Oregon State University

CS_557_X001_W2022 COMPUTER GRAPHICS SHADERS

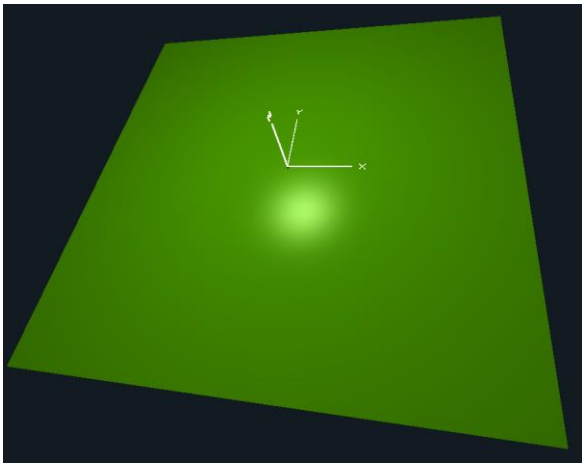
Project #3

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

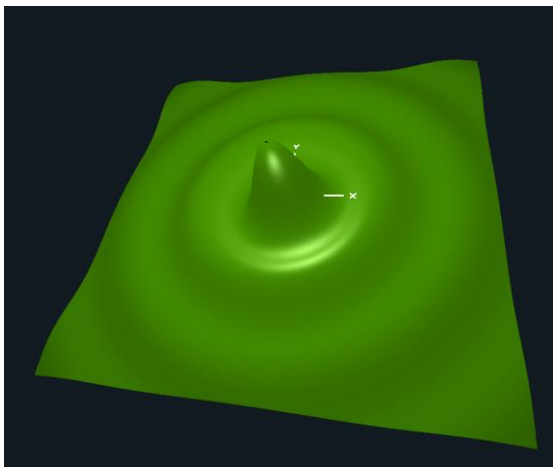
I used the custom Sinc function and DerivSinc function to achieve a sine wave effect by changing the parameters of the z-axis, and to change the brightness of the light source by adjusting the parameters of uShininess.
Also, I adjusted the uNoiseAmp and uNoiseFreq values to generate different noise effects as last assignment.

Screen Shots：
Kaltura link: https://media.oregonstate.edu/media/t/1_rts9c0wd

