RenderMan and OpenGL Shaders

CS557

Project # 7

Chao Zhang

1. Source listings

```
##OpenGL GLIB 8
Perspective 70
LookAt  2 -1 5  0 2 -1  0 0 1
Vertex          chaozhang.vert
TessControl     chaozhang.tcs
TessEvaluation chaozhang.tes
Geometry        chaozhang.geom
Fragment        chaozhang.frag
Program          MidTriangle \
                        uOuter01 <1 5 20> \
                        uOuter12 <1 5 20> \
                        uOuter20 <1 5 20> \
                        uInner <1 5 20> \
                        uZ01 <-5 2 10.> \
                        uZ12 <-5 2 10.> \
                        uZ20 <-5 2 10> \
                        uAdaptToZs <false> \
                        uShrink <0. 0.9 1.> \
                        uKa <0. 0.1 1.0> \
                        uKd <0. 0.7 1.0> \
                        uKs <0. 0.2 1.0> \
                        uShininess <3. 10. 1000.> \
                        uLightX <-10. 0. 10.>   uLightY <-10. 8. 10.>   uLightZ <-10. 8. 10.>\
Color 1. .5 0.
NumPatchVertices 3
glBegin gl_patches
        glVertex 0. 0. 0.
        glVertex 2. 0. 0.
        glVertex 0. 2. 0.
glEnd
```

chaozhang.glib

This file includes all the definition I need to use. The uOuter01, uOuter12 and uOuter12uOuter20 will change the number of the edges in the side, and the default is 5. With increase of those numbers, the side will look more smooth. The inner is the control value of the inside edges. uZ01, uZ12 and uZ20 are the control point's value on the Z coordinate of each side, so it can be negative. The shrink will change the size of the tessellation.

glBegin gl_patches

glVertex 0. 0. 0.

glVertex 2. 0. 0.

glVertex 0. 2. 0.

glEnd

those are the three corner vertices.

```
#version 330 compatibility


out vec3  vMCposition;
out vec4  vColor;
out float vLightIntensity;
out vec2  vST;
out float z;

const vec3 LIGHTPOS = vec3( -2., 0., 10. );

void
main( )
{
gl_Position = gl_Vertex;
}
```

chaozhang.vert

only the gl_position included.

```
#version 400 compatibility
in vec3 gNs;
in vec3 gLs;
in vec3 gEs;
uniform float uKa, uKd, uKs;
uniform float uShininess;


void
main( )
{

        vec3 Normal;
        vec3 Light;
        vec3 Eye;

        Normal = normalize(gNs);
        Light = normalize(gLs);
        Eye = normalize(gEs);

        vec4 ambient = uKa * vec4 (1., 0.5, 0., 1.);

        float d = max( dot(Normal, Light), 0.);
        vec4 diffuse = uKd * d * vec4 (1., 0.5, 0., 1.);

        float s = 0.;
        if(dot(Normal, Light) > 0.)
        {
                vec3 ref = normalize(2. * Normal * dot(Normal, Light) - Light);
                s = pow(max(dot(Eye, ref), 0.), uShininess);
        }


        vec4 specular = uKs * s * vec4(1., 1., 1., 1.);

        gl_FragColor = vec4 (ambient.rgb + diffuse.rgb + specular.rgb, 1.);
}
```

chaozhang.freq

This is the .freq file. This fragment shader is used to compute the normal and use the lighting to show it is right.

```
#version 400 compatibility
#extension GL_EXT_gpu_shader4: enable
#extension GL_EXT_geometry_shader4: enable

layout( triangles ) in;
layout( triangle_strip, max_vertices=32 ) out;
uniform float uShrink;
uniform float uLightX, uLightY, uLightZ;
in vec3 teECposition[3];
in vec3 teNormal[3];
out vec3 gNs;
out vec3 gLs;
out vec3 gEs;
vec3 LightPos = vec3( uLightX, uLightY, uLightZ );
vec3 V[3];
vec3 CG;

void
ProduceVertex( int v )
{
        gNs = teNormal[v];
        gLs = LightPos - teECposition[v];
        gEs = vec3(0.,0.,0.) - teECposition[v];
        gl_Position = gl_ProjectionMatrix * vec4( CG + uShrink * ( V[v] - CG ), 1. );
        EmitVertex( );
}

void
main( )
{
        V[0] = gl_PositionIn[0].xyz;
        V[1] = gl_PositionIn[1].xyz;
        V[2] = gl_PositionIn[2].xyz;
        CG = ( V[0] + V[1] + V[2] ) / 3.;
        ProduceVertex( 0 );
        ProduceVertex( 1 );
        ProduceVertex( 2 );
}
```

chaozhang.goem

This is a new shader I used first time. This shader will control the geometry.

void

ProduceVertex(int v)

{

gNs = teNormal[v];

gLs = LightPos - teECposition[v];

gEs = vec3(0.,0.,0.) - teECposition[v];

gl_Position = gl_ProjectionMatrix * vec4(CG + uShrink * (V[v] - CG), 1.);

EmitVertex( );

}

Those lines will produce the vertices, that is how the ushrink works. The geometry shader is kind of more powerful vertex shader.

V[0] = gl_PositionIn[0].xyz;

V[1] = gl_PositionIn[1].xyz;

V[2] = gl_PositionIn[2].xyz;

CG = (V[0] + V[1] + V[2]) / 3.;

ProduceVertex(0);

ProduceVertex(1);

ProduceVertex(2); Those will finish the produce.

```
#version 400 compatibility
#extension GL_ARB_tessellation_shader : enable

uniform float uZ01, uZ12, uZ20;
uniform int uOuter01, uOuter12, uOuter20, uInner;
uniform bool uAdaptToZs;
layout( vertices = 3 ) out;

void
main( )
{
	gl_out[ gl_InvocationID ].gl_Position = gl_in[ gl_InvocationID ].gl_Position;
	if( uAdaptToZs )
	{
		gl_TessLevelOuter[0] = float( uOuter12 ) + uZ12;
		gl_TessLevelOuter[1] = float( uOuter20 ) + uZ20;
		gl_TessLevelOuter[2] = float( uOuter01 ) + uZ01;
		gl_TessLevelInner[0] = gl_TessLevelInner[1] = float(uInner) + (uZ12 + uZ20 + uZ01);
	}
	else
	{
		gl_TessLevelOuter[0] = float(uOuter12);
		gl_TessLevelOuter[1] = float(uOuter20);
		gl_TessLevelOuter[2] = float(uOuter01);
		gl_TessLevelInner[0] = gl_TessLevelInner[1] = float(uInner);
	}
}
```

chaozhang.tcs

This is the tessellation control shader. In this shader I can define the control point to the tessellation. If the uAdaptToZs is true, the outer will add the uZ value and the same to the inner. Else, the outer will be the value of the uOuter.

```
#version 400 compatibility
#extension GL_ARB_tessellation_shader : enable

layout( triangles, equal_spacing, ccw) in;
uniform float uZ01, uZ12, uZ20;
out vec3 teNormal;
out vec3 teECposition;

void
main( )
{
        vec4 p0 = gl_in[0].gl_Position;
        vec4 p1 = gl_in[1].gl_Position;
        vec4 p2 = gl_in[2].gl_Position;
        vec4 p3 = gl_in[3].gl_Position;


        vec4 p01 = vec4 ((p0.x + p1.x)/2,(p0.y + p1.y)/2, uZ01, 1.);
        vec4 p12 = vec4 ((p1.x + p2.x)/2,(p1.y + p2.y)/2, uZ12, 1.);
        vec4 p20 = vec4 ((p2.x + p0.x)/2,(p2.y + p0.y)/2, uZ20, 1.);



        float u = gl_TessCoord.x;
        float v = gl_TessCoord.y;
        float w = gl_TessCoord.z;
        float b0 = u * u;
        float b1 = v * v;
       float b2 = w * w;
        float b01 = 2 * u * v;
        float b12 = 2 * v * w;
        float b20 = 2 * w * u;
        float db0du = 2.*u;
        float db0dv = 0.;
        float db1du = 0.;
        float db1dv = 2.*v;
        float db2du = -2.*(1.-u-v);
        float db2dv = -2.*(1.-u-v);
        float db01du = 2.*v;
        float db01dv = 2.*u;
        float db12du = -2.*v;
        float db12dv = 2.*(1.-u-2.*v);
        float db20du = 2.*(1.-2.*u-v);
        float db20dv = -2.*u;
        teECposition = ( gl_ModelViewMatrix * ( b0*p0 + b01*p01 + b1*p1 + b12*p12 + b2*p2 + b20*p20 )
).xyz;
        gl_Position = vec4( teECposition, 1. );
        vec4 dpdu = db0du*p0 + db01du*p01 + db1du*p1 + db12du*p12 + db2du*p2 + db20du*p20;
        vec4 dpdv = db0dv*p0 + db01dv*p01 + db1dv*p1 + db12dv*p12 + db2dv*p2 + db20dv*p20;
        teNormal = gl_NormalMatrix * normalize( cross( dpdu.xyz, dpdv.xyz ) );
}^M
```

chaozhang.tes

This is the tessellation evaluation shader. In this shader, I will computing all the vertices position.

