Space Combat Kit

Radar Documentation

By VSXGames

Contents

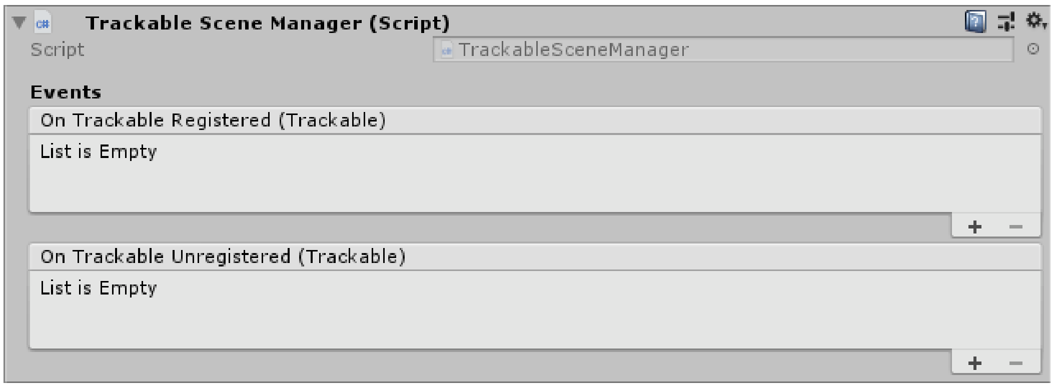
1. Creating A Radar

QUICK START

1) Add a **TrackablesSceneManager** script anywhere in the scene.
2) Add a **Tracker** component to the root transform of the vehicle that you wish to enable to track other targets. Customize the settings in the inspector.
3) You can now access the tracked targets in the **Tracker**'s *Targets* list via code.

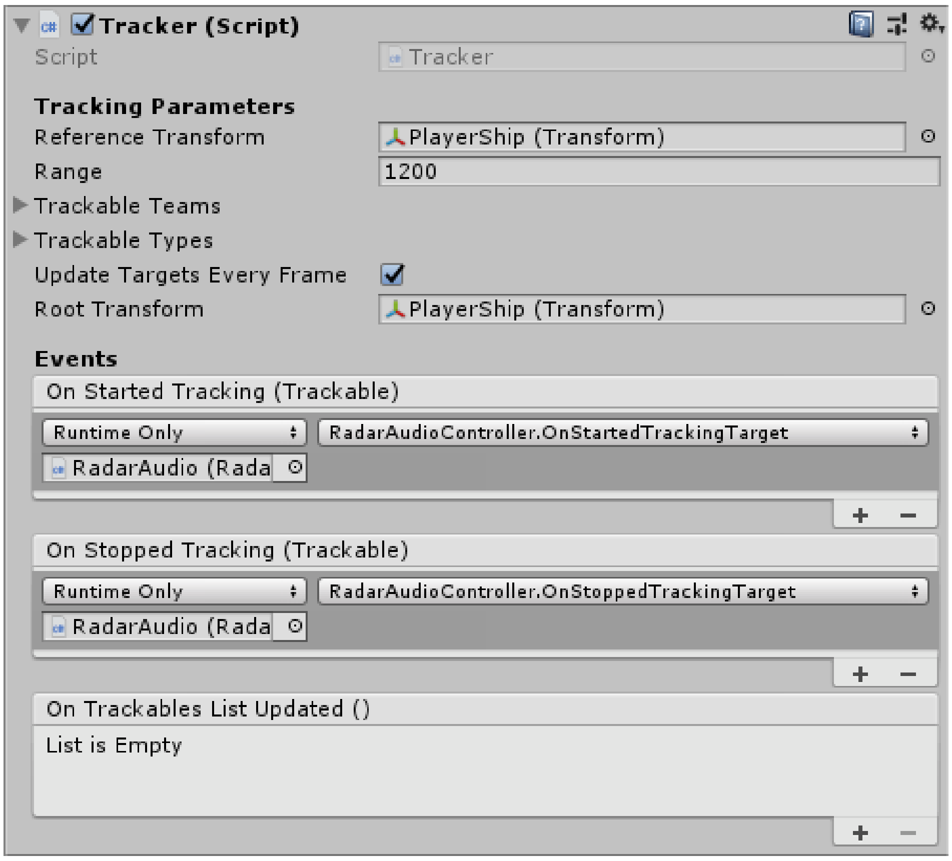
The Trackables Scene Manager component provides a way to efficiently keep a record of all trackable objects in the scene. All trackable objects register with this component when they are created.

A Trackables Scene Manager is a singleton class, and one must exist in every scene that involves target tracking.


Trackable Scene Manager (Script)
Script TrackableSceneManager
Events
On Trackable Registered (Trackable)
List is Empty
On Trackable Unregistered (Trackable)
List is Empty

- **On Trackable Registered (Unity Event)**: Event called when a new trackable is registered in the scene.
- **On Trackable Unregistered (Unity Event)**: Event called when a trackable is removed from the scene.

A Tracker component represents a radar which tracks targets in the scene according to its tracking parameters, and stores them in a list.


Tracker (Script)

Script: Tracker

Tracking Parameters

Reference Transform: PlayerShip (Transform)

Range: 1200

Trackable Teams

Trackable Types

Update Targets Every Frame: ✓

Root Transform: PlayerShip (Transform)

Events

On Started Tracking (Trackable)

Runtime Only | RadarAudioController.OnStartedTrackingTarget

RadarAudio (Rada

On Stopped Tracking (Trackable)

Runtime Only | RadarAudioController.OnStoppedTrackingTarget

RadarAudio (Rada

On Trackables List Updated ()

List is Empty

- **Reference Transform**: The transform that represents the radar origin, for example when calculating if a target is in range.
- **Range**: The tracking range, in Unity units.
- **Trackable Teams**: All the teams that this radar is capable of tracking.
- **Trackable Types**: All the types of targets that this radar can track.
- **Update Targets Every Frame**: Whether this radar is updated every frame (can be unchecked and updated externally at a different frequency, e.g. for performance reasons).
- **Root Transform**: This is used for identification purposes, to prevent the radar from tracking itself when it is also set up as a target for other radars.
- **On Started Tracking (Unity Event)**: Event called when a new target is tracked by the radar. If the radar loses the target and then re-acquires it, this will be called again.
- **On Stopped Tracking (Unity Event)**: Event called when a target stops being tracked by this radar.
- **On Trackables List Updated (Unity Event)**: Event called when the radar 'scans' and updates its targets list.

