Using Noise to Automatically Generate Generic Terrain
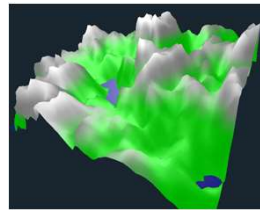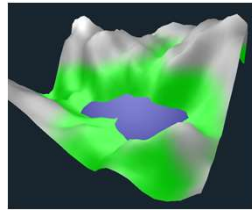
Oregon State University

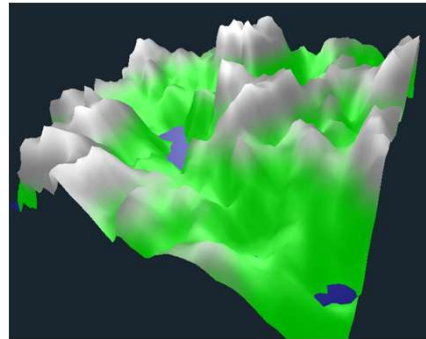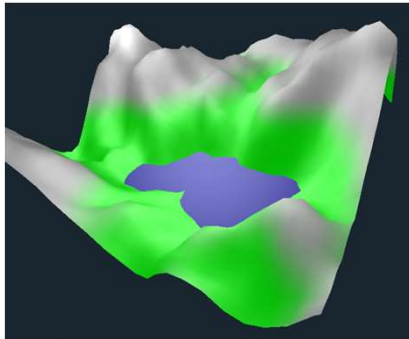**Mike Bailey**

mjb@cs.oregonstate.edu

Oregon State University
Computer Graphics

noiseterrain.pptx

mjb – December 17, 2021

The General Idea

Use noise to determine terrain heights. Utilize as many different parameters as we can to give a variety of terrain.

Oregon State University
Computer Graphics

mjb – December 17, 2021

terrainnoise.glib

```
##OpenGL GLIB
Perspective 70
LookAt 0 0 3  0 0 0  0 1 0

Vertex      noiseterrain.vert
Fragment    noiseterrain.frag
Program     NoiseTerrain                              \
            uNoiseAmp <0. 0. 5.>                      \
            uNoiseFreq <0.1 0.2 0.5>                  \
                    uDelta 0.1                        \
                    uBiasx <-2. 0. 2.>                \
                    uBiasy <-2. 0. 2.>                \
                    uBiasz <0. 0.1 1.>                \
                    uLevel1 <0.1 0.2 0.8>             \
                    uLevel2 <0.4 0.6 1.0>             \
                    uTol <0. 0.2 01.0>                \
                    uKa <0. 0.1 1.0>                  \
                    uKd <0. 0.6 1.0>                  \
                    uKs <0. 0.3 1.0>                  \
                    uShininess <3. 10. 200.>          \
                    uLightX <-20. 0. 20.>             \
                    uLightY <-20. 0. 20.>             \
                    uLightZ <5. 10. 20.>              \
                    uSpecularColor {1. 1. 1. 1.}

QuadXY  -0.2  1.  1000 1000
```

---

terrainnoise.vert, I

```
#version 330 compatibility

uniform sampler3D Noise3;
uniform float uNoiseAmp;
uniform float uNoiseFreq;
uniform float uBiasx, uBiasy, uBiasz;
uniform float uLightX, uLightY, uLightZ;

out vec3 vNs;
out vec3 vLs;
out vec3 vEs;
out vec3 vMC;
uniform float uDelta;


vec3 DELTAX = vec3( uDelta, 0., 0. );
vec3 DELTAY = vec3( 0., uDelta, 0. );


float
Height( vec3 mc )
{
        vec3 newmc = vec3( mc.x+uBiasx, mc.y+uBiasy, mc.z );
        vec4 nv = texture( Noise3, uNoiseFreq * newmc );
        float n = nv.r + nv.g + nv.b + nv.a;
        n = n - 2.;
        n = n + uBiasz;
        n *= uNoiseAmp;
        return n;
}
```

Reading a texture from within the vertex shader

terrainnoise.vert, II

```
void
main( )
{
        float h00 = Height( gl_Vertex.xyz );
        float hp0 = Height( gl_Vertex.xyz + DELTAX );
        float hm0 = Height( gl_Vertex.xyz - DELTAX );
        float h0p = Height( gl_Vertex.xyz + DELTAY );
        float h0m = Height( gl_Vertex.xyz - DELTAY );

        float dzdx = hp0 - hm0;
        vec3 xtangent = vec3( 1., 0., dzdx );
        float dzdy = h0p - h0m;
        vec3 ytangent = vec3( 0., 1., dzdy );
        //vNs = normalize( gl_NormalMatrix * cross( xtangent, ytangent )  );
        vNs = normalize( cross( xtangent, ytangent )  );

        vec3 new = gl_Vertex.xyz;
        new.z += h00;            // displace the point
        if( new.z < 0. )
                new.z = 0.;
        vMC = new;

        vec4 ECposition = gl_ModelViewMatrix * vec4( new, 1. );
        vec3 eyeLightPosition = vec3( uLightX, uLightY, uLightZ );

        vLs = normalize( eyeLightPosition - ECposition.xyz );
        vEs = normalize( vec3( 0., 0., 0. ) - ECposition.xyz );

        gl_Position = gl_ModelViewProjectionMatrix * vec4( new, 1. );
}
```

Cross product to get a normal vector

It's always a heated discussion about how much quality lighting to put on terrain. We usually don't multiply by the normal matrix because you generally don't turn a landform around in your hands.

