Day 5

Taraash

1 Introduction

I'll first talk about some recommended ways to collaborate on the project. Then we'll go over some optimization for foliage, particularly culling, world position offset and using materials (with wind). **These are rough, improve them after the class**

2 Collaboration

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

Git (with LFS) Unreal Engine has built-in (though in Beta) for revision control using Git.

1. Install Git if haven't already.
2. Get GitLFS (large file system) (it's included by default in Windows).
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.
5. Read more about everything on git-lfs.com.

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.

2.1 Uploading your game

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

3 Assignment 1

Talk about assignment 1

4 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 it's 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 (geometry nodes), and then show how to create it.

Show the different collision meshes in the mesh editor. Talk about use complex collision as simple.

Make trees collidable in foliage mode. Then talk about optimization.

5 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 if 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 can 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.

We also get a simple `World Position Offset Disable Distance` parameter that does exactly what it says; stops calculating this offset (here, wind) after the specified distance.

6 Materials

I covered materials in my week 1 recap, I'll just construct a basic material and talk about instances and parameters.

6.1 Foliage Materials

Last thing we'll talk about is giving it the wind animation, get the simplewindanimation node, multiply it by a bounding box node, maybe show a lil gradient in blender? 0 at bottom, multiplying by zero nullifies our wind, keeps the bottom stationary, what we needed.

7 View modes

Once again, Epic's documentation on viewport modes. These are helpful to get a more detailed look at your scene, with hopes to optimize it and find bottlenecks or resolve artifacts. The important ones are

1. Light complexity
2. Shader complexity
3. Collision
4. Nanite
5. Scene depth

8 End

End?