vec4 p0 = gl_in[0].gl_Position;

vec4 p1 = gl_in[1].gl_Position;

vec4 p2 = gl_in[2].gl_Position;

vec4 p3 = gl_in[3].gl_Position; Those are the gl_positions.

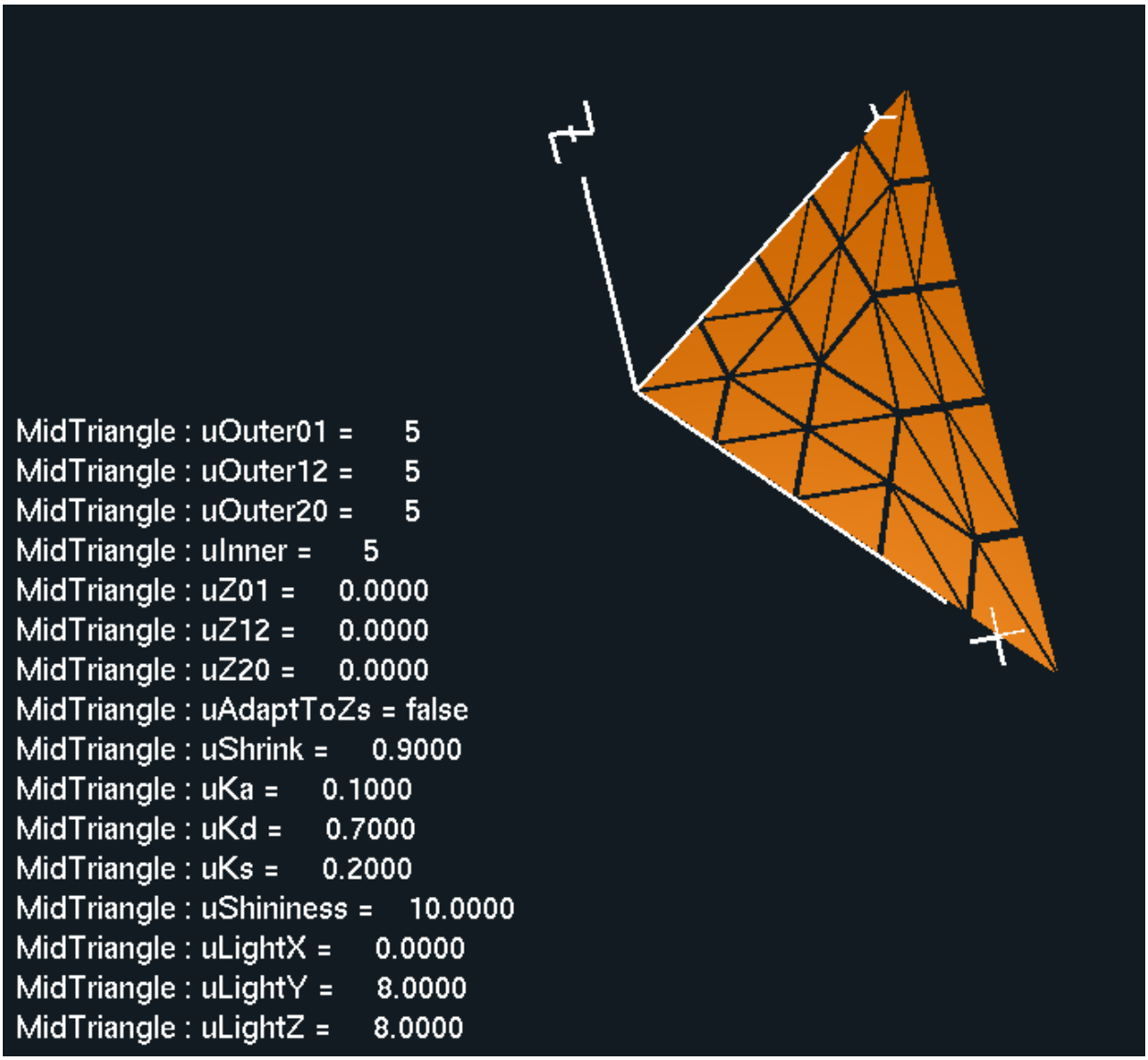
teECposition = (gl_ModelViewMatrix * (b0*p0 + b01*p01 + b1*p1 + b12*p12 + b2*p2 + b20*p20)).xyz;

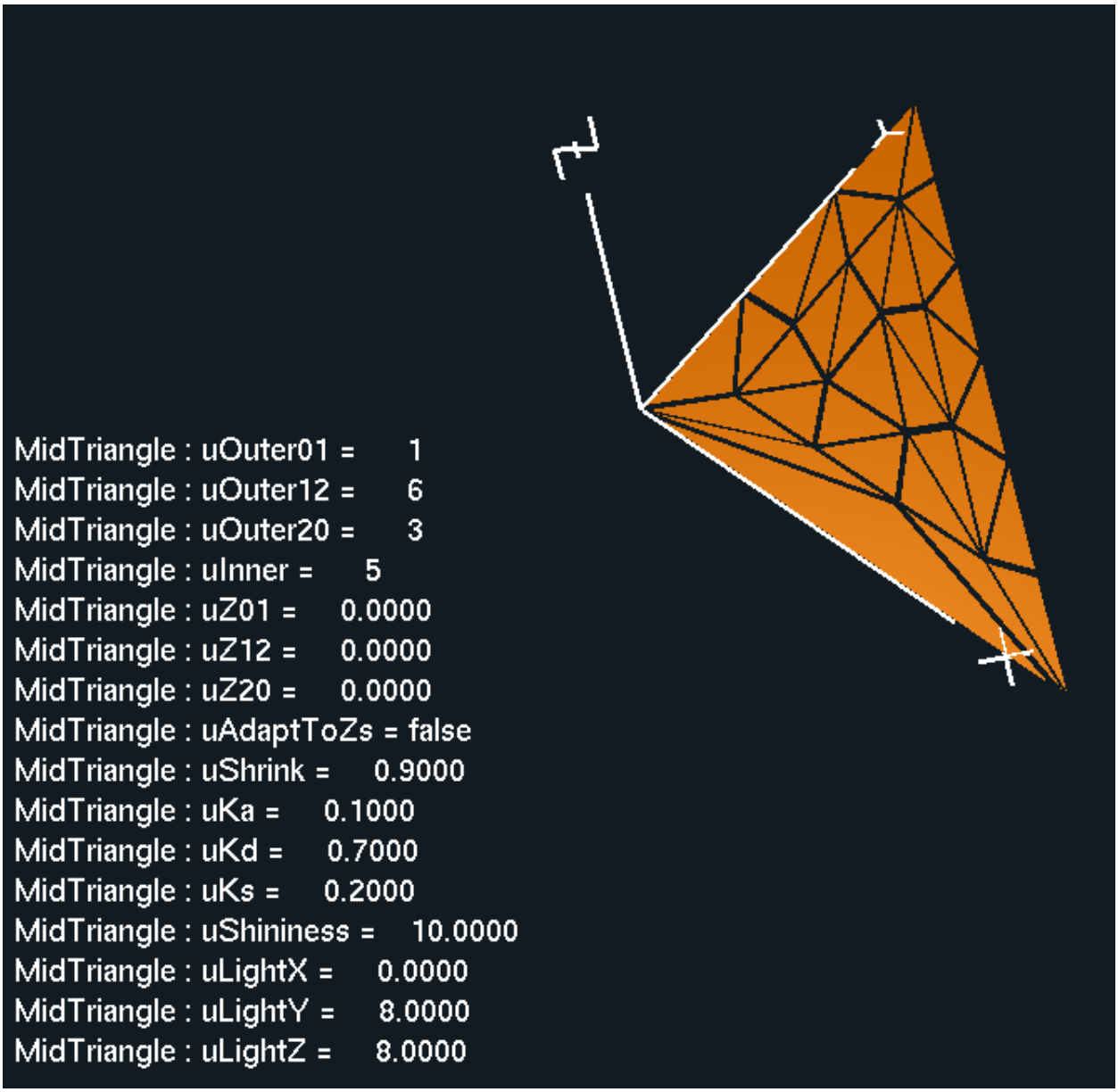
gl_Position = vec4(teECposition, 1.);

