Day 5

Taraash

Introduction

I'll first talk about some recommended ways to collaborate on the project. I'll then go over landscape splies, which we'll use to create a procedural road. I'll then end with some talk about optimization — culling and world position offset.

Collaboration

I would've preferred markdown for this, but here goes:

Git (with LFS) Unreal Engine has built-in support for Git (albeit in beta).

1. Install Git if haven't already (You could run (in powershell or similar) `winget install Git.Git` for a quick install, but I'd recommend using the installer).
2. Get GitLFS (large file system) (it's included by default in Windows, otherwise, git-lfs.com).
3. I'd recommend putting `Content/Fab` or similar heavy assets in the `.gitignore`. Could even put it in `.gitattributes`, but I'd recommend just telling your fellow teammates to download whatever you downloaded locally, there's no need to upload them.
4. I wouldn't really recommend using LFS because it'll be difficult to tell it which files it should track, every `.uasset` will be a waste, but you can ofcourse see if it works for you.

Diversion There's a new up-and-coming technology, Diversion that has a ton of storage for free. You'll need to get it's plugin here. (Linux is not officially supported.) You can use this if you find it easier, but I'd still recommend not uploading heavy assets, your teammates can download that themselves.

Subversion SVN is the Git's predecessor and has much better integration with Unreal. That being said, it is much slower than Git (but we probably don't need to create branches and merge stuff here, so not really that big of a deal). You might wanna try it out, download it here.

Perforce There's perforce. A nightmare to set up and you need a local server. Of course, not recommended, but worth mentioning since it's the industry standard for version control in games.

