Oregon State University

CS_557_X001_W2022 COMPUTER GRAPHICS SHADERS

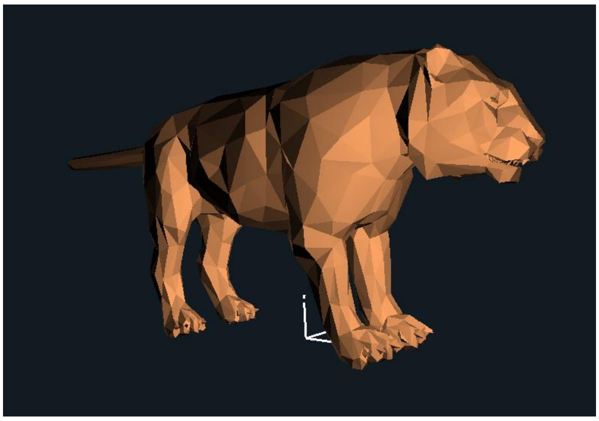
Project #7

Professor: Mike Bailey
Student: Chengxu Xu
(xucheng@oregonstate.edu)

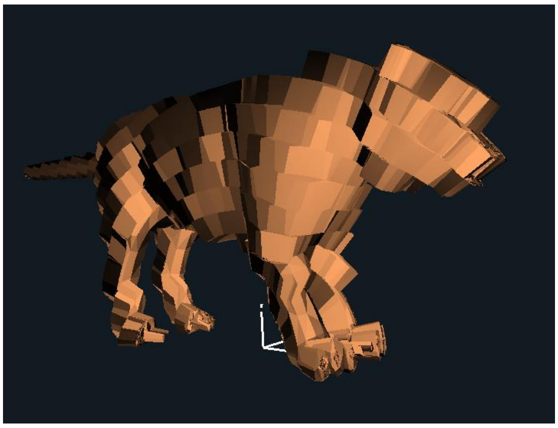
For this project, I defined uLevel and uQuantize to implement an obj model Quantize into a lego-like look. I also set the bool type uRadiusOnly to distinguish between the radius parameter or all parameters in the Quantize spherical coordinate.

