NEW DOCUMENTATION IS AVAIALBLE

HERE

Advanced Third Person Camera

ATPCCameraComponent

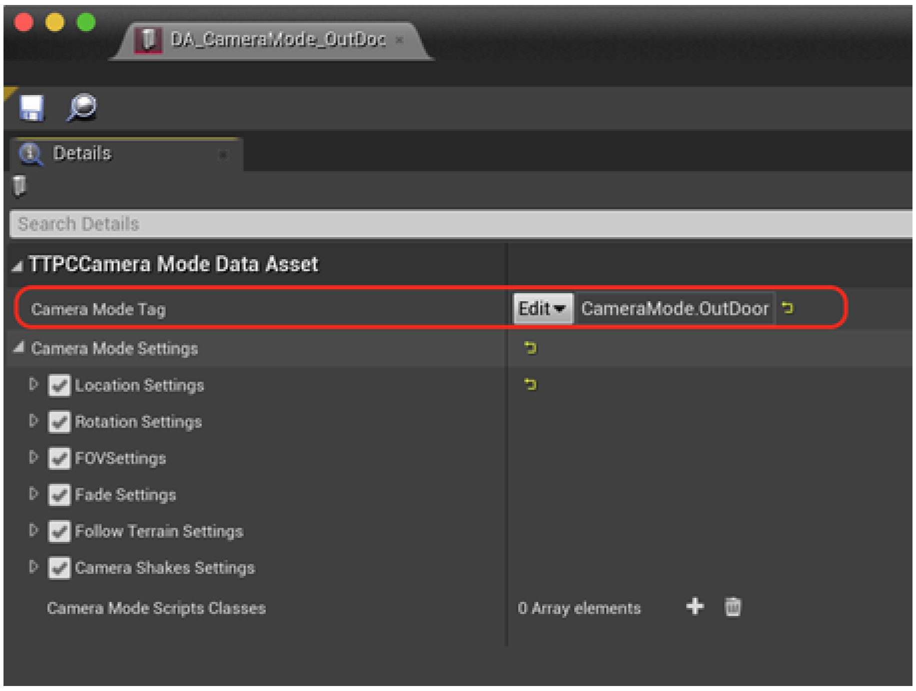
The main component of the plugin is **UATPCCameraComponent**. It controls the switching of the camera modes, camera objects, and scripts.

To add a camera to the character, add component (or replace the standard ACharacter) **UATPCCameraComponent.**

The main adjustment unit is Camera mode. It contains many parameters that fully describe the camera behavior.

UATPCCameraComponent contains a set of special data assets that store parameters of camera modes and are identified by the gameplay tags (FGameplayTag). You can read more about the data sets and adding them to the project in the documentation from Epic Games https://docs.unrealengine.com/en-US/Gameplay/Tags/index.html

To add a camera mode to a component, create a ATPCCameraModeDataAsset and specify the tag in it, which should correspond to this camera mode:



DA_CameraMode_OutDoo

Details

Search Details

TTPCCamera Mode Data Asset

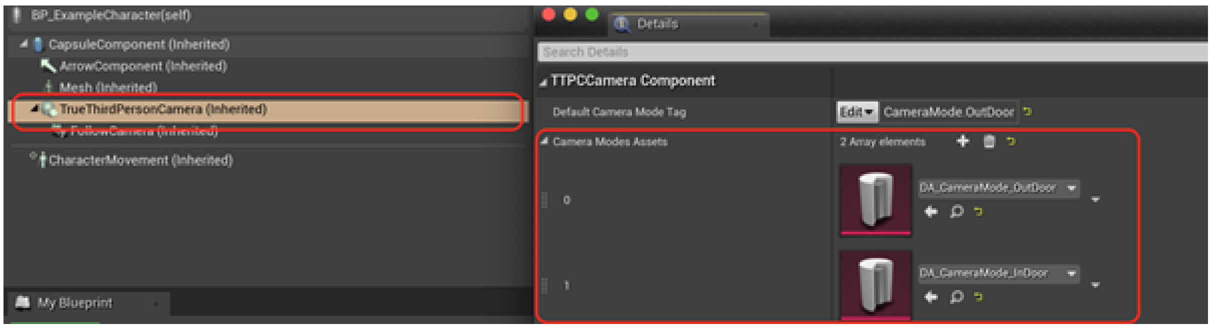
Camera Mode Tag — Edit — CameraMode.OutDoor

