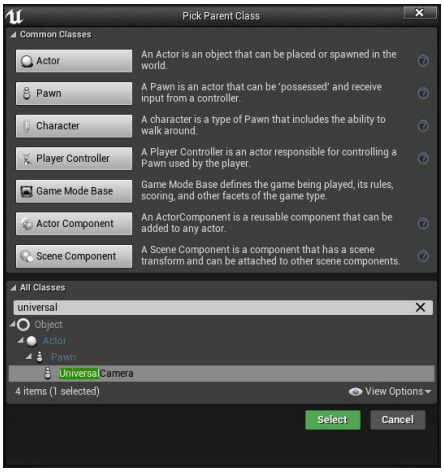
Universal Camera Plugin

Documentation

Table of Contents

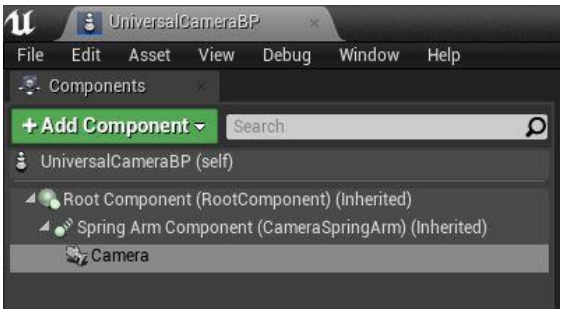
Setup

Create a new **Blueprint** from the **UniversalCamera** class and set it as your default pawn class.



Now Attach a Camera Component to your SpringArmComponent.

You can use the **CineCameraComponent** instead of the **CameraComponent** if you want to.



*The Camera has a RootComponent which I will be referring to as the **"Origin"** in the defaults.*

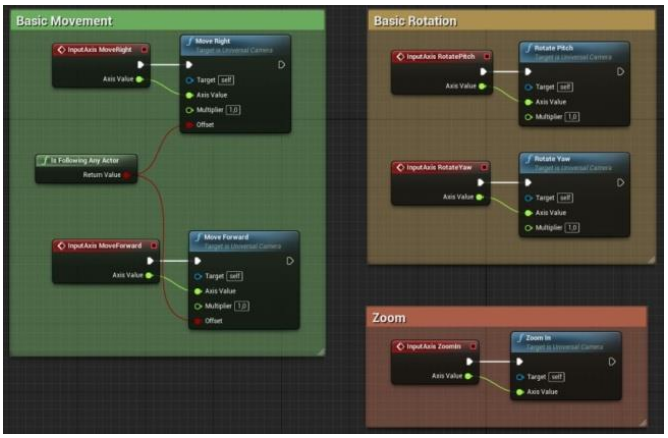
Now set it as your **default pawn class**, set up the **Inputs** and start moving around!

Also, don't forget to enable **ShowMouseCursor** in your **PlayerController** if you're using the mouse.

Inputs (Basic Setup)

Set up your inputs in Project Settings -> Input and use the functions provided in the “Basic Movement” category.

Here is an example of a basic setup that supports Keyboard/Mouse and Controller:



“IsFollowingAnyActor” on MoveRight() and MoveForward() allows you to move the Offset instead of the Origin if you’re following an Actor.

Camera Infos

DesiredLocation, **DesiredOffset**, **DesiredRotation** and **DesiredZoom** represent the desired position of your camera. Using the lag, the camera will try to catch up to these values.

Even with the lag disabled, these will always be updated and should be used instead of GetActorLocation(), GetActorRotation(), etc.

In order to **manually modify the camera position**, use **SetDesiredLocation**(), **SetDesiredOffset**(), **SetDesiredYaw**(), **SetDesiredPitch**(), **SetDesiredRoll**() and **SetDesiredZoom**(). You can also use **SetDesiredPosition**() to set them all in one function.

Offset

There are two types of Offset: Target Offset and Socket Offset.

- **Target Offset**: Offset at start of the Spring Arm Component.
- **Socket Offset** : Offset at end of the Spring Arm Component.

You can read more about how Spring Arm Components work here.

If you don't know how these work, I'd recommend trying both offsets one by one in your movement functions and play the level with "DrawDebugSpheres" enabled. Try to move around and rotate, and get a general feel of the two behaviors.

Camera Settings

You can define the camera behavior by tweaking its default values directly inside of the class defaults.

All of the information that you enter in the Defaults can be modified during runtime.

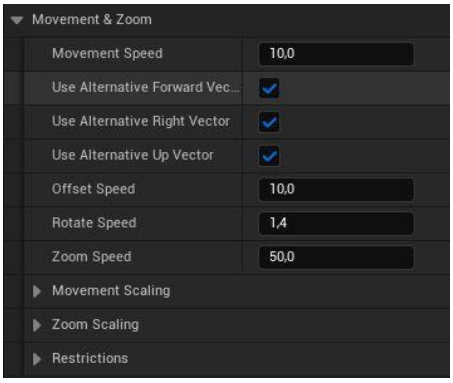
Starting Position

These information will set the starting position of your camera and will override the PlayerStart Location and Rotation.