At this point, the Tracker will be able to track targets in the scene. However, there are no targets to track yet. In the next section, we'll create a target.

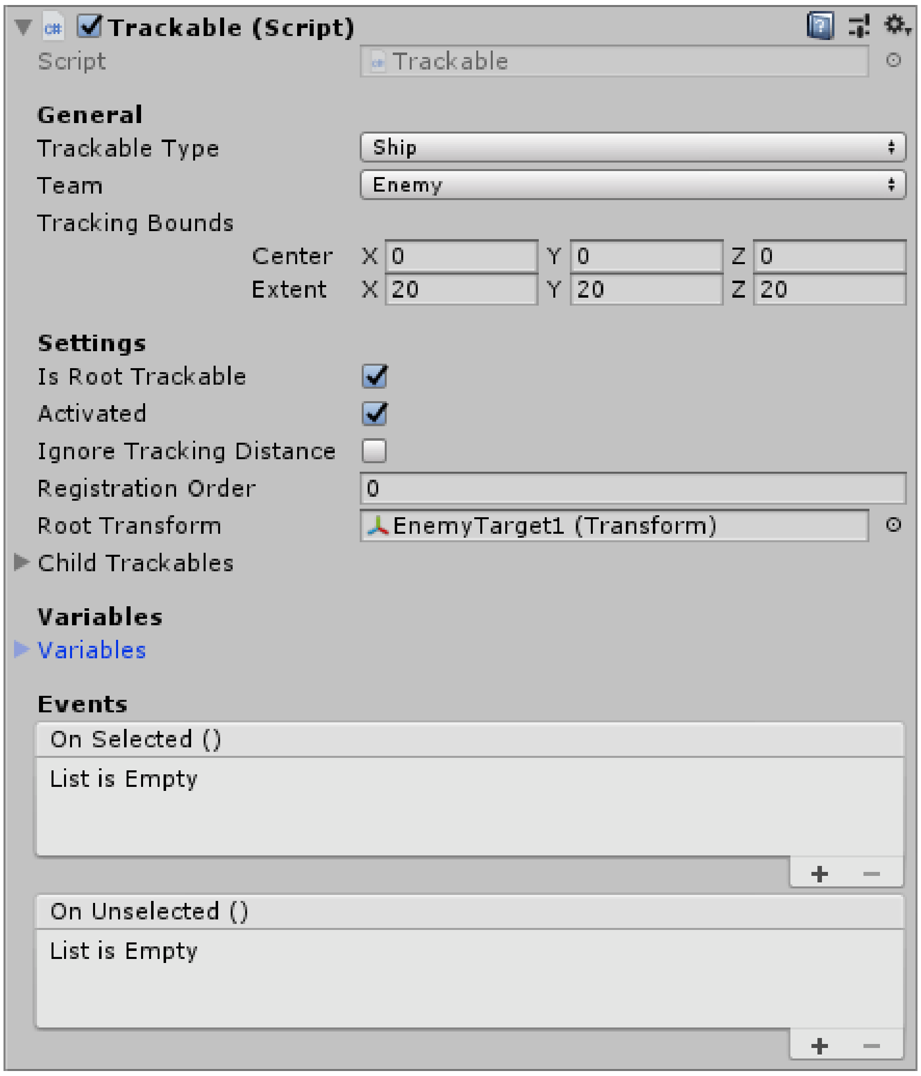
2. Creating A Target

2.1 The Trackable Component

QUICK START

1. Add a **Trackable** component to the gameobject you wish to track.

2. Customize the inspector settings.



Trackable (Script)

Script: Trackable

General

Trackable Type: Ship

Team: Enemy

Tracking Bounds

Center X 0 Y 0 Z 0

Extent X 20 Y 20 Z 20

Settings

Is Root Trackable: ✓

Activated: ✓

Ignore Tracking Distance: ☐

Registration Order: 0

Root Transform: EnemyTarget1 (Transform)

Child Trackables

Variables

Variables

Events

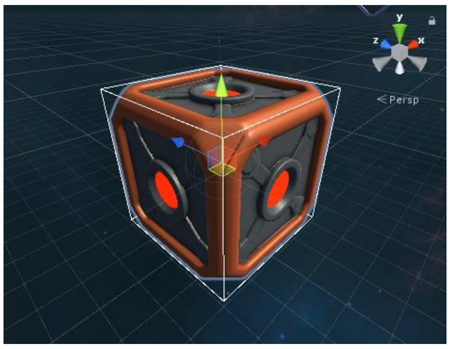
On Selected ()

List is Empty

On Unselected ()

List is Empty

- **Trackable Type**: The type of target this is.
- **Team**: The team that this target belongs to.
- **Tracking Bounds**: The rectangular prism that defines the shape of this target for tracking purposes (e.g. when using size-responsive target boxes). You can see the bounds in the editor as a white box.



- **Is Root Trackable**: Whether this trackable is the root parent trackable, or a child trackable that is part of a bigger object (e.g. a subsystem on a ship).
- **Activated**: Whether this trackable is active.

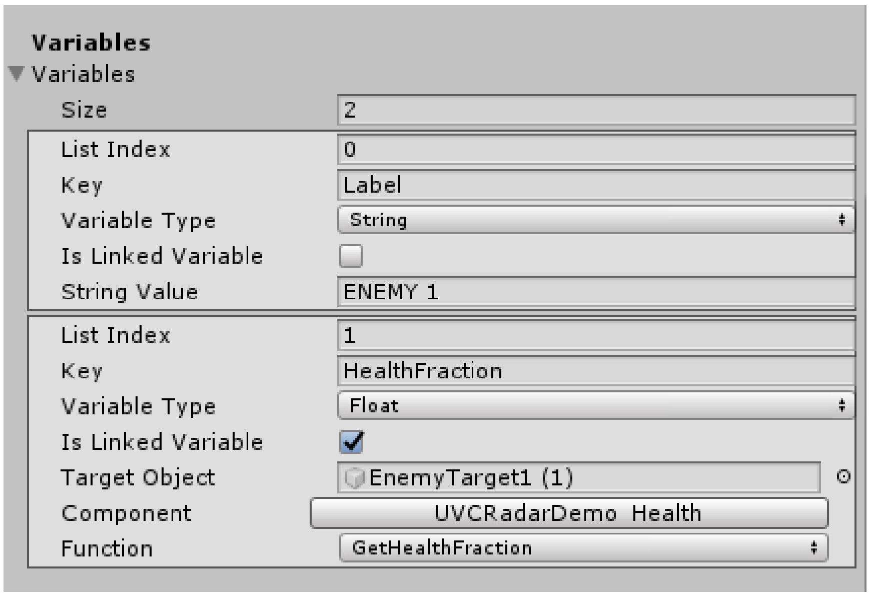
- **Ignore Tracking Distance**: When checked, this enables Trackers to track this target at any distance. Useful for waypoints etc.
- **Registration Order**: The position in the list of targets stored in the Trackables Scene Manager component. Defines the order of target selection when cycling back and forth.
- **Root Transform**: Used to identify the object and gain access its vital components.
- **Child Trackables**: The child trackables (e.g. vehicle subsystems or unique damageable parts that can be selected/targeted individually).
- **Variables**: All the variables that must be able to be accessed on the target (e.g. name, health etc). This powerful feature will be discussed in more detail in the next section.
- **On Selected (Unity Event)**: Event called when this object is selected (e.g. to add an emissive highlight).
- **On Unselected (Unity Event)**: Event called when this object is unselected (e.g. to switch off the emissive highlight).

