* Proper stride length when you initialize a compute buffer is important so you don’t get stragglers (particles that are not acted on by the compute shader). I had a bug where a single particle would stay stationary even though the rest moved. Turns out it was because of an incorrect stride.
* If you don’t want to use a geometry shader, use a mesh in Unity to create particles (Unity doesn’t have a “particle creator” per se). However, you can’t control the size of the particles if they are individual verts. You can use a mesh to create triangles and then create your own custom size, but since Unity limits all objects to have 65,000 verts, you have at most 21,666 particles with this method.
* Dynamic height map from the terrain function is incredibly slow in Unity. Just create the geometry in Maya instead
* Maya does not make it easy to stitch two textures together when two different texture are applied to different faces of an object. I found a crazy workaround where I generated a “flattened” texture and then applied that texture in place of the original two.
* A compute shader in Unity does not allow you to use certain functions to sample a texture that are valid in other shaders
* Other differences between compute shader and fixed pipeline in Unity. For example, time is a predefined macro in the fixed pipeline. Time must be passed into a compute shader from the GPU
* OpenGL and DirectX have different terminology for compute shaders, which is confusing. For example, per DirectX, “SV\_DispatchThreadID is the sum of SV\_GroupID \* numthreads and GroupThreadID.” And per OpenGl, “The value of gl\_GlobalInvocationID is equal to gl\_WorkGroupID \* gl\_WorkGroupSize + gl\_LocalInvocationID.” These are both simply how you index the buffers passed into the compute shader.
* Seems like if you try to request data every frame from the compute shader, it doesn’t do anything for the fluid sim because it hasn’t completed calculations yet. I have it requesting data every 10th frame.
* What are the x, y, z, and w of DispatchThreadID? May encounter issues with this being different than the OpenGL indexing.
* Density outputting 0 when delta looks at xyz instead of xy
* I made one large struct with position, velocity, force, density, and pressure all belonging in one massive particle. Is there a reason to split this into 5 buffers instead of one?