Movement & Zoom

To use basic movement, you can use the following functions:

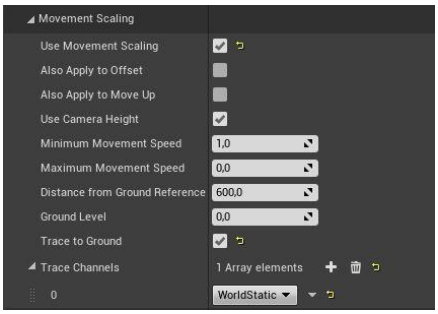
- MoveRight()
- MoveForward()
- MoveUp()
- RotateYaw()
- RotatePitch()
- ZoomIn()



- **Use Alternative Forward Vector** will use a forward vector independent of the **Pitch** for the **MoveForward**() function
- **Use Alternative Right Vector** will use a right vector independent of the **Roll** for the **MoveRight**() function
- **Use Alternative Up Vector** will make **MoveUp**() only move on the **Z axis**.

Movement Scaling

You can use Movement Scaling to adapt the Camera speed depending on its distance from the ground.



You can override “GetMovementScaling()” to implement your own Movement Scaling method if you prefer (you can still use these settings inside your own method).

Zoom Scaling

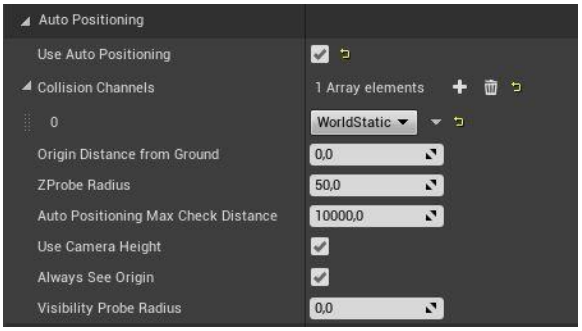
You can also use Zoom Scaling inside Basic Movement.

You can override “GetZoomInScaling()” to implement your own Movement Scaling method if you prefer (you can still use these settings inside your own method).

Auto Positioning

This is used in order to adapt the Z axis to terrain using the specified CollisionChannels.

If you don’t want to use the default Origin Positioning and want to implement your own, you can override the function “SetZDesiredLocation()”.



Edge Scrolling

Edge Scrolling allows you to move the camera whenever your cursor is near the edges of the screen.

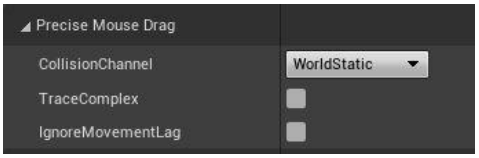
Use the Boolean IsEdgeScrolling to see if the player is currently Edge Scrolling.

You can react to Edge Scrolling (to manage the mouse widget for example) with the events OnBeginEdgeScrolling(), OnUpdateEdgeScrolling(), OnEndEdgeScrolling() which you can override in Blueprints.

Precise Mouse Drag

A built-in alternative to the setup shown on page 3.

Use TogglePreciseMouseDrag() to activate and deactivate it.



Screen Sliding

Screen sliding allows you to drag your mouse in order to move or rotate the camera. The SlidingOriginWidgetClass is the widget that will be displayed at the location of your mouse when you trigger Screen Sliding.

Use the function ToggleScreenSliding() to activate and deactivate Screen Sliding with your inputs.

Use the Boolean IsScreenSliding to see if the player is currently ScreenSliding.

You can react to Screen Sliding (to manage the mouse widget for example) with the events OnBeginScreenSliding(), OnUpdateScreenSliding(), OnEndScreenSliding() which you can override in Blueprints.

Lag

Lag allows you to have smooth movement for your camera.

