One Man's Sky

Shawn Wu and Chenyang Wang

December 1, 2016

1 Introduction

One Man's Sky, the hit* sequel* to No Man's Sky, is a flight simulator that takes place in a procedurally generated world. Players can soar through the air across beautifully* rendered landscapes that go on as far as the eye can see.

2 Procedural Generation

The landscape is divided up into a grid of chunks. Each chunk has a unique position in the x-z plane that determines its random seed. This seed is then passed into the Perlin noise function to generate a heightmap for that chunk.

Perlin noise is a composition of random noise of exponentially increasing frequency and exponentially decreasing weight. At each octave (frequency and weight combination), random noise is upsampled to the desired resolution using linear interpolation. Edge pixels are upsampled using the noise from neighboring chunks, so that the terrain between chunks is smooth.

The resulting noise is then passed into a weighting function that transforms the values into more terrain-like heights:

$$y_f = \begin{cases} \frac{0.5}{1+2.8^{-7*y}} - 0.25 & y < 0.5 \\ 2.8^{y-0.5} - 0.763 & y >= 0.5 \end{cases}$$

New chunks are generated when the player is no longer over the center of the generated chunk grid. At this time, any unnecessary chunks (far behind the player) are freed from memory, all faces and normals are generated, and the data is passed to the GPU.

Water is a plane at $y = 0$.

3 Rendering

3.1 Terrain

The terrain is rendered using a simple Lambertian lighting model. The color is determined from the world y position and a small random factor; this randomness is polled from a Perlin noise texture that has been interpolated with itself. By using this texture, the randomness has structure and loops properly.

3.2 Skybox

The skybox is simply 6 textures mapped to a large cube at infinity and drawn behind all other objects. We achieve this by temporarily disabling the depth check, drawing the skybox first, then reenabling the depth check and drawing the rest of the scene.

3.3 Water

The water has a reflective component. Reflections are drawn using a second render pass from the position of the reflected camera, which is underneath the water:

$$p_r = p_c * \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$

During this second render pass, the skybox and all terrain above $y = 0$ is rendered to a framebuffer, then bound to a texture. This texture is passed into the water fragment shader so that reflected pixels can be sampled.

The water's reflectivity is determined by the Fresnel effect, which is calculated using Schlick's Approximation, where θ is the angle of incidence and $n1, n2$ are the indices of refraction:

$$R(\theta) = R_0 + (1 - R_0)(1 - \cos\theta)^5$$

$$R_0 = (\frac{n_1 - n_2}{n_1 + n_2})^2$$

4 Physics

The plane's physics take into account gravitational, drag, and lift forces. However, the user has full control over the pitch and roll of the plane, making the simulation more like flying a perfectly controllable rectangle than a real plane with flaps. The position of the plane is updated at each frame according to the time delta from the last frame; this is so that the plane obeys the laws of physics regardless of framerate.

The plane can crash into the ground, where it will release a large amount of particle effects and soft-lock the game.