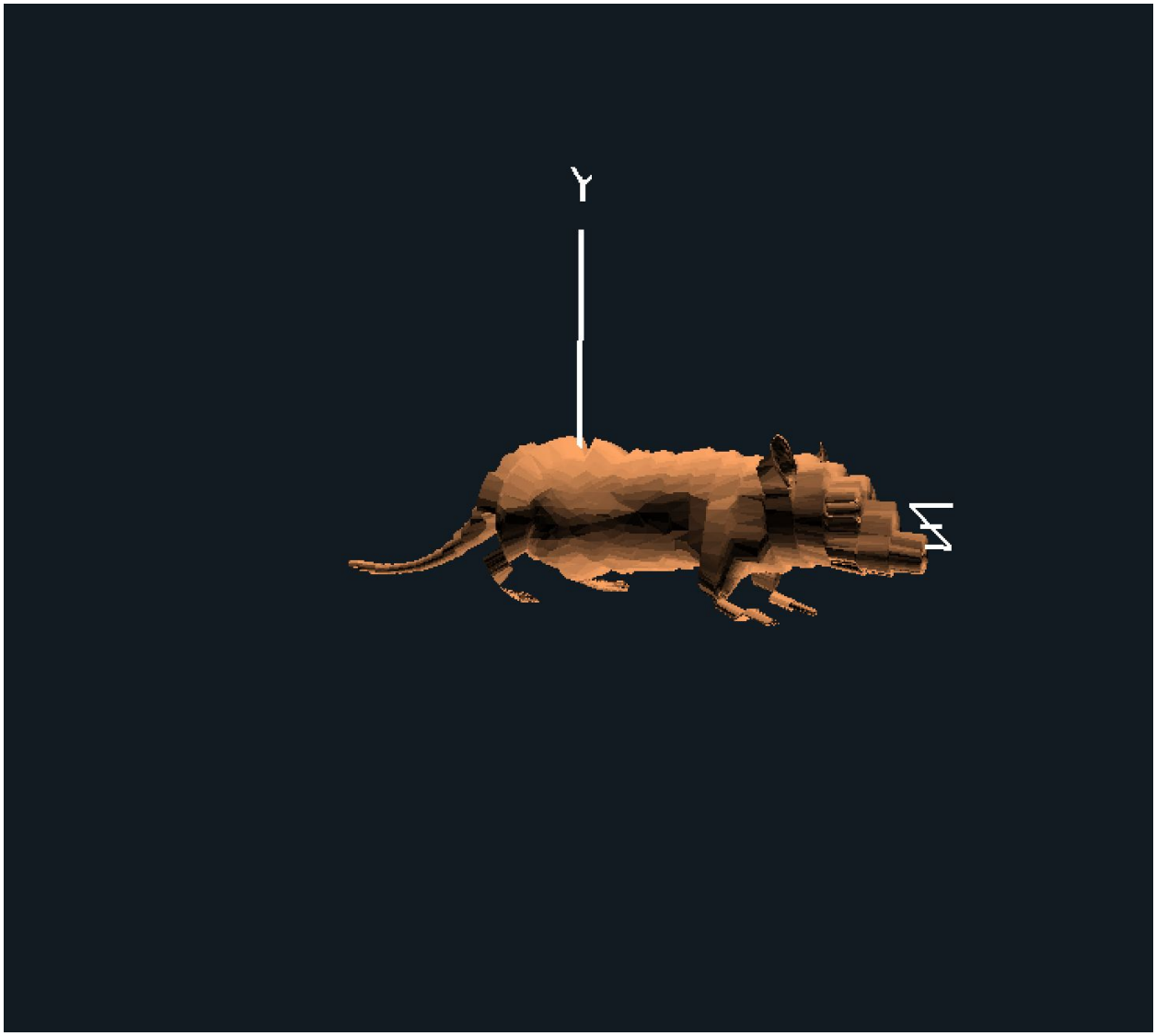
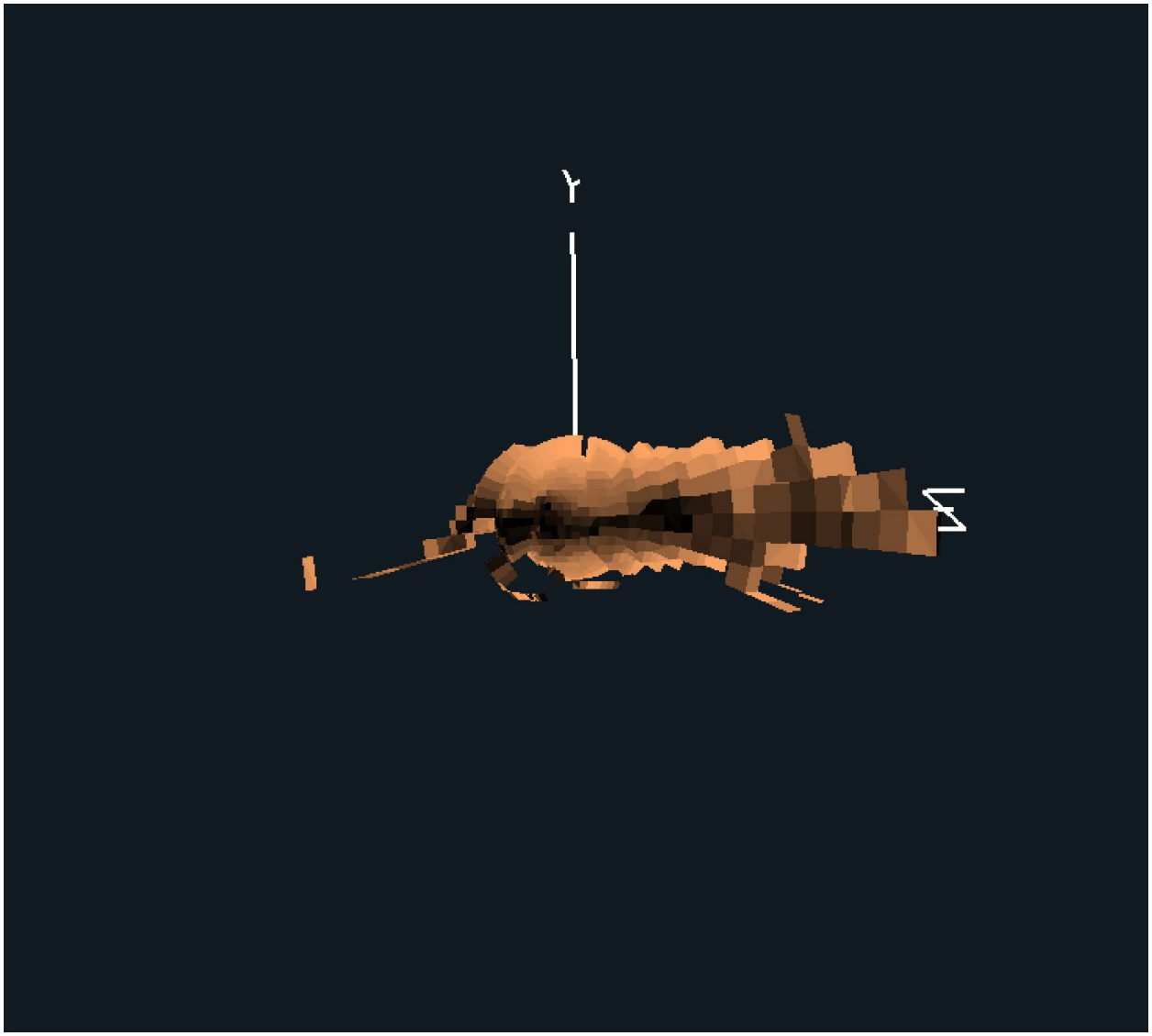
Title : Geometry Shaders
Video : https://media.oregonstate.edu/media/t/0_hcbkvzzt
Name : Si Thu Lin
ID : 933-957-884
Email : linsi@oregonstate.edu