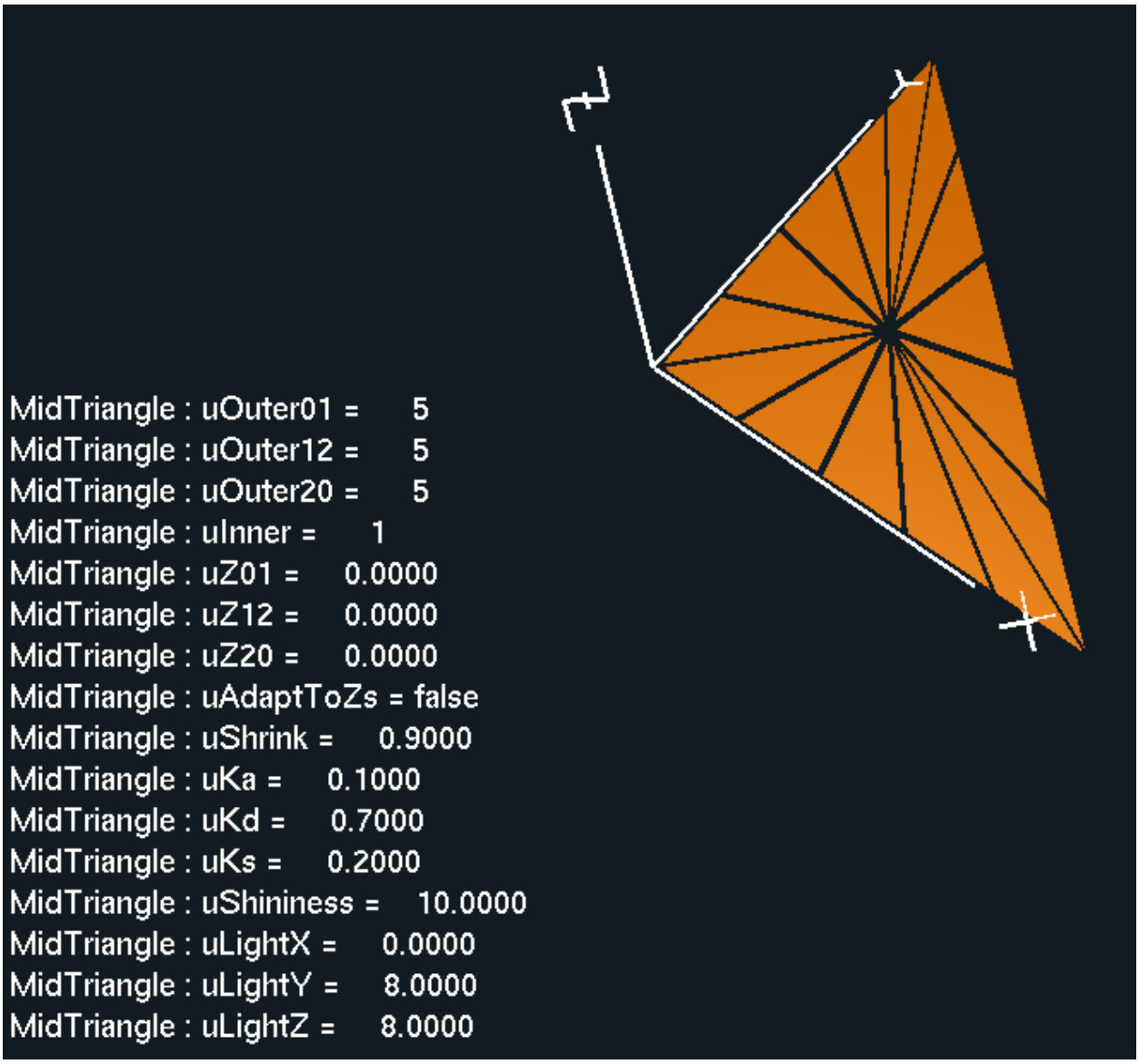
```
    vec4 dpdu = db0du*p0 + db01du*p01 + db1du*p1 + db12du*p12 + db2du*p2 + db20du*p20;
    vec4 dpdv = db0dv*p0 + db01dv*p01 + db1dv*p1 + db12dv*p12 + db2dv*p2 + db20dv*p20;
    teNormal = gl_NormalMatrix * normalize( cross( dpdu.xyz, dpdv.xyz ) );
}
```

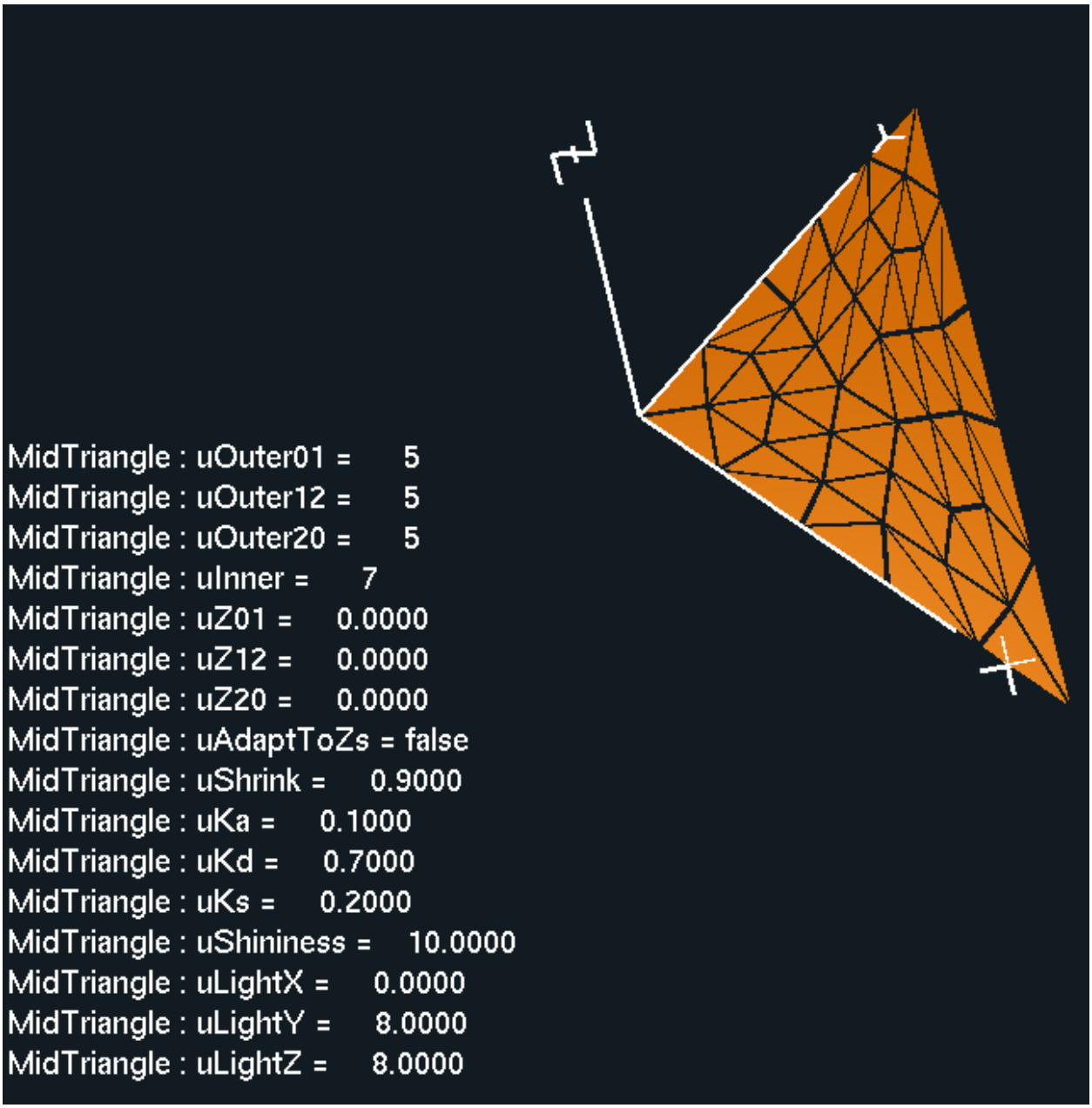
Those are use the position are computed in the previous equation to get the teECposition and teNormal. With those the whole project will work correctly.

