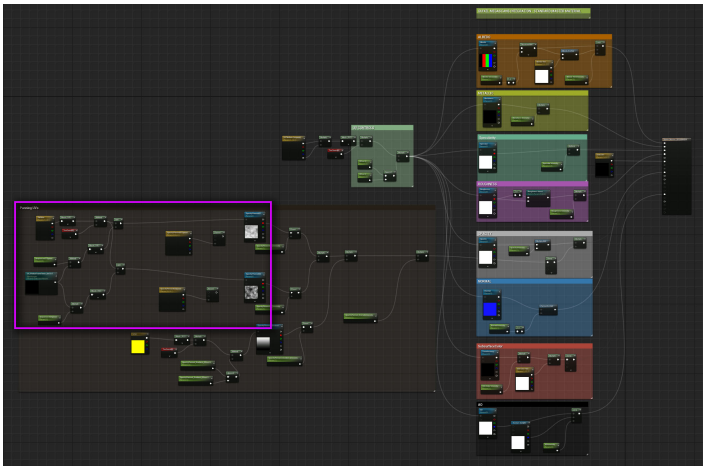
Path Traced Panning Textures in UE5

Chris Lomaka https://chrislomaka.com/

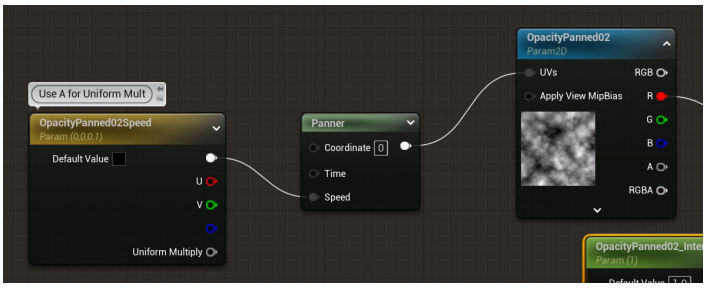
Let's imagine that you have this beautiful level in Unreal. You're 'Lit' mode while working on it, occasionally going to 'Path Tracing' since you know you'll be rendering out for sharing. Looking nice. You set up blinking lights and steam cards animated with the usual 'Panner' node in the material. You make a camera sequence so you can render it out, switch to 'Path Tracing', and render out a few seconds into a video file. You're ready to celebrate being done when you open the video file and.... wait, the blinking lights are staying on and the moving steam is just a barely-there blur. What the heck?

What's happening is this: path tracing takes a while to render each frame (a couple of minutes for me in this case), BUT that panner node in the material is moving that texture at 30 bits per second. Sequencer, (as far as I could tell) does not freeze everything while it waits for the path tracing to finish before moving on. So while pth tracing is taking its time to do all its passes, it's catching that blinking light on and off, and that steam keeps on moving. Each path traced frame was essentially several minutes worth of everything blinking and moving all smooshed into one frame. And then again for the next frame.

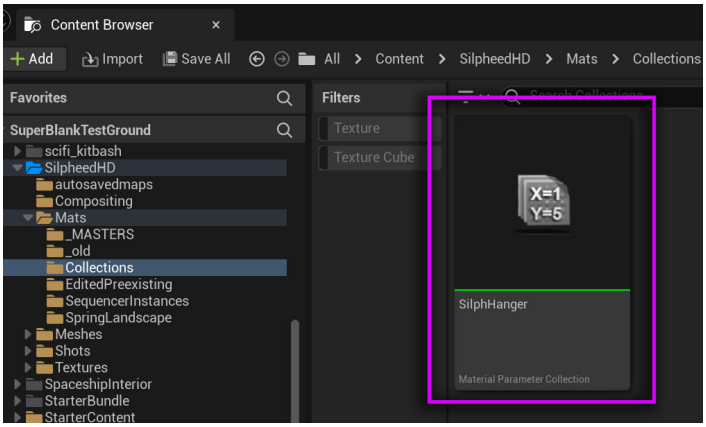
How do we solve this? Long story short-ish, I replaced the panner node in my material with something I can control in sequencer – a material parameter collection.



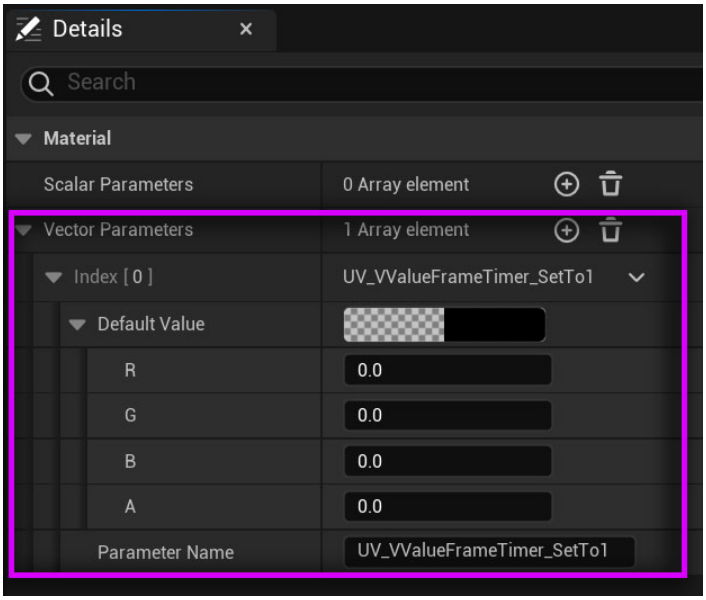
1. First things first, let's take a look at the material. Most of this is from MegaScan's basic material with a few tweeks. What we're going to look at is in the hilighted box. For the steam material I'm animating the opacity, so that's what we need to fix. (Technically there's three of four opacity layers all moving at different speeds and opacity levels all multiplied together. We'll just tackle one here. The blinking light has a similar setup but the moving part is the emissive)