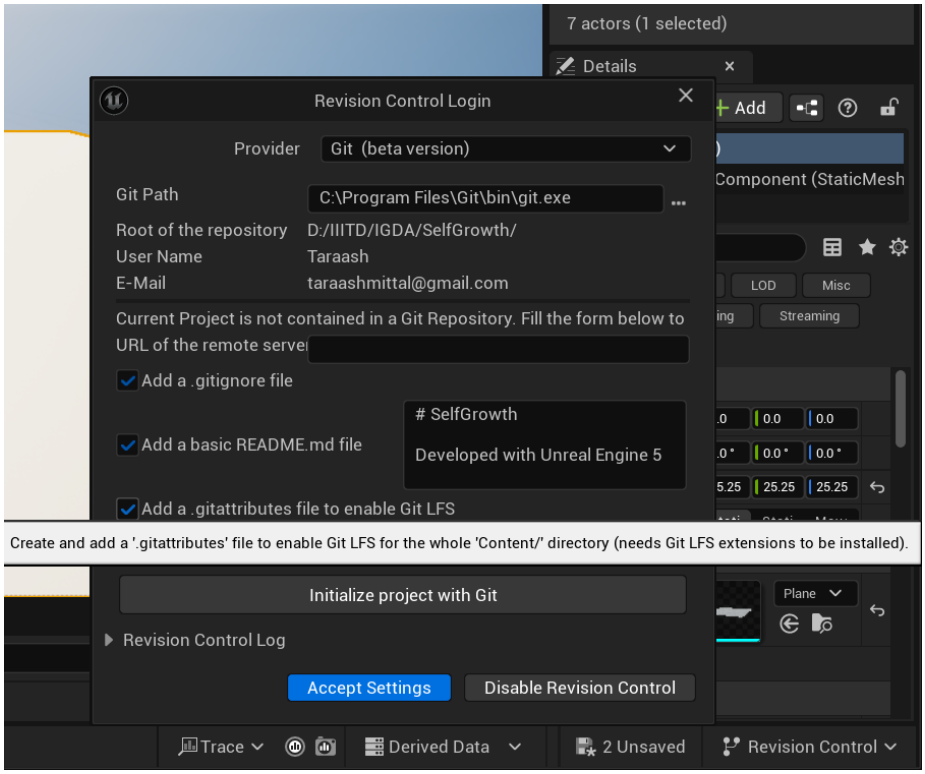


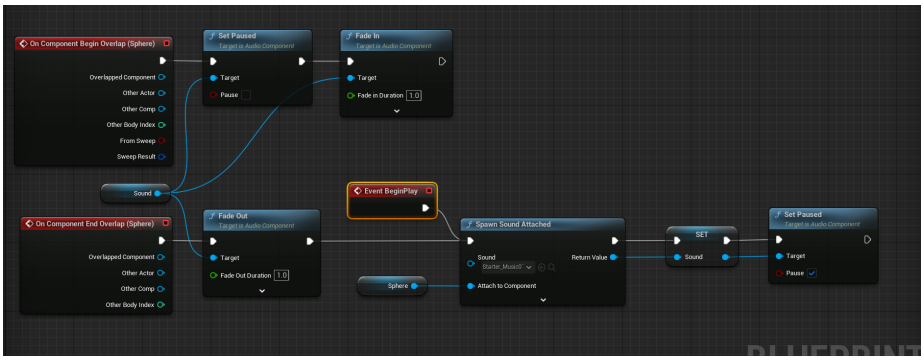
Figure 1: Bottom left in the editor, the revision control tab.

Uploading your game

Check out itch.io or indiexpo to host your game if you'd like.

Assignment 1

The sound component might've given you some trouble, in particular, it probably wasn't fading out. I demoed this in the class, take a look at this screenshot:



I demoed a different way to do this in the class, using the `Do Once` node and spawning the sound at begin overlap, but the one above comes from error-driven-development. I tried the simplest way to do it, got some errors, saw what they were and fixed them. The error I got was fading in a sound after we've faded it out (going in for the second time). This happened because once we've ran the fade out node, Unreal thinks we don't need it anymore and sets it off to be garbage collected. So I spawn the sound after fading it out once again. (When multiple exec pins are connected to the input of another node, that node is run whenever it receives *either* of the exec pins. Here, on BeginPlay, we spawn the sound so we have something on our first BeginOverlap, after which we fade it out and spawn it so we have it in the next BeginOverlap.)

Landscape splines

I then talked about a slighly niche thing, landscape splines. This is a non-destructive, procedural way to create roads and paths in your landscape. The landscape automatically deforms to match the height of the spline, but don't support collisions very well. If you need a big straight line or something, I'd recommend the Ramp brush.

This gets you proper collision. I then talked about adding a road to a landscape spline, take a look at Epic's documentation on landscape splines for more information. Let's now create a road.

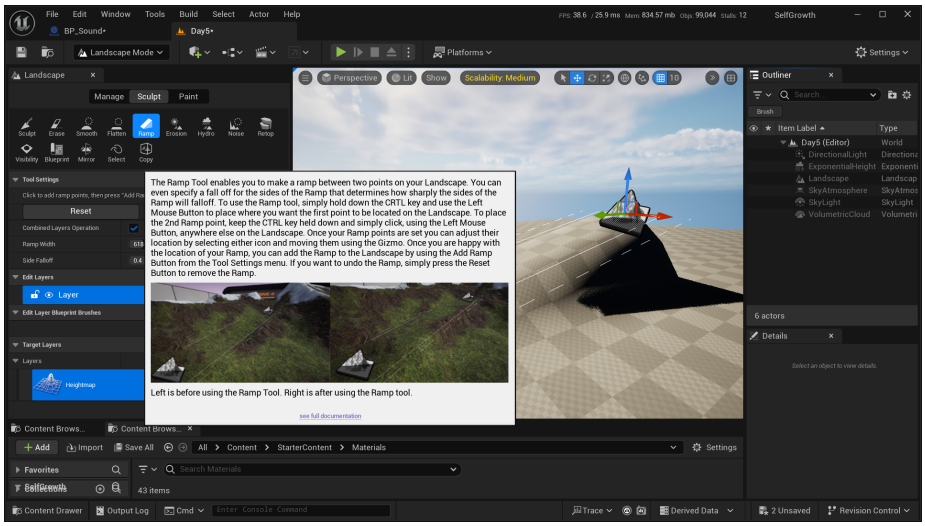


Figure 2: Hold Ctrl+Alt while hovering for more information

Creating a road

We need a material, I downloaded one off of Fab. Dragged off a plane from the add menu and applied this material to it. We know have a road, but we need it as a static mesh to be able to use it in our landscape spline.

