CS557 Project #4

Yunfan Li

[liyunf@oregonstate.edu](mailto:liyunf@oregonstate.edu)

1. **Source Listings**
   1. ***liyunf.glib***

##OpenGL GLIB

LookAt 0 0 3 0 0 0 0 1 0

Perspective 70

Noise3D 128

Vertex liyunf.vert

Fragment liyunf.frag

Program CosCos \

uA <0. 0. 1.> \

uB <0. 0. 10.> \

uC <0. 0. 10.> \

uNoiseAmp <0. 0. 10.> \

uNoiseFreq <0. 0.15 2.> \

uKa <0. 0.1 1.0> \

uKd <0. 0.6 1.0> \

uKs <0. 0.3 1.0> \

uShininess <1. 10. 50.> \

uLightX <-20. 5. 20.> \

uLightY <-20. 10. 20.> \

uLightZ <-20. 20. 20.> \

uColor {1. .7 0. 1.} \

uSpecularColor {1. 1. 1. 1.}

QuadXY -0.2 1. 50 50

* 1. ***liyunf.vert***

#version 330 compatibility

out vec2 vST;

out vec3 vMC;

out vec3 vNf;

out vec3 vLf;

out vec3 vEf;

uniform float uA;

uniform float uB;

uniform float uC;

uniform float uLightX;

uniform float uLightY;

uniform float uLightZ;

vec3 eyeLightPosition = vec3(uLightX, uLightY, uLightZ);

void

main( )

{

vST = gl\_MultiTexCoord0.st;

vec4 p = vec4(gl\_Vertex.x, gl\_Vertex.y, uA\*cos(uB\*gl\_Vertex.x)\*cos(uC\*gl\_Vertex.y), 1.);

vMC = p.xyz;

vec3 Tx = vec3(1., 0., (0-uA)\*uB\*sin(uB\*p.x)\*cos(uC\*p.y));

vec3 Ty = vec3(0., 1., (0-uA)\*uC\*cos(uB\*p.x)\*sin(uC\*p.y));

vec3 normal = normalize(cross(Tx, Ty));

vec4 ECposition = gl\_ModelViewMatrix \* p;

vNf = normalize(gl\_NormalMatrix \* normal);

vLf = eyeLightPosition - ECposition.xyz;

vEf = vec3(0., 0., 0.) - ECposition.xyz;

gl\_Position = gl\_ModelViewProjectionMatrix \* p;

}

* 1. ***liyunf.frag***

#version 330 compatibility

in vec2 vST;

in vec3 vMC;

in vec3 vNf;

in vec3 vLf;

in vec3 vEf;

uniform float uKa;

uniform float uKd;

uniform float uKs;

uniform float uShininess;

uniform vec4 uColor;

uniform vec4 uSpecularColor;

uniform float uNoiseAmp;

uniform float uNoiseFreq;

uniform sampler3D Noise3D;

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 );

}

void

main ()

{

vec3 Normal, Light, Eye;

vec4 nvx = texture3D(Noise3D, uNoiseFreq \* vMC);

vec4 nvy = texture3D(Noise3D, uNoiseFreq \* vec3(vMC.xy, vMC.z+0.5));

float angx = (nvx.r + nvx.g + nvx.b + nvx.a – 2) \* uNoiseAmp;

float angy = (nvy.r + nvy.g + nvy.b + nvy.a – 2) \* uNoiseAmp;

Normal = RotateNormal(angx, angy, vNf);

Light = normalize(vLf);

Eye = normalize(vEf);

vec4 ambient = uKa \* uColor;

vec4 diffuse = uKd \* max(dot(Normal, Light), 0.) \* uColor;

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 \* uSpecularColor;

gl\_FragColor = vec4(ambient.rgb + diffuse.rgb + specular.rgb, 1.);

}

1. **What I did & Why it worked**

***2.1 What I did***

In my liyunf.glib file, I created a Quad extended in XY plane and connect this .glib flie with liyunf.vert vertex shader and liyunf.frag fragment shader.