Kaltura link: https://media.oregonstate.edu/media/t/1_5wcus331
Screen Shots：

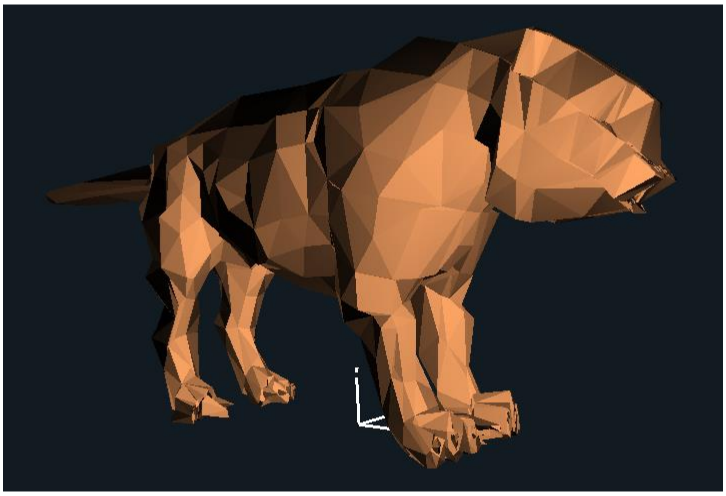
Original



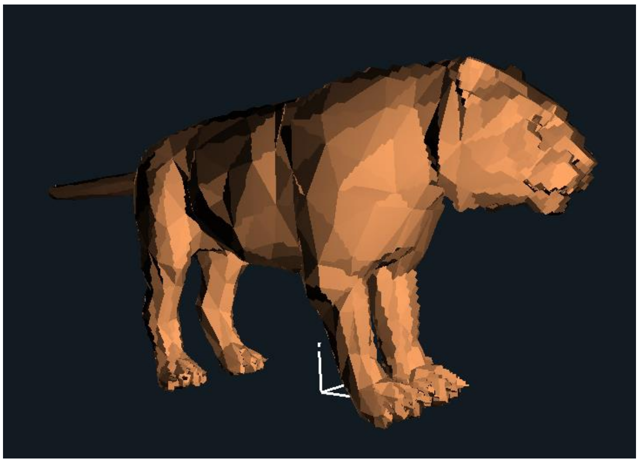
adjust uQuantize

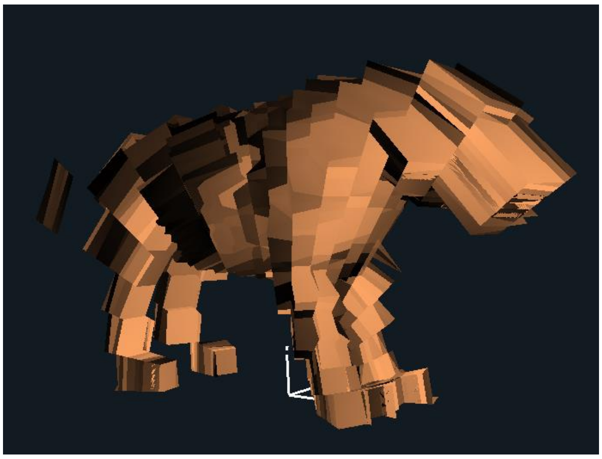


adjust uLevel



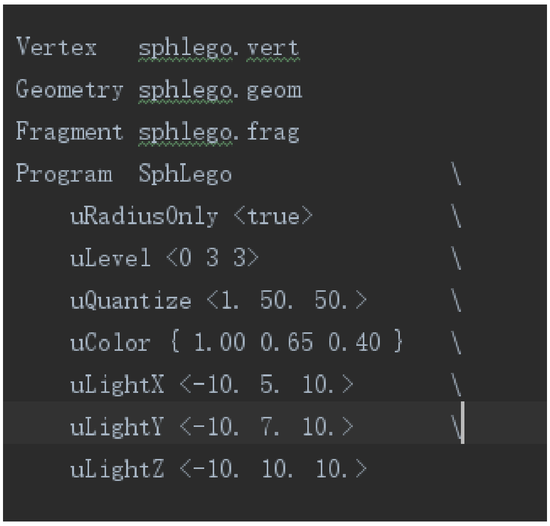
adjust uRadiusOnly





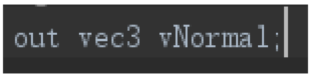
Key snippets:

Special parameter predefined：

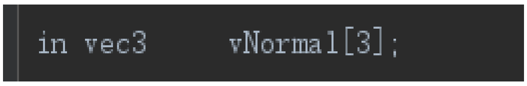


```
Vertex   sphlego.vert
Geometry sphlego.geom
Fragment sphlego.frag
Program  SphLego                       \
    uRadiusOnly <true>                 \
    uLevel <0 3 3>                     \
    uQuantize <1. 50. 50.>             \
    uColor { 1.00 0.65 0.40 }          \
    uLightX <-10. 5. 10.>              \
    uLightY <-10. 7. 10.>              \
    uLightZ <-10. 10. 10.>
```