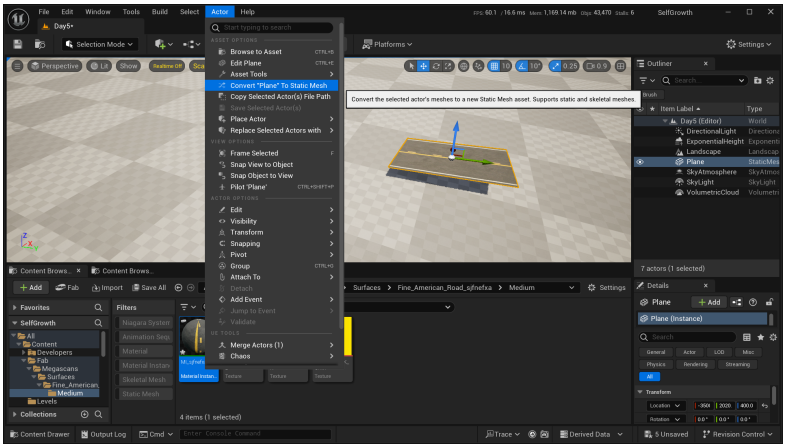


Figure 3: Converting to a static mesh

It'll ask you to save it, so naviagte to where you wish to save it, give it a name and click save. I named mine `SM_Road`, let's now apply it to our spline. (Don't forget, while creating a spline you should be in the landscape spline layer, not the regular edit layer.)

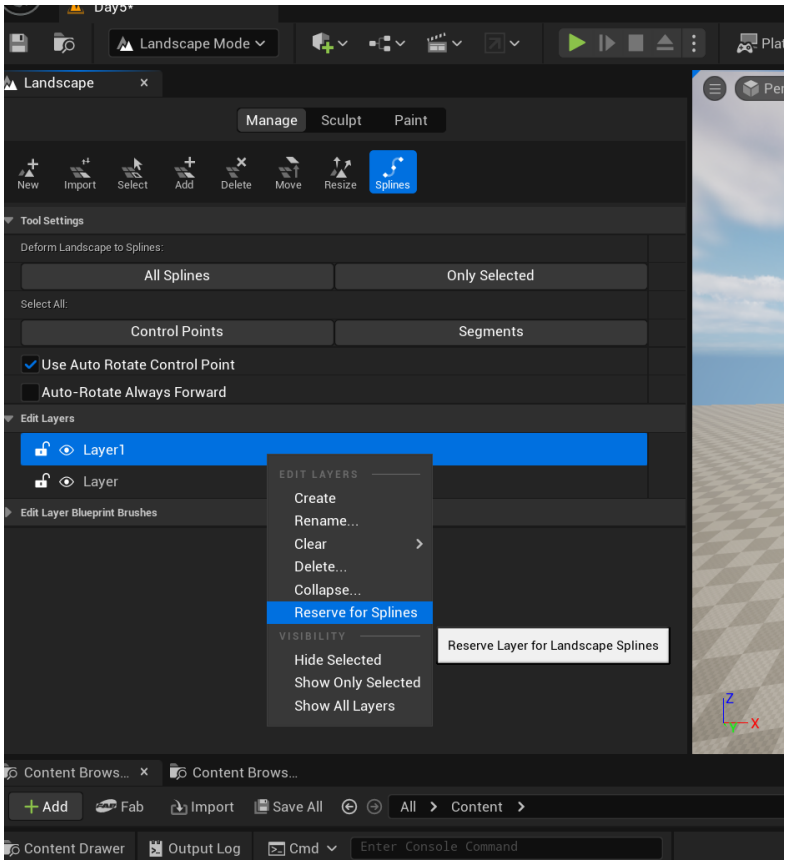
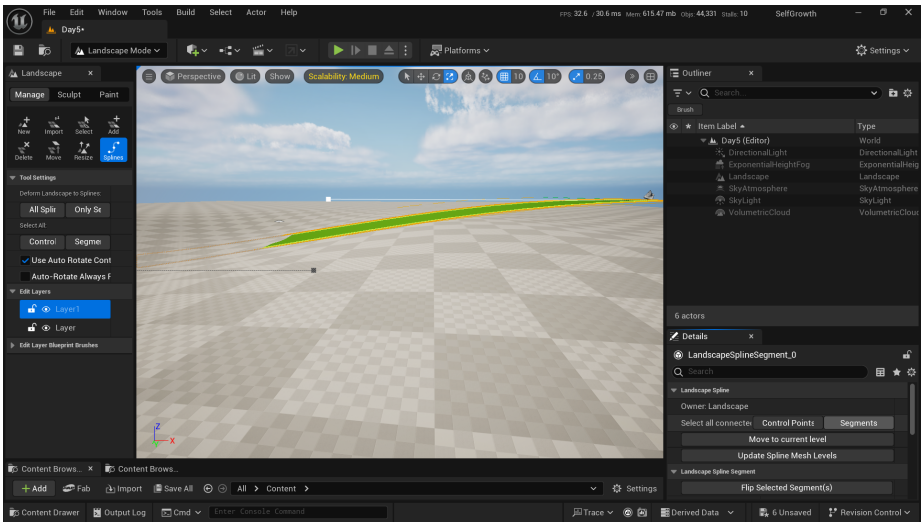