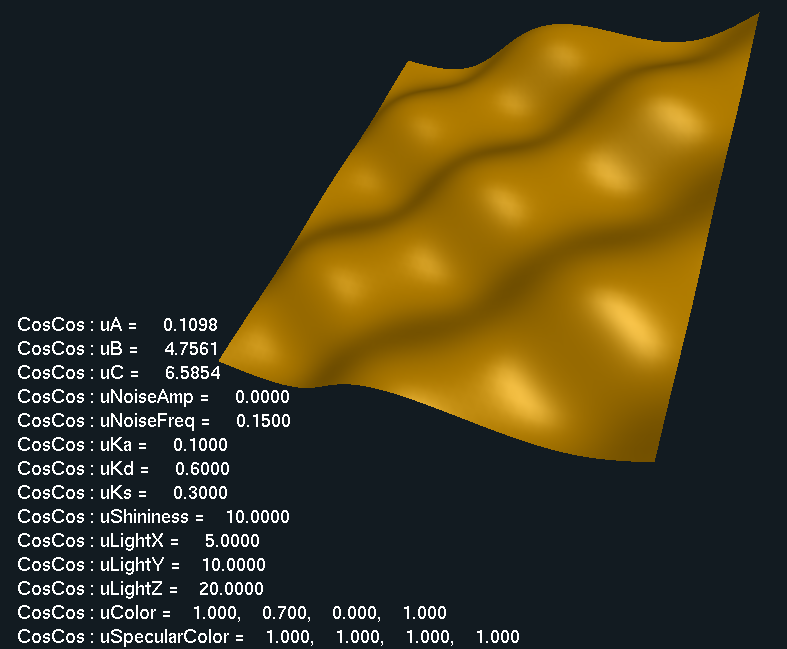
In my liyunf.vert file, I used original vertex position to displace the quad by using the equation shown on project website. And I used new vertex position to calculate new normal, light vector and eye vector. These three vec3 variable would be sent through rasterizer and pass into fragment shader.

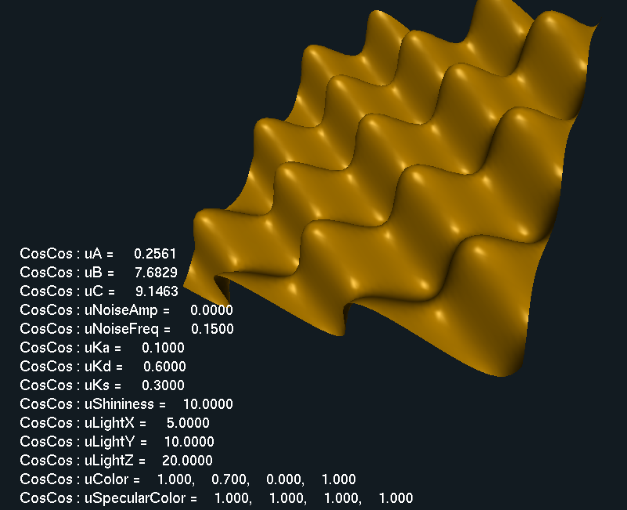
In my liyunf.frag flie, I created two vec4 variable nvx and nvy in order to perturb my normal. Then I used RotateNormal function which is given on website to rotate my normal that passed from vertex shader and normalize my light and eye vector. Finally I calculate ambient, diffuse and specular color, added them together to get my final fragment color.

