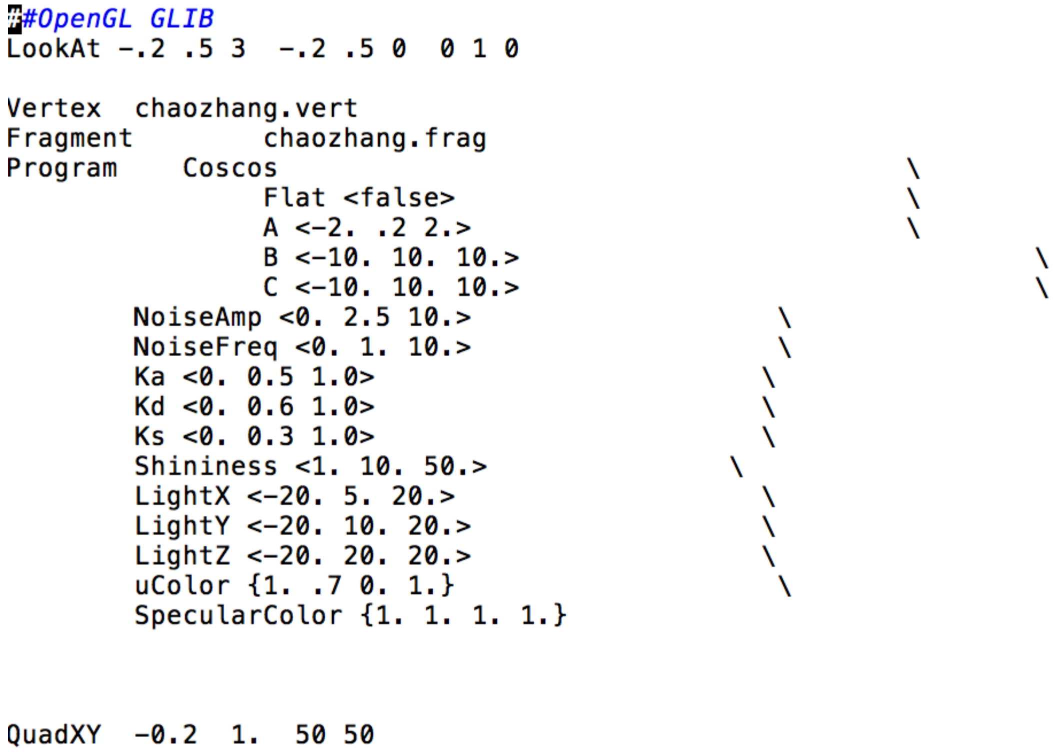
RenderMan and OpenGL Shaders

CS557

Project # 4

Chao Zhang

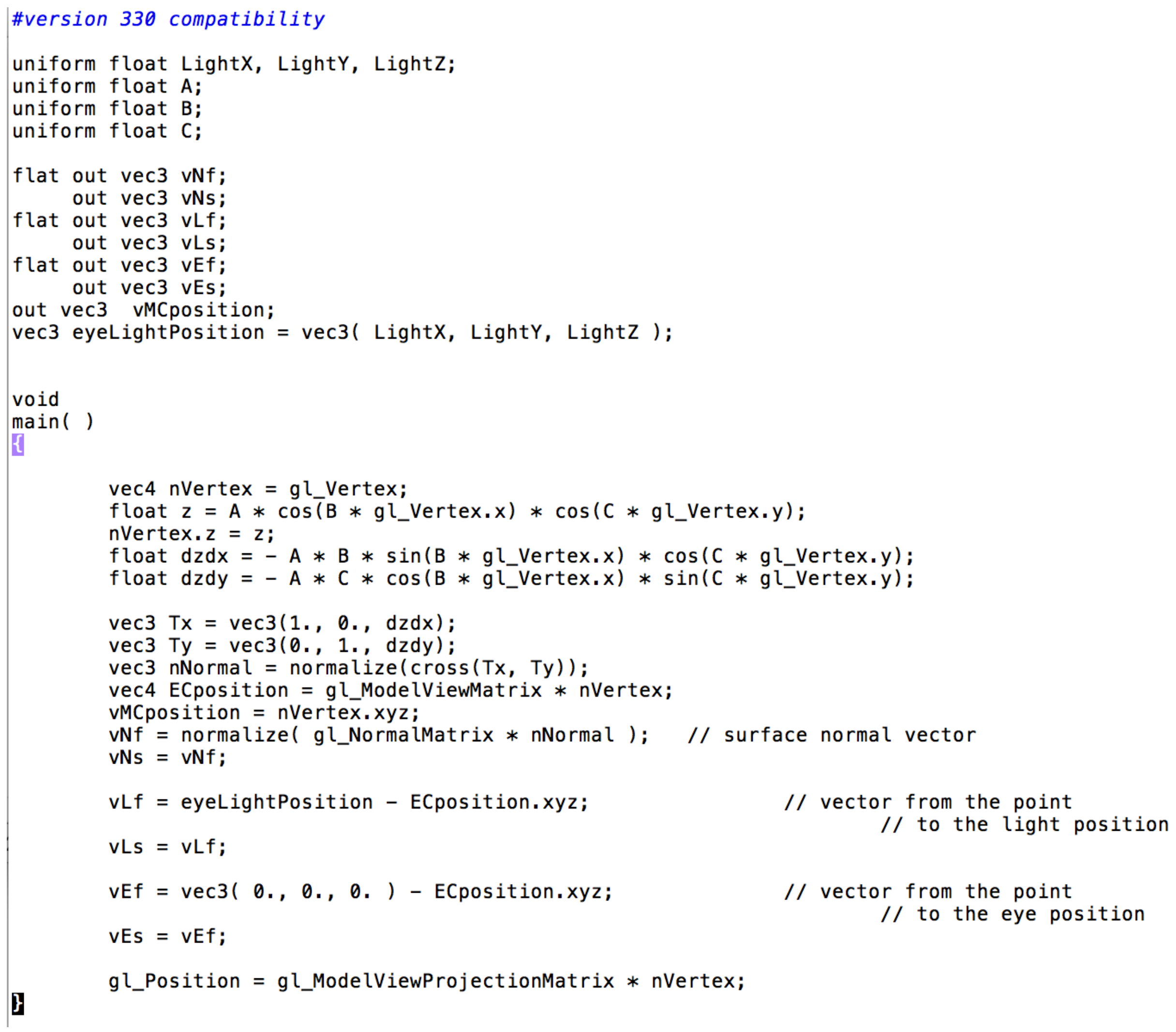
1. Source listings



```
##OpenGL GLIB
LookAt -.2 .5 3  -.2 .5 0  0 1 0

Vertex  chaozhang.vert
Fragment        chaozhang.frag
Program    Coscos                                        \
                Flat <false>                             \
                A <-2. .2 2.>                            \
                B <-10. 10. 10.>                                 \
                C <-10. 10. 10.>                                 \
        NoiseAmp <0. 2.5 10.>                    \
        NoiseFreq <0. 1. 10.>                    \
        Ka <0. 0.5 1.0>                         \
        Kd <0. 0.6 1.0>                         \
        Ks <0. 0.3 1.0>                         \
        Shininess <1. 10. 50.>                \
        LightX <-20. 5. 20.>                    \
        LightY <-20. 10. 20.>                   \
        LightZ <-20. 20. 20.>                   \
        uColor {1. .7 0. 1.}                     \
        SpecularColor {1. 1. 1. 1.}



QuadXY  -0.2  1.  50 50
```