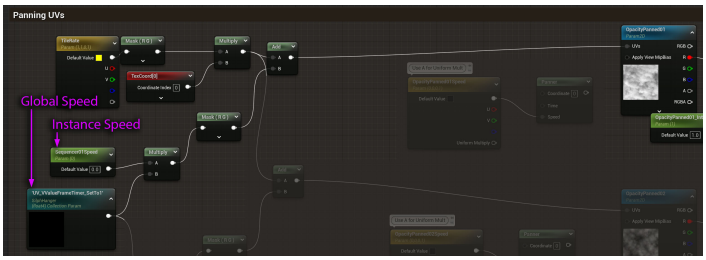
2. This is what you would normally use to animate a texture:
   1. A vector parameter to control the U and V speed, plugged into
   2. a panner node, plugged into
   3. the image UV input

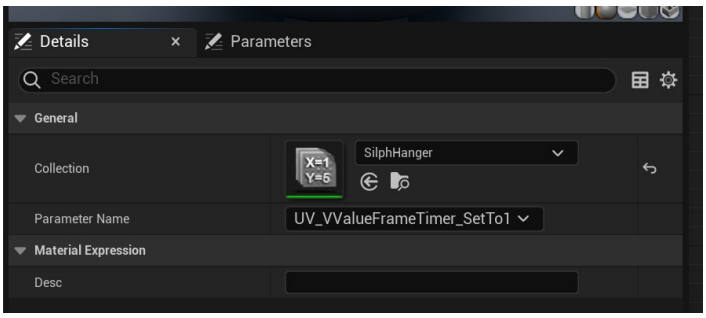


3. Let's create a Material Parameter Collection



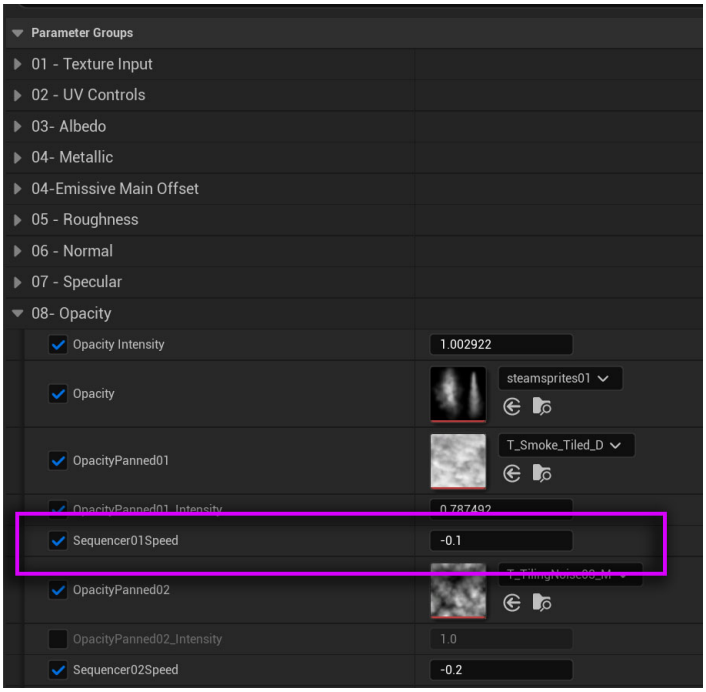
4. Open the collection and add a vector parameter. **This** is what we are able to control in sequencer. (please ignore my typo and the note of set to 1 in the name)





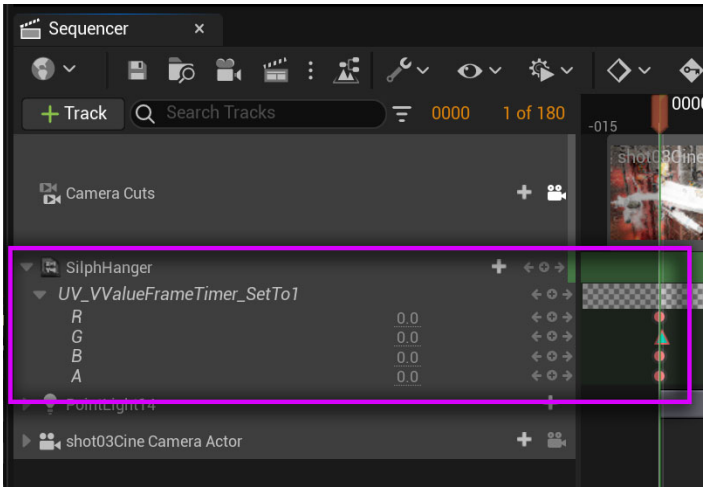
5. Let's go back to our material and
    1. add in the material parameter collection (you can just drag it over from the content browser)
    2. Go to the details of the material parameter collection and set the parameter name to the one we just created.