2.2 Adding Target Information

When tracking a target, it is useful to access information on it without having to search for a specific component. This is straightforward with static (non-changing) variables such as the target name, simply by adding it as a property to the Trackable component, but how about accessing something like the current health, which can change over time?

This radar system comes with a powerful Linkable Variable system where you can add functions from other components as variables on the Trackable component, accessing target data via the Variables dictionary without having to look through other components.

In the following image, you'll see the Variables list in the inspector of a Trackable component, showing how you can add variables directly (such as the target label) or add functions (in this case the *GetHealthFraction* function from the *UVCRadarDemo_Health* component on the target).


Variables
▼ Variables

Size	2
List Index	0
Key	Label
Variable Type	String
Is Linked Variable	☐
String Value	ENEMY 1
List Index	1
Key	HealthFraction
Variable Type	Float
Is Linked Variable	✓
Target Object	EnemyTarget1 (1)
Component	UVCRadarDemo Health
Function	GetHealthFraction

(The Trackable component inspector)

- **List Index**: The position of this variable in the list (used to remove items by putting them at the end of the list and shortening the list).
- **Key**: The string key that is used to access the variable.

- **Variable Type**: The type of variable, currently supported are Unity Objects, Booleans, Integers, Floats, Strings and Vector3s.
- **Is Linked Variable**: Whether the variable is linked as a function, or set directly here in the inspector.
- **Target Object**: The gameobject with the script that has the function to be linked as a variable.
- **Component**: The component with the function to be linked as a variable.
- **Function**: The function which returns the desired data.

Now if you have a target reference, you can get the information by accessing the *Variables* dictionary on it, and searching for the key value. This returns an instance of the *LinkableVariable* class, which has functions such as *ObjectValue(), BoolValue(), IntValue()* etc to return the data you need.

3. Target Selection

QUICK START

1) Add a **TrackerTargetSelector** component to the vehicle that you wish to be able to select targets with.

2) Open the inspector and drag the **Tracker** component you added earlier into the *Tracker* field.

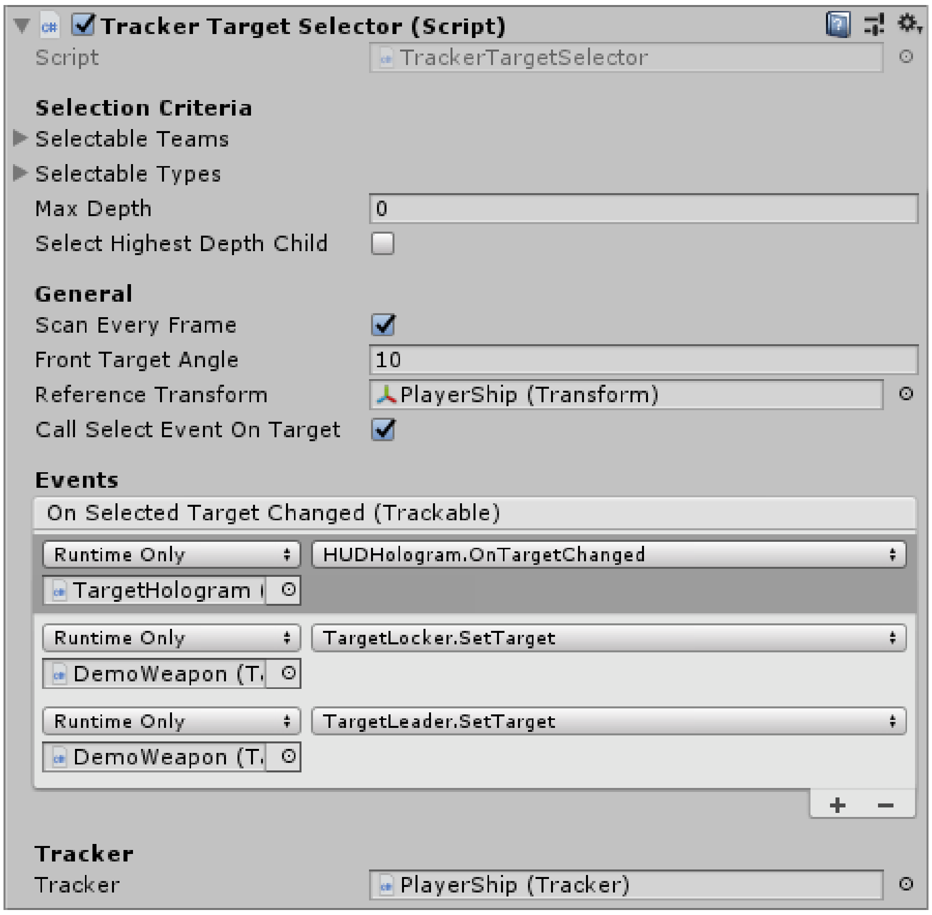
3) Customize the rest of the settings in the inspector.

A **Tracker Target Selector** component selects targets from the *Targets* list of a **Tracker** component.

To select between targets, this component has a variety of functions you can call:

- **GetFirstSelectableTarget()**: returns the first target in the list that is selectable.
- **Cycle(bool forward)**: Cycle forward or (if forward = false) backward through the targets list.
- **SelectNearest()**: Select the nearest target.
- **SelectFront()**: Select the target in front.

