In order to create procedurally generated planets, I had to first find the right geometry to do it. As It turned out using Unity’s terrains is a bad idea because they can’t be rotated, therefore I can’t connect them in the form of a sphere (which I originally tried).

So instead I found a better way by using an octahedron approximation of a sphere (I’ll reference the code I found for it), because it has much more vertices than ordinary sphere and because the polar regions are not twisted. At this point I had to find a way to implement noise on the surface to create the terrain. I found LibNoise library that provides a lot of noise modules. Basically by combining noises such as Perlin, Billow and Multifractal I can achieve complex terrains. Applying just pure noise to displace the vertices of the sphere ends up in sine-like waves everywhere on the surface so that is not a good solution. Instead I created a 2D spherical projection of the noise, thus creating a height map. I applied the map to displace the vertices of the sphere and it worked very wells, and I can create different maps depending on the seed. I’m not creating very complex terrains at the moment because that involves combining many modules in different ways, which I’ll do later on.

After that I implemented spherical gravity that attracts bodies to the surface of the planet, which worked as expected. Also implemented basic FPS controller so it allows me to walk on the planet. For the water I used another sphere that goes inside the planet and has a water material and when the surface elevates the lower regions are covered in the water. I also used a space-like skybox and I used directional light for sun, which I made temporarily to circle around the planet to create the effect on a day-night cycle.

The problems I face now are the following:

1. The more noise functions I combine with Libnoise, the more calculations it performs and it takes a lot of processing power and slows down everything and unity sometime crashes. I know I have to implement threads to solve this, but it didn’t quite work and it turned out Unity’s threads are very limited.
2. Creating textures that blend depending on terrain elevation seem only possible with shaders and not C# script.
3. Creating realistic atmosphere and water material requires using shaders as well. I don’t know much about shaders or how to program them. It might require learning HLSL…