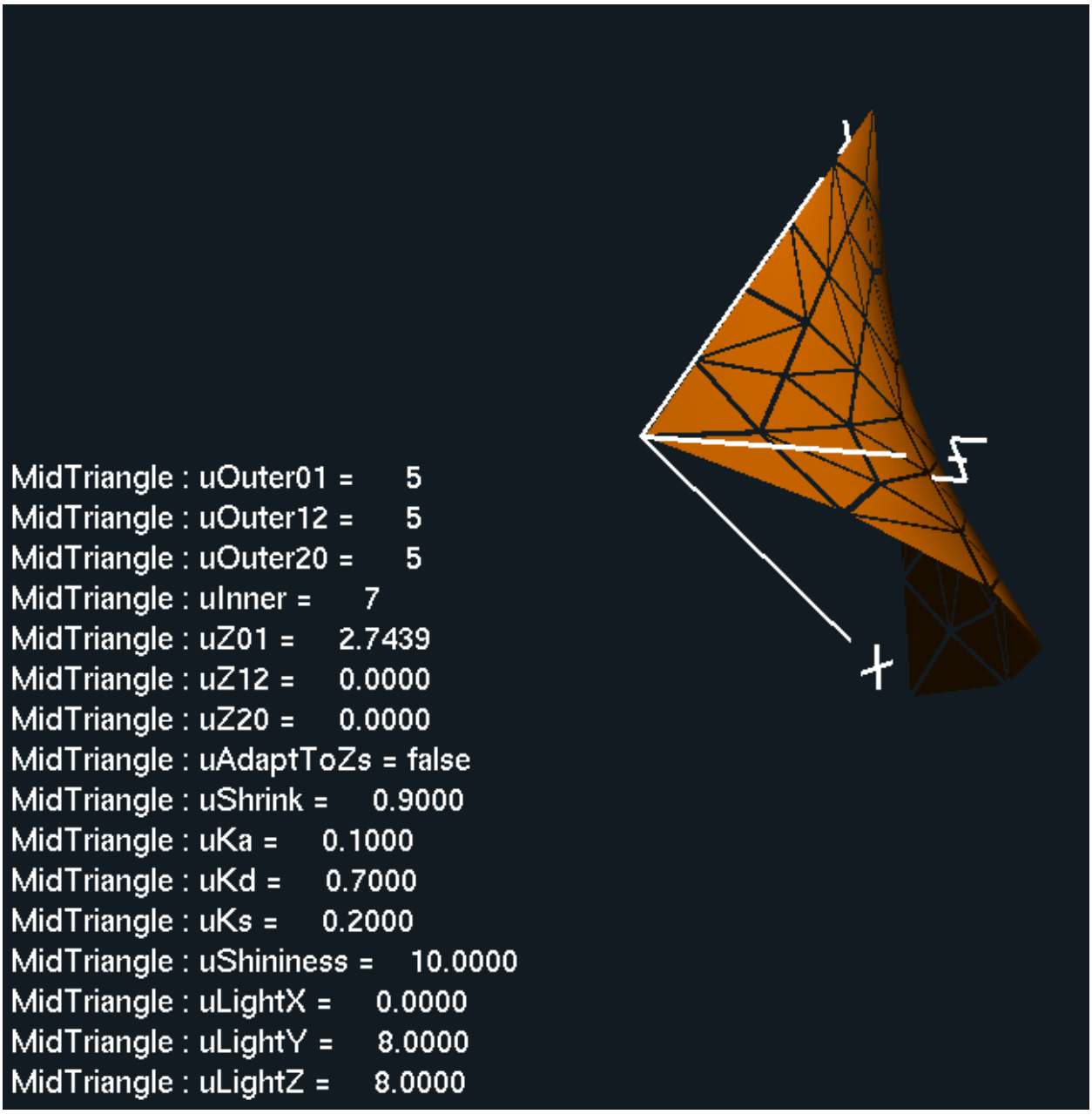
2. Results

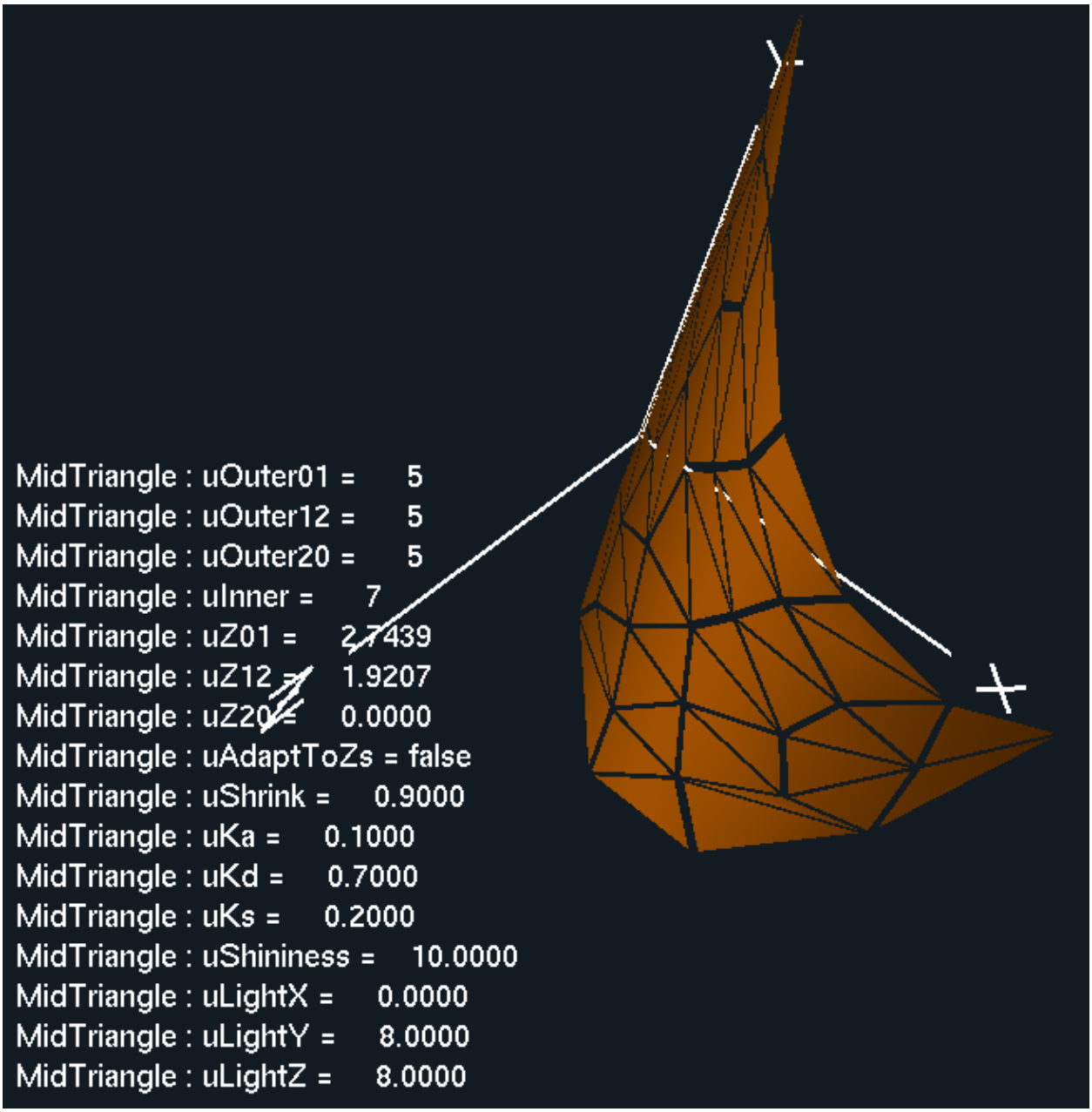


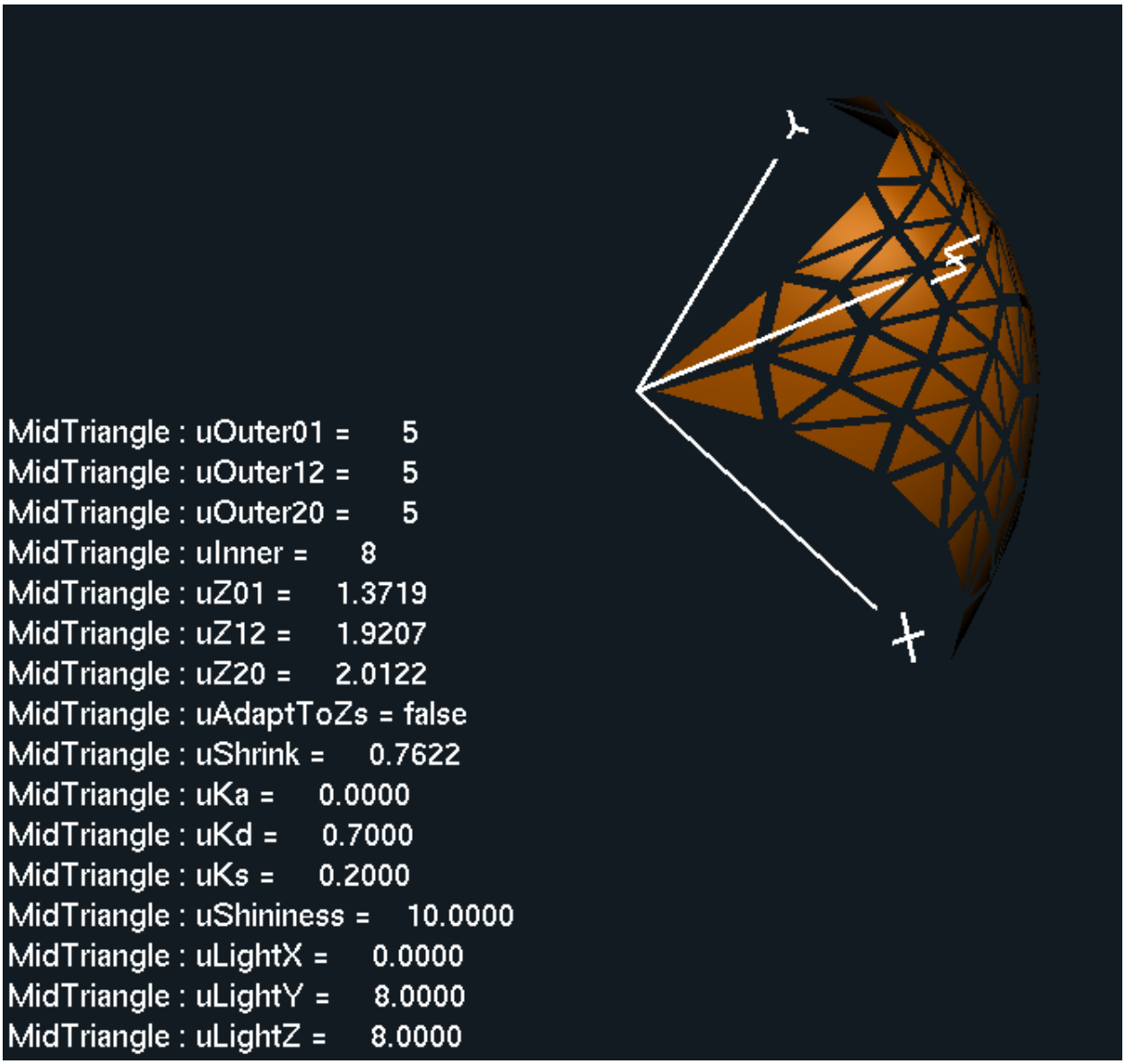

MidTriangle : uOuter01 = 1
MidTriangle : uOuter12 = 6
MidTriangle : uOuter20 = 3
MidTriangle : uInner = 5
MidTriangle : uZ01 = 0.0000
MidTriangle : uZ12 = 0.0000
MidTriangle : uZ20 = 0.0000
MidTriangle : uAdaptToZs = false
MidTriangle : uShrink = 0.9000
MidTriangle : uKa = 0.1000
MidTriangle : uKd = 0.7000
MidTriangle : uKs = 0.2000
MidTriangle : uShininess = 10.0000
MidTriangle : uLightX = 0.0000
MidTriangle : uLightY = 8.0000
MidTriangle : uLightZ = 8.0000