The currently selected target can be accessed via the *SelectedTarget* accessor.


Tracker Target Selector (Script)

Script	TrackerTargetSelector
Selection Criteria	
Selectable Teams	
Selectable Types	
Max Depth	0
Select Highest Depth Child	☐
General	
Scan Every Frame	✓
Front Target Angle	10
Reference Transform	PlayerShip (Transform)
Call Select Event On Target	✓
Events	
On Selected Target Changed (Trackable)	
Runtime Only / TargetHologram	HUDHologram.OnTargetChanged
Runtime Only / DemoWeapon (T	TargetLocker.SetTarget
Runtime Only / DemoWeapon (T	TargetLeader.SetTarget
Tracker	
Tracker	PlayerShip (Tracker)

- **Selectable Teams**: The teams that this target selector can select.
- **Selectable Types**: The types of trackables that this target selector can select.
- **Max Depth**: When set to 0, this selector will only select root Trackables. Values above this will enable it to select child trackables (e.g. subsystems) at different levels of the hierarchy.
- **Select Highest Depth Child**: Automatically looks for the child at the deepest level of the target's Trackable hierarchy.
- **Scan Every Frame**: Whether to scan for a new target every frame when none is selected.
- **Front Target Angle**: The maximum angle to target that is allowed when selecting the target in front.
- **Reference Transform**: The reference transform used for searching the target in Front, or the Nearest target.
- **Call Select Event On Target**: Whether to call the target's OnSelected method when that target becomes selected.
- **On Selected Target Changed (Unity Event)**: Event called when the selected target changes.
- **Tracker**: The Tracker component to get the list of targets to select from.

4. Target Locking

QUICK START

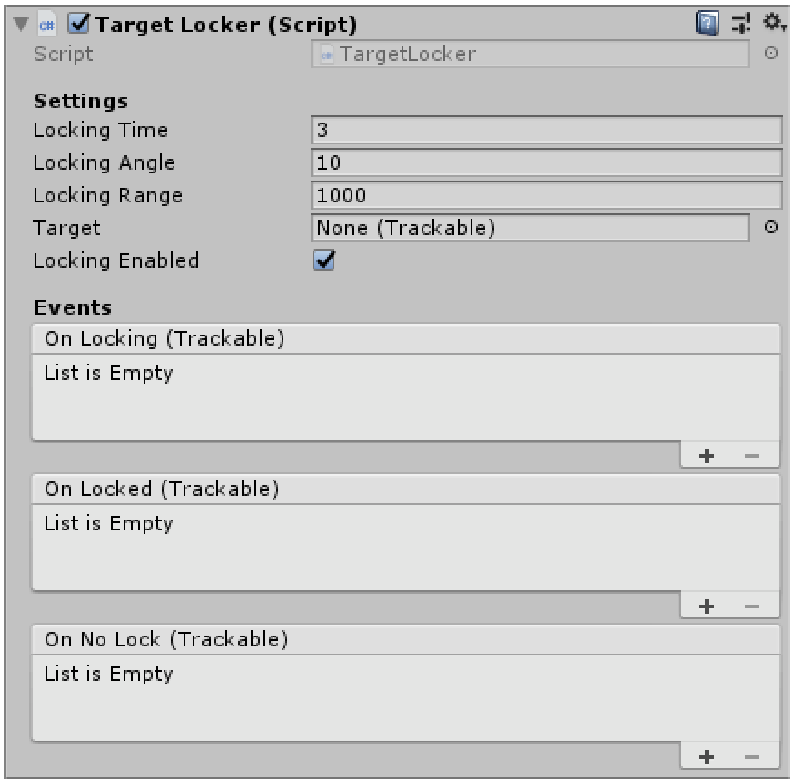
1) Add a **Target Locker** component to the object (e.g. a missile weapon) that should lock onto a selected target.

2) Customize the settings in the inspector.

3) Add the **Target Locker**'s *SetTarget* method to the *On Selected Target Changed* event in the inspector of the **Tracker Target Selector** component you added earlier.

A target locker represents something (e.g. a missile weapon) that locks onto a target based on its distance and relative position. A target locker has 3 states: *Locking*, *Locked* and *NoLock*.

The current state can be accessed via the *LockState* accessor.



Target Locker (Script)

Script	TargetLocker
Settings	
Locking Time	3
Locking Angle	10
Locking Range	1000
Target	None (Trackable)
Locking Enabled	✓

Events

On Locking (Trackable)
List is Empty

On Locked (Trackable)
List is Empty

On No Lock (Trackable)
List is Empty

- **Locking Time**: How long it takes to lock onto the target once it becomes lockable.
- **Locking Angle**: The angle from the forward vector that the target must be within to be able to be locked onto.
- **Locking Range**: The maximum distance to target within which the target can be locked onto.
- **Target**: The currently selected target (may be set manually when not using a target selector).
- **Locking Enabled**: Toggle for enabling/disabling locking (may be set via code too).
- **On Locking (Unity Event)**: Event called when this Target Locker starts locking onto a target.
- **On Locked (Unity Event)**: Event called when this Target Locker becomes locked onto a target.
- **On No Lock (Unity Event)**: Event called when target lock is lost after the locking sequence begins or is completed.

5. Target Leading

QUICK START

1) Add a **Target Leader** component to the object that requires a lead target position calculation.

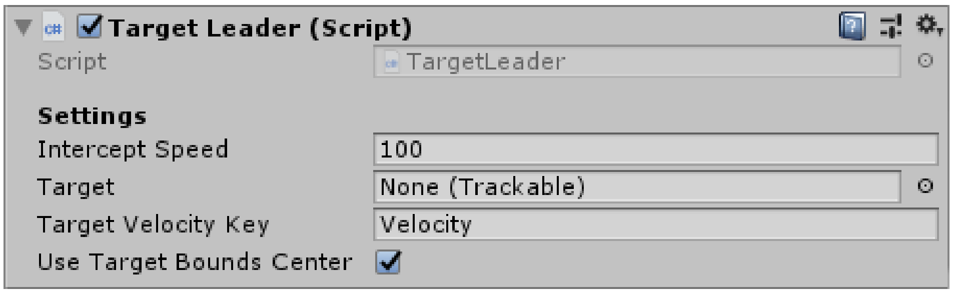
2) Customize the settings in the inspector.

3) Add the **Target Leader**'s *SetTarget* method to the *On Selected Target Changed* event in the inspector of the **Tracker Target Selector** component you added earlier.