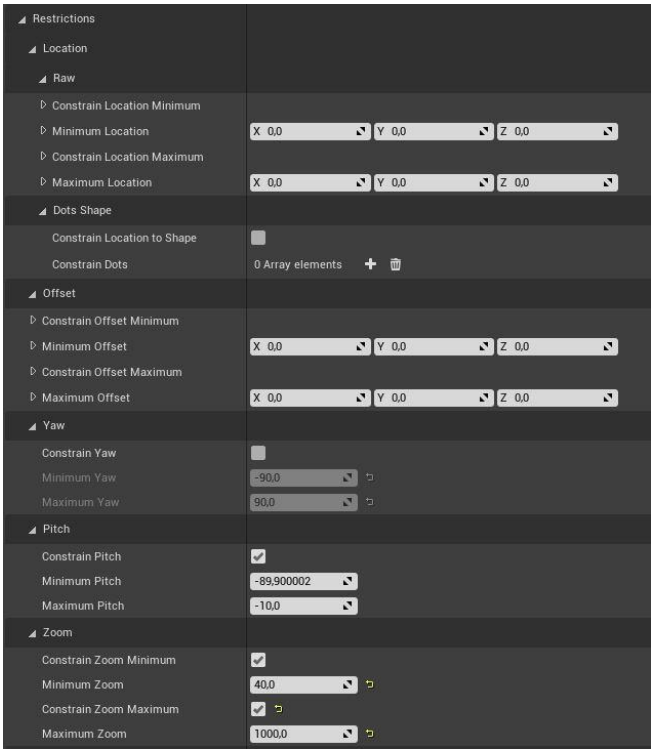
The camera will transition from its current position to the Desired one using Lag_GetTickDesiredLocation(), Lag_GetTickDesiredOffset(), Lag_GetTickDesiredHeight(), Lag_GetTickDesiredRotation(), Lag_GetTickDesiredZoom ().

All of these functions can be overridden in Blueprints if you want to implement your own Lag method.

Restrictions

Restrictions are used to constrain the camera position.

You can use it for Location, Offset, Yaw, Pitch and Zoom.

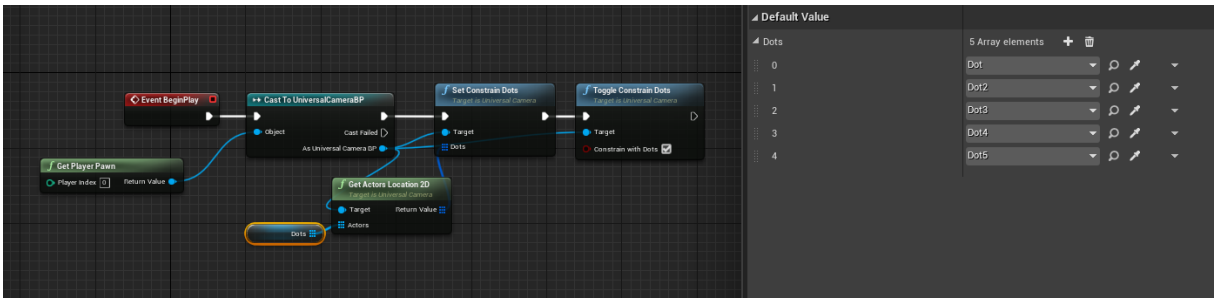


Location Restrictions

There's an additional Restriction method for the Location of the Camera: Dots Shape.

This allows you to draw a shape using an array of dots where the last element will connect to the first one (ConstrainDots). The camera won't be able to cross any segment of the shape but will smoothly slide along them.

This allows you to draw the shape of your level using these ConstrainDots. You can either set their location manually or place multiple actors into your level and set ConstrainDots from these actors using the function GetActorsLocation2D() in your Level Blueprints (see below).



Here I created a BP class "Dot" and placed some of them inside of the level to draw its shape.

I then add them inside an Array of Dots and feed it inside the ConstrainDots of my camera.

Debug

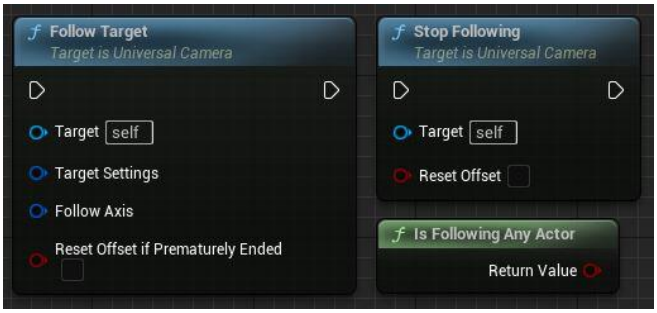
You can enable Draw Debug Spheres to draw spheres in different colors:

- Location (Green)
- Desired Location (Yellow)
- Desired Target Offset (Orange)
- Desired Socket Offset (Red)

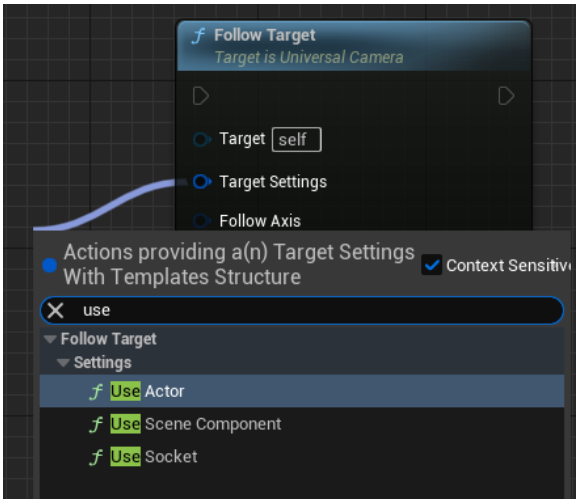
Print Desired Position will print the desired values on the top-left corner of the screen.