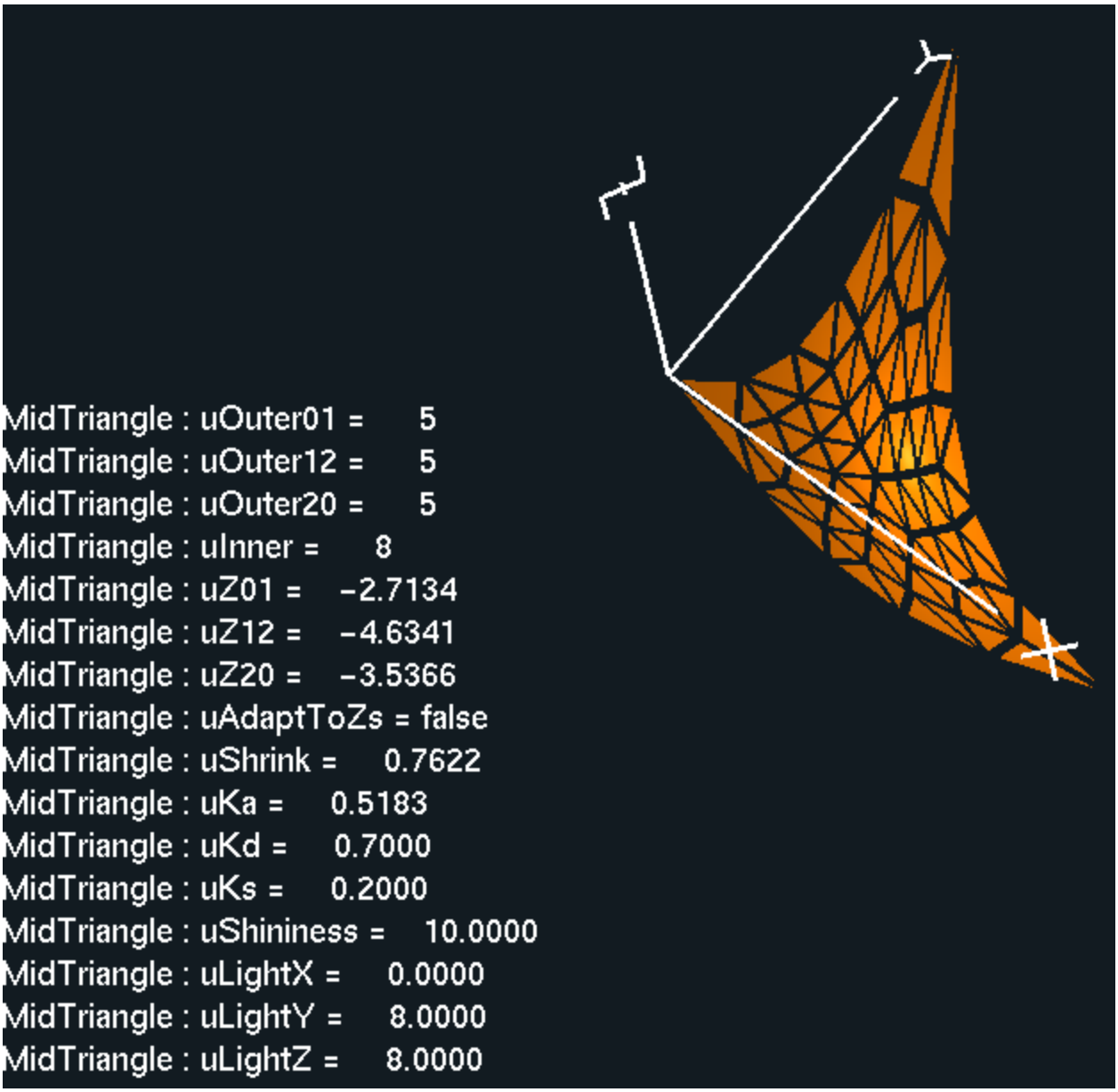

Z
Y
X
MidTriangle : uOuter01 = 5
MidTriangle : uOuter12 = 5
MidTriangle : uOuter20 = 5
MidTriangle : uInner = 1
MidTriangle : uZ01 = 0.0000
MidTriangle : uZ12 = 0.0000
MidTriangle : uZ20 = 0.0000
MidTriangle : uAdaptToZs = false
MidTriangle : uShrink = 0.9000
MidTriangle : uKa = 0.1000
MidTriangle : uKd = 0.7000
MidTriangle : uKs = 0.2000
MidTriangle : uShininess = 10.0000
MidTriangle : uLightX = 0.0000
MidTriangle : uLightY = 8.0000
MidTriangle : uLightZ = 8.0000

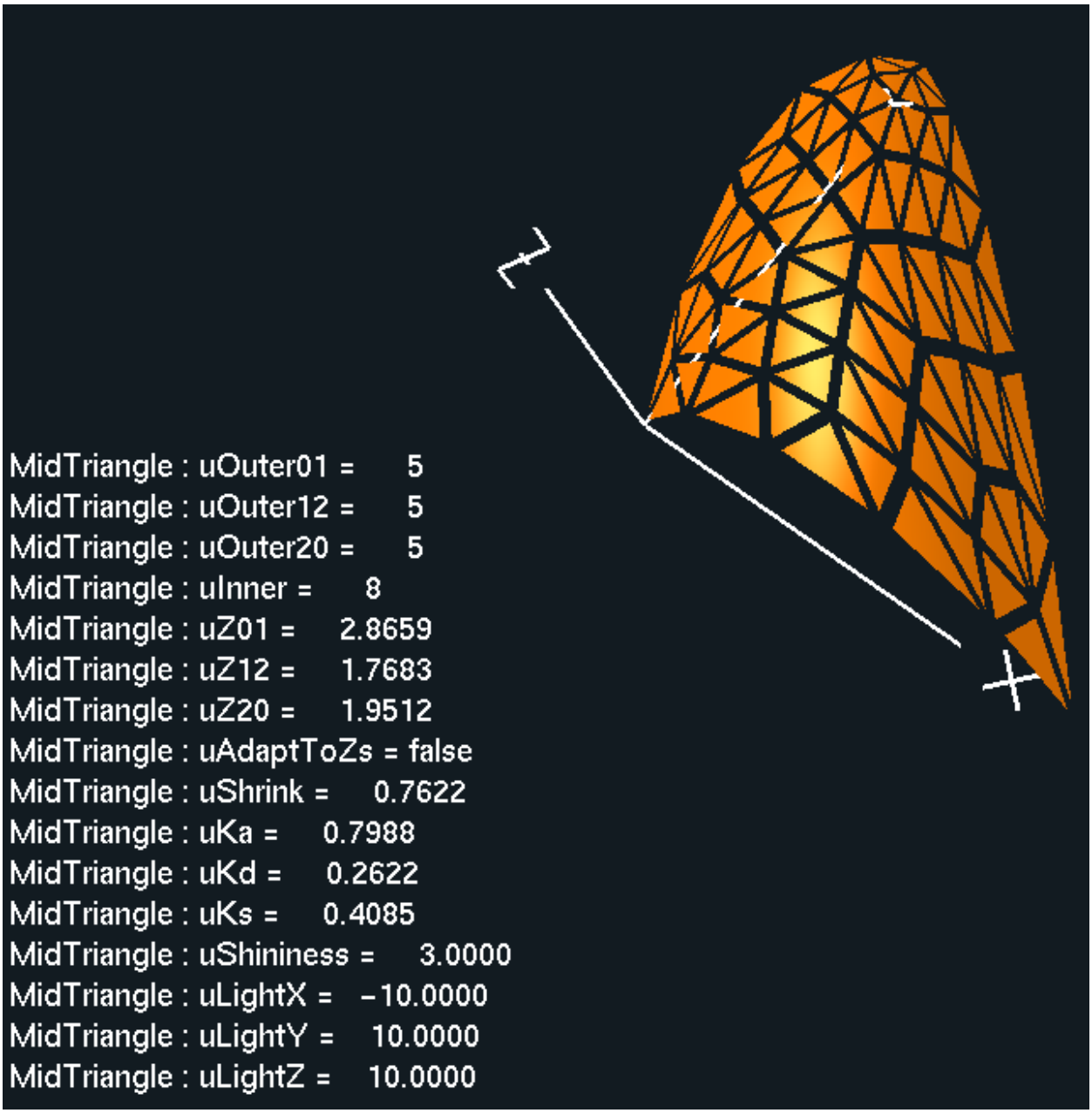

Z
Y
X
MidTriangle : uOuter01 = 5
MidTriangle : uOuter12 = 5
MidTriangle : uOuter20 = 5
MidTriangle : uInner = 7
MidTriangle : uZ01 = 0.0000
MidTriangle : uZ12 = 0.0000
MidTriangle : uZ20 = 0.0000
MidTriangle : uAdaptToZs = false
MidTriangle : uShrink = 0.9000
MidTriangle : uKa = 0.1000
MidTriangle : uKd = 0.7000
MidTriangle : uKs = 0.2000
MidTriangle : uShininess = 10.0000
MidTriangle : uLightX = 0.0000
MidTriangle : uLightY = 8.0000
MidTriangle : uLightZ = 8.0000