Target leading refers to the calculation of a 'lead target position' which represents a position to aim at to intercept a target with a specific speed. This is useful for hitting moving targets with a projectile weapon.

The Target Leader script represents something (such as a projectile weapon) which requires calculation of an intercept position to aim at to hit the target with a specified intercept speed.

The current lead target position can be accessed via the *LeadTargetPosition* accessor.


Target Leader (Script)
Script: TargetLeader
Settings
Intercept Speed: 100
Target: None (Trackable)
Target Velocity Key: Velocity
Use Target Bounds Center: ✓

- **Intercept Speed**: The intercept speed to use when calculating the lead target position.
- **Target**: May be manually set when not using a target selector.
- **Target Velocity Key**: The string key to use when accessing the velocity from the target **Trackable**'s *Variables* dictionary.
- **Use Target Bounds Center**: Whether to use the center of the target bounds defined in the target's **Trackable** component, or to use its transform position directly.

6. Target Boxes

6.1 Setting Up Target Boxes

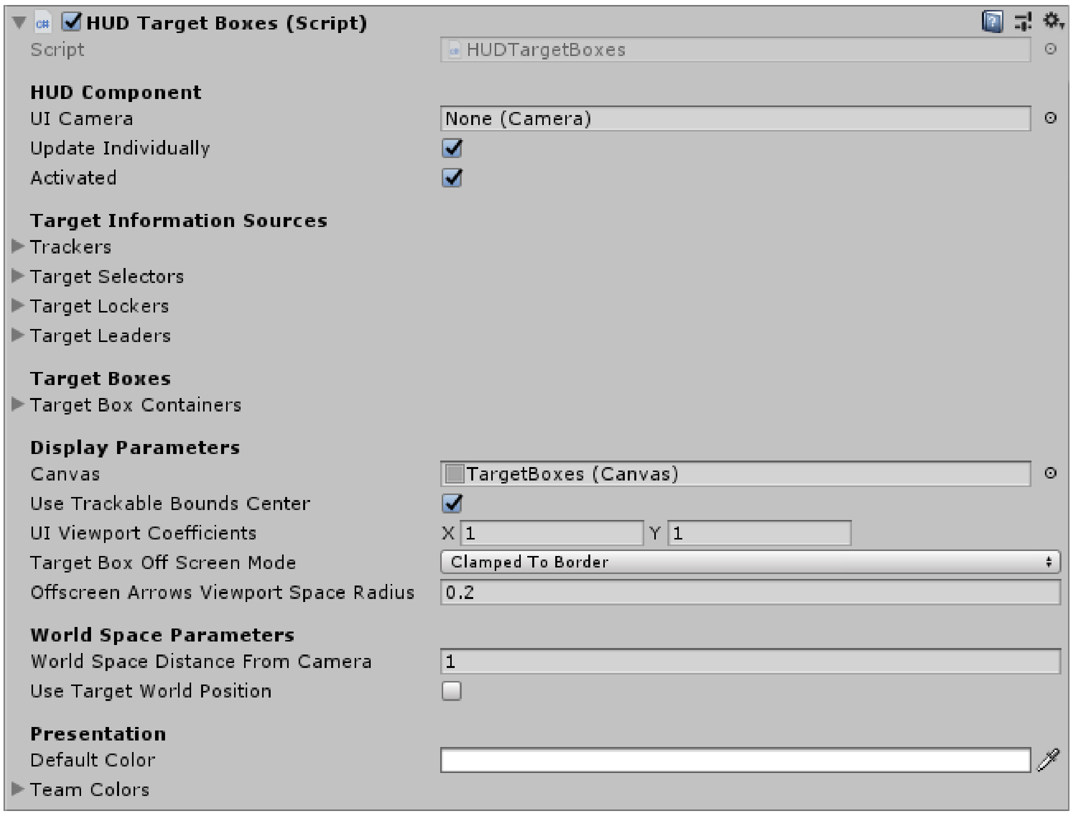
QUICK START

1) Add a UI Canvas to the scene.

2) Add a HUDTargetBoxes component to it and customize the settings in the inspector.

The HUD Target Boxes component manages the display of target boxes over targets on the screen.

For World Space target boxes (such as for a VR game), simply switch the Canvas render mode to World Space. The target boxes will automatically switch over to the new mode.



HUD Target Boxes (Script)

Script	HUDTargetBoxes
HUD Component	
UI Camera	None (Camera)
Update Individually	✓
Activated	✓
Target Information Sources	
▶ Trackers	
▶ Target Selectors	
▶ Target Lockers	
▶ Target Leaders	
Target Boxes	
▶ Target Box Containers	
Display Parameters	
Canvas	TargetBoxes (Canvas)
Use Trackable Bounds Center	✓
UI Viewport Coefficients	X 1 Y 1
Target Box Off Screen Mode	Clamped To Border
Offscreen Arrows Viewport Space Radius	0.2
World Space Parameters	
World Space Distance From Camera	1
Use Target World Position	☐
Presentation	
Default Color	⬜
▶ Team Colors	

- **UI Camera**: The camera used to display the UI elements.
- **Update Individually**: Whether to update this part of the HUD individually or do it externally using the HUDManager component.
- **Activated**: Whether this HUD component is activated or not.
- **Trackers**: All the Tracker components to get targets from.
- **Target Selectors**: All the target selectors (selected targets may be displayed differently on the HUD).
- **Target Lockers**: All the target lockers to get locking information from.
- **Target Leaders**: All the target leaders to get lead target data from.
- **Target Box Containers**: A target box container is used to associate a target box with a specific type of trackable object.
- **Canvas**: The canvas used to display target boxes on.
- **Use Trackable Bounds Center**: Whether to center the target boxes on the target Trackable's bounds center, or on the Trackable's transform position.
- **UI Viewport Coefficients**: The amount of the viewport used to display target boxes in (VR games may require it to be reduced).
- **Target Box Off Screen Mode**: How to display the off-screen arrows pointing at targets (may be clamped to screen border, or in a circular pattern around the screen center).
- **Offscreen Arrows Viewport Space Radius**: The radius (in viewport space) of the circular pattern of arrows pointing at off-screen targets.
- **World Space Distance From Camera**: The distance from the camera to display the target boxes when using a World Space canvas.
- **Use Target World Position**: Whether to position target boxes at the same world position as the target (scaled up and down to maintain the same apparent size).
- **Default Color**: The default color for target boxes.