Quantizing only in radius with the level 3

Quantizing only in radius with the level 0



Quantizing in radius, theta and phi with the level 3



Quantizing in radius, theta and phi with the level 0

```
void
ProduceVertex( float s, float t )
{
      const vec3 lightPos = vec3(0., 10., 0.);
     vec3 v = V0 + s*V01 + t*V02;

      vec3 n = N0 + s*N01 + t*N02;

      vec3 tnorm = normalize(gl_NormalMatrix * n);    //the
transform normal
      vec4 ECposition = gl_ModelViewMatrix * vec4(v,1.);
      gLightIntensity = abs( dot( normalize(lightPos -
ECposition.xyz), tnorm )  );



      if (uRadiusOnly == false){
float r = length( v );
float theta = atan( v.z, v.x );
float phi   = atan( v.y, length( v.xz ) );
v.xyz = QuantizeVec3(vec3(r,theta,phi));
v=RTOC(v);
v = (gl_ModelViewMatrix * vec4(v, 1.)).xyz;
      }


      if (uRadiusOnly == true){
float r = length( v );
float theta = atan( v.z, v.x );
float phi   = atan( v.y, length( v.xz ) );
r=Quantize(r);
v=RTOC(vec3(r,theta,phi));
```

```
float r = length( v );
float theta = atan( v.z, v.x );
float phi   = atan( v.y, length( v.xz ) );
r=Quantize(r);
v=RTOC(vec3(r,theta,phi));
v = (gl_ModelViewMatrix * vec4(v, 1.)).xyz;
      }
      gl_Position = gl_ProjectionMatrix * vec4 (v, 1.);
      EmitVertex();
}

void
main()
{
      V01 = (gl_PositionIn[1] - gl_PositionIn[0]).xyz;
      V02 = (gl_PositionIn[2] - gl_PositionIn[0]).xyz;
      V0  =  gl_PositionIn[0].xyz;

      N0  = vNormal[0];
      N01 = vNormal[1] - vNormal[0];
      N02 = vNormal[2] - vNormal[0];

      int numLayers = 1 << uLevel;

      float dt = 1. / float( numLayers );
      float t_top = 1.;

      for( int it = 0; it < numLayers; it++ )
      {
            float t_bot = t_top - dt;
            float smax_top = 1. - t_top;
            float smax_bot = 1. - t_bot;
```

```
for( int it = 0; it < numLayers; it++ )
{
	float t_bot = t_top - dt;
	float smax_top = 1. - t_top;
	float smax_bot = 1. - t_bot;

	int nums = it + 1;
	float ds_top = smax_top / float( nums - 1 );
	float ds_bot = smax_bot / float( nums );

	float s_top = 0.;
	float s_bot = 0.;

	for( int is = 0; is < nums; is++ )
	{
		ProduceVertex( s_bot, t_bot );
		ProduceVertex( s_top, t_top );
		s_top += ds_top;
		s_bot += ds_bot;
	}

	ProduceVertex( s_bot, t_bot );
	EndPrimitive();

	t_top = t_bot;
	t_bot -= dt;
}
```