Oregon State University
Computer Graphics

mjb – December 17, 2021

---

terrainnoise.frag, I

```
#version 330 compatibility

uniform float          uLevel1, uLevel2, uTol;
uniform float          uKa, uKd, uKs;
uniform vec4           uSpecularColor;
uniform float          uShininess;

in vec3 vNs;
in vec3 vLs;
in vec3 vEs;
in vec3 vMC;

const vec3 BLUE   =     vec3( 0.1, 0.1, 0.5 );
const vec3 GREEN  =     vec3( 0.0, 0.8, 0.0 );
const vec3 BROWN =      vec3( 0.6, 0.3, 0.1 );
const vec3 WHITE  =     vec3( 1.0, 1.0, 1.0 );
const vec3 GRAY   =     vec3( 0.5,  0.5, 0.5 );
```

Oregon State
University
Computer Graphics

mjb – December 17, 2021

3

terrainnoise.frag, II

```
void
main( )
{
            vec3 Normal = vec3( 0., 0., 1. );
            vec3 color = BLUE;
            if( vMC.z > 0. )
            {
                        float t = smoothstep( uLevel1-uTol, uLevel1+uTol, vMC.z);
                        color = mix( GREEN, GRAY, t );
                        Normal = normalize( vNs );
            }
            if( vMC.z > uLevel1+uTol )
            {
                        float t = smoothstep( uLevel2-uTol, uLevel2+uTol, vMC.z);
                        color = mix( GRAY, WHITE, t );
                        Normal = normalize( vNs );
            }

            vec3 Light = normalize( vLs );
            vec3 Eye = normalize( vEs );
            vec3 ambient = uKa * color;
            float d = dot(Normal,Light);
            vec3 diffuse = uKd * d * color;

            float s = 0.;
            if( d > 0. )                        // only do specular if the light can see the point
            {
                        vec3 ref = normalize( 2. * Normal * dot(Normal,Light) - Light );
                        s = pow( max( dot(Eye,ref),0. ), uShininess );
            }
            vec3 specular = uKs * s * uSpecularColor.rgb;

            gl_FragColor = vec4( ambient.rgb + diffuse.rgb + specular.rgb, 1. );
}
```
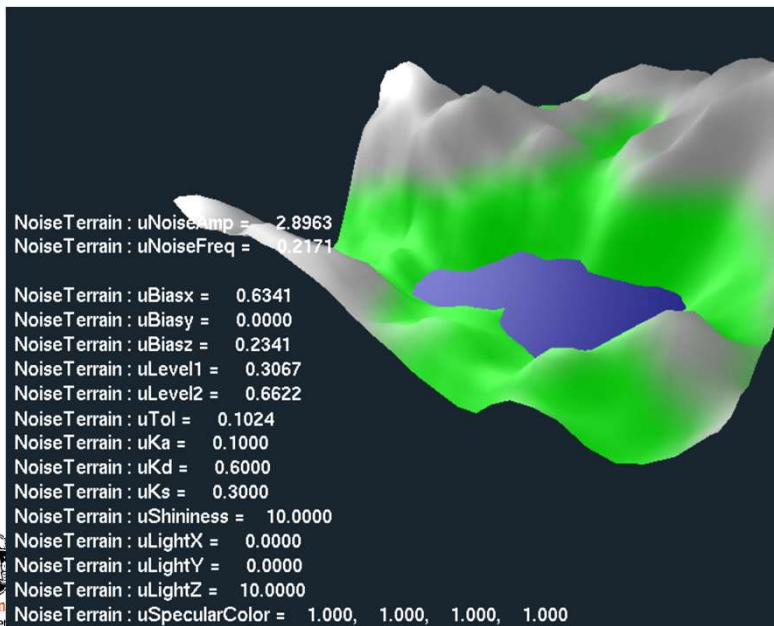
What does it mean to do specular lighting on *terrain*? No, I don't know either, but here it is if you want it.

Oregon State
University
Computer Graphics

mjb – December 17, 2021

Lots of Parameters Can Be Changed

```
NoiseTerrain : uNoiseAmp =      2.8963
NoiseTerrain : uNoiseFreq =      0.2171

NoiseTerrain : uBiasx =     0.6341
NoiseTerrain : uBiasy =     0.0000
NoiseTerrain : uBiasz =     0.2341
NoiseTerrain : uLevel1 =     0.3067
NoiseTerrain : uLevel2 =     0.6622
NoiseTerrain : uTol =    0.1024
NoiseTerrain : uKa =    0.1000
NoiseTerrain : uKd =    0.6000
NoiseTerrain : uKs =    0.3000
NoiseTerrain : uShininess =   10.0000
NoiseTerrain : uLightX =    0.0000
NoiseTerrain : uLightY =    0.0000
NoiseTerrain : uLightZ =   10.0000
NoiseTerrain : uSpecularColor =  1.000,   1.000,   1.000,   1.000
```

Oregon State
University
Computer Graphics

mjb – December 17, 2021