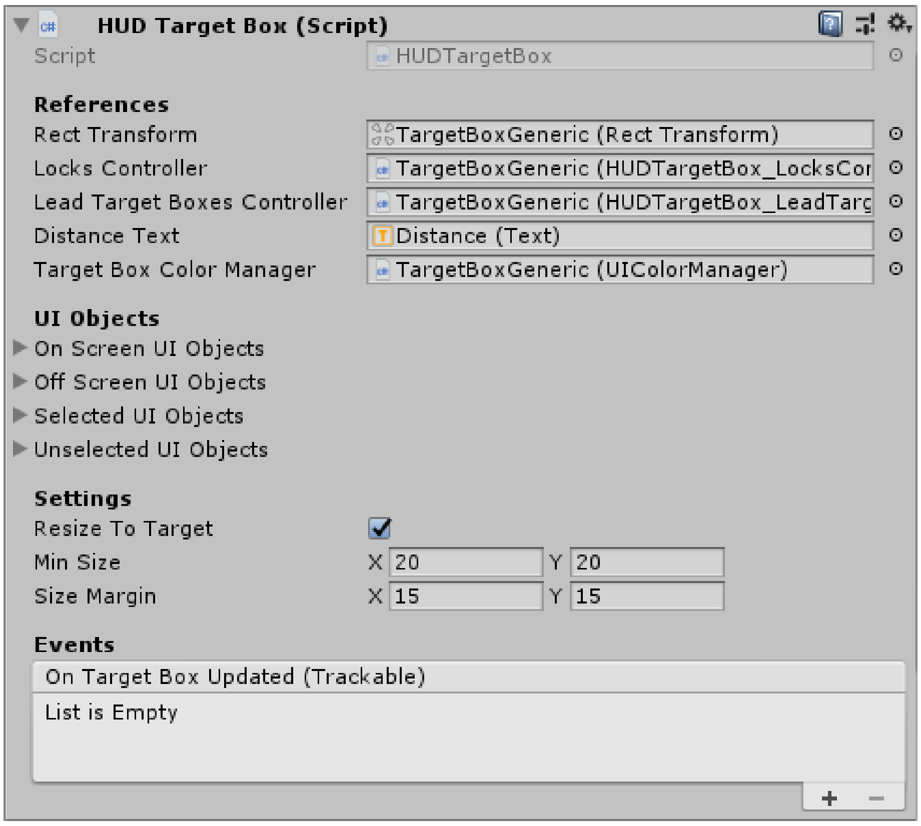
- **Team Colors**: The colors of the target box for each team (uses default color for any team not appearing in this list).

6.2 Creating A Target Box

QUICK START

1) Create a new empty UI object.
2) Add a HUDTargetBox component.
3) Customize the settings in the inspector, adding UI elements to your target box as desired.

Target boxes are fully customizable and are represented by a UI object with a HUDTargetBox component attached.



HUD Target Box (Script)

Script	HUDTargetBox
References	
Rect Transform	TargetBoxGeneric (Rect Transform)
Locks Controller	TargetBoxGeneric (HUDTargetBox_LocksCor
Lead Target Boxes Controller	TargetBoxGeneric (HUDTargetBox_LeadTarc
Distance Text	Distance (Text)
Target Box Color Manager	TargetBoxGeneric (UIColorManager)
UI Objects	
On Screen UI Objects	
Off Screen UI Objects	
Selected UI Objects	
Unselected UI Objects	
Settings	
Resize To Target	✓
Min Size	X 20 Y 20
Size Margin	X 15 Y 15
Events	
On Target Box Updated (Trackable)	List is Empty

- **Rect Transform**: The root rect transform of the target box.
- **Locks Controller**: A reference to the HUDTargetBox_LocksController component that controls locking UI.
- **Lead Target Boxes Controller**: A reference to the HUDTargetBox_LeadTargetBoxesController component that controls display of lead target box UI.
- **Distance Text**: The text for displaying the distance to the target.
- **Target Box Color Manager**: A reference to the UIColorManager component that manages the color of the different parts of the target box.
- **On Screen UI Objects**: These gameobjects will be activated when the target is on screen.
- **Off Screen UI Objects**: These gameobjects will be activated when the target is off screen.

- **Selected UI Objects**: These objects will be activated when the target is selected by a target selector.
- **Unselected UI Objects**: These objects will be activated when a target is not selected by any target selector.
- **Resize To Target**: Whether the target box should conform to the size of the target, or remain a constant size.
- **Min Size**: The minimum size for the target box (e.g. used when the target is very far away and very small).
- **Size Margin**: The padding added to the size of the target box.
- **On Target Box Update (Unity Event)**: Event called when the target information for this target box is updated. Used to update other components with target information.

7. HUD Radar

7.1 Setting Up HUD Radar

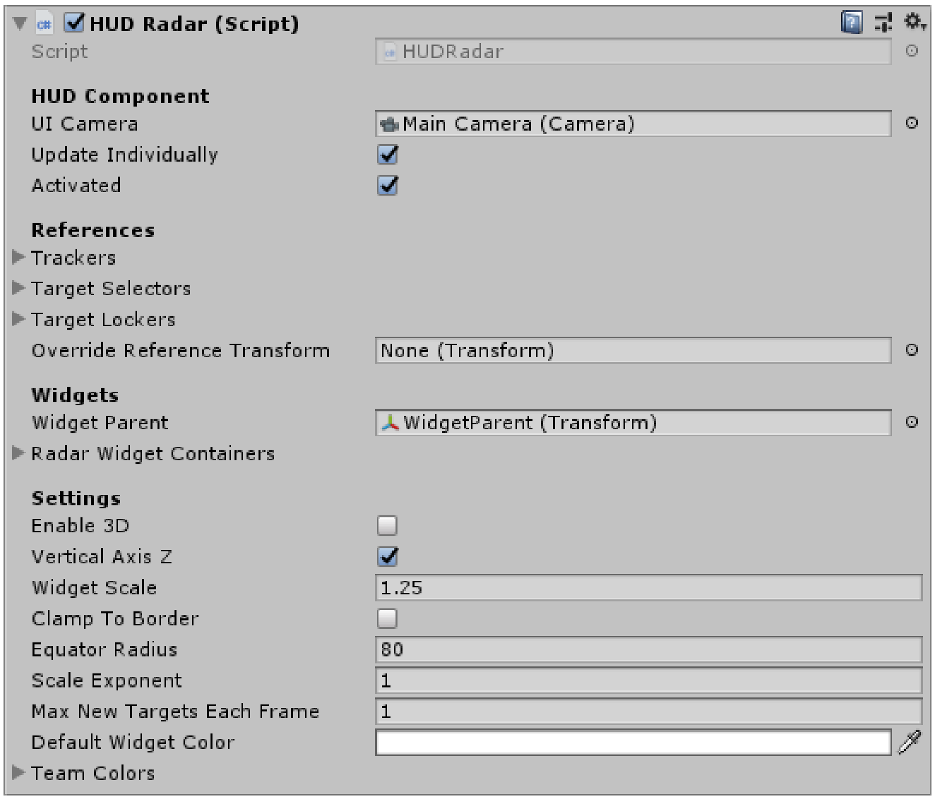
QUICK START

1) Create a new world-space UI Canvas.