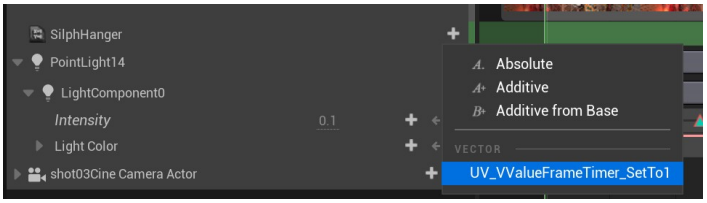
Graph explaination: this parameter is going to be shared by ALL the steam cards (and the blinking light mat). It essentially is a global speed. I then multiply it by a parameter that i can change for each material instance (this probably should be a vector param). I then mask it to R,G so it's just U and V info, and then add it to some tiling control nodes.



6. Next we make material instances. It will have a parameter that we can adjust for each one so we can have our steam materials going at different speeds. Remember – this value gets multiplied with the material collection parameter.

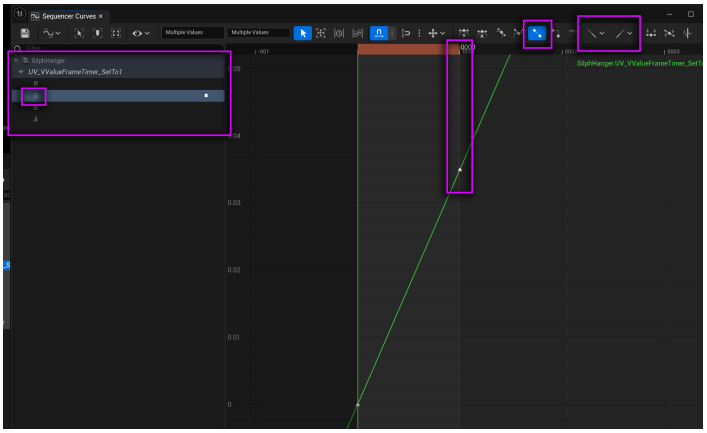
    My goal was to make it so that I could plug in the same values I was using for the panner node into this value. I spent all this time trying to do the maths for changing my (old material) panner values into the correct material instance value, but I got something wrong so I don't have much advice except whatever calculation you use, use the same math for ALL the material instances you have. As long as you do that, if you miscalculate (like me), everything will be equally incorrect and all you have to is adjust only the material collection parameter in sequencer (step 8)





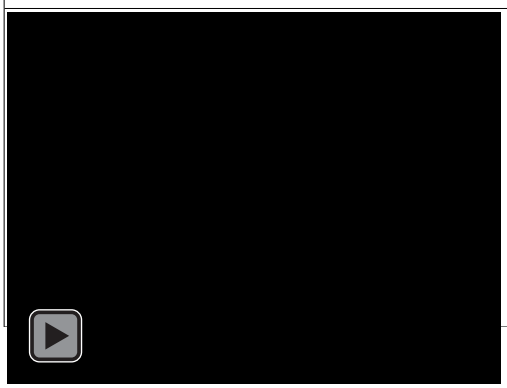
7. Now we can get back to our sequencer shot.
    1. add in the material parameter collection (you can just drag it over from the content browser)
    2. Click the + and select the vector parameter we made from the bottom
    3. expand it so we can see the RGBA values (remember, we're just using R and G for U and V values)
    4. we want a keyframe at 0 and at 1 (or whatever you want, but i thought it would make the math easier)

For once in this whole process something went my way: all the images that I'm panning are all going the same direction (down), so I just need to key the G. Things could be altered if I have to pan in different directions, but I didn't need to so I didn't



8. Open the curve editor and set your keys/curve to linear tangents and linear before and after behaviour. That should do it except for controlling how steep that curve is.

    Your numbers will vary depending on your needs, but here's some guidelines I used:
    1. I animated the G value to increase. This way a negative value in the material instance (step 6) would do the same thing as a negative value plugged into the old panner node I was using.
    2. To adjust the speed of **everything**, adjust the material collection parameter in sequencer. Once that looks good for most things, then go into each material instance to adjust anything. Thankfully, since I did the same calculation for all the material instances, all I needed to change was the sequencer value.
    3. You can do *quick* tests of the speed of everything by (temproarily):
    - smaller image output size
    - lowering the settings on the path tracing
    - (untried, but one should also be able to render out in normal lit mode. Things *should* move on a per-frame basis, no matter what render mode you're in)



9. Render out your sequence! Yea!

    It's a little hard to see at this size (the blinking light is the most obvious), but I hope this guides you through all the bits that need to be set up and how they work together.
    -Chris