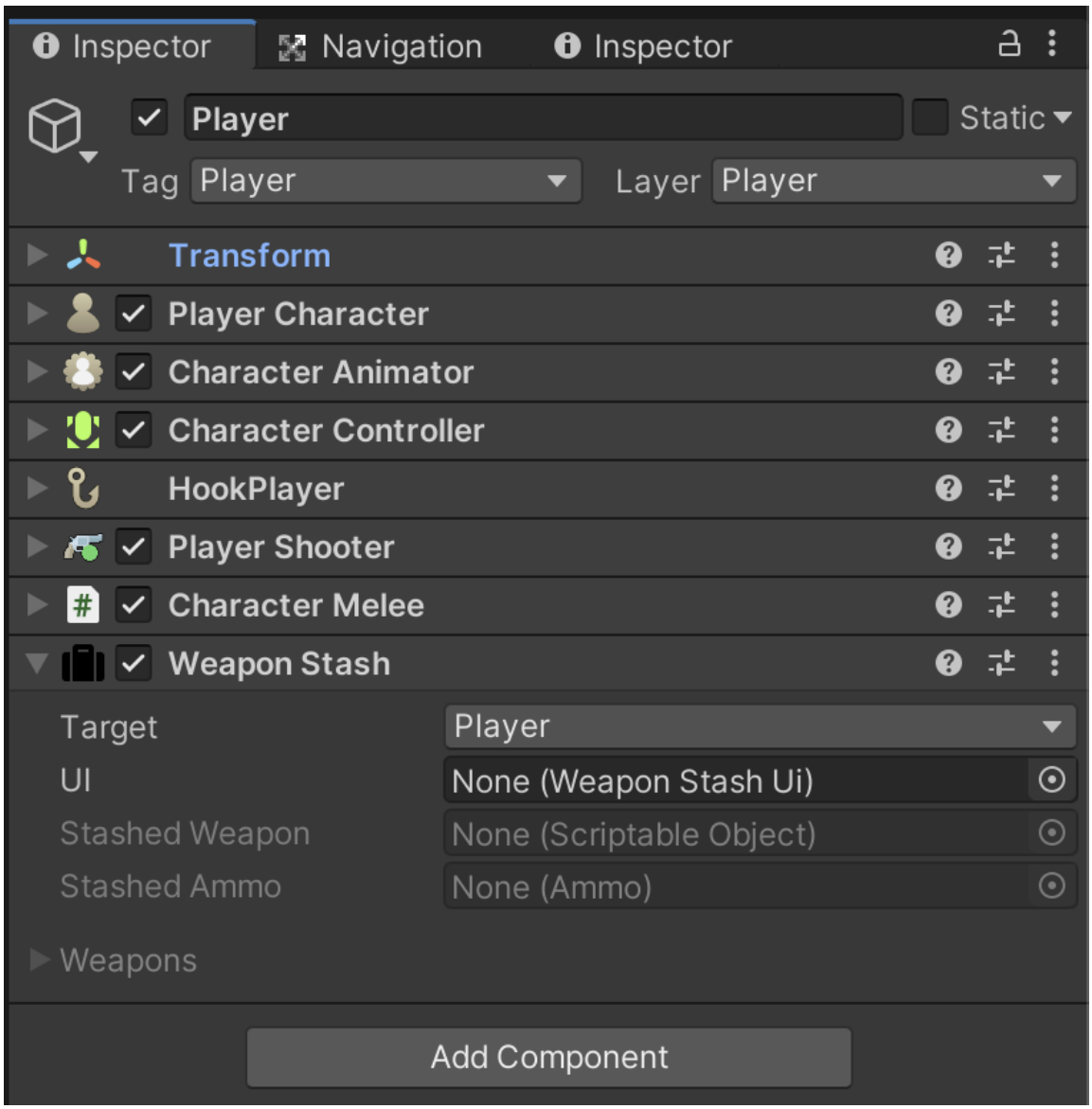

Inspector
Navigation
OnNotActiveTargetActions
Static
Tag
Untagged
Layer
Default
Transform
Position
X 0.9536481
Y 1.01
Z -0.8130533
Rotation
X 0
Y 178.361
Z 0
Scale
X 1
Y 1
Z 1
Actions
Reset ReactToTargetingCharacter state in Layer1
Character
Character
ReactToTargetingCharacter (Character)
Action
Reset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

Weapon Stashes

Weapon Stashes is a lightweight inventory system to assign weapons to a Player Character and allow the player to switch between them. It is compatible with both Shooter and Melee module weapons. Currently, only a weapon's default ammo is supported.

Weapon Stash Component

A **Weapon Stash** must be added to a game object (usually the player):



Adding a Weapon

There are two actions for adding weapons: Give Shooter Weapon and Give Melee Weapon. When configuring these actions, be sure to select the **Target Game Object** (usually the player) with a **Weapon Stash**.

Changing Weapons

The current weapon can be switched to the next (or previous) weapon in the stash with the **Cycle to Next Weapon** action.