Figure 4: Getting to spline mode

Do follow the documentation I've linked above, but simply

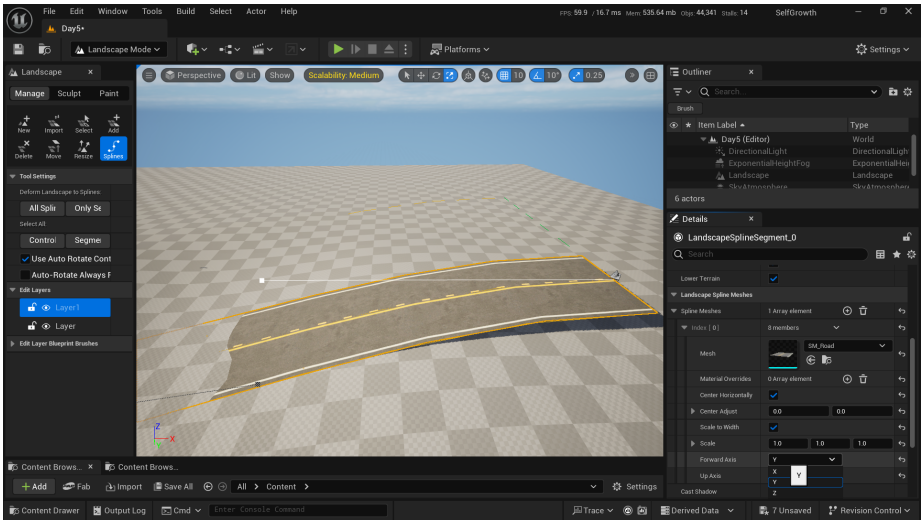
1. `Ctrl+Click` to place a *control point*, this allows you to manipulate the spline.
2. Add as many points as you need, and manipulate the control points to get what you want.

3. Find and press the `Control Points` button (in the details panel, select all connected option)

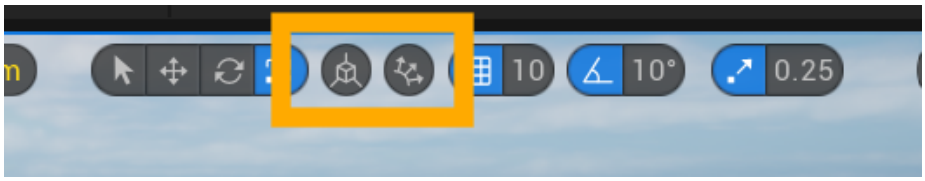


4. Scroll down and find the `Spline Meshes`, click the $\oplus$ button, expand the drop down you would see and slot in the static mesh we created earlier.

