Executable Objects can be thought of as designer-friendly tasks that can be played, stopped paused and reversed.

Executable Objects are owned and controlled by a **Managing Object**. Depending on the needs of your project, the managing object can also be a Gameplay Task, Ability Task, Actor, Component, or any other Object.

A **subsystem** will keep track of all executed objects, and make sure that no conflicting objects are active together.

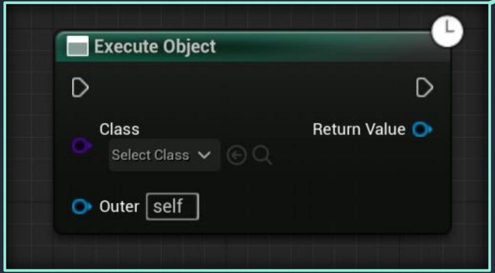
JON VAN DAM

Executable Objects

Overview

Outer Object

Creates, activates and controls an **Executable Object**



Can optionally implement *IManagingObjectInterface*

- ▶ *On Execution Started* (implementable)
- ■ *On Execution Ended* (implementable)
- ⚑ *On Execution Updated* (implementable)

Executable Object

Applies asynchronous effects to the game world.

UAsyncActionBase

Blueprintable

- ▶ *On Execution Start* (implementable)
- ⟳ *On Execution Tick* (implementable)
- ■ *On Execution End* (implementable)
- ⚑ *Process Execution Event* (callable)
- List of *Reference Objects*
- Fixed *Storage Slot*

Executable Object Subsystem

Keeps track of all active *Executable Objects*

UGameInstanceSubsystem (automatically spawned and destroyed with Game Instance)

Stored Executable Objects can be accessed based on:

- The class of the Executable Object
- Its Reference Objects
- Its Storage Slot

If a new Executable Object is started, the Subsystem will end all Executable Objects with the same Storage Slot and one or more matching Reference Objects.