Follow Target

You can follow any Actor, Scene Component or Socket with the function **FollowTarget**(). You can stop following the actor with the function **StopFollowing**().

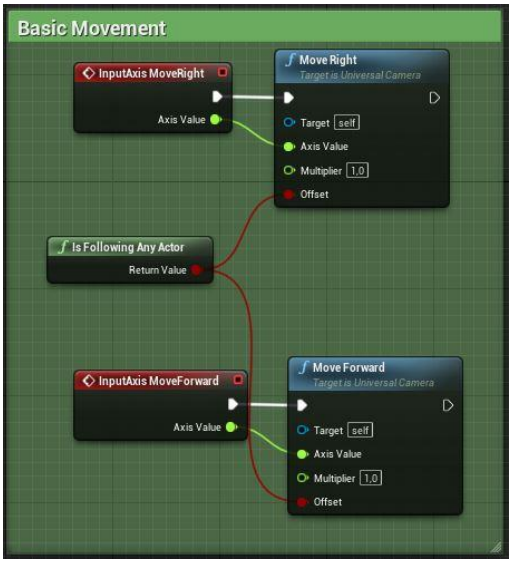


By dragging the Target Settings and using the keyword “Use” with Context Sensitive enabled, you can choose to follow an Actor, a Scene Component or a Socket.

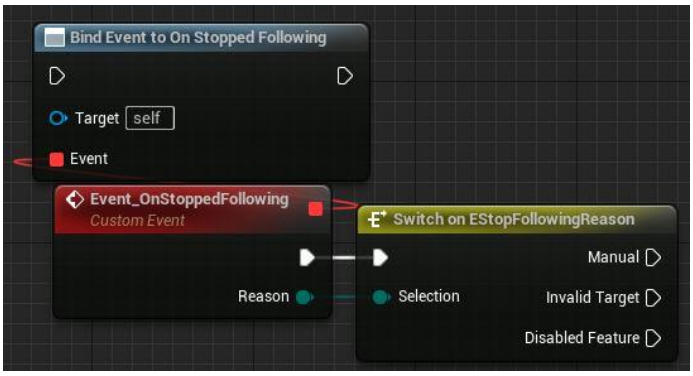


- “Follow Axis” allow you to track only specified axis.
- “Reset Offset If Prematurely Ended” allow you to reset the Target Offset and Socket Offset back to (0;0;0) if the task ended before calling StopFollowing() (e.g., target is no longer valid, FollowActor feature is disabled…)

You can feed the Boolean “IsFollowingAnyActor” into your basic movement “Offset” parameter to automatically switch to moving the Offset whenever you’re following an Actor.

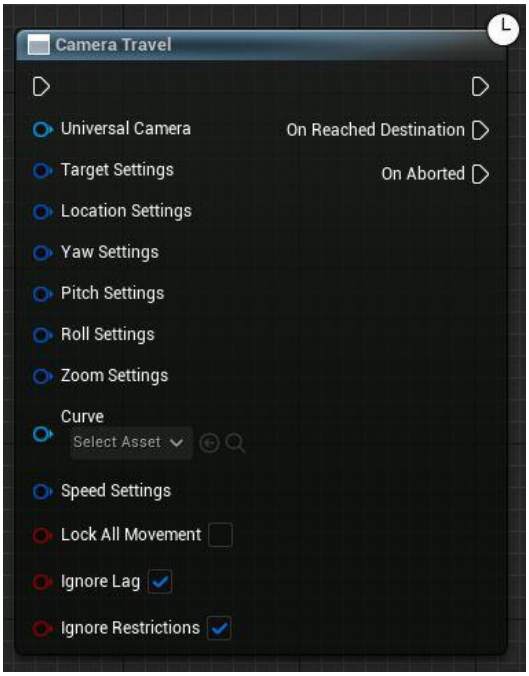


You can also use the dispatcher “OnStoppedFollowing”.