chaozhang.glib

This is the glib file. This file includes all the basic information about this project. The shader type which is vertex shader and the fragment shader. The A, B, C are the changes in the x, y, z axle. The Ka, Kd and Ks are the Ambient, Diffuse, and Specular Color. The LightX, LightY and LightZ are the position of the light in the axles.



```
#version 330 compatibility

uniform float LightX, LightY, LightZ;
uniform float A;
uniform float B;
uniform float C;

flat out vec3 vNf;
     out vec3 vNs;
flat out vec3 vLf;
     out vec3 vLs;
flat out vec3 vEf;
     out vec3 vEs;
out vec3  vMCposition;
vec3 eyeLightPosition = vec3( LightX, LightY, LightZ );


void
main( )
{

        vec4 nVertex = gl_Vertex;
        float z = A * cos(B * gl_Vertex.x) * cos(C * gl_Vertex.y);
        nVertex.z = z;
        float dzdx = - A * B * sin(B * gl_Vertex.x) * cos(C * gl_Vertex.y);
        float dzdy = - A * C * cos(B * gl_Vertex.x) * sin(C * gl_Vertex.y);

        vec3 Tx = vec3(1., 0., dzdx);
        vec3 Ty = vec3(0., 1., dzdy);
        vec3 nNormal = normalize(cross(Tx, Ty));
        vec4 ECposition = gl_ModelViewMatrix * nVertex;
        vMCposition = nVertex.xyz;
        vNf = normalize( gl_NormalMatrix * nNormal );   // surface normal vector
        vNs = vNf;

        vLf = eyeLightPosition - ECposition.xyz;                // vector from the point
                                                                        // to the light position
        vLs = vLf;

        vEf = vec3( 0., 0., 0. ) - ECposition.xyz;              // vector from the point
                                                                        // to the eye position
        vEs = vEf;

        gl_Position = gl_ModelViewProjectionMatrix * nVertex;
}
```

chaozhang.vert

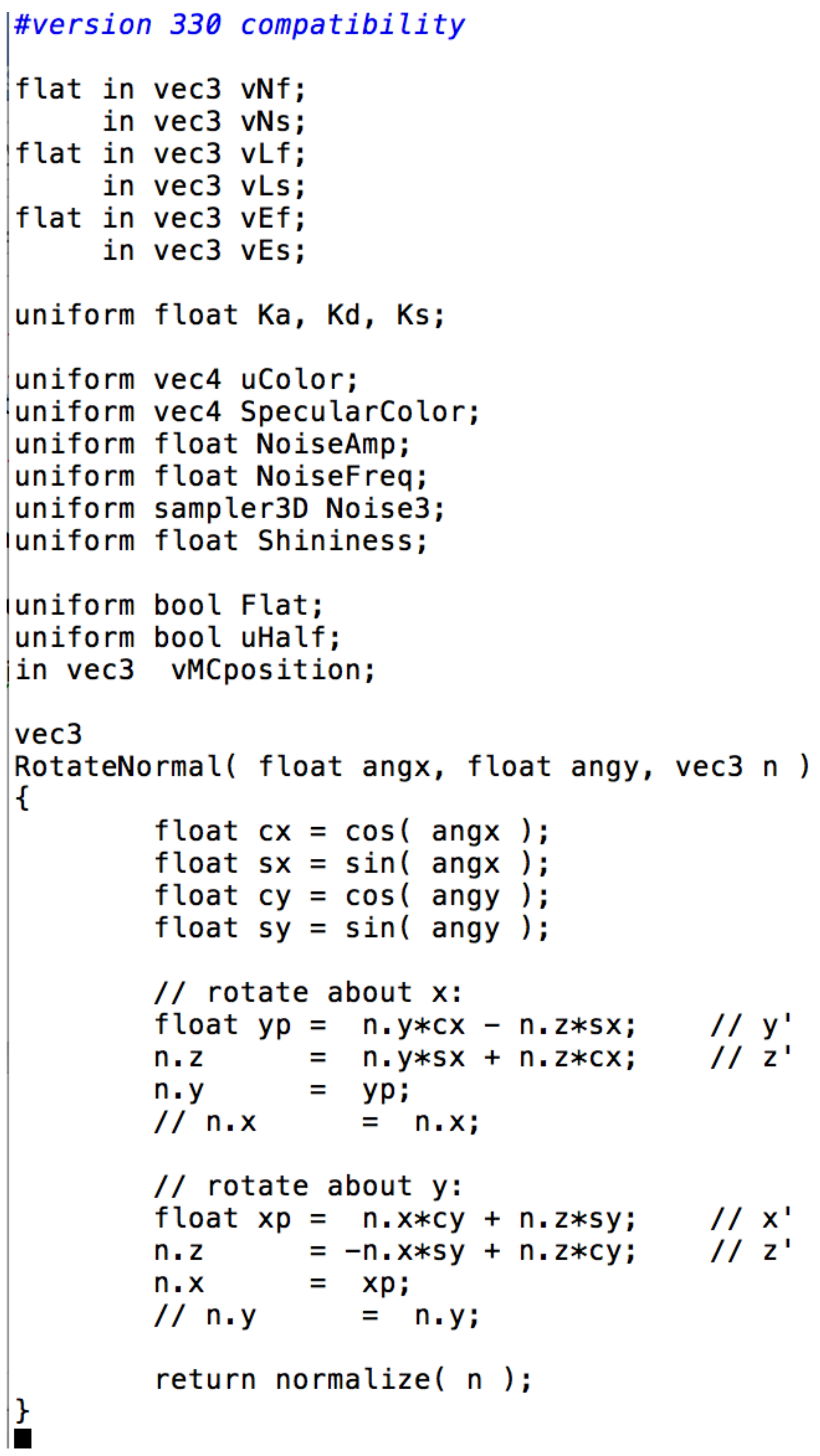
This is the vertex shader file. This file has all the vertex shader information. The compute of the normal, position of the lights. The nVertex mean the new vertex position, it is equal to gl_vertex in the beginning. Then I use the float z = A * cos(B * gl_Vertex.x) * cos(C * gl_Vertex.y); to compute the position in the z axle of the gl_vertex and give this new z value to the nVertex. This make the change of A works.

