Terrain Instancing using height-maps

Research and Development
Subhir Rao

Step 1: Create a mesh of triangles, with vertices at every unit point, of size 33x33. That should give you a model with 1089 vertices and 2048 triangles. This is the mesh we are going to instance. The reason for 33x33 is to allow the instanced meshes to overlap, so we don't have to be concerned with stitching the meshes together.

Step 2: Create your vertex declaration for instancing. In our implementation, we use four float4 variables (fog, bi-normal, tangent and blend-weights) in the second stream to store our world matrix. The method of recreating the world matrix is of little importance, so it can also be done differently if desired. The first stream is also up to the user and in our case, has two vector3 variables for position and normal and one vector2 for the texture coordinate.

Step 3: Create the instance data, instance buffer and store the relative world matrices in them. We decided to instance the terrain 64 times, centered on the player's current position, in an 8x8 grid. This gave us enough coverage so you can see the terrain at all times without open holes and without too much of a performance hit. The result of instancing was to give us a total of 65,536 vertices and 123,008 triangles (not including overlap).

Step 4: Then just draw the model along with the instance data and you are done. Well not quite, you still need to do some things with this instanced mesh in the shader. The first thing is to unpack the world matrix from the second stream. Then in the vertex shader, multiply each incoming vertex's position with the unpacked world matrix. Check the new x and z values of the vertex position to ensure that it is between 0.0f and the game environment size.

If it isn't, set the texture coordinates to -1.0f and then discard the pixel, in the fragment shader.

If it is, then we have got ourselves a vertex which will be a part of the terrain. Get the vertex's actual texture coordinates by dividing its new position by the size of your terrain. Then using the new texture coordinates, lookup the vertex's normal from a texture which stores the terrain normals. And use the new texture coordinates to retrieve the vertex's height from our height map.

That's it!