Camera Mode Settings

- ✓ Location Settings
- ✓ Rotation Settings
- ✓ FOVSettings
- ✓ Fade Settings
- ✓ Follow Terrain Settings
- ✓ Camera Shakes Settings

Camera Mode Scripts Classes — 0 Array elements

Next, add this data asset to ATPCCameraComponent in CameraModes:


BP_ExampleCharacter(self)
CapsuleComponent (Inherited)
ArrowComponent (Inherited)
Mesh (Inherited)
TrueThirdPersonCamera (Inherited)
FollowCamera (Inherited)
CharacterMovement (Inherited)
My Blueprint
Details
Search Details

TTPCCamera Component

Default Camera Mode Tag: Edit CameraMode.OutDoor

Camera Modes Assets: 2 Array elements

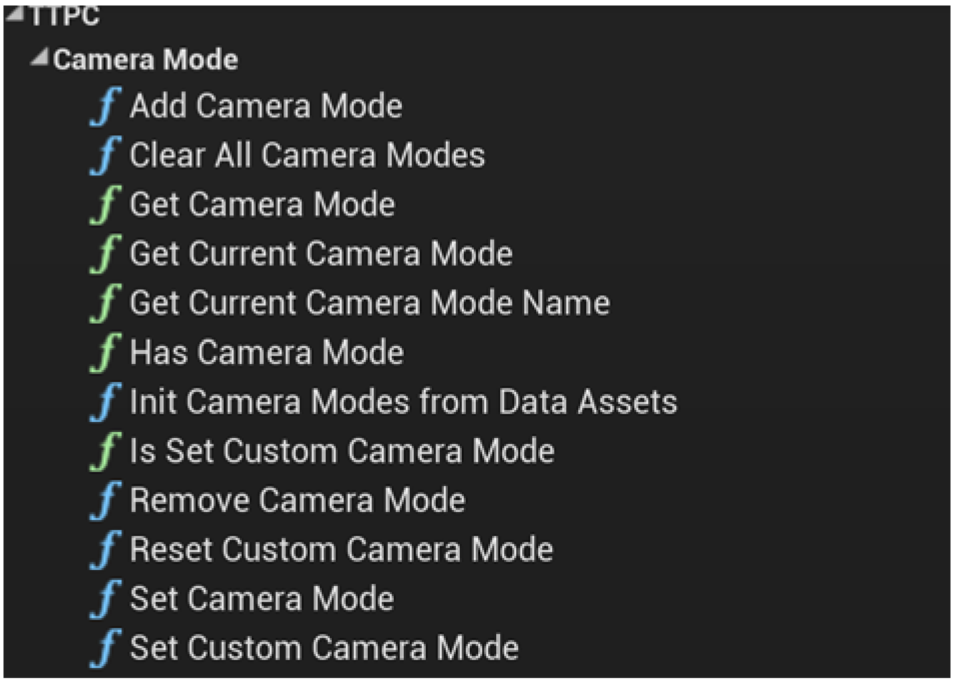
0: DA_CameraMode_OutDoor

1: DA_CameraMode_InDoor

The camera mode to be automatically activated when the game is started is specified with the *DefaultCameraModeName* parameter. With that, a mode with this name should be added to ATPCCameraModeComponent.

Cameras may also be added during the game. The current camera mode may be overridden without changing it using the *SetCustomCameraMode* function, and reset using the *ResetCustomCameraMode* function.