float dzdx = - A * B * sin(B * gl_Vertex.x) * cos(C * gl_Vertex.y);
float dzdy = - A * C * cos(B * gl_Vertex.x) * sin(C * gl_Vertex.y);
This two line can get the tangent and then I use
vec3 Tx = vec3(1., 0., dzdx);
vec3 Ty = vec3(0., 1., dzdy); to get the tangent vectors. The Tx and Ty can help us to get the new normal. vec3 nNormal = normalize(cross(Tx, Ty));

Because of the change of the A, I need to change the light position.

vNf = normalize(gl_NormalMatrix * nNormal);

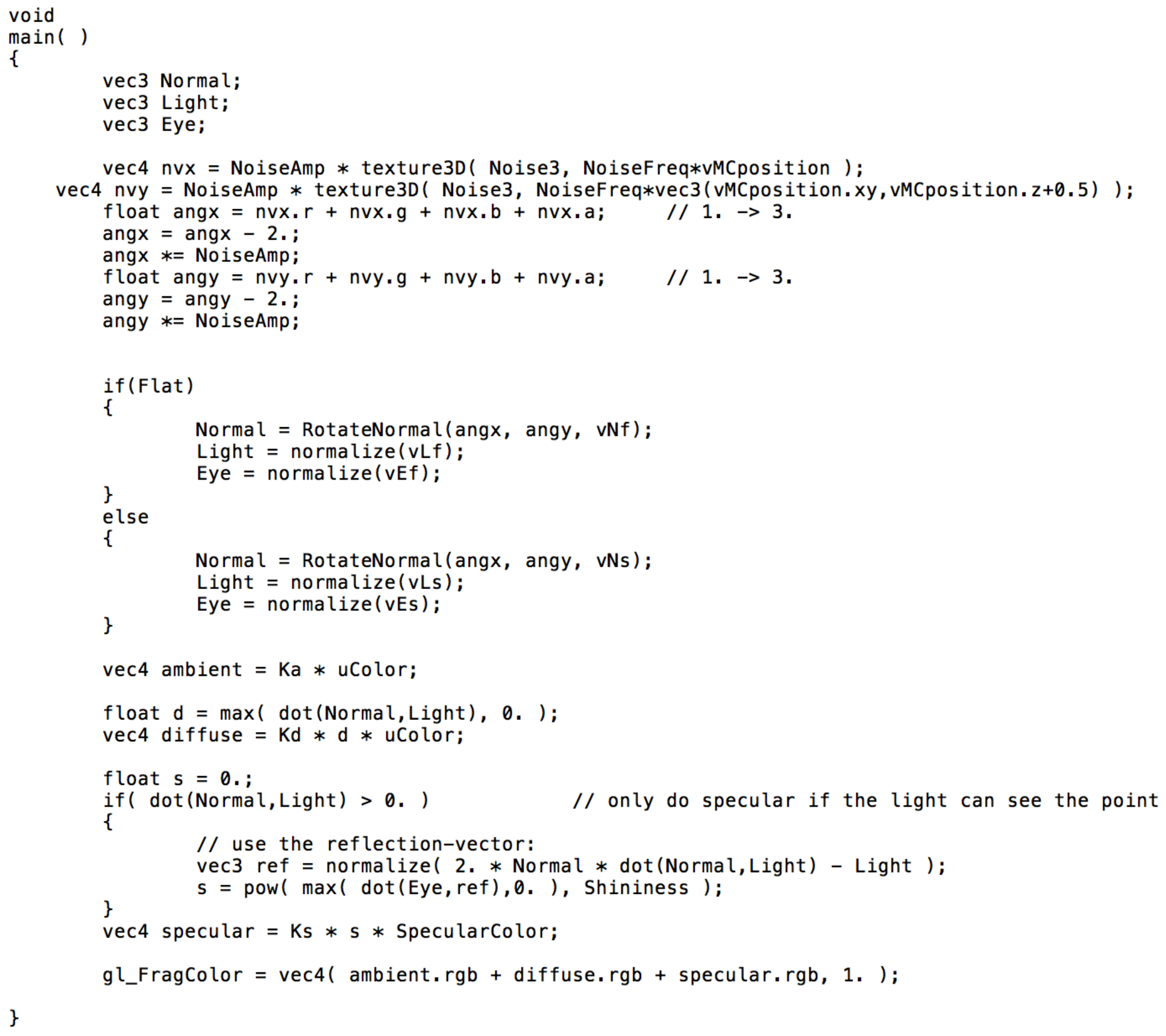
gl_Position = gl_ModelViewProjectionMatrix * nVertex;



```
#version 330 compatibility

flat in vec3 vNf;
     in vec3 vNs;
flat in vec3 vLf;
     in vec3 vLs;
flat in vec3 vEf;
     in vec3 vEs;

uniform float Ka, Kd, Ks;

uniform vec4 uColor;
uniform vec4 SpecularColor;
uniform float NoiseAmp;
uniform float NoiseFreq;
uniform sampler3D Noise3;
uniform float Shininess;

uniform bool Flat;
uniform bool uHalf;
in vec3  vMCposition;

vec3
RotateNormal( float angx, float angy, vec3 n )
{
        float cx = cos( angx );
        float sx = sin( angx );
        float cy = cos( angy );
        float sy = sin( angy );

        // rotate about x:
        float yp =  n.y*cx - n.z*sx;    // y'
        n.z      =  n.y*sx + n.z*cx;    // z'
        n.y      =  yp;
        // n.x      =  n.x;

        // rotate about y:
        float xp =  n.x*cy + n.z*sy;    // x'
        n.z      = -n.x*sy + n.z*cy;    // z'
        n.x      =  xp;
        // n.y      =  n.y;

        return normalize( n );
}
```

part1 chaozhang.frag

This is the first part of the fragment shader. This part do a rotate of the normal. The reason I do this is to implement the bump-mapping. This function is to rotate the normal in two ways, the angle with x axle and the angle with y axle.



```
void
main( )
{
        vec3 Normal;
        vec3 Light;
        vec3 Eye;

        vec4 nvx = NoiseAmp * texture3D( Noise3, NoiseFreq*vMCposition );
    vec4 nvy = NoiseAmp * texture3D( Noise3, NoiseFreq*vec3(vMCposition.xy,vMCposition.z+0.5) );
        float angx = nvx.r + nvx.g + nvx.b + nvx.a;     // 1. -> 3.
        angx = angx - 2.;
        angx *= NoiseAmp;
        float angy = nvy.r + nvy.g + nvy.b + nvy.a;     // 1. -> 3.
        angy = angy - 2.;
        angy *= NoiseAmp;


        if(Flat)
        {
                Normal = RotateNormal(angx, angy, vNf);
                Light = normalize(vLf);
                Eye = normalize(vEf);
        }
        else
        {
                Normal = RotateNormal(angx, angy, vNs);
                Light = normalize(vLs);
                Eye = normalize(vEs);
        }

        vec4 ambient = Ka * uColor;

        float d = max( dot(Normal,Light), 0. );
        vec4 diffuse = Kd * d * uColor;

        float s = 0.;
        if( dot(Normal,Light) > 0. )            // only do specular if the light can see the point
        {
                // use the reflection-vector:
                vec3 ref = normalize( 2. * Normal * dot(Normal,Light) - Light );
                s = pow( max( dot(Eye,ref),0. ), Shininess );
        }
        vec4 specular = Ks * s * SpecularColor;

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

}
```