Weapon Stash UI Component

The Weapon Stash UI component allows a stash's current weapon and ammo (if any) to be displayed on screen. This component differs the Shooter module's **AmmoUI** component in that it contains weapon information, not just ammo. Also, it will display the name of Melee module weapons (not just Shooter weapons).

The **Combat Examples** module includes a **WeaponStashUI** prefab that demonstrates how to use the component:

Homing Projectiles

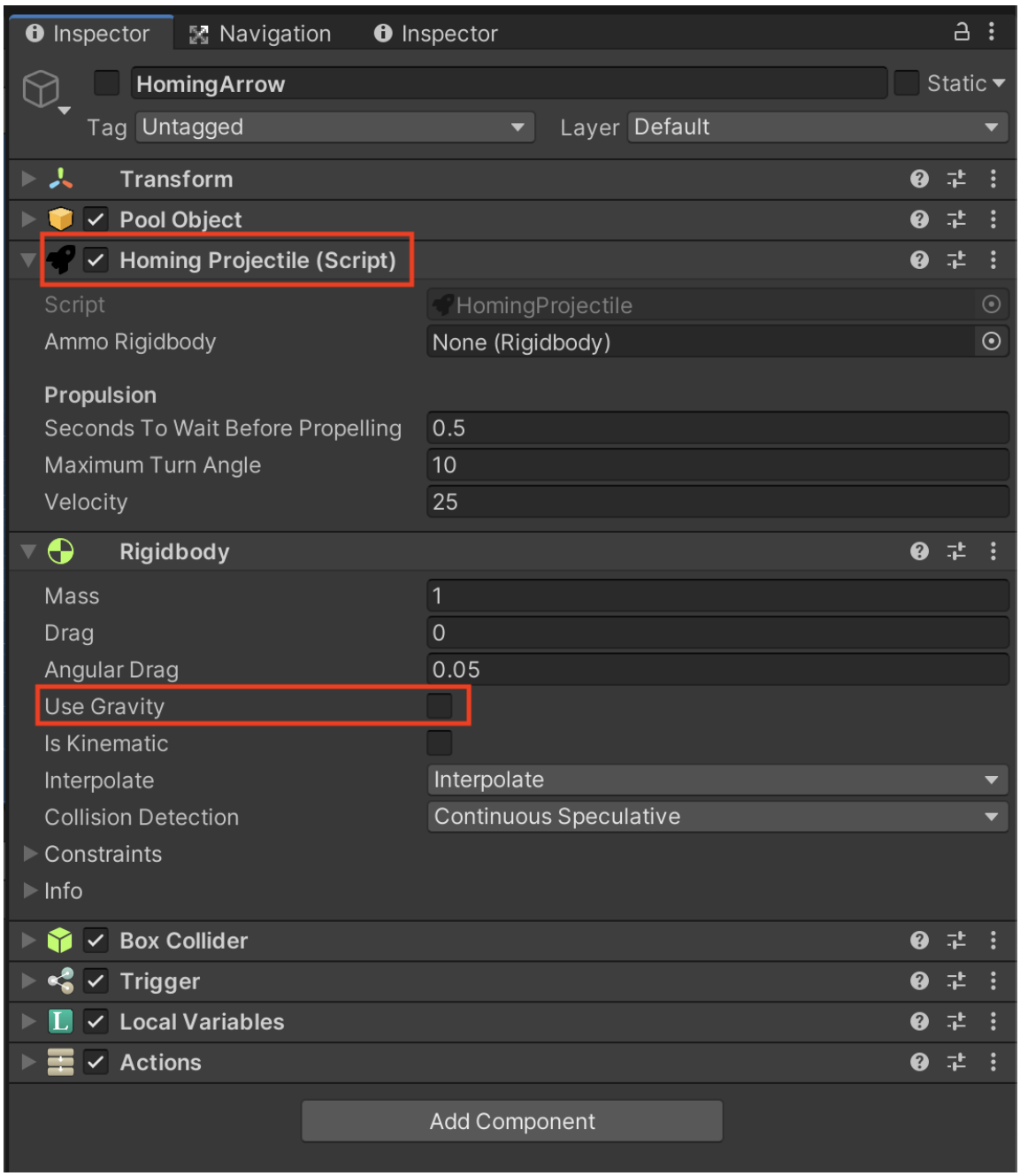
As the name suggests, a **Homing Projectile** seeks its target even if the weapon firing the projectile is not pointed directly at the intended target.

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

Roadmap

v0.5.0

- Melee targeting integration.
- Weapon Stashes (weapon carrying/switching feature).

v0.4.1

- Removed .blend files that caused an issue when Blender was not installed.

v0.4.0

- Targeting by visibility.
- Support for destructible targets.

v0.3.0

- Proximity-based weapon targeting.
- Target indicator.
- Homing projectile.

Possible Future Features

- Aim-assist.
- Headshots/body part targeting.
- Simplified death action for targetable characters.
- In editor weapon positioning.
- Proximity mines.

- Dual-wielding.
- When a target is defeated, switch to its nearest neighboring target (instead of closest to the player).
- Left/right target cycling (instead of nearest/farthest from player).
- Enemy spawn system.
- AI targeting.