Dynamic SideScroller Camera Plugin

Contents

(Note: there is also a video version of this tutorial available on YouTube.)

NOTE: If migrating a UE4 project to UE5, reset the child actor by clearing the child actor class, and setting it back to SideScrollerCameraActor. There is a bug when migrating that will offset the camera from the character.

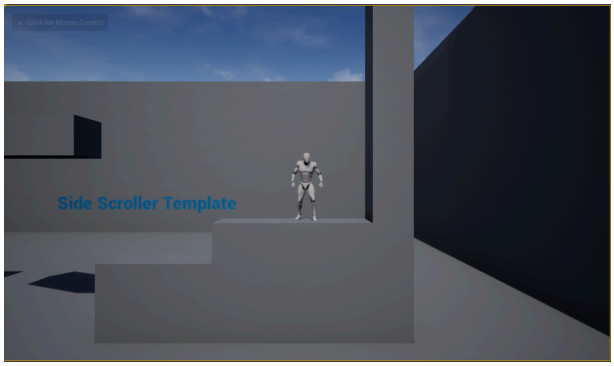
Introduction

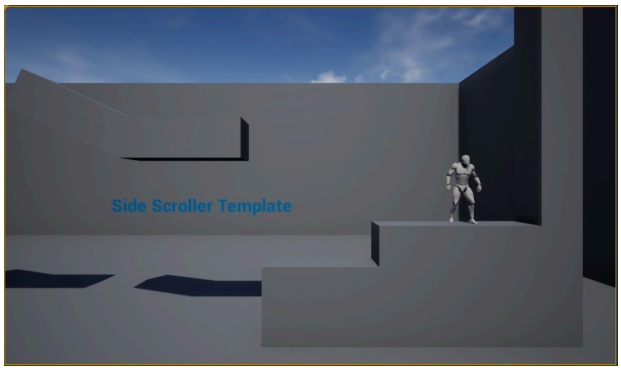
A common solution for camera handling in 2.5D games is to have camera volumes where the player swaps between volumes as they move through the level. However this is limited in terms of both flexibility (cannot use complex room shapes) and requires a lot of work to get the correct bounds and position.

So, we decided to create one based on collisions. This camera system works by probing right, left, up and down and moving the camera appropriately. This means that if a camera blocking volume exists on the right, it will hit it and move the camera to the left in response. It can work with multiple blocking volumes at once, so any polygonal room is feasible.

This allows to hide areas offscreen such as dirt or walls or unwanted hidden areas.

Here is a screen shot of the default setup camera with a camera blocking volume on the right without the plugin (left picture) and with the plugin (right picture).



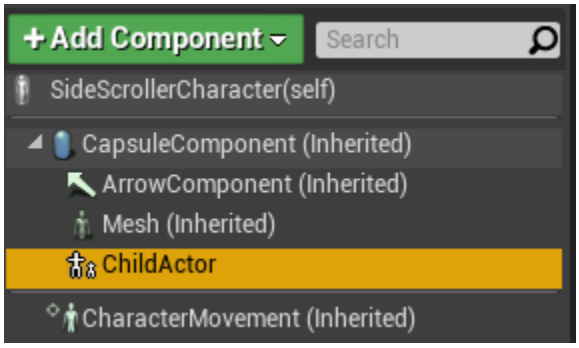
>


How to set up

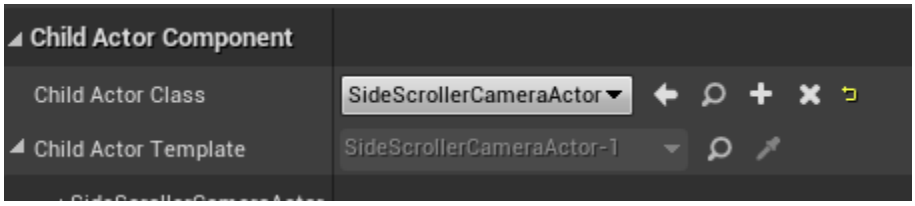
The following steps assumes you create a blank SideScroller project and you use the Y and Z axes for movement.