part2 chaozhang.frag

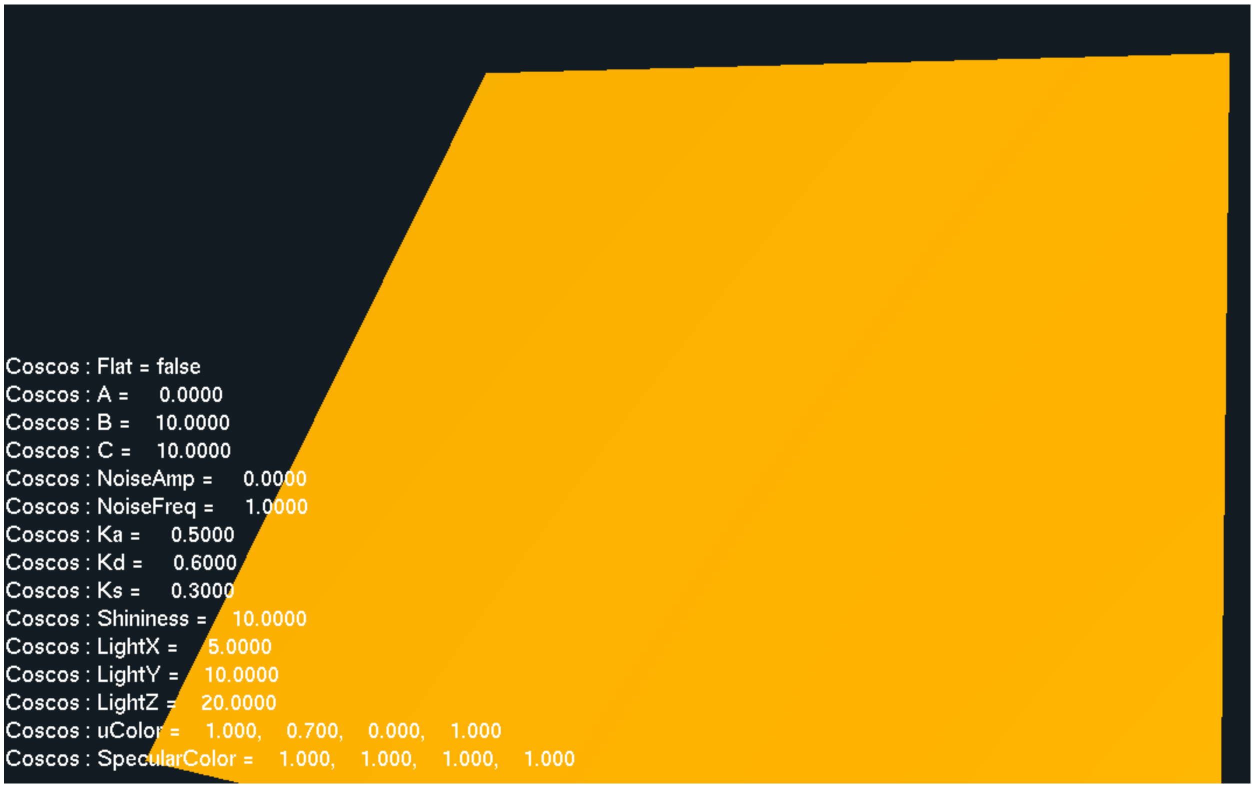
This part is to implement the rotate normal.

vec4 nvx = uNoiseAmp * texture3D(Noise3, uNoiseFreq*vMC);

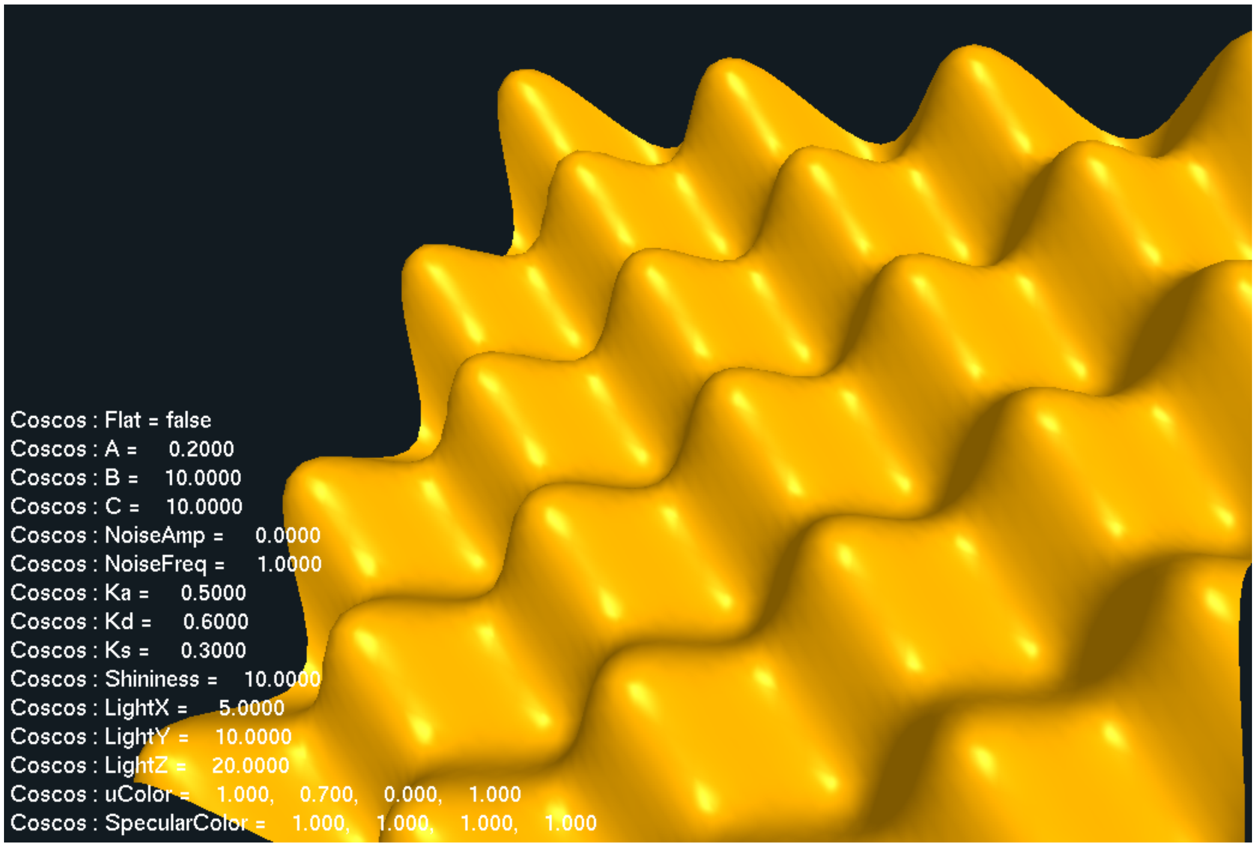
vec4 nvy = uNoiseAmp * texture3D(Noise3, uNoiseFreq*vec3(vMC.xy,vMC.z+0.5));

this two line is to calculate the noise vectors. This part is the same as project 3 noise part. After that, I make the Normal = RotateNormal(angx, angy, vNf); to get the result.