5. You'll see the road is oriented the wrong way, find and change the `Forward Axis` to something that fixes it, Y in my case. (And probably yours too.)



While we're here, let me bring your attention to these two options:

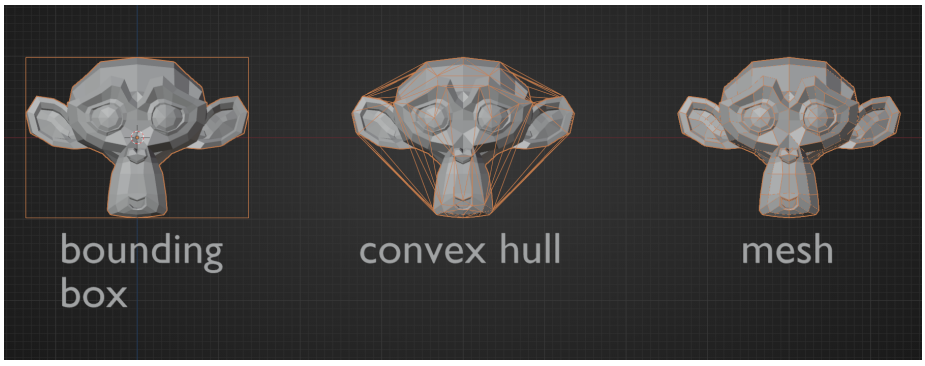


- The first one allows you to change your coordinate system from local (object) to world (global). You'll find it helpful when you've rotated an object and want to move it in the direction it's facing, rather than the global axes. Play around with it, you'll get the hang of it.
- The second one is the snapping option. If you've got a rough/curved surface and want to place an object cleanly on it (respecting the average surface normal), you can use this. It will appropriately rotate the object to match the surface normal, so it looks like it's placed on it. I quickly demoed this in the class by placing a car on a slanted surface.

Collision

I'll talk a little about different collision meshes since I didn't cover them very well earlier. You should aim to keep as little faces you possible in a mesh that is to be considered for collision since its algorithm has to loop through each face. The simplest collision meshes are often the most inaccurate, but inexpensive to work with. You can use your actual mesh as the collision mesh, which gives the most realistic result (we often don't need it to be so, it'll be difficult for us to properly utilize this fidelity! Remember that our default ThirdPerson character also has a simple cylindrical collision).

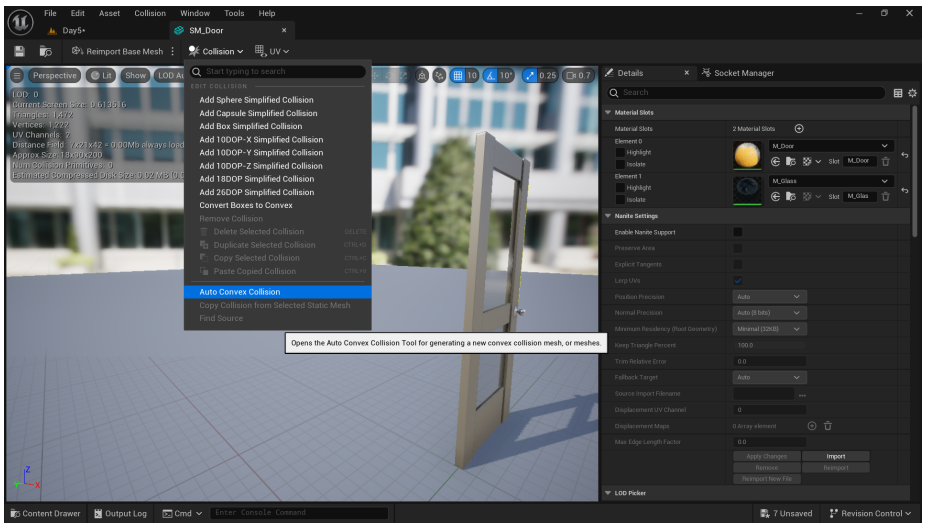
A usually good middle ground is a convex hull. I'll show them in Blender first, and then show how to create it.