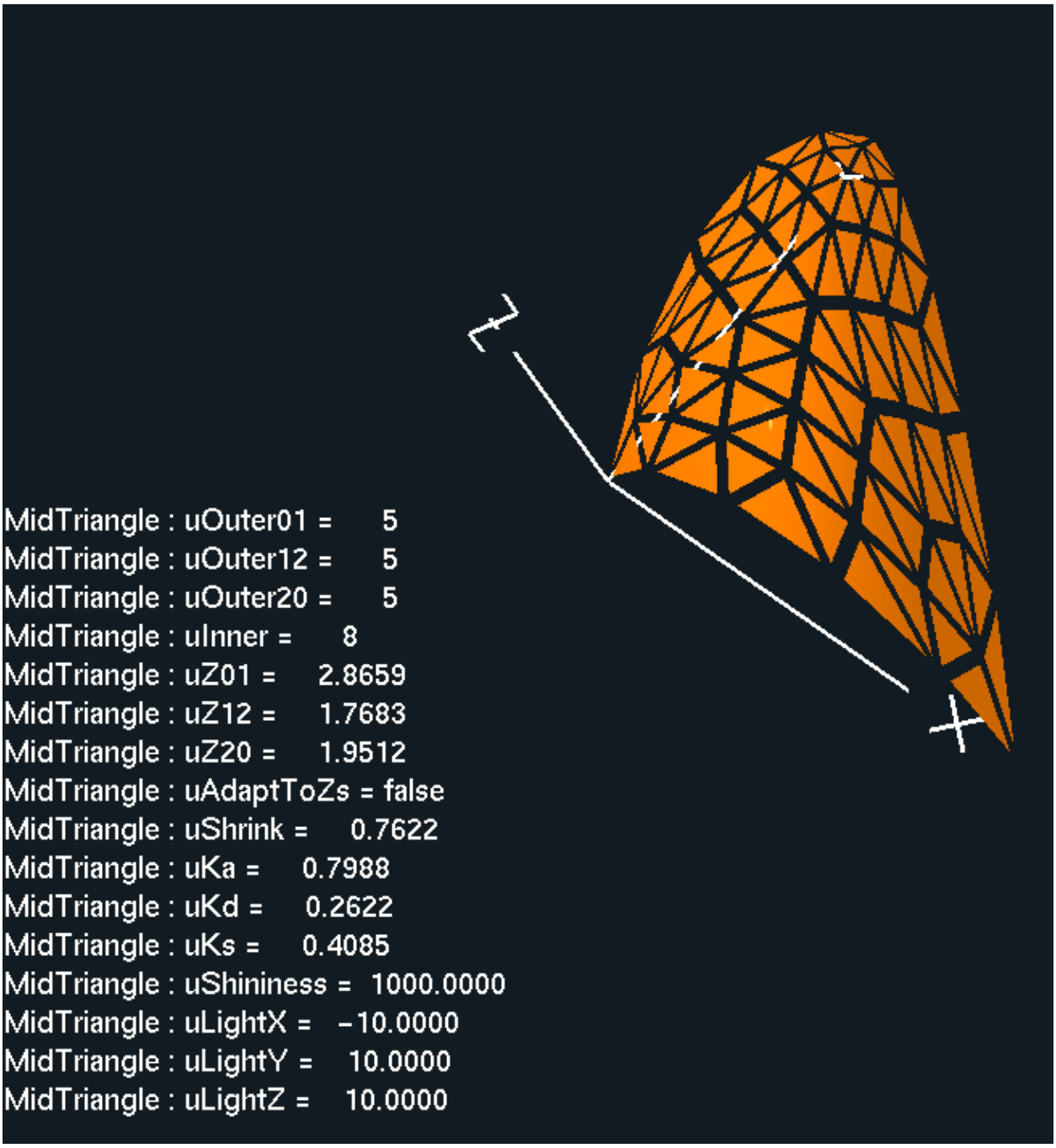

MidTriangle : uOuter01 = 5
MidTriangle : uOuter12 = 5
MidTriangle : uOuter20 = 5
MidTriangle : uInner = 7
MidTriangle : uZ01 = 2.7439
MidTriangle : uZ12 = 0.0000
MidTriangle : uZ20 = 0.0000
MidTriangle : uAdaptToZs = false
MidTriangle : uShrink = 0.9000
MidTriangle : uKa = 0.1000
MidTriangle : uKd = 0.7000
MidTriangle : uKs = 0.2000
MidTriangle : uShininess = 10.0000
MidTriangle : uLightX = 0.0000
MidTriangle : uLightY = 8.0000
MidTriangle : uLightZ = 8.0000


MidTriangle : uOuter01 = 5
MidTriangle : uOuter12 = 5
MidTriangle : uOuter20 = 5
MidTriangle : uInner = 7
MidTriangle : uZ01 = 2.7439
MidTriangle : uZ12 = 1.9207
MidTriangle : uZ20 = 0.0000
MidTriangle : uAdaptToZs = false
MidTriangle : uShrink = 0.9000
MidTriangle : uKa = 0.1000
MidTriangle : uKd = 0.7000
MidTriangle : uKs = 0.2000
MidTriangle : uShininess = 10.0000
MidTriangle : uLightX = 0.0000
MidTriangle : uLightY = 8.0000
MidTriangle : uLightZ = 8.0000

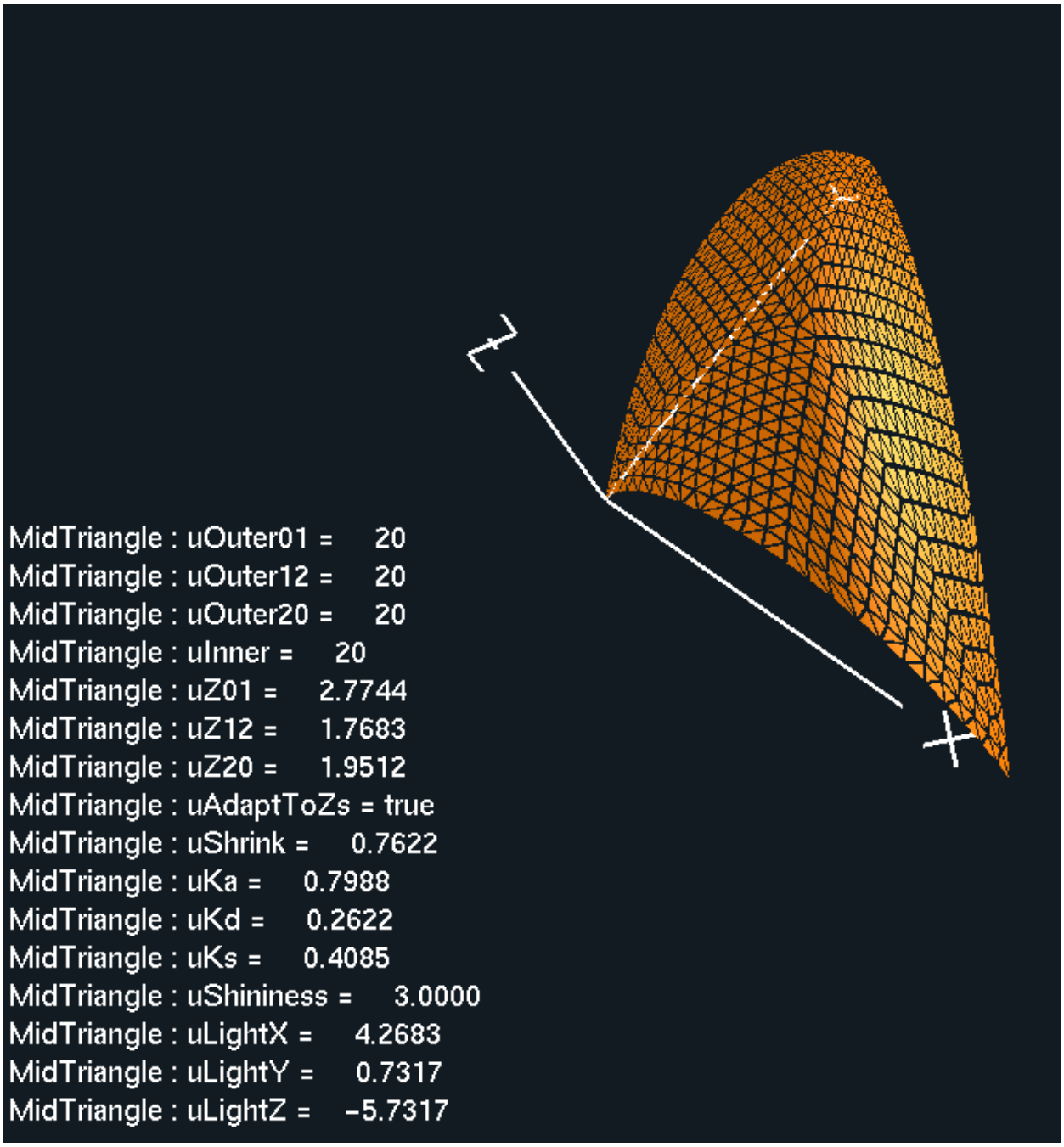

Y
S
T
MidTriangle : uOuter01 = 5
MidTriangle : uOuter12 = 5
MidTriangle : uOuter20 = 5
MidTriangle : uInner = 8
MidTriangle : uZ01 = 1.3719
MidTriangle : uZ12 = 1.9207
MidTriangle : uZ20 = 2.0122
MidTriangle : uAdaptToZs = false
MidTriangle : uShrink = 0.7622
MidTriangle : uKa = 0.0000
MidTriangle : uKd = 0.7000
MidTriangle : uKs = 0.2000
MidTriangle : uShininess = 10.0000
MidTriangle : uLightX = 0.0000
MidTriangle : uLightY = 8.0000
MidTriangle : uLightZ = 8.0000