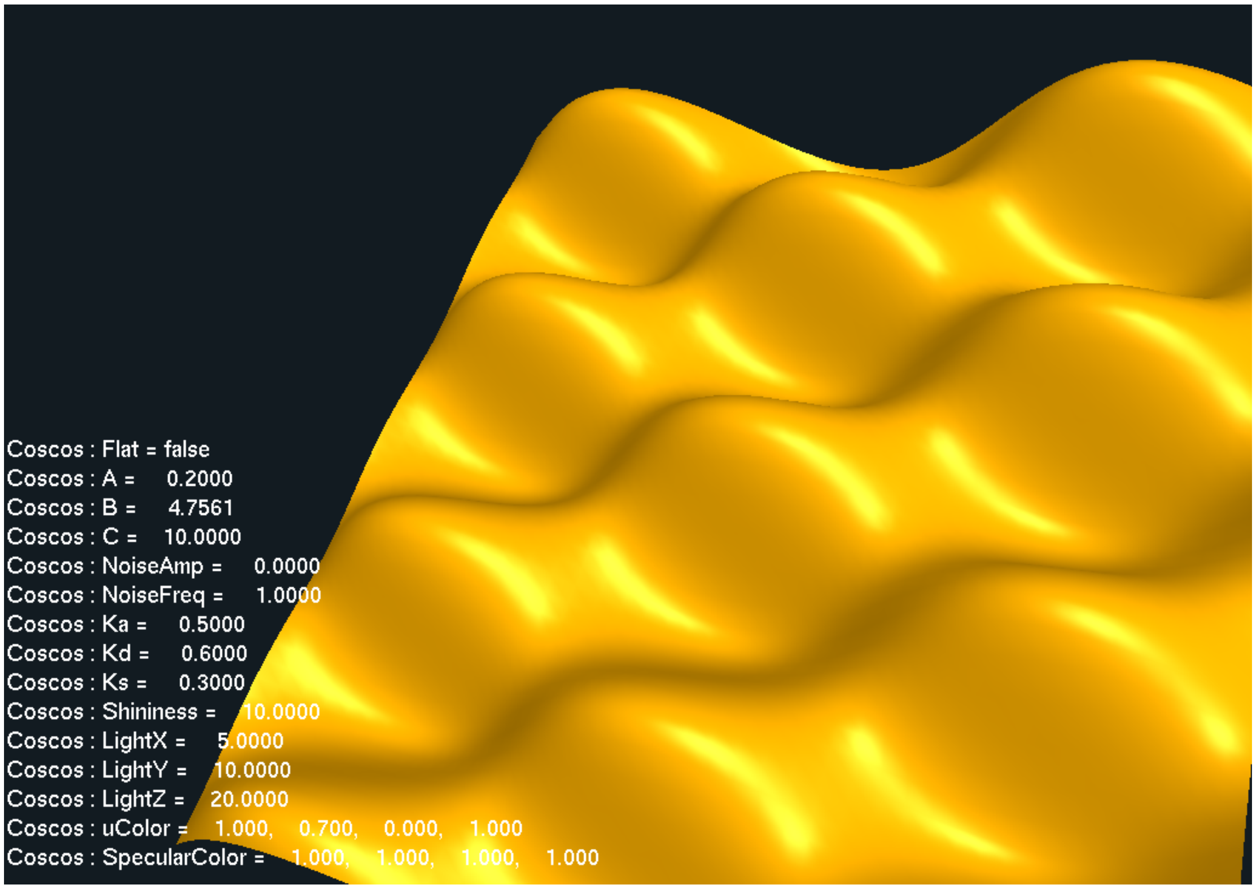
2. Result



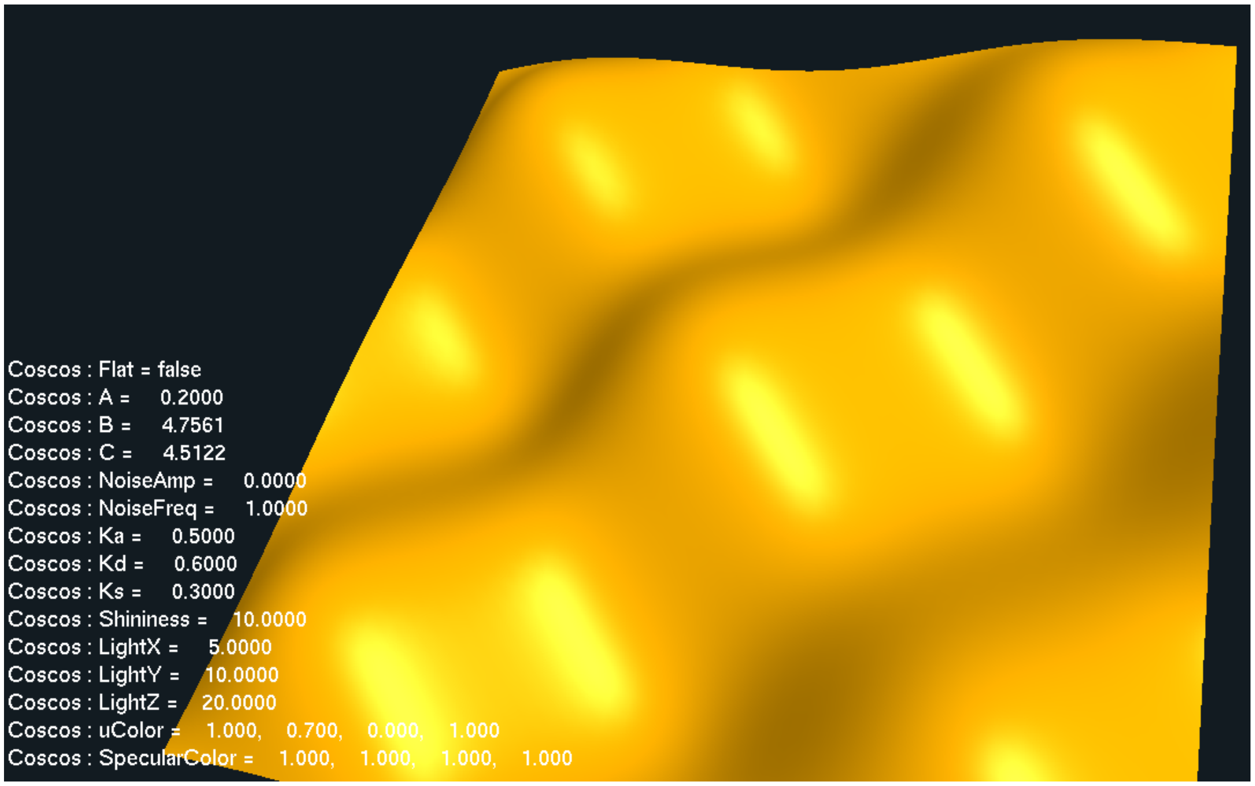
```
Coscos : Flat = false
Coscos : A =      0.0000
Coscos : B =     10.0000
Coscos : C =     10.0000
Coscos : NoiseAmp =      0.0000
Coscos : NoiseFreq =      1.0000
Coscos : Ka =      0.5000
Coscos : Kd =      0.6000
Coscos : Ks =      0.3000
Coscos : Shininess =     10.0000
Coscos : LightX =      5.0000
Coscos : LightY =     10.0000
Coscos : LightZ =     20.0000
Coscos : uColor =     1.000,     0.700,     0.000,     1.000
Coscos : SpecularColor =     1.000,     1.000,     1.000,     1.000
```



