Developer
/ Documentation
/ Unreal Engine
/ Unreal Engine 5.4 Documentation
/ Making Interactive Experiences
/ Gameplay Targeting System
/ Gameplay Targeting System Reference

Gameplay Targeting System Reference

A Debugging and Troubleshooting guide of the Gameplay Targeting System plugin framework in Unreal Engine.

Gameplay Targeting System Reference

TargetingRequestHandle

A **handle** is created when a user begins an **Async Targeting Request**. The handle is used to interface with the targeting **Data stores**. Data stores are templated classes around generic data structs that the system and tasks use to accomplish a targeting request.

> ⓘ When a targeting handle is created it will not implicitly release the handle. It is up to the creator to either grab an Async Task Data or Immediate Task Data and set a flag indicating the system should release the handle for them after the callback fires, otherwise it is up to you (the user) to release the handle when you are finished with it.

TargetingSubsystem

The **Targeting Subsystem** is responsible for keeping track of Async Targeting requests as well as being the entry point for immediate targeting requests. The Targeting Subsystem contains code to help display debug information about active targeting requests.

TargetingDataStore

The **Targeting Data Store** is a Templated struct that defines targeting data stores. The goal is to provide a flexible way for targeting tasks to add, remove, and update generic sets of data they wish to work with.

This allows future growth in games to write tasks that are either new or extending off from the base framework archetypes to achieve the targeting goals they need.

> Creating your own custom data stores requires some boiler-plate code provided in `#define` libraries

The targeting system has 3 mandatory data stores, and 1 is required for async targeting requests. These data stores need to be set up before the system can properly run a targeting request.

TargetingTask

Targeting Tasks make up a Targeting Preset. Targeting Tasks are expected to be small reusable parts of a **Targeting Sequence**. Targeting Tasks have virtual `Init` and `Execute` functions to define behavior.

They contain a virtual function for drawing debug information to the screen. Targeting Tasks will generally fall into one of the following categories:

Category	Description
`Selection`	Selection tasks select Actors from the world and puts them in the Targets list. Simple examples provided by the plugin include selecting all Actors hit by a line trace, selecting all Actors inside an area of effect, and selecting the Source Actor.
`Sorting`	Sorting tasks reorder the list of Targets. A simple example provided by the plugin is sorting by distance from the source Actor.

Category	Description
`Filter`	Filter tasks reduce the target result data set to targets that match certain criteria. A simple example provided by the plugin is sorting by Actor Class. The plugin also provides a basic template for Filter tasks with a virtual ShouldFilterTarget function.

Targeting Tasks provided by the plugin include the following:

`TargetingFilterTask_ActorClass`	A simple filtering task that removes Actors that DON'T match classes in the `RequiredActorClassFilters` list or Actors that DO match classes in the `IgnoredActorClassFilters` list.
`TargetingFilterTask_BasicFilterTemplate`	A template for Filter tasks containing a virtual function `ShouldFilterTarget` which returns `true` if a target should be removed from the list.
`TargetingFilterTask_SortByDistance`	Sorts the target list by their distance from the source actor location. Stores the distance in the `TargetDefaultResultsData` as a `Score` variable.
`TargetingSelectionTask_AOE`	A selection task that grabs all Actors that collide with an Area centered at the source Actor location with an optional offset. Supports multiple shapes (Box, Cylinder, Sphere, Capsule) as well as using a component on the source Actor, specified by the component tag.
`TargetingSelectionTask_SourceActor`	A selection task that adds the source Actor to the target list.

`TargetingSelectionTask_Trace`	A selection task that adds all Actors hit by a line trace or sweep up to the first blocking hit (or its endpoint). The Trace starts at the source Actor location and traces in the direction of the source Actor's forward vector (or if it's a pawn, its control rotation) unless an explicit direction is provided.
`TargetingTaskSet`	A set of tasks used by targeting requests to find and process targets.
`TargetingDefaultResultData`	A simple struct containing an `FHitResult` and a `score` variable that represents a single target.
`TargetingDefaultResultsSet`	The base targeting result data used by the tasks implemented at the framework level. Provides an array of data (hit result / score) that tasks can add, remove, and sort to complete a targeting request.
`TargetingSourceContext`	A struct that holds context data for a Targeting Request including the Source Actor, Instigator, and Location.
`TargetingRequestData`	A struct that holds data and delegates related to a Targeting Request, including whether it is complete or not and both native and Blueprint callbacks for on completion.
`TargetingAsyncTaskData`	A struct that holds bookkeeping data for async targeting requests.
`TargetingImmediateTaskData`	A struct that holds bookkeeping data for immediate targeting requests.