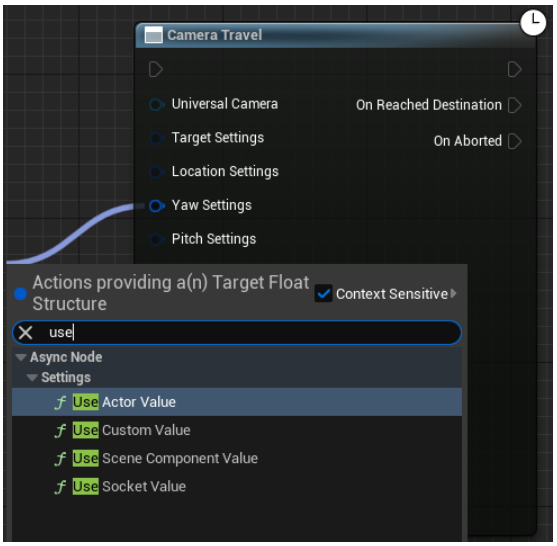
Camera Travel

Camera Travel allows you to smoothly change the camera position.



"Target Settings" allow you to store a reference to a Character, a Scene Component and a Socket. This will allow you to track their location while traveling.

For Location, Yaw, Pitch, Roll and Zoom settings, you have four options:



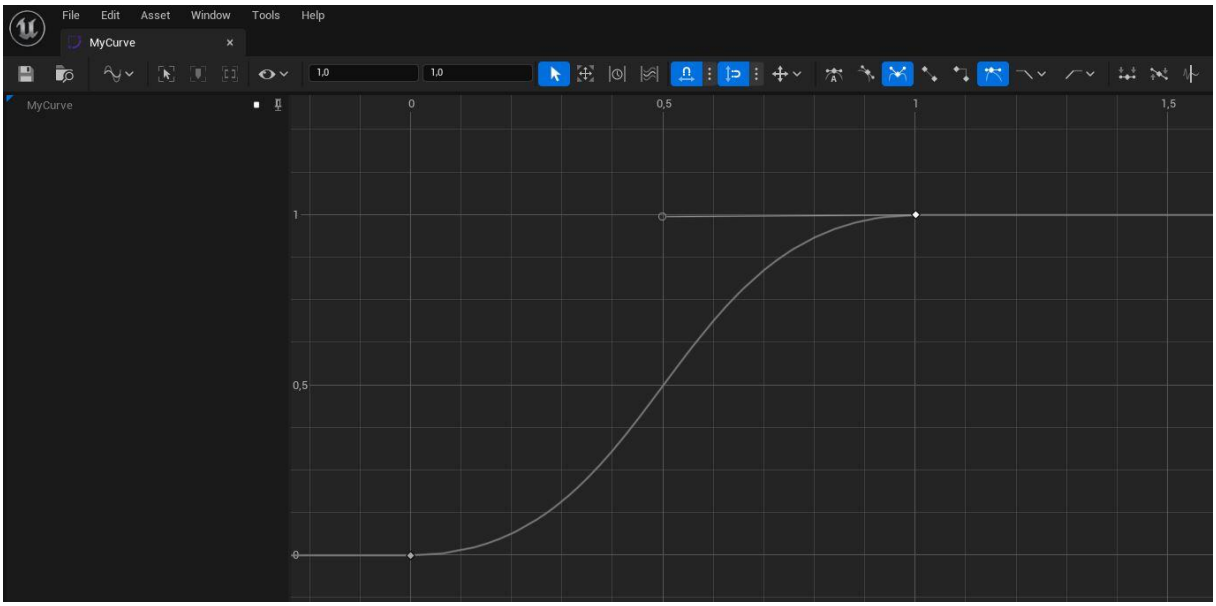
Those options will appear while dragging a setting and typing the keyword "use" with Context Sensitive enabled.

Keep in mind that all of these settings are optional. However, for the travel task to trigger, the function will need at least one valid setting.

Curve

You can also use a CurveFloat, which will determine the variations of speed during the traveling process.
To create a CurveFloat: right click in your content browser -> Miscellaneous -> Curve -> CurveFloat.

You will need two keys in for your curve: one at (0;0) and one at (1;1).



In this example, the camera will go slower at the beginning and the end of the travel task, and faster in the middle.

Speed Settings

You have two options for the speed settings:

- Use Duration
- Use Speed

While the duration is pretty straightforward, the speed tries to normalize the different settings. Since there are different types of values and all of them should end their traveling at the same time, the speed setting will try to pick the greatest distance between the initial value and the target value.

For example, if you feed a location close to the initial location, and feed a yaw value that will result in a 180° rotation, the speed will be used for the yaw instead of the location.