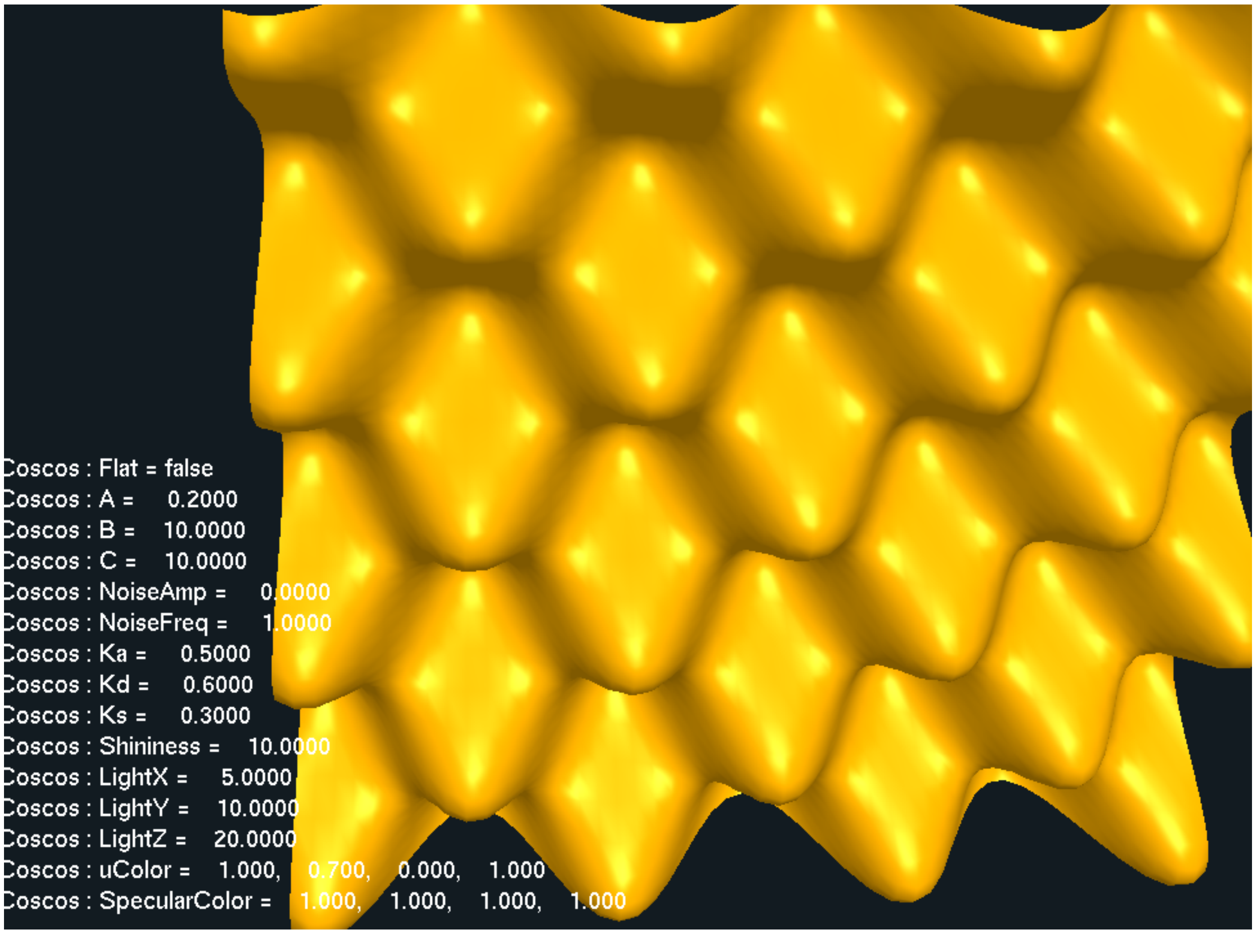
```
Coscos : Flat = false
Coscos : A =      0.2000
Coscos : B =     10.0000
Coscos : C =     10.0000
Coscos : NoiseAmp =      0.0000
Coscos : NoiseFreq =      1.0000
Coscos : Ka =      0.5000
Coscos : Kd =      0.6000
Coscos : Ks =      0.3000
Coscos : Shininess =     10.0000
Coscos : LightX =      5.0000
Coscos : LightY =     10.0000
Coscos : LightZ =     20.0000
Coscos : uColor =     1.000,     0.700,     0.000,     1.000
Coscos : SpecularColor =     1.000,     1.000,     1.000,     1.000
```