The complete list of functions for working with the camera modes:



TTPC

Camera Mode

- Add Camera Mode
- Clear All Camera Modes
- Get Camera Mode
- Get Current Camera Mode
- Get Current Camera Mode Name
- Has Camera Mode
- Init Camera Modes from Data Assets
- Is Set Custom Camera Mode
- Remove Camera Mode
- Reset Custom Camera Mode
- Set Camera Mode
- Set Custom Camera Mode

The following functions are also available for overriding in Blueprints:

1. *CanSetCameraMode* specifies whether the camera mode with the specified tag may be used (the camera mode is to be added to the ATPCCameraComponent list).
2. *GetDesiredCameraModeTag* returns the required camera mode (for example, checks whether the character exists in the special triggers that can modify the camera mode, or returns the current camera mode).
3. *UpdateCameraMode* invokes GetDesiredCameraModeTag and sets the received camera mode using SetCameraMode

All camera mode parameters are stored in the **FATPCCameraMode** structure.

All parameters are grouped by type. They are also represented by structures.

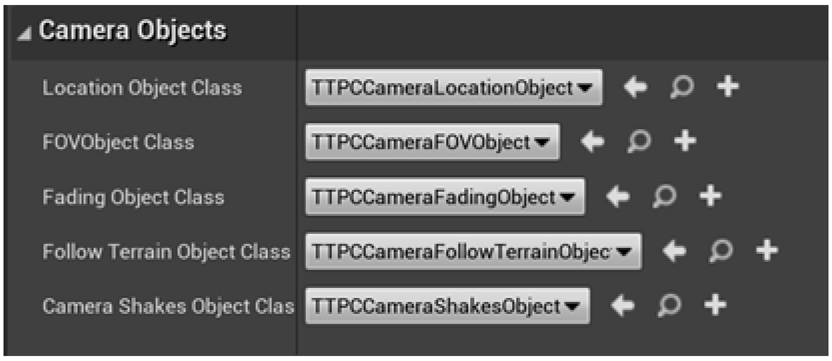
The main groups of parameters:

- Camera position control (**FATPCLocationSettings**)
- Camera rotation control (**FATPCRotationSettings**)
- Camera field of view control (**FATPCFOVSettings**)
- Hiding the objects between the camera and the character (**FATPCFadeSettings**)
- Changing the camera behavior depending on the terrain (**FATPCFollowTerrainSettings**)
- Shake control (**FATPCCameraShakesSettings**)

The main features of the camera

UATPCCameraComponent is divided into several objects, each working with one of the groups of the camera mode parameters and implementing a particular feature.

They can be overridden or supplemented both in C ++ and in Blueprint:



Camera Objects

Location Object Class	TTPCCameraLocationObject
FOVObject Class	TTPCCameraFOVObject
Fading Object Class	TTPCCameraFadingObject
Follow Terrain Object Class	TTPCCameraFollowTerrainObjec
Camera Shakes Object Clas	TTPCCameraShakesObject

Camera Base Object

The base class for all objects that are used in **UATPCCameraComponent** is **UATPCCameraBaseObject**. All the objects that are used in the component are inherited from it. It contains a set of basic functions that are overridden in the child objects.

Camera Location Object

It is represented as the UATPCCameraLocationObject class. This object monitors the camera position and rotation on the scene.

Its parameters are split into 2 structures: **FATPCLocationSettings** and **FATPCRotationSettings**.