Lock All Movement

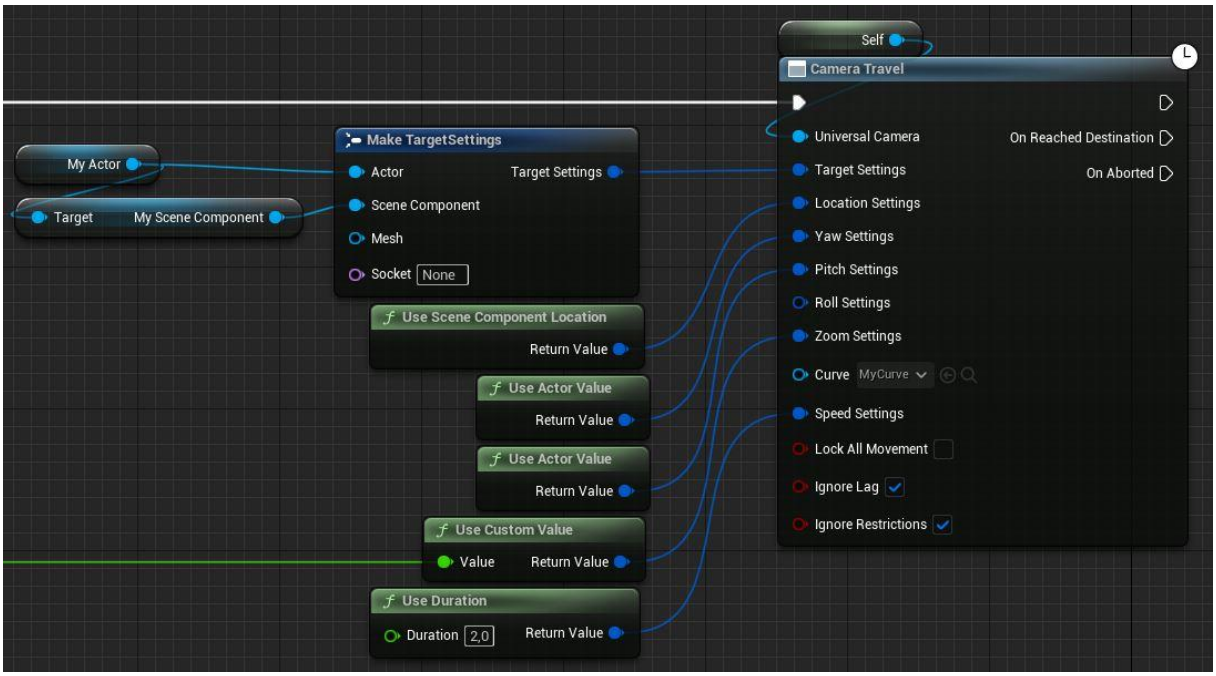
By default, only valid settings will lock their respective type during the traveling process (meaning that if you input a valid pitch, you won't be able to rotate the pitch during the traveling).

Setting "Lock All Movement" to true will prevent all movement/rotation/zoom during the traveling.

Abort Travel Task

Use the function “Abort Travel Task” to abort the current Travel Task. By doing so, the “On Abort” pin of the Camera Travel function will be triggered.

Example



In this example:

- My camera will track and travel to the **Location** of “**MySceneComponent**”.
- The **Yaw** and **Pitch** values will track and travel to the **Yaw** and **Pitch** values of “**MyActor**”.
- The **Roll** setting won’t move. Since it’s an invalid setting and “**LockAllMovement**” is disabled, I’ll be able to rotate the **Roll** during the traveling.
- The **Zoom** value will travel to my custom value.
- All of this will happen in 2 seconds, following the shape of “**MyCurve**”.

Focus Actor

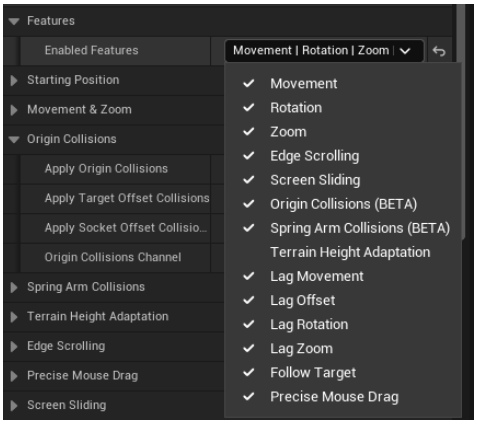
Considering your Rotation, you can find the ideal Location and Zoom to focus an Actor with the function **GetActorFocusLocation**().

Collisions

Origin Collisions

Enabling “**Origin Collisions**” in your **features** will apply collision between the Origin of the Universal Camera and the environment, using the channel in the “Origin Collisions” category.

Right now, it doesn’t work with moving objects.



Spring Arm Collisions

Enabling “Spring Arm Collisions” will control the Zoom so that your camera doesn’t end up in a wall and will assure you to always be able to see the Origin of the Universal Camera.

It uses the Collision Channel specified in the “Spring Arm Collisions” in the defaults.