2) Add a HUDRadar component.

3) Customize the settings in the inspector.

The HUD Radar can easily switch between 2D and 3D modes depending on your needs.



HUD Radar (Script)

Script	HUDRadar
HUD Component	
UI Camera	Main Camera (Camera)
Update Individually	✓
Activated	✓
References	
▶ Trackers	
▶ Target Selectors	
▶ Target Lockers	
Override Reference Transform	None (Transform)
Widgets	
Widget Parent	WidgetParent (Transform)
▶ Radar Widget Containers	
Settings	
Enable 3D	☐
Vertical Axis Z	✓
Widget Scale	1.25
Clamp To Border	☐
Equator Radius	80
Scale Exponent	1
Max New Targets Each Frame	1
Default Widget Color	
▶ Team Colors	

- **UI Camera**: The camera used to render the UI for the HUD Radar.
- **Update Individually**: Whether to update the HUD Radar every frame, or update externally from a different script.

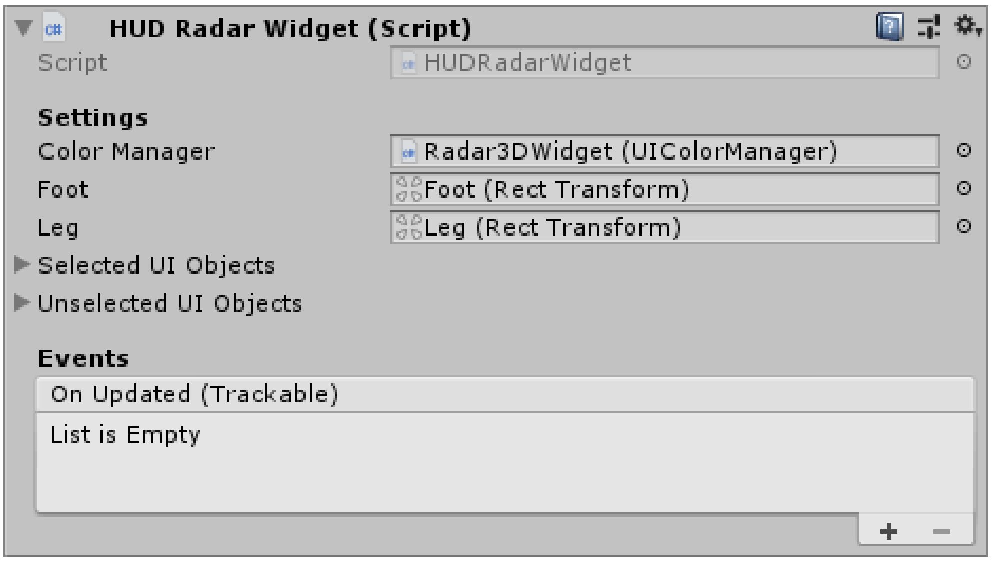
- **Activated**: Whether this HUD component is activated.
- **Trackers**: All the Tracker components to get targets from.
- **Target Selectors**: All the target selectors (selected targets may be displayed differently on the HUD).
- **Target Lockers**: All the target lockers to get locking information from.
- **Override Reference Transform**: The override reference transform for calculating the widget position, will use Tracker's Reference Transform if not provided.
- **Widget Parent**: The parent for each of the radar widgets (the UI objects representing each target).
- **Radar Widget Containers**: Each radar widget container associates a radar widget prefab with a type of target.
- **Enable 3D**: Whether the radar is 3D (leave unchecked for 2D).
- **Vertical Axis Z**: Whether the radar widget should be oriented with its Z axis 'up'. Most cases leave checked for 3D, unchecked for 2D.
- **Widget Scale**: The local scale for the radar widgets.
- **Clamp To Border**: Whether to clamp tracked targets that are currently outside the display range to the edge (leave unchecked to simply make targets disappear outside of the display range).
- **Equator Radius**: The radius of the 3D radar plane.
- **Scale Exponent**: The exponent for displaying target position exponentially (leave at 1 for linear display).
- **Max New Targets Each Frame**: The number of new targets shown can be limited to prevent lag (for example at the beginning of a scene with many targets).
- **Default Widget Color**: The default color for the radar widgets.
- **Team Colors**: The colors for each of the teams (default color is used when a team is not included in the list).

7.2 Creating A New Radar Widget

QUICK START

1) Create a new empty UI object.
2) Add a HUDRadarWidget component.
3) Customize the settings in the inspector, adding UI elements to the widget as desired.

A radar widget is represented by a UI object with a HUDRadarWidget component attached.


HUD Radar Widget (Script)

Script HUDRadarWidget

Settings

Color Manager Radar3DWidget (UIColorManager)

Foot Foot (Rect Transform)

Leg Leg (Rect Transform)

Selected UI Objects

Unselected UI Objects

Events

On Updated (Trackable)