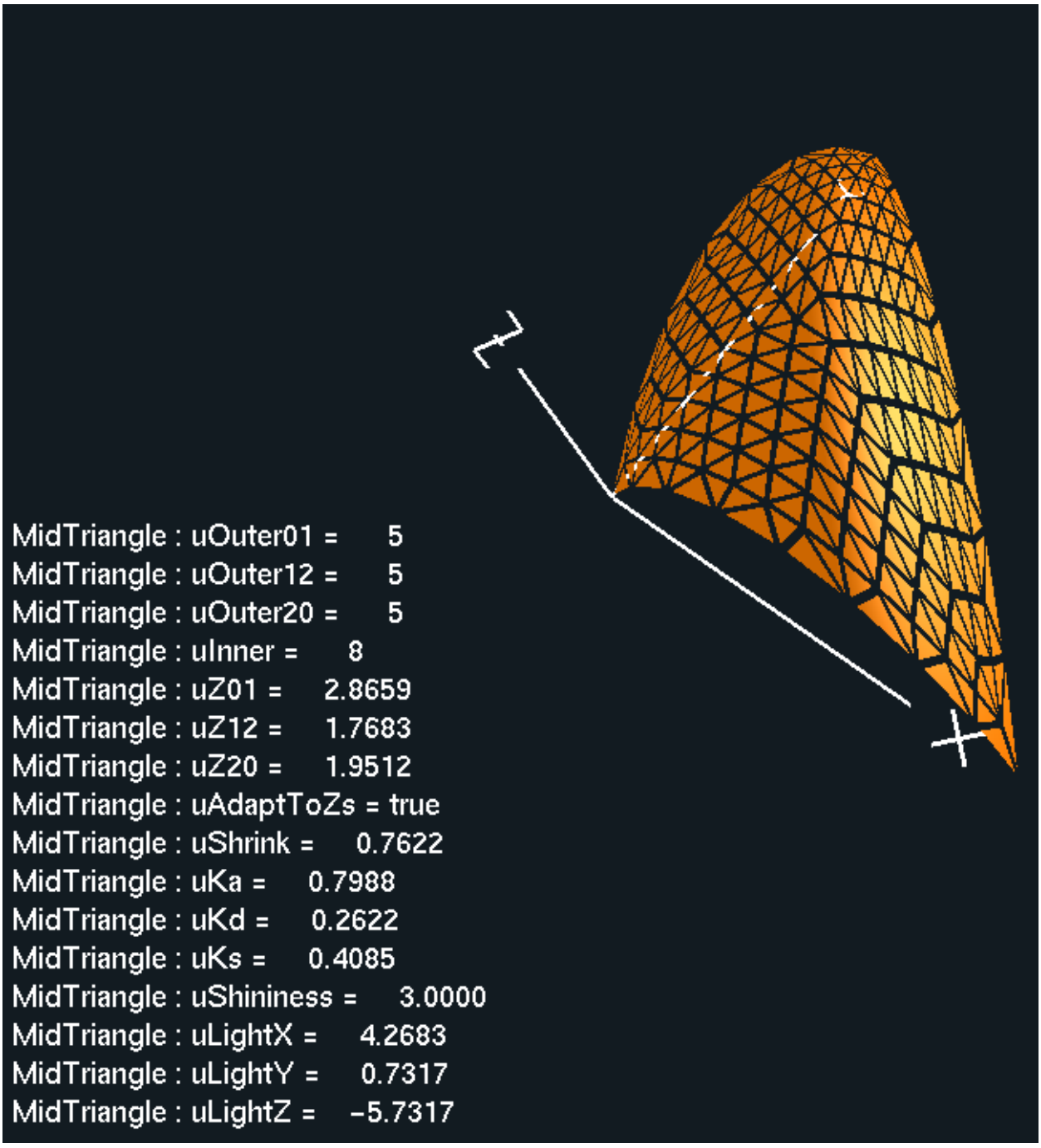

Y
Z
X
MidTriangle : uOuter01 = 5
MidTriangle : uOuter12 = 5
MidTriangle : uOuter20 = 5
MidTriangle : uInner = 8
MidTriangle : uZ01 = -2.7134
MidTriangle : uZ12 = -4.6341
MidTriangle : uZ20 = -3.5366
MidTriangle : uAdaptToZs = false
MidTriangle : uShrink = 0.7622
MidTriangle : uKa = 0.5183
MidTriangle : uKd = 0.7000
MidTriangle : uKs = 0.2000
MidTriangle : uShininess = 10.0000
MidTriangle : uLightX = 0.0000
MidTriangle : uLightY = 8.0000
MidTriangle : uLightZ = 8.0000


MidTriangle : uOuter01 = 5
MidTriangle : uOuter12 = 5
MidTriangle : uOuter20 = 5
MidTriangle : uInner = 8
MidTriangle : uZ01 = 2.8659
MidTriangle : uZ12 = 1.7683
MidTriangle : uZ20 = 1.9512
MidTriangle : uAdaptToZs = false
MidTriangle : uShrink = 0.7622
MidTriangle : uKa = 0.7988
MidTriangle : uKd = 0.2622
MidTriangle : uKs = 0.4085
MidTriangle : uShininess = 3.0000
MidTriangle : uLightX = -10.0000
MidTriangle : uLightY = 10.0000
MidTriangle : uLightZ = 10.0000


MidTriangle : uOuter01 = 5
MidTriangle : uOuter12 = 5
MidTriangle : uOuter20 = 5
MidTriangle : uInner = 8
MidTriangle : uZ01 = 2.8659
MidTriangle : uZ12 = 1.7683
MidTriangle : uZ20 = 1.9512
MidTriangle : uAdaptToZs = false
MidTriangle : uShrink = 0.7622
MidTriangle : uKa = 0.7988
MidTriangle : uKd = 0.2622
MidTriangle : uKs = 0.4085
MidTriangle : uShininess = 1000.0000
MidTriangle : uLightX = -10.0000
MidTriangle : uLightY = 10.0000
MidTriangle : uLightZ = 10.0000


MidTriangle : uOuter01 = 5
MidTriangle : uOuter12 = 5
MidTriangle : uOuter20 = 5
MidTriangle : uInner = 8
MidTriangle : uZ01 = 2.8659
MidTriangle : uZ12 = 1.7683
MidTriangle : uZ20 = 1.9512
MidTriangle : uAdaptToZs = false
MidTriangle : uShrink = 0.7622
MidTriangle : uKa = 0.7988
MidTriangle : uKd = 0.2622
MidTriangle : uKs = 0.4085
MidTriangle : uShininess = 3.0000
MidTriangle : uLightX = 4.2683
MidTriangle : uLightY = 0.7317
MidTriangle : uLightZ = -5.7317


MidTriangle : uOuter01 = 20
MidTriangle : uOuter12 = 20
MidTriangle : uOuter20 = 20
MidTriangle : uInner = 20
MidTriangle : uZ01 = 2.7744
MidTriangle : uZ12 = 1.7683
MidTriangle : uZ20 = 1.9512
MidTriangle : uAdaptToZs = true
MidTriangle : uShrink = 0.7622
MidTriangle : uKa = 0.7988
MidTriangle : uKd = 0.2622
MidTriangle : uKs = 0.4085
MidTriangle : uShininess = 3.0000
MidTriangle : uLightX = 4.2683
MidTriangle : uLightY = 0.7317
MidTriangle : uLightZ = -5.7317


MidTriangle : uOuter01 = 5
MidTriangle : uOuter12 = 5
MidTriangle : uOuter20 = 5
MidTriangle : uInner = 8
MidTriangle : uZ01 = 2.8659
MidTriangle : uZ12 = 1.7683
MidTriangle : uZ20 = 1.9512
MidTriangle : uAdaptToZs = true
MidTriangle : uShrink = 0.7622
MidTriangle : uKa = 0.7988
MidTriangle : uKd = 0.2622
MidTriangle : uKs = 0.4085
MidTriangle : uShininess = 3.0000
MidTriangle : uLightX = 4.2683
MidTriangle : uLightY = 0.7317
MidTriangle : uLightZ = -5.7317