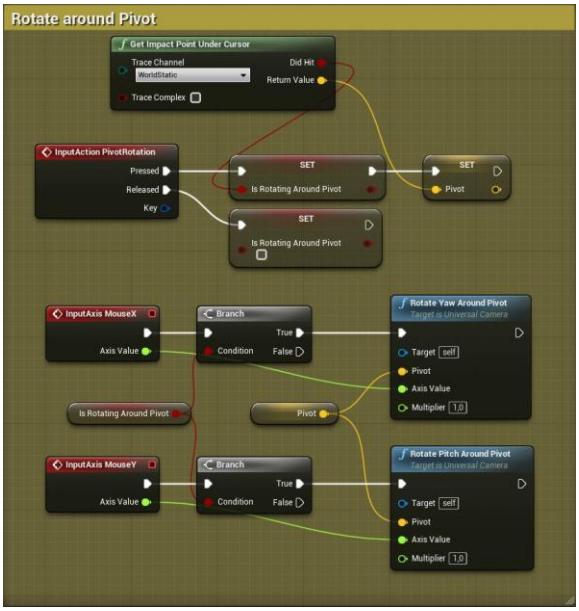
Zoom At Cursor

You can Zoom at the location under your cursor (tracing to the ground) using **ZoomAtCursor**().

You can also feed your own location using **ZoomAtLocation**().

Rotate around Pivot

You can rotate the Yaw and the Pitch around a pivot point using this kind of setup:

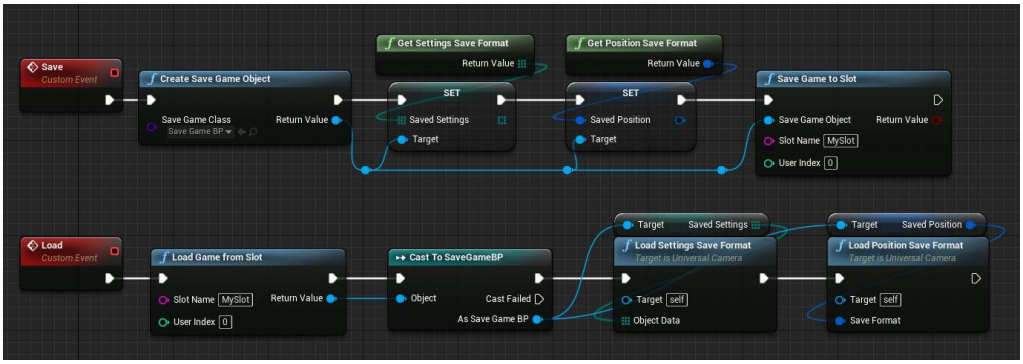


Save & Load

You can save the settings or the position of the camera at any moment.

The Settings of the camera are all of the data defining its behavior. You can save these settings inside of you SaveGame object with the function GetSettingsSaveFormat() and load them with the function LoadSettingsSaveFormat().

Likewise, you can save and load the Camera Position with the functions GetPositionSaveFormat() and LoadPositionSaveFormat().

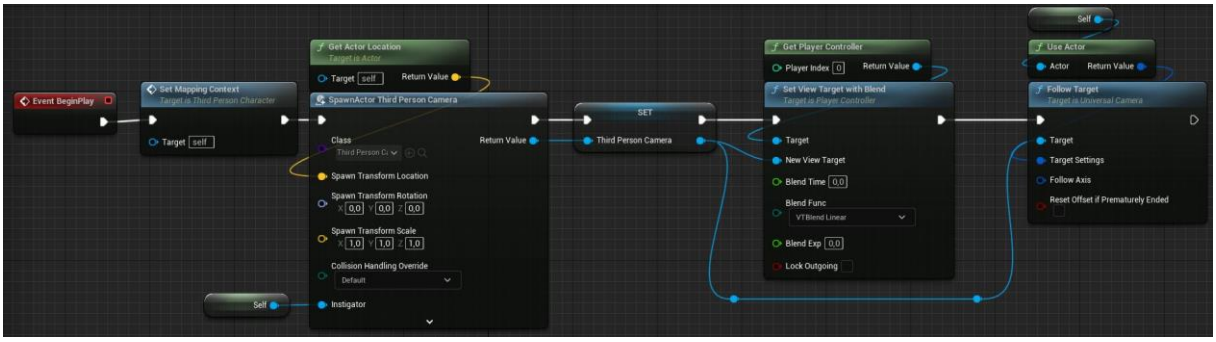


In your SaveGame class, use the “Universal Camera Position Save Format” structure for the Position, and an array of Bytes for the Settings.