If you are using a character class for your main character, you simply need to do the following:

1. Open your character's blueprint
2. Remove the side arm and camera component
3. Add a child actor component



4. Define the child as "SideScrollerCameraActor"



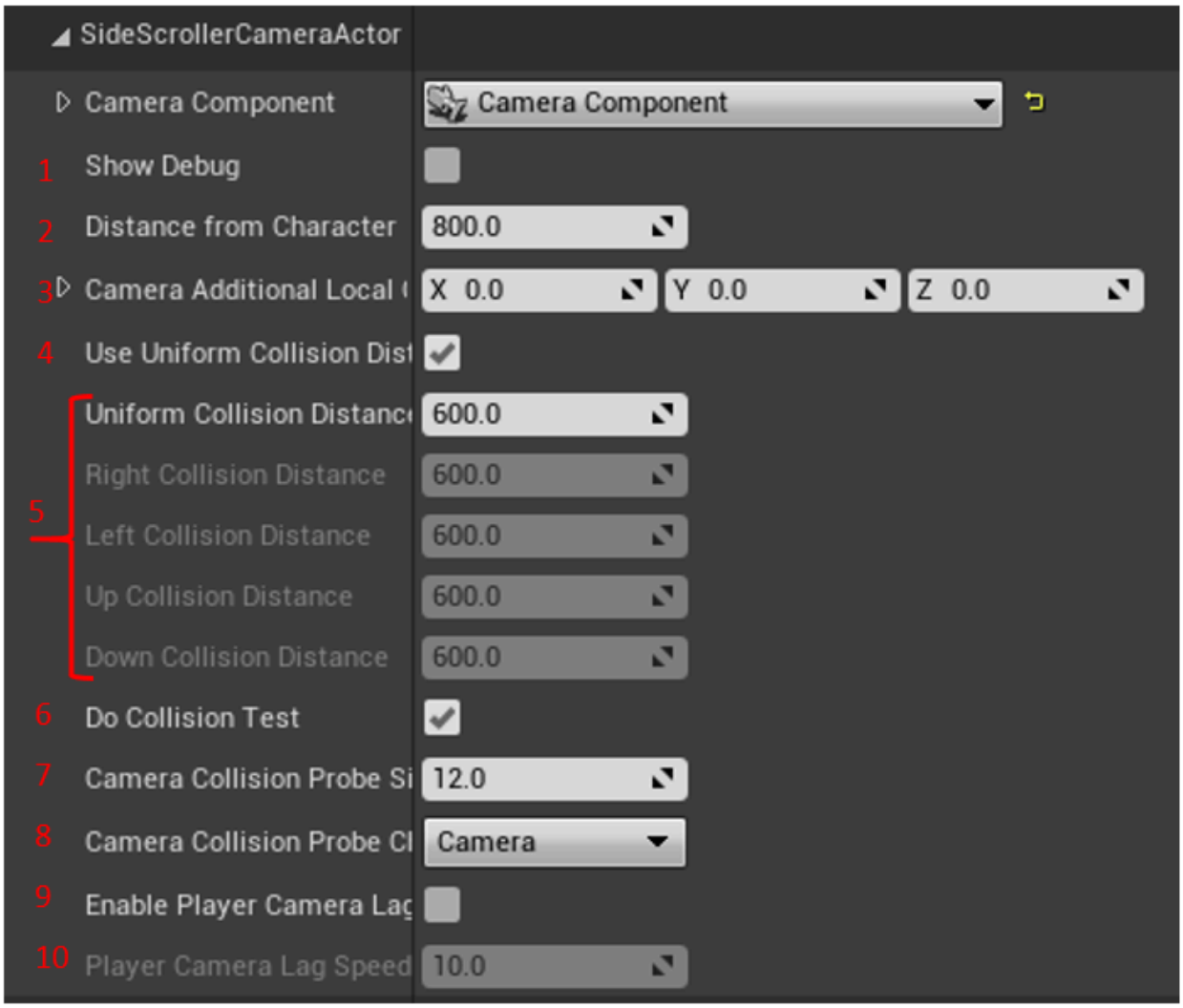
5. Define the properties of your Camera as required. (See "Modifying it to fit your ends")

6. Place a CameraBlockingVolumes in your level and test against it.

If you use the default template, you will see it spring upwards. That is because the ground is colliding with it. You can either change the camera's collision channel or make it so that the ground overlaps it instead.

Modifying it to fit your ends

If you click on your child actor in the player character, you will see the following default properties. See below on what they do.



1. Show Debug: If true, will show camera collision probes with arrows indicating where the bound is headed.
2. Distance From Character: smaller value means the camera is closer to the character.