Key features:

- Obtaining the camera location and rotation
  - GetCameraLocation
  - GetCameraRotation
- Setting the distance from the camera to the actor
  - The maximum and the minimum distance and the scroll speed are stored in the **FATPCLocationSettings** structure
  - SetCameraDistance — setting the distance from the camera to the player
  - GetCameraDistance — obtaining the current distance to the camera without considering SocketOffset and TargetOffset

  - GetRealCameraDistance — obtaining the current real distance to the camera
  - GetCameraTargetDistance — obtaining the distance the camera is trying to reach (can be used if the camera does not instantly change the distance but interpolates it)
- Setting SocketOffset and TargetOffset
  - The values are taken from the **FATPCLocationSettings structure**
  - Obtaining the current values
    - GetSocketOffset
    - GetTargetOffset
  - They may be modified for the current camera mode using special functions.
    - SetCustomSocketOffset, GetCustomSocketOffset, ResetCustomSocketOffset, HasCustomSocketOffset
    - SetCustomTargetOffset, GetCustomTargetOffset, ResetCustomTargetOffset, HasCustomTargetOffset
- Camera lag implementation for the x, y, and z axes, the parameters are stored in CameraLocationLagSettings
- Changing the camera snap point
  - AttachCamera To Component — set a new camera snap point
- Check collision between the camera and the character
  - bDoCollisionTest — enables and disables this option
  - ProbeSize — the size of the collision sphere to check
  - ProbeChannel — the collision channel for collision check
- Individual collision check for a particular object during the camera movement. It is used for objects, such as fences, trees, and poles. While the camera is moving, the collisions will not be considered for them; after the camera stops, the collisions will be processed in the standard way. The parameters for this option are stored in TODO
  - bSkipCollisionTestInMovement — enables or disables this option
  - SkipCollisionTestDuration — the period of ignoring the collisions after the camera stops
  - MinLocationDeltaForSkipCollisiontest — the minimum delta of changing the camera location for disabling collisions
  - ObjectCollisionChannelForSkipCollision — the collision channel used for determining the object for collision disabling. Collision Response is to be set to Overlap
- Setting the minimum and the maximum camera rotation angle around the Yaw and Pitch axes. Parameters ViewPitchMin, ViewPitchMax, ViewYawMin, ViewYawMax

- Changing the distance from the camera to the character depending on the camera rotation angle around the Pitch axis. The distance modifier is stored in the Pitch Distance Curve float curve, where value is the change of the distance relative to the current one, and time is the camera tilt angle around Pitch.
- The automatic change of the camera rotation angle around the Pitch axis in the absence of input. The parameters of this option are stored in RotationOffsetSettings (type FATPCRotationOffsetSettings)
- The automatic change of the camera rotation angle around the Pitch axis to avoid collision on top of the camera (useful for ceilings, doorways, etc.). The parameters of this option are stored in RoofCollisionCheckSettings (type FATPCRoofCollisionCheckSettings)
- The automatic camera rotation along the Yaw axis upon character rotation and in the absence of input for the camera. The parameters of this option are stored in ViewRotationToActorRotationSettings (type FATPCViewRotationToActorRotationSettings)

Camera FOV Object

It is represented as class UATPCCameraFOVObject. This object controls the FOV of the camera. Its parameters are stored in the **FATPCFOVSettings** structure

Key features:

- Setting and obtaining the current FOV of the camera.
    - If the FOV value is not explicitly overridden, it will be set to FATPCFOVSettings**::**CameraFOV from the current camera mode.
    - GetCurrentFOV returns the current FOV
    - Any change does not occur instantly; it occurs smoothly with the speed of interpolation FATPCFOVSettings::InterpolationSpeed
- FOV is overridden from the camera mode
    - SetOverrideFOV — set a new FOV
    - ResetOverrideFOV — reset the FOV overriding

Camera Fading Object

It is represented as class UATPCCameraFadingObject. This object controls the hiding of the objects between the game character and the camera and hiding of the game character if the camera collides with him. Its parameters are stored in the **FATPCFadeSettings** structure

Key features:

- Searching for objects between the character and the camera

    - The objects are searched for in the FindCollidedActors function, using boxTrace, the size of which is calculated based on the size of the collision of the character
    - If additional filtering of the found actors is required, a special FilterCollidedActors function exists, which is available for overriding in the child classes
- Hiding/showing objects. To change the object visibility, all static and skeletal meshes are searched for the actor, and the parameters for all the materials in the meshes found are changed.
    - MaterialFadeParamNames stores the names of the float parameters of the material to be changed.
    - MaterialFadeMinValue, MaterialFadeMaxValue are the minimum and the maximum values of the material parameters for completely hiding or showing the object, respectively.
    - FadeInTime, FadeOutTime are the durations of hiding and showing the object from 0 % to 100 %
    - FadeChannels the collision channel for searching for the objects. Collision response for this channel should be set with the objects to Overlap.
- Hiding the own character upon its collision with the camera
    - bFadeSelfIfCollision — enables and disables the option
    - bUseCustomFadeOutTimeForSelfFade and SelfFadeCustomFadeOutTime are responsible for overriding the character hiding duration
    - SelfFadeCheckRadius is the size of the collision in the camera for detecting a collision with the character
    - bSelfFadeAttachedActors - whether the attached actors are to be hidden when the character is hidden
- To add the actor to exclusions to avoid detection and hiding, use the following functions:
    - AddIgnoredActor
    - RemoveIgnoredActor
    - ClearIgnoredActors
- To force hide the actor, use the following functions:
    - AddManualFadeActor
    - RemoveManualFadeActor
    - ClearManualFadeActors

Camera Follow Terrain Object

It is represented as the UATPCCameraFollowTerrainObject class. This object calculates the position displacement and the camera rotation depending on the surface inclination angle under the character and transfers this data to UATPCCameraLocationObject (see SocketOffset and RotationOffsetSettings). Its parameters are stored in the **FATPCFollowTerrainSettings** structure

Key features:

- Changing the camera offset. The offset position is stored in the TerrainAngleSocketOffsetCurve vector curve, where x, y, and z are the offset, and time is the angle of surface inclination. The position change is interpolated at the speed of SocketOffsetInterpSpeed
- Changing the camera rotation angle. The offset angle is stored in the TerrainAngleRotationOffsetCurve float curve, where value is the additional camera rotation around the pitch, and time is the angle of the surface inclination. Changing the angle is interpolated with the speed of AngleRotationInterpSpeed
- AngleDelayChange is the time that must elapse from the beginning of changing the surface incarnation angle until the offset settings are applied.

Camera Shakes Object

It is represented as the UATPCCameraShakesObject class. This object controls the camera shake objects. Its parameters are stored in **FATPCCameraShakesSettings**

Key features:

- EnterToModeCameraShake is the camera shake that is activated upon invoking a new camera mode.
- RegularCameraShake - camera shake, which is activated after the end of ModeInCameraShake, or immediately after activating a new camera mode, if ModeInCameraShake is not set. IMPORTANT: This shake is looping.