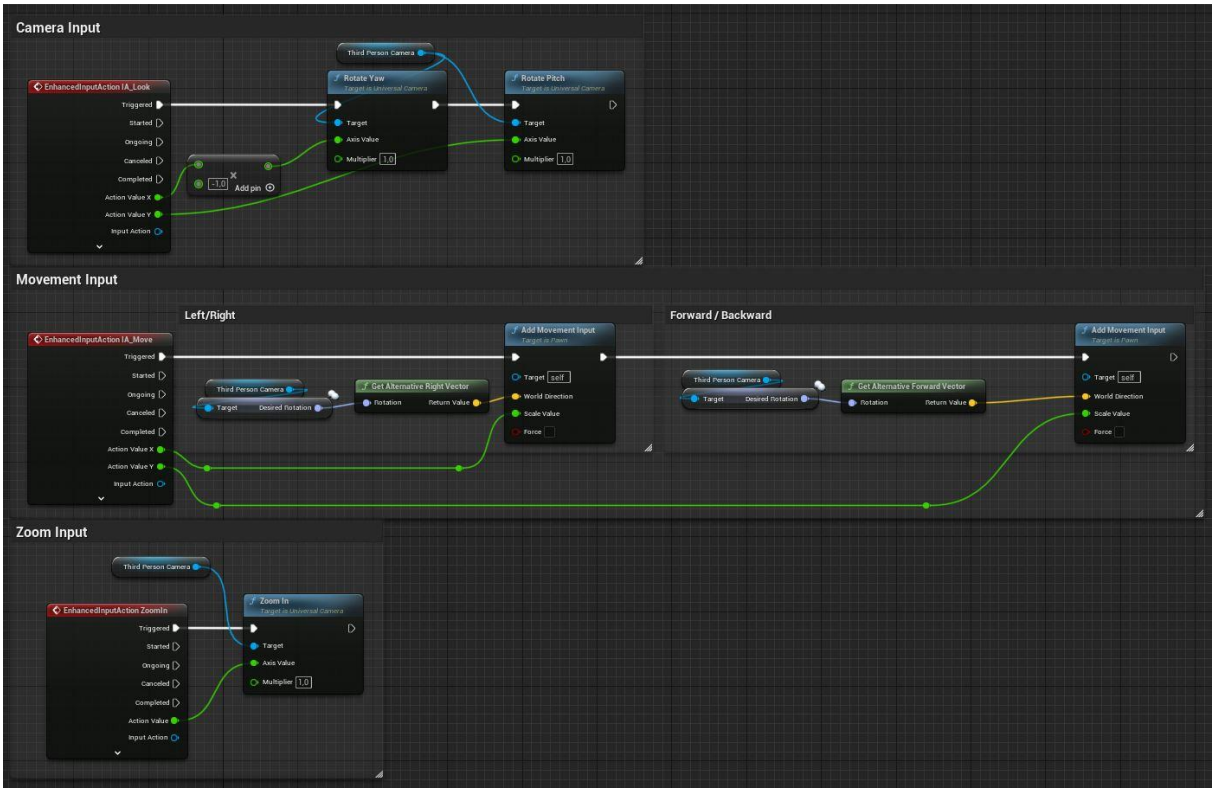
Control a Character

You can control a Character while using the Universal Camera by using “SetViewTargetWithBlend()”:

Third Person Example



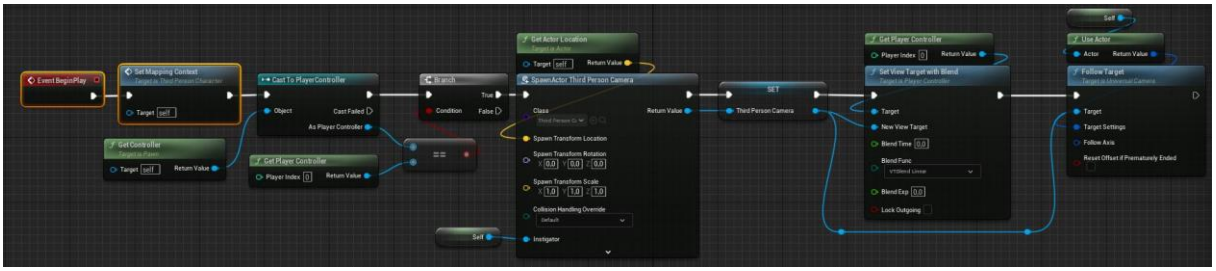
In this example from the Blueprints of my Character, I spawn the Universal Camera on BeginPlay and make it follow my Character and then set it as my view target.



The Character uses the Forward and Right alternative vectors of the Camera for its movement.

Multiplayer example with Characters

In Multiplayer, you could use this setup to only spawn the Universal Camera on controllers that own the pawn.



If you want to use RPCs, know that DesiredLocation, DesiredTargetOffset, DesiredSocketOffset, DesiredRotation and DesiredZoom are replicated.

More

I am willing to release multiple tools to help you work with the camera, so don't hesitate to send me feedback and suggestions on what features you want to see implemented! (Selection tools, traveling tools, etc.). You can email me at **heac.unreal@gmail.com**.

I really hope you enjoy this camera!