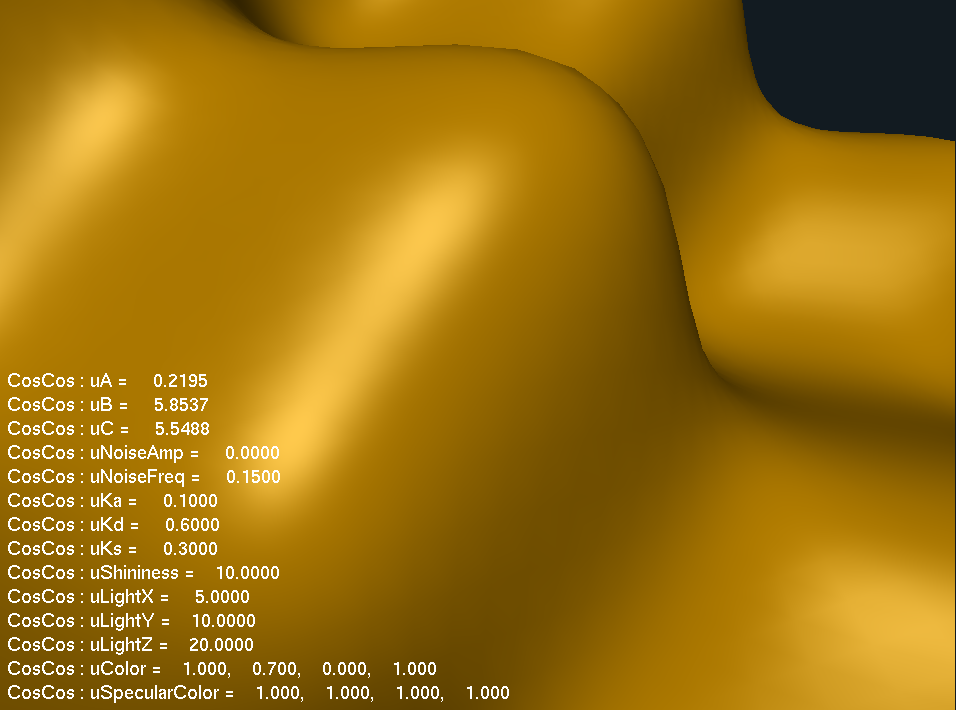
***2.2 Why it worked***

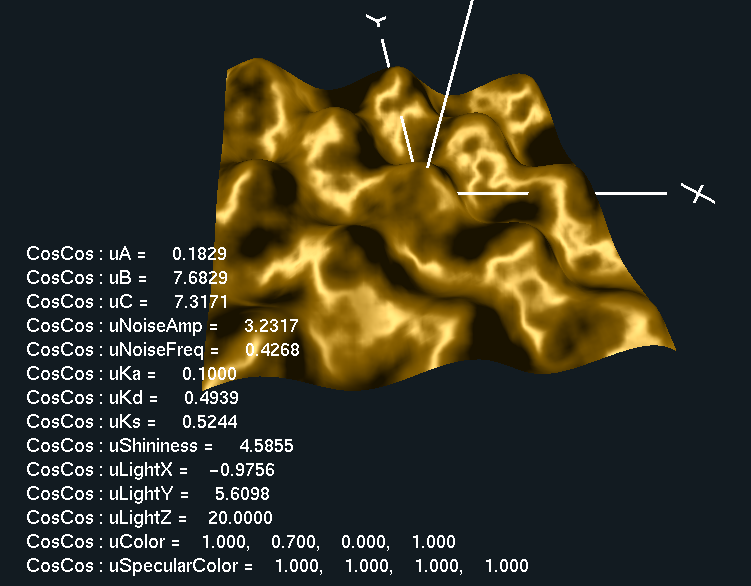
The noise is a variable that we can control. So we use texture to pass vec3 and then get a vec4. We can use this vec4 variable as a noise value and perturb my normal to get my bump-mapping effect.

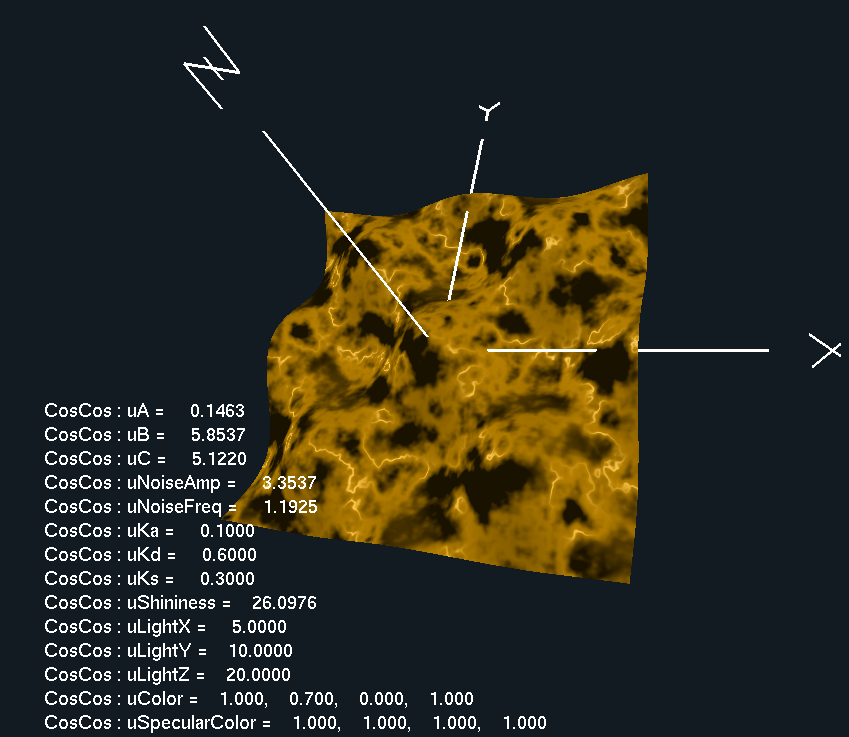
1. **Images**

****

****

****

****

****