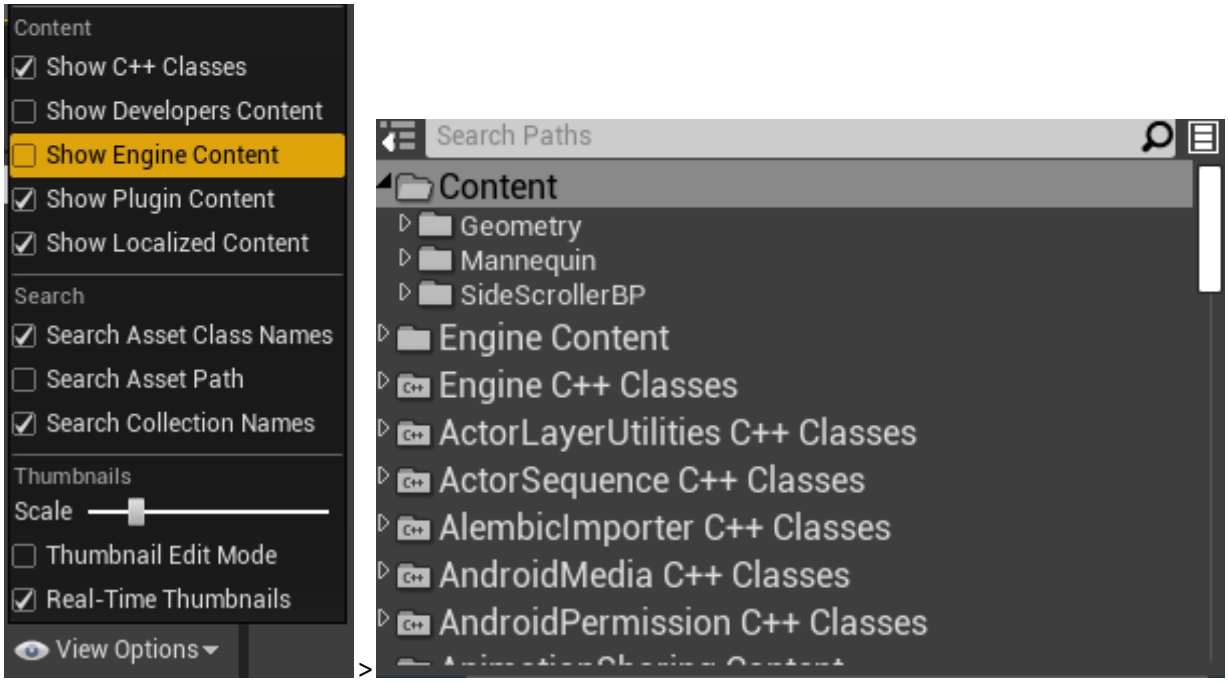
3. Camera Additional Local Offset: Vector. If you wish to give your character an offset when moving (e.g. camera trailing behind the character), adjust here.
4. Use Uniform Collision Distance:Boolean. False: You set each probes' distances individually. True: set them all equally in one go.
5. Probe distances in each direction.
6. Do Collision Test: Boolean to use this camera system. Turn to false if you don't want to use it anymore.
7. Camera Collision Probe size: the radius of the collision probe at the end of an arrow. The bigger it is, the farther it probes. Set between 5 to 12 for optimal results.
8. Camera Collision Probe channel: it can only do one so choose carefully. This is the channel it will probe into, so make sure your camera blocking volumes are set to it. Whatever you don't want to collide with should be set to overlap or ignore the camera channel.
9. Enable Player Camera Lag: Boolean if you want to give the camera lag or "inertia" when moving. Settings 9) and 10) are independent of the camera lag settings in AC_TransitionLogic.
10. Player Camera Lag Speed: determines how quickly the camera will follow the player when moving around. Higher value -> "snappier" follow.