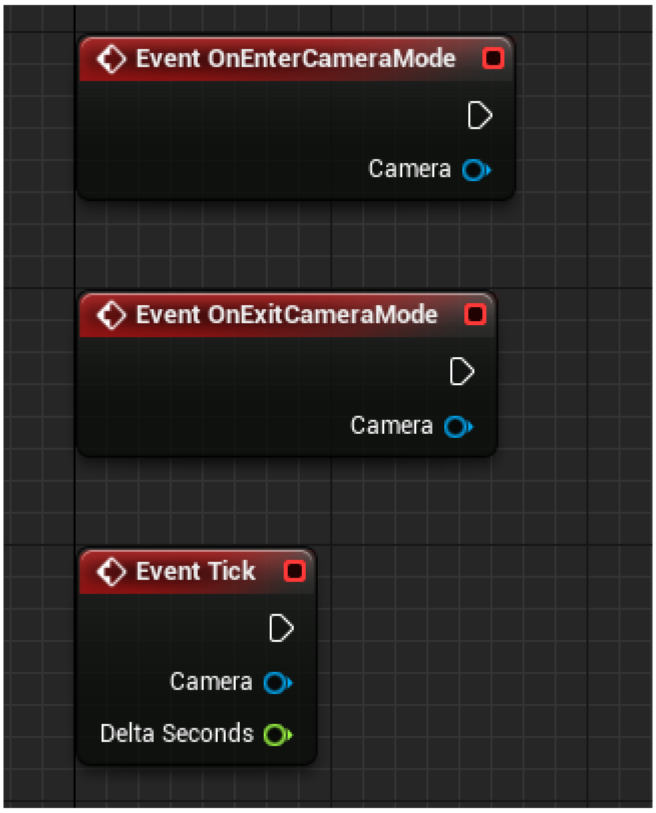
Scripts for the camera modes

Camera mode script is a special object that is created upon activation of the camera mode. It is convenient for writing user logic, which should be linked to the camera mode.

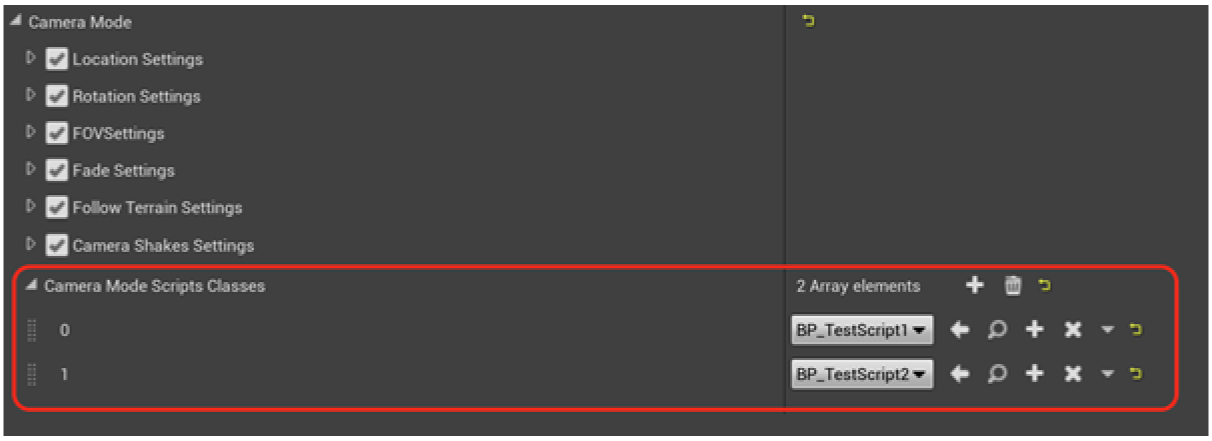
To create this object, to create a blueprint inherited from the ATPCCameraModeScript class.

It has three functions available for overriding:

- OnEnterCameraMode
- OnExitCameraMode
- EventTick



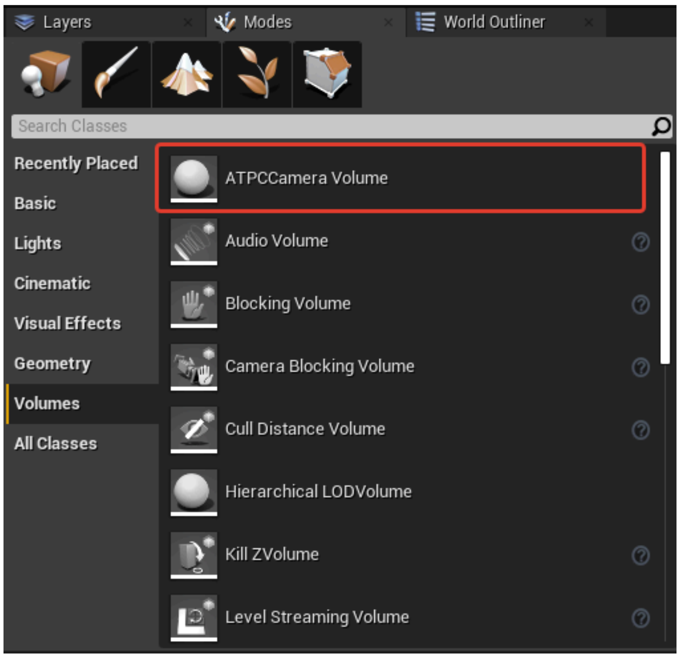
For the same camera mode, it supports an unlimited number of scripts; they will be activated in the order in which they had been added.