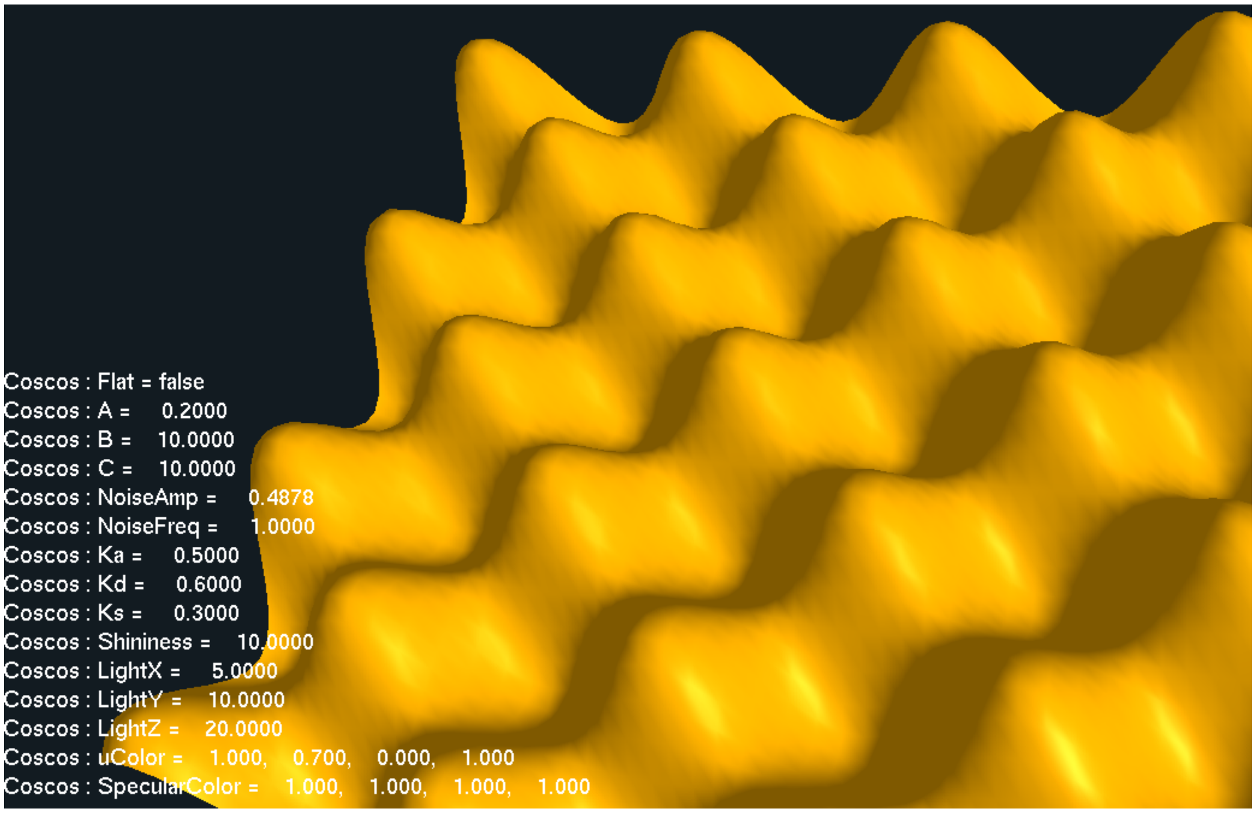
```
Coscos : Flat = false
Coscos : A =     0.2000
Coscos : B =     4.7561
Coscos : C =    10.0000
Coscos : NoiseAmp =     0.0000
Coscos : NoiseFreq =     1.0000
Coscos : Ka =     0.5000
Coscos : Kd =     0.6000
Coscos : Ks =     0.3000
Coscos : Shininess =    10.0000
Coscos : LightX =     5.0000
Coscos : LightY =    10.0000
Coscos : LightZ =    20.0000
Coscos : uColor =    1.000,    0.700,    0.000,    1.000
Coscos : SpecularColor =    1.000,    1.000,    1.000,    1.000
```



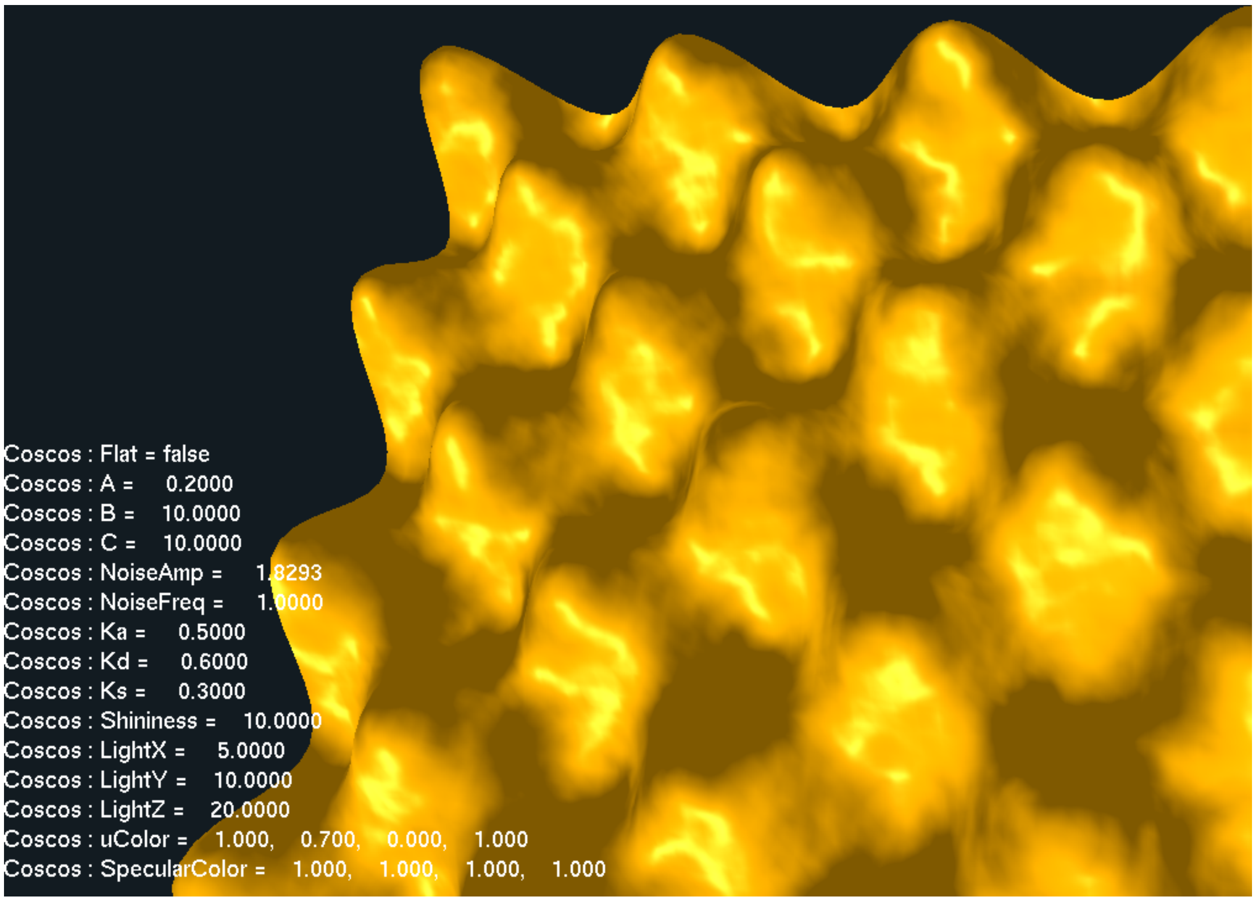
```
Coscos : Flat = false
Coscos : A =     0.2000
Coscos : B =     4.7561
Coscos : C =     4.5122
Coscos : NoiseAmp =     0.0000
Coscos : NoiseFreq =     1.0000
Coscos : Ka =     0.5000
Coscos : Kd =     0.6000
Coscos : Ks =     0.3000
Coscos : Shininess =    10.0000
Coscos : LightX =     5.0000
Coscos : LightY =    10.0000
Coscos : LightZ =    20.0000
Coscos : uColor =    1.000,    0.700,    0.000,    1.000
Coscos : SpecularColor =    1.000,    1.000,    1.000,    1.000
```