Take a look at this, the first one is a simple box, fewest of faces but least accurate. The second is a convex hull, it wraps our mesh fairly well and still uses

relatively few faces. The third is our actual mesh, which is the most accurate but also the most expensive to work with.

In Unreal

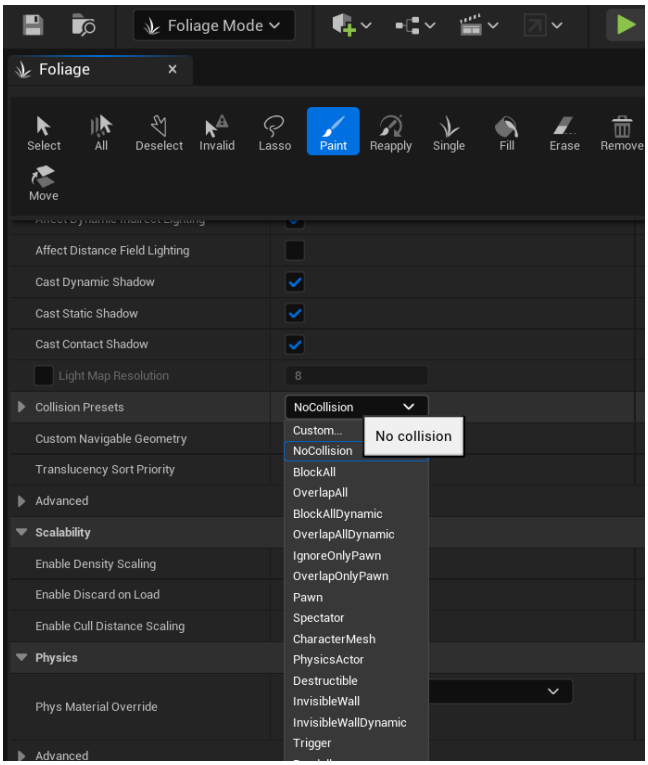


Press the auto convex collision button to open up a new menu, where you can tweak a few settings which will get you a different convex mesh to be used for collisions. Note that the *simple collision mesh* is the one used in games (real-time) and the *complex collision mesh* is always the actual mesh. We can set to use the complex collision mesh and the simple one from the collision complexity setting.

You could view these collision meshes by going to the `Show` panel (right next to where it says Lit) and selecting what you wish to preview. You can also use the `Collision` view mode to see the collision meshes in your scene. This is helpful to debug collision issues, but also to see how many faces your collision mesh has.

Collision in foliage

By default, when you put in trees in your scene using Foliage, they won't have collision and you can walk straight through them, this isn't what we usually want. To fix this, find this setting when you scroll down in the foliage panel (after selecting the foliage type you want to edit):



You've probably seen this menu before. I quickly wanted to differentiate between `No Collision` and `Overlap` since they seem to do the same thing. This is true, they both let our character pass through and don't stop it. But, in the blueprints editor, we can get an event that fires off when our character overlaps if we've set the preset to `OverlapAll`, which we do not get if we've set it to `No Collision`. This is useful if we want to trigger some event when our character walks through a tree, for example, we can play a sound or something.

Foliage optimization

Foliage is quite difficult to run in real time, mainly for two reasons:

1. Number of instances. For good looking grass, you need a lot of it.
2. Wind animations. Calculating and applying these transformations isn't very quick!