Additional Features

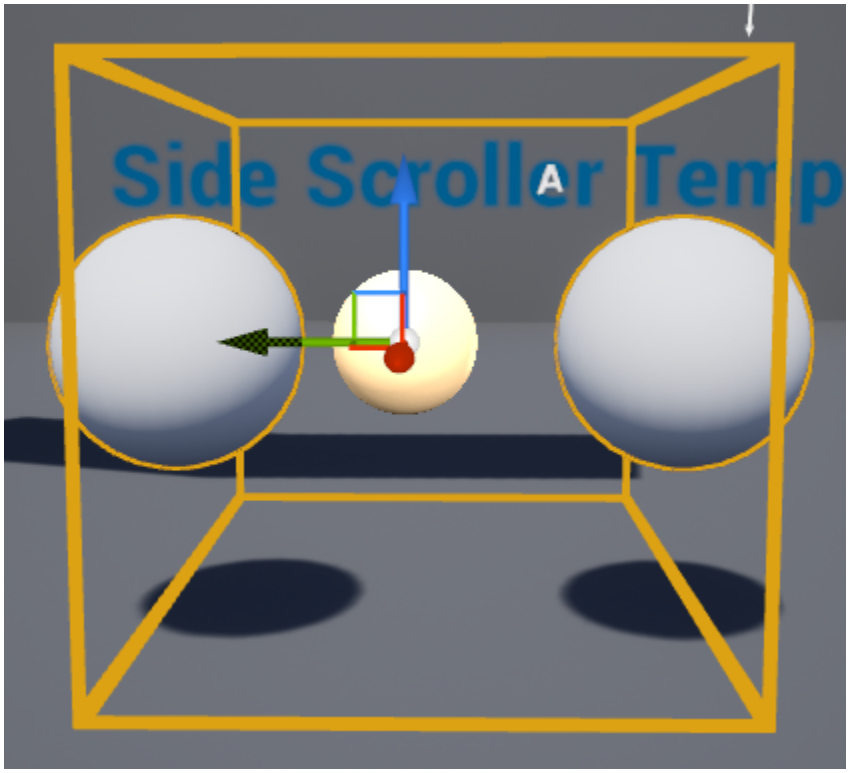
In the spirit of sidescrollers, we implemented room transitions using overlap volumes. There are two variants. The first one is a panning motion that may be useful for hidden areas or if you need a transition between streamed areas.

The second one mimics a typical fade transition as seen in games like Kirby or Hollow Knight when unloading and loading between rooms.

To use it, in the lower right handside of the contents folder, there is a view option. Click on it and check "Show Engine Content". This will expand the folder menu on the left hand side.

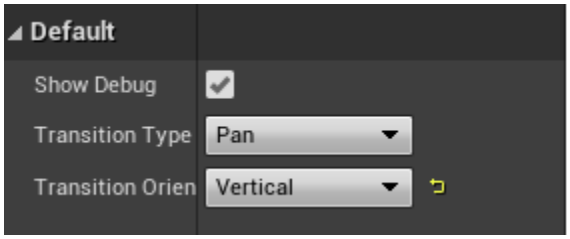


Search for the folder “SideScrollerDynamicCamera Content”.
In it you will find the BP_TransitionVolume. Simply drag and drop it into the level.
It will contain a cube with 2 spheres:



The spheres are the entry and exit points. Rotate as needed. You can have left<->right transitions, up<-> down ones, even diagonal. Note that when editing the per-instance position of the entry and exit points,

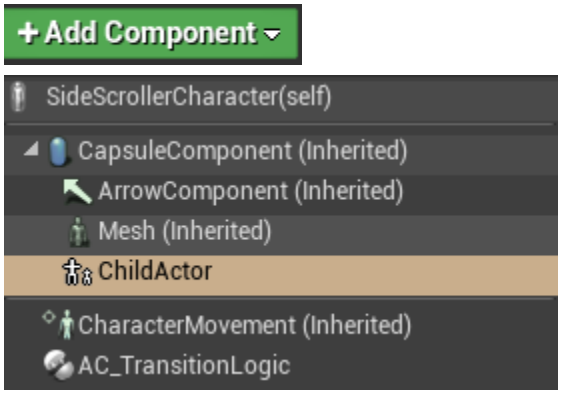
the collision box will not immediately update its bounds due to a UE bug. However, when in-game, the collision box will be updated.
The actor will have the following settings:



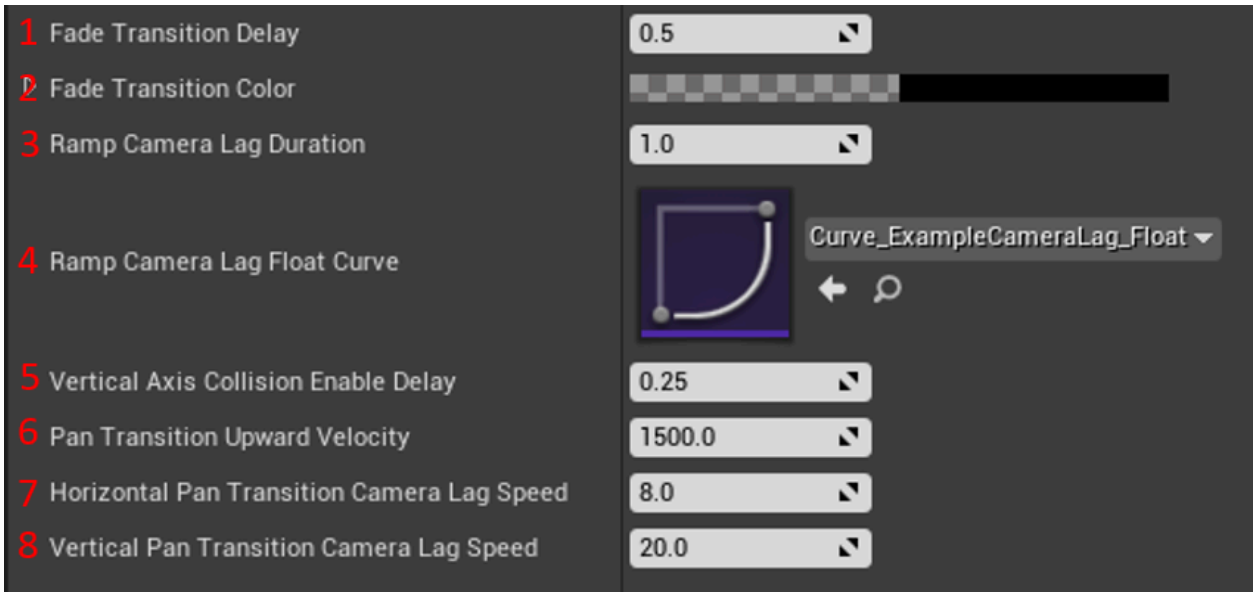
The transition type drives the look and feel of it. For Pan, the camera will pan across as your character moves across the transition. For Fade, it will fade the camera out, teleport the character to the other side of the transition volume, and fade back in.
The transition orientation will result in special behavior if it's set to Vertical. For vertical transitions, if the actor component logic detects that the entry point is lower in the Z axis than the exit point, it will launch the character upwards.
Debug will show the bounds while in-game.

/!_IMPORTANT_/!

Your player character must contain the “AC_TransitionLogic” actor component. To add it, go to your player character and click on “Add Component” and search for AC_TransitionLogic.



Once you have this component, you can further control what happens with these transition actors:



1. The fade transition delay determines how fast the fading occurs.
2. You can also change the fade color from the default black.
3. The ramp camera lag duration determines how long the camera will have lag enabled after coming out of a panning transition. Make sure it’s not too short for the float curve you set, otherwise the camera will suddenly “snap” to the player after coming out of a transition.
4. The float curve is used to determine how quickly the camera catches up to the player after a panning transition. By default, it’s set to a linear ramp from 10 to 100 over the course of 1 second. The higher the number, the more quickly the camera catches up to the player. For a smooth transition, start from a relatively low number and ramp up over time to a high number (e.g. 50 or up)
5. The Vertical Axis Collision Enable Delay is how long to delay enabling vertical collisions after exiting a panning transition. When moving through a panning transition, the camera probe collision is temporarily disabled. If it’s re-enabled too quickly, the up and down facing collision

probes can interact with the camera blocking volume, causing sudden jolts. If you experience this, try gradually increasing this value (max of 1s).

6. Pan Transition Upward Velocity is the magnitude of the upward “push” given to the character when going through a vertical panning transition.
7. The Horizontal Pan Transition Camera Lag Speed determines how quickly the camera will follow the player when they go into a horizontal panning transition. Higher value -> “snappier” follow.
8. The Vertical Pan Transition Camera Lag Speed is similar to 7), just for vertical panning transitions. This value will typically be higher than the horizontal, as the character will jump/fall at higher speeds than running.
11.

Troubleshooting Common Issues

Camera is pushed upwards

The camera works physically, meaning that if your floor is set to collide with the camera channel, it will push it upwards. Simply set your floor collision to custom and make the camera overlap or ignore.

Camera randomly jerks

The camera works physically, meaning that any movement is due to one of the probes hitting a target. The recommendation is to probe the area and check the collision settings in order to see what causes it. If you see jerks during transitions, try setting the Vertical Axis Collision Enable Delay to a slightly higher value.