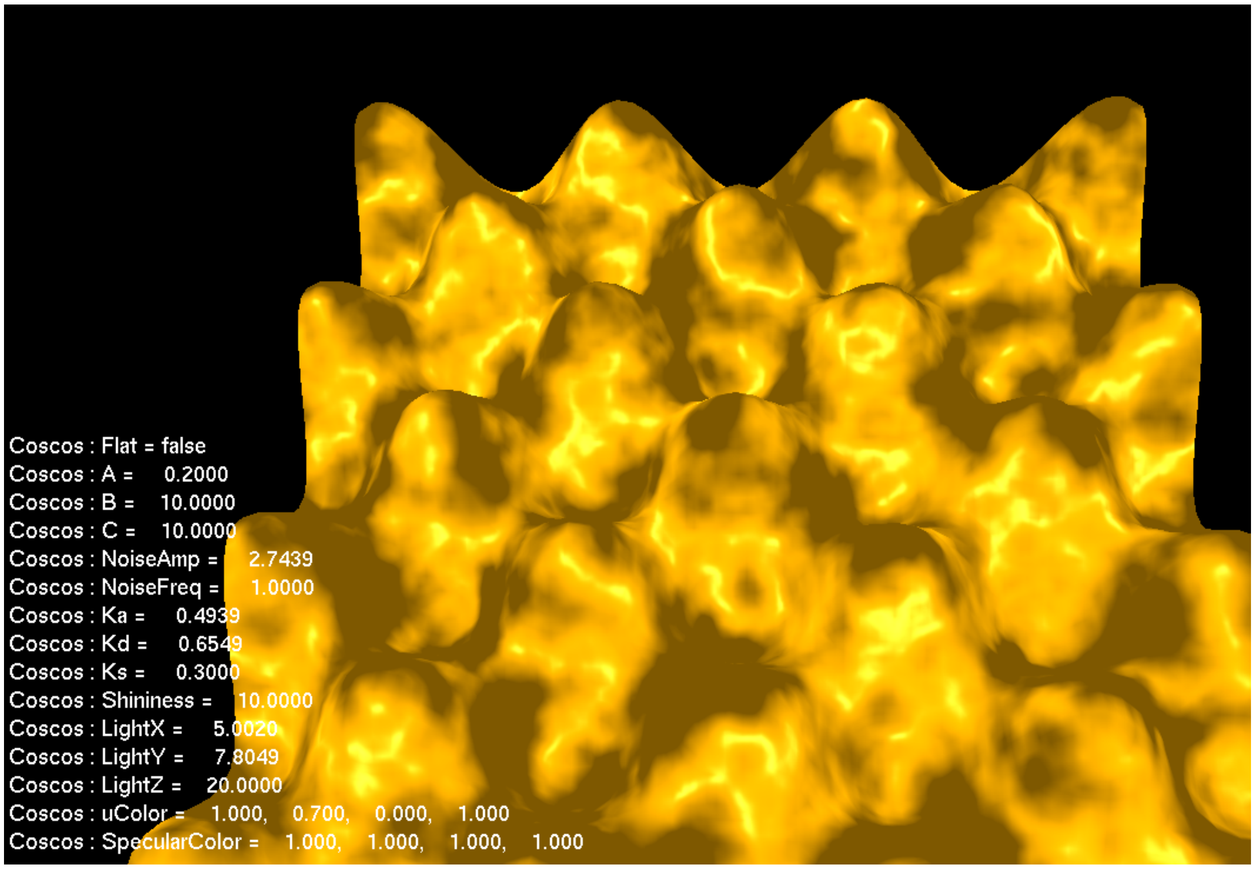
```
Coscos : Flat = false
Coscos : A =      0.2000
Coscos : B =     10.0000
Coscos : C =     10.0000
Coscos : NoiseAmp =      0.0000
Coscos : NoiseFreq =      1.0000
Coscos : Ka =      0.5000
Coscos : Kd =      0.6000
Coscos : Ks =      0.3000
Coscos : Shininess =     10.0000
Coscos : LightX =      5.0000
Coscos : LightY =     10.0000
Coscos : LightZ =     20.0000
Coscos : uColor =     1.000,     0.700,     0.000,     1.000
Coscos : SpecularColor =     1.000,     1.000,     1.000,     1.000
```



```
Coscos : Flat = false
Coscos : A =     0.2000
Coscos : B =     10.0000
Coscos : C =     10.0000
Coscos : NoiseAmp =     0.4878
Coscos : NoiseFreq =     1.0000
Coscos : Ka =     0.5000
Coscos : Kd =     0.6000
Coscos : Ks =     0.3000
Coscos : Shininess =     10.0000
Coscos : LightX =     5.0000
Coscos : LightY =     10.0000
Coscos : LightZ =     20.0000
Coscos : uColor =     1.000,     0.700,     0.000,     1.000
Coscos : SpecularColor =     1.000,     1.000,     1.000,     1.000
```



```
Coscos : Flat = false
Coscos : A =     0.2000
Coscos : B =     10.0000
Coscos : C =     10.0000
Coscos : NoiseAmp =     1.8293
Coscos : NoiseFreq =     1.0000
Coscos : Ka =     0.5000
Coscos : Kd =     0.6000
Coscos : Ks =     0.3000
Coscos : Shininess =     10.0000
Coscos : LightX =     5.0000
Coscos : LightY =     10.0000
Coscos : LightZ =     20.0000
Coscos : uColor =     1.000,     0.700,     0.000,     1.000
Coscos : SpecularColor =     1.000,     1.000,     1.000,     1.000
```



```
Coscos : Flat = false
Coscos : A =     0.2000
Coscos : B =    10.0000
Coscos : C =    10.0000
Coscos : NoiseAmp =     2.7439
Coscos : NoiseFreq =     1.0000
Coscos : Ka =     0.4939
Coscos : Kd =     0.6549
Coscos : Ks =     0.3000
Coscos : Shininess =    10.0000
Coscos : LightX =     5.0020
Coscos : LightY =     7.8049
Coscos : LightZ =    20.0000
Coscos : uColor =    1.000,    0.700,    0.000,    1.000
Coscos : SpecularColor =    1.000,    1.000,    1.000,    1.000
```