Running Pre and Post Render Callback Scripts

Introduction to the new ways of adding Callbacks to the Movie Render Graph.

Introduction to Callbacks

Callbacks are a great way of adding pre/post logic to your render jobs. The Movie Render Graph has a callback system for handling scripts to run before and after jobs/shots are run which allows adding additional logic to renders.

With the new render graph comes a node called "Execute Script Node" which makes it easier and more visible to create callbacks, as previously callbacks were only supported as part of the executor in the project settings, which is now supported inside of graph configs.

Callbacks can be created by subclassing MovieGraphScriptBase and overriding the functions for the dedicated callbacks.

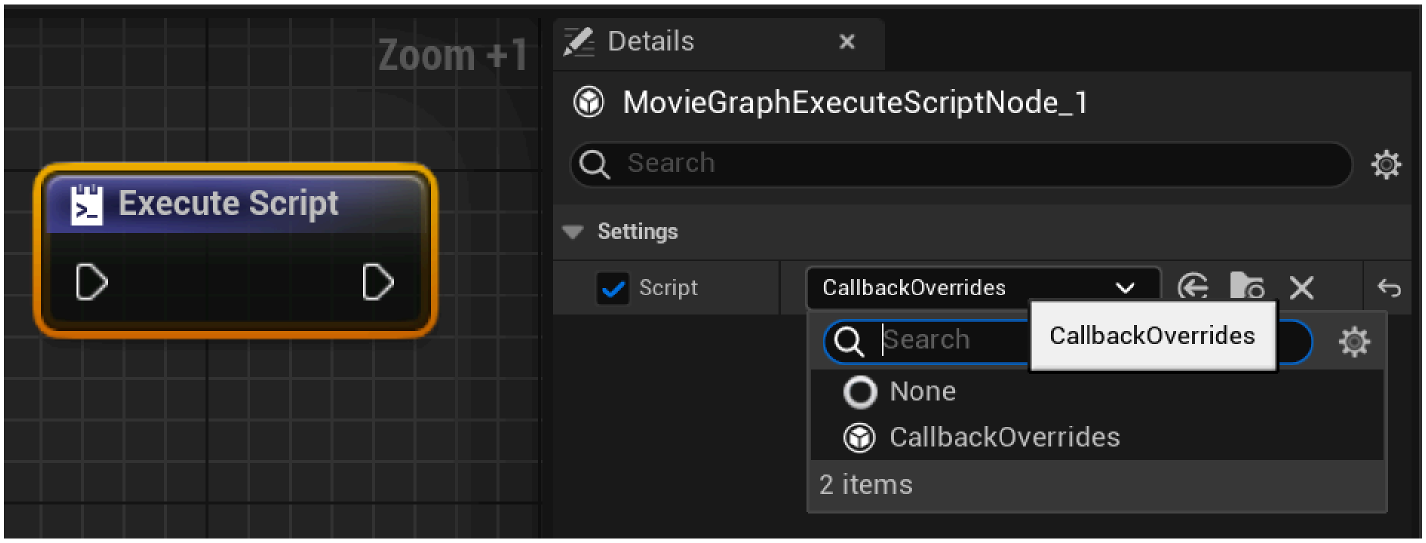
The following snippet is an example that shows how the unreal specific decorator `@unreal.uclass()` is used to tag a class for the UObject handling system, so it is aware and ready to be referenced from the Execute Script Node.

```
@unreal.uclass()
```

```
class CallbackOverrides(unreal.MovieGraphScriptBase):

def _post_init(self):

print("Callback Overrides Class")

```

Copy full snippet

Once the code snippet is run, the "CallbackOverrides" UClass will be selectable in the items menu of the Execute Script Node.



The individual functions are overridden by using the decorator `@unreal.ufunction(override=True)` which indicates that we are interested in overriding the underlying function.

On Job Start Callback:

This is called early in the process after an individual Job is started. It provides a reference to the job that is about to be run, and the Movie Graph configurations are available through the reference to the job. Both the job and the Movie Graph configurations provided in the callback are duplicates of the one originally specified in the Queue. This means that you do not have to worry about leaking modifications into the original assets (which may be shared by multiple jobs).

```
@unreal.uclass()
```

```

class CallbackOverrides(unreal.MovieGraphScriptBase):

@unreal.ufunction(override=True)

def on_job_start(self, in_job_copy):

super().on_job_start(in_job_copy)

print("This is run before the render starts")

```

Copy full snippet

On Job Finished Callback:

This is called at the very end of the Movie Graph Pipeline just before shutting down, by this point all image data should be written to disk. This function is called both on success and on cancellation. The second parameter has an attribute "success" that can be checked to see if the job actually succeeded, or if it was canceled either due to a configuration error or user input.

```
@unreal.ufunction(override=True)

def on_job_finished(self, in_job_copy, in_output_data):

super().on_job_start(in_job_copy, in_output_data)

print("This is run after the render jobs are all finished")




#Sample code that showcases how to access shot render layer data

for shot in in_output_data.graph_data:

for layerIdentifier in shot.render_layer_data:

unreal.log("render layer: " + layerIdentifier.layer_name)
```

```
for file in shot.render_layer_data[layerIdentifier].file_paths:

unreal.log("file: " + file)

```

Copy full snippet

On Shot Start Callback:

```
@unreal.ufunction(override=True)

def on_shot_start(self, in_job_copy, in_shot_copy):

super().on_shot_start(in_job_copy, in_shot_copy)

print("This is run before every shot rendered")

```

Copy full snippet

Similar to Job Start but called before a shot is set up and is useful if you need to interact at the beginning of individual shots.