Pass vNormal from the vertex shader to the geometry shader：

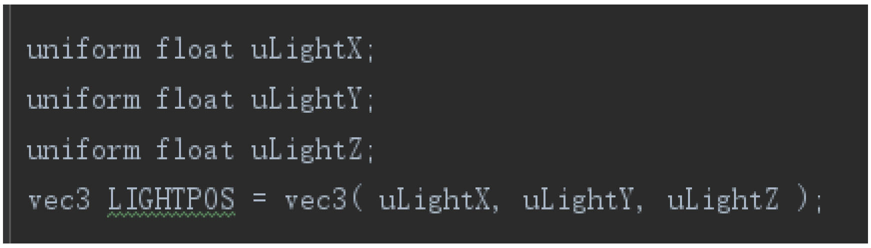


```
out vec3 vNormal;
```



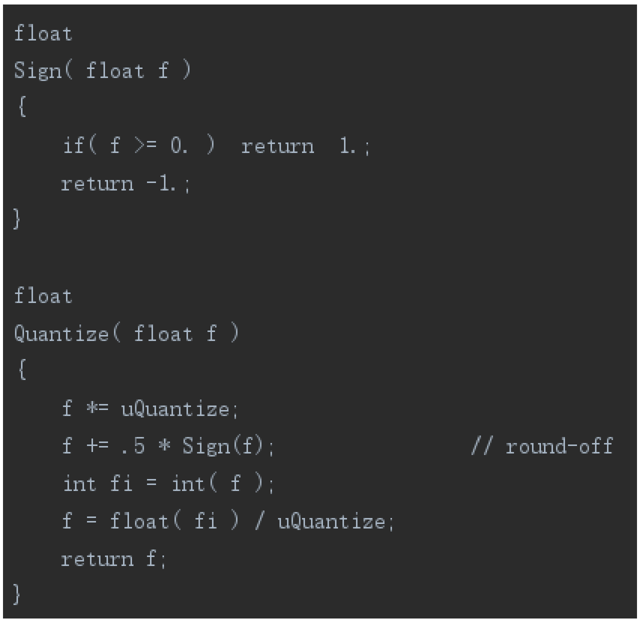
```
in vec3     vNormal[3];
```

Light position：



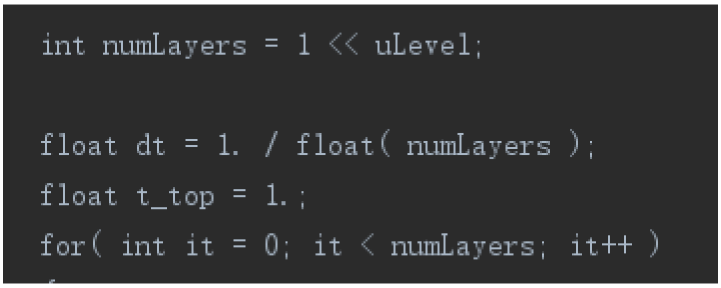
```
uniform float uLightX;
uniform float uLightY;
uniform float uLightZ;
vec3 LIGHTPOS = vec3( uLightX, uLightY, uLightZ );
```

Quantize function：



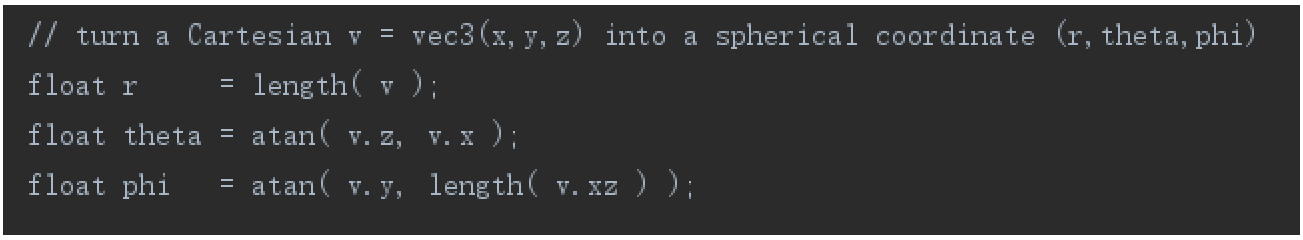
```
float
Sign( float f )
{
    if( f >= 0. )  return  1.;
    return -1.;
}

float
Quantize( float f )
{
    f *= uQuantize;
    f += .5 * Sign(f);              // round-off
    int fi = int( f );
    f = float( fi ) / uQuantize;
    return f;
}
```

subdivide triangle with uLevel:



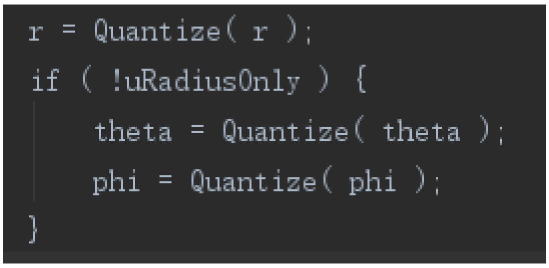
```
int numLayers = 1 << uLevel;

float dt = 1. / float( numLayers );
float t_top = 1.;
for( int it = 0; it < numLayers; it++ )
```

Turn a Cartesian v into a spherical coordinate:



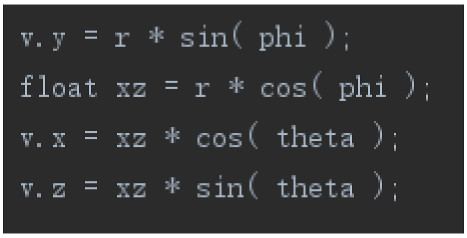
```
// turn a Cartesian v = vec3(x,y,z) into a spherical coordinate (r,theta,phi)
float r     = length( v );
float theta = atan( v.z, v.x );
float phi   = atan( v.y, length( v.xz ) );
```

Control change of entire spherical coordinate with uRadiusOnly:



```
r = Quantize( r );
if ( !uRadiusOnly ) {
    theta = Quantize( theta );
    phi = Quantize( phi );
}
```

Turn a spherical coordinate back to a Cartesian v:



```
v.y = r * sin( phi );
float xz = r * cos( phi );
v.x = xz * cos( theta );
v.z = xz * sin( theta );
```