Camera Mode
Location Settings
Rotation Settings
FOVSettings
Fade Settings
Follow Terrain Settings
Camera Shakes Settings
Camera Mode Scripts Classes — 2 Array elements
0 — BP_TestScript1
1 — BP_TestScript2

Triggers for automatic switching camera modes

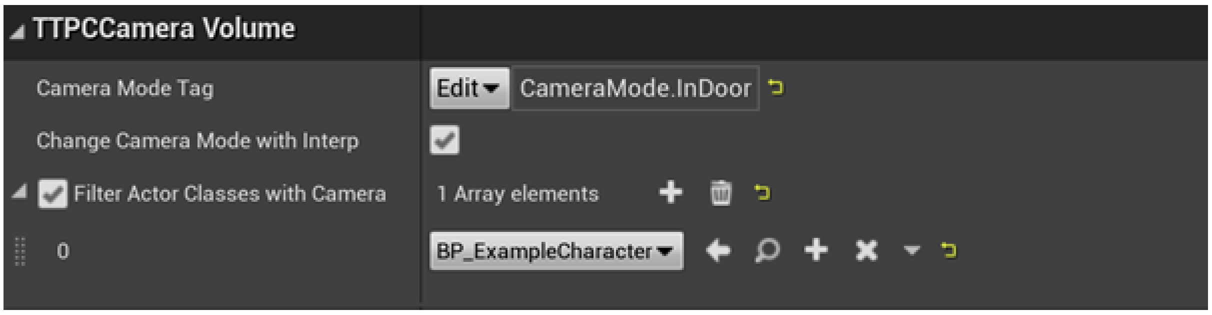
To automatically switch the camera modes in certain zones (for example, rooms or corridors). There are special triggers, which, when entered into, will activate the specified camera mode.

To add a trigger to the scene, open the Modes window in the editor, select the Volumes tab, find ATPCCameraVolume, and drag it to the location.



ATPCCameraVolume is a class derived from ABrush; it means what its shape and size may be edited as the standard Brush Volume.

Trigger parameters:



TTPCCamera Volume

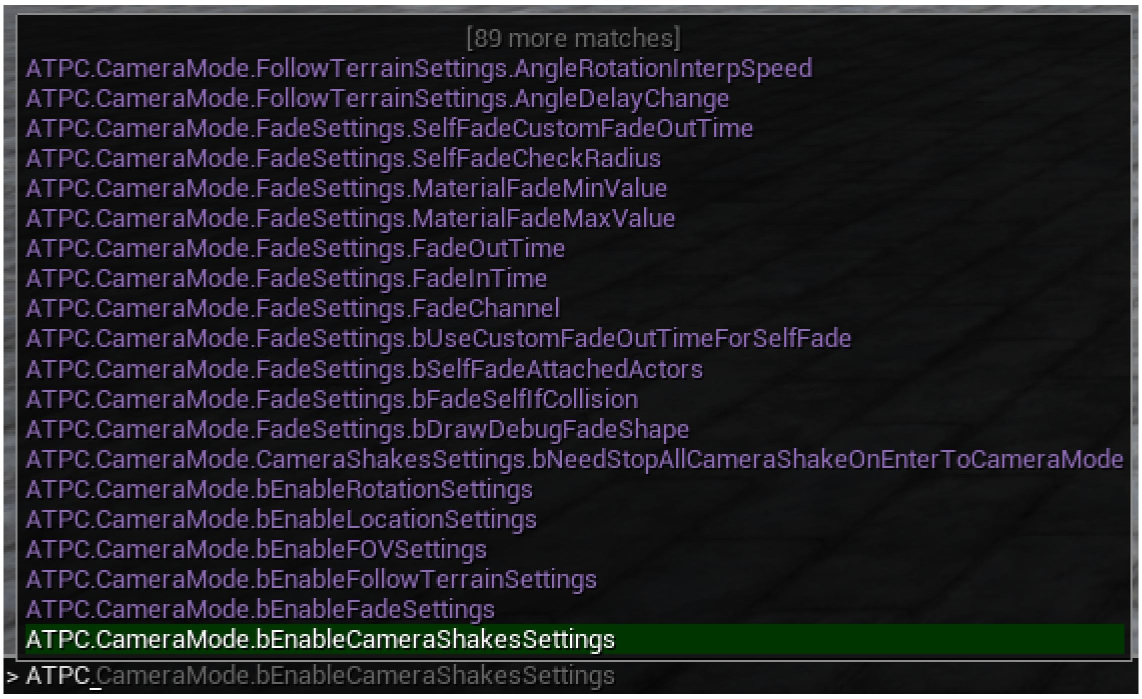
Camera Mode Tag	Edit CameraMode.InDoor
Change Camera Mode with Interp	✓
✓ Filter Actor Classes with Camera	1 Array elements
0	BP_ExampleCharacter

- CameraModeTag is the camera mode to be activated when the character crossed the trigger. The specific camera mode is to be added to ATPCCameraComponent.
- ChangeCameraModeWithInterp activates a smooth transition from the current camera mode to the one specified in the trigger.
- FilterActorClassesWithCamera is an optional parameter that allows enabling and disabling triggers for certain classes. If the parameter is disabled, the trigger will be enabled for all characters.

Console commands

For editing and debugging the camera modes, there is a set of special console commands that start with the ATPC prefix.

- ATPC.SetCameraMode [CameraModeName] [bChangeWithInterp] activates the specified camera mode via the CameraModeName argument
- ATPC.PrintCameraMode returns the current camera mode tag and all parameters to the prompt
- Here are the commands for editing all the parameters of the current camera mode:



```
[89 more matches]
ATPC.CameraMode.FollowTerrainSettings.AngleRotationInterpSpeed
ATPC.CameraMode.FollowTerrainSettings.AngleDelayChange
ATPC.CameraMode.FadeSettings.SelfFadeCustomFadeOutTime
ATPC.CameraMode.FadeSettings.SelfFadeCheckRadius
ATPC.CameraMode.FadeSettings.MaterialFadeMinValue
ATPC.CameraMode.FadeSettings.MaterialFadeMaxValue
ATPC.CameraMode.FadeSettings.FadeOutTime
ATPC.CameraMode.FadeSettings.FadeInTime
ATPC.CameraMode.FadeSettings.FadeChannel
ATPC.CameraMode.FadeSettings.bUseCustomFadeOutTimeForSelfFade
ATPC.CameraMode.FadeSettings.bSelfFadeAttachedActors
ATPC.CameraMode.FadeSettings.bFadeSelfIfCollision
ATPC.CameraMode.FadeSettings.bDrawDebugFadeShape
ATPC.CameraMode.CameraShakesSettings.bNeedStopAllCameraShakeOnEnterToCameraMode
ATPC.CameraMode.bEnableRotationSettings
ATPC.CameraMode.bEnableLocationSettings
ATPC.CameraMode.bEnableFOVSettings
ATPC.CameraMode.bEnableFollowTerrainSettings
ATPC.CameraMode.bEnableFadeSettings
ATPC.CameraMode.bEnableCameraShakesSettings
> ATPC_CameraMode.bEnableCameraShakesSettings
```