List is Empty

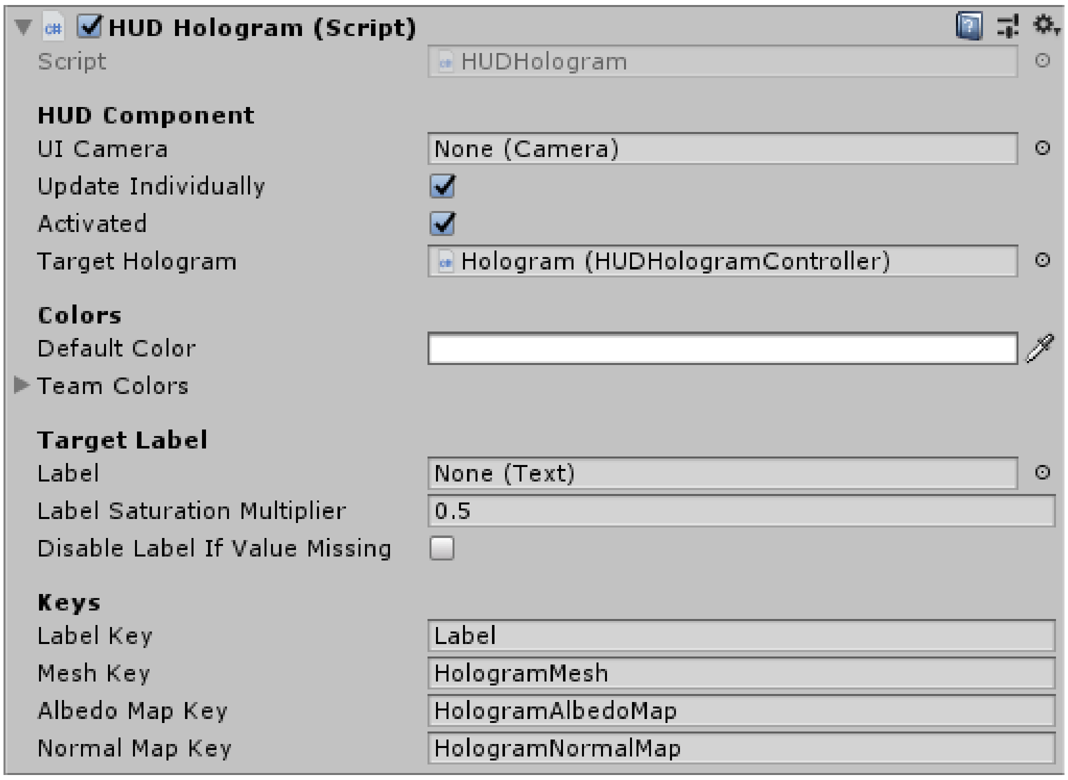
- **Color Manager**: The UI Color Manager component that controls the color of the UI elements on this widget.
- **Foot**: The foot UI for a 3D widget.
- **Leg**: The leg UI for a 3D widget.
- **Selected UI Objects**: A list of gameobjects to activate when the widget is showing a selected target (deactivate if not selected).
- **Unselected UI Objects**: A list of gameobjects to activate when the widget is showing a target not selected by a target selector (deactivated if target is selected).
- **On Updated (Unity Event)**: Event called when the widget is updated with target information.

8. HUD Target Hologram

8.1 Setting Up The HUD Target Hologram

QUICK START

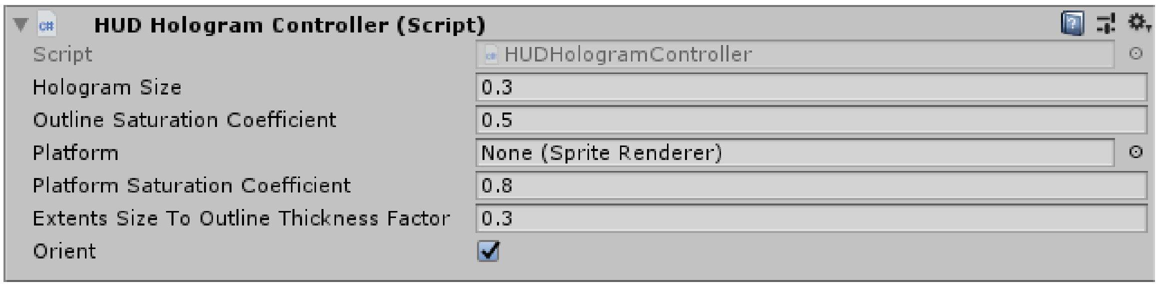
1) Create a new gameobject and add a **HUD Hologram** component.
2) Create a new gameobject that represents where the target hologram will be shown, and add a **HUD Hologram Controller** component.
3) Add the **HUD Hologram Controller** component into the *Target Hologram* field in the inspector of the **HUD Hologram** component.
4) Customize the settings of both scripts in their inspectors.



HUD Hologram (Script)

Script	HUDHologram
HUD Component	
UI Camera	None (Camera)
Update Individually	✓
Activated	✓
Target Hologram	Hologram (HUDHologramController)
Colors	
Default Color	
Team Colors	
Target Label	
Label	None (Text)
Label Saturation Multiplier	0.5
Disable Label If Value Missing	☐
Keys	
Label Key	Label
Mesh Key	HologramMesh
Albedo Map Key	HologramAlbedoMap
Normal Map Key	HologramNormalMap

- **UI Camera**: The camera used to render thus HUD component.
- **Update Individually**: Whether to update this HUD component on its own (leave unchecked if updating externally).
- **Activated**: Whether this HUD component is activated.
- **Target Hologram**: A reference to the HUD Hologram Controller that controls the target hologram.
- **Default Color**: The default color for the hologram.
- **Team Colors**: The hologram color for each team (default color is used for any team that doesn't appear in the list).
- **Label**: The text that shows the label of the trackable.
- **Label Saturation Multiplier**: Increase or decrease the saturation of the label.
- **Disable If Value Missing**: Whether to disable the label text if label information is not found on the target.
- **Label Key**: The dictionary key to access the label information from the *Variables* dictionary on the target's **Trackable** component.
- **Mesh Key**: The dictionary key to access the hologram mesh from the *Variables* dictionary on the target's **Trackable** component.
- **Albedo Map Key**: The dictionary key to access the albedo map for the hologram from the *Variables* dictionary on the target's **Trackable** component.
- **Normal Map Key**: The dictionary key to access the normal map for the hologram from the *Variables* dictionary on the target's **Trackable** component.



HUD Hologram Controller (Script)

Script	HUDHologramController
Hologram Size	0.3
Outline Saturation Coefficient	0.5
Platform	None (Sprite Renderer)
Platform Saturation Coefficient	0.8
Extents Size To Outline Thickness Factor	0.3
Orient	✓

- **Hologram Size**: The size of the hologram displayed for the target.
- **Outline Saturation Coefficient**: The saturation level of the edge outline on the target hologram.
- **Platform**: The platform displayed under the hologram.
- **Platform Saturation Coefficient**: The saturation of the platform UI.
- **Extents Size To Outline Thickness Factor**: Control the relationship between the mesh size and the outline thickness.
- **Orient**: Whether to orient the target hologram relative to this transform (leave unchecked for no hologram rotation).