To combat the number of instances, we can use something referred to as `Culling`, this simply deletes our grass if it's some distance away from our character (we also get a gradual fade, so it isn't a strict cutoff). Take a look at Epic's documentation for PerInstanceFade if you'd like to learn more, the PerInstanceFade node in the material editor gives us a simple gradient from 0 to 1 between the

character the maximum set cull distance. Usually, setting the max cull distance

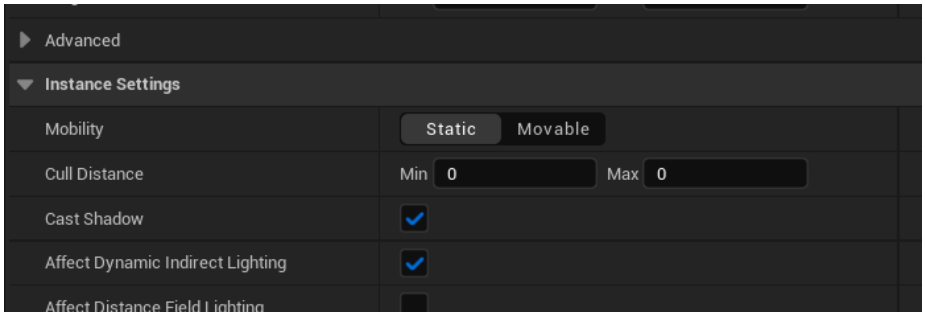


Figure 5: Min and max cull distance in the foliage settings.

to about 4000 is good enough, and we keep the minimum to zero.

Wind animations

The world position offset is a parameter in the material editor of Unreal Engine that allows us to manipulate the position of the vertices of a mesh in the vertex shader. This is useful for creating effects like wind animations, where we want to move the vertices of a mesh in a way that simulates the effect of wind blowing through it.

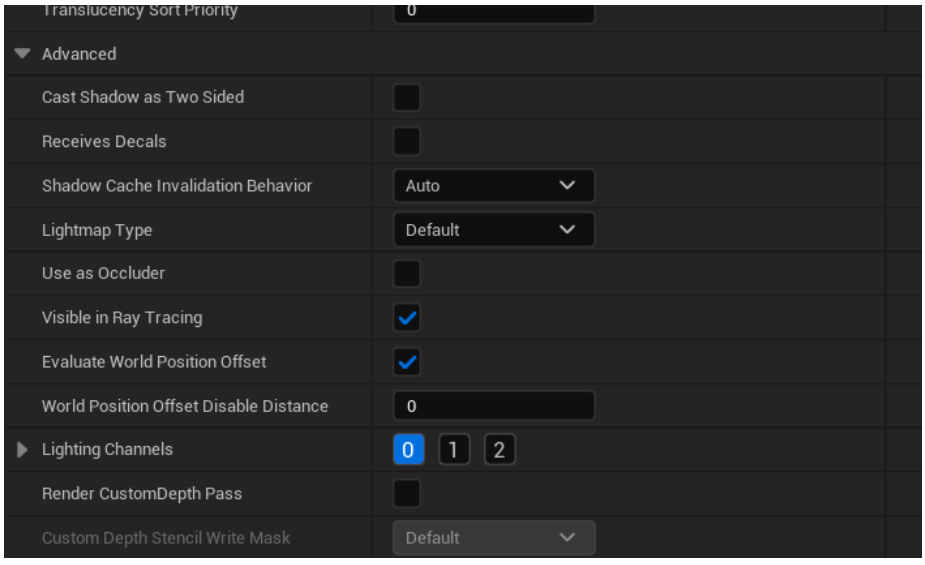


Figure 6: WPO in the foliage setting, you'll have to expand the Advanced section to see it (of instance settings).

Fairly self-explanatory, the wind animations will not be calculated after the distance specified in the `World Position Offset Disable distance` (setting to 0 will take the maximum possible value, or disables the setting, whatever you prefer to call it). This is useful to save on performance, since we don't need the wind animations for grass that is far away from the player. The wind animations are calculated in the vertex shader, so they can be quite expensive to run in real time, especially if we have a lot of grass in our scene.
The enable switch right on top of this setting is used to enable or disable the wind animations for the foliage. If we disable it, the grass will not move at all, similar to disabling real-time mode in our viewport.

Conclusion

I believe this does it, I covered everything I talked about in the class. If you've got any questions (or suggestions!), let me know.