On Shot Finished Callback:

```
@unreal.ufunction(override=True)

def on_shot_finished(self, in_job_copy, in_shot_copy, in_output_data):

super().on_shot_finished(in_job_copy, in_shot_copy, in_output_data)

print("This is called after every shot is finished rendering")

```

Copy full snippet

Similar to Job Finished but called after the end of every shot. The output data will only contain data from the shot that this callback is for.

Enable Per Shot Callbacks:

`on_shot_start` and `on_shot_finished` callbacks will only be called if we enable it explicitly as they may increase render times due to the fact that the render will stall at the end of every shot and wait until all files have been written to disk before invoking the callback.

There are two ways to enable per shot writes.

1. On the Global Output Settings Node set "Flush Disk Writes Per Shot" to checked.
2. While defining your callbacks, in the same manner you can override the function `is_per_shot_callback_needed` to enable it.

```
@unreal.ufunction(override=True)

def is_per_shot_callback_needed():

return True

```

Copy full snippet

Permanently defining the Python UClass:

Registering a callback class will only persist for the current session, to ensure your custom UClass is registered permanently, you need to add an import statement to one of the `/Content/Python/init_unreal.py` files. Importing your class inside of this python file will make sure that it is run on every startup and hence will be available permanently.

Callbacks on Executor:

Apart from the Script Node in the Graph which allows overwriting per job/shot pre/post callbacks, the executor itself also allows overriding callbacks. The most useful one is `on_executor_finished_delegate` which is the final step after all jobs are done and is useful to notify other applications that the jobs are finished. Even though there are other callbacks that work similar to the ones described in the Execute Script Node, we recommend keeping per/post job/shot callbacks as part of the Execute Script Node in the Movie Render Graph.

```
def on_queue_finished_callback(executor: unreal.MoviePipelineExecutorBase, suc

print("Run before a Render job starts: ")

subsystem_executor.on_individual_job_started_delegate.add_callable_unique(on_i

```

Copy full snippet

A full Example can be found in the `MovieGraphScriptNodeExample.py` file which will be available in `/Engine/Plugins/MovieScene/MovieRenderPipeline/ Content/Python/` and is attached for the time being to this document.

```
# Copyright Epic Games, Inc. All Rights Reserved.

import unreal

# This example showcases the creation of a UClass that overrides the
Pre/Post Shot

# Job Callbacks which is necessary for the Execute Script Node. Running

# register_callback_class() will display Object named "CallbackOverrides"
in the graph's

# Execute Script Node details. This object replaced the available available
Job/Shot

# Callbacks with basic statements

# USAGE:

```

```
#

# import MovieGraphScriptNodeExample

# MovieGraphScriptNodeExample.register_callback_class()

#

# After running this function, ou will be able to select this UObject

# within a Movie Graph Config's Execute Script Node.

#

# You can add the above snippet to any of the /Content/Python/__init__ py
file

# to make this UObject permanently available at engine startup.

def register_callback_class():

"""This helper function creates a sample UClass called CallbackOverrides

that overrides MovieGraphsScriptBase functions, demonstrating how
individual

callbacks run before/after jobs and shots within a Movie Graphs Execute

Script Node.

Call this function to register your CallbackOverrides class so it becomes

available for selection in the Execute Script Node.

"""

@unreal.uclass()

class CallbackOverrides(unreal.MovieGraphScriptBase):

@unreal.ufunction(override=True)

```

```
def on_job_start(self, in_job_copy):

super().on_job_start(in_job_copy)

unreal.log("This is run before the render starts")

@unreal.ufunction(override=True)

def on_job_finished(self, in_job_copy:unreal.MoviePipelineExecutorJob,

in_output_data:unreal.MoviePipelineOutputData):

super().on_job_finished(in_job_copy, in_output_data)

unreal.log("This is run after the render jobs are all finished")

for shot in in_output_data.graph_data:

for layerIdentifier in shot.render_layer_data:

unreal.log("render layer: " + layerIdentifier.layer_name)

for file in shot.render_layer_data[layerIdentifier].file_paths:

unreal.log("file: " + file)

@unreal.ufunction(override=True)

def on_shot_start(self, in_job_copy:unreal.MoviePipelineExecutorJob,

in_shot_copy: unreal.MoviePipelineExecutorShot):

super().on_shot_start(in_job_copy, in_shot_copy)

unreal.log(" This is run before every shot rendered")

@unreal.ufunction(override=True)

def on_shot_finished(self, in_job_copy:unreal.MoviePipelineExecutorJob,

in_shot_copy:unreal.MoviePipelineExecutorShot,

in_output_data:unreal.MoviePipelineOutputData):

super().on_shot_finished(in_job_copy, in_shot_copy, in_output_data)

unreal.log(" This is called after every shot is finished rendering")

@unreal.ufunction(override=True)

def is_per_shot_callback_needed(self):

"""

Overriding this function and returning true enables per-shot disk

flushes, which has the same affect as turning on Flush Disk Writes

Per Shot on the Global Output Settings Node.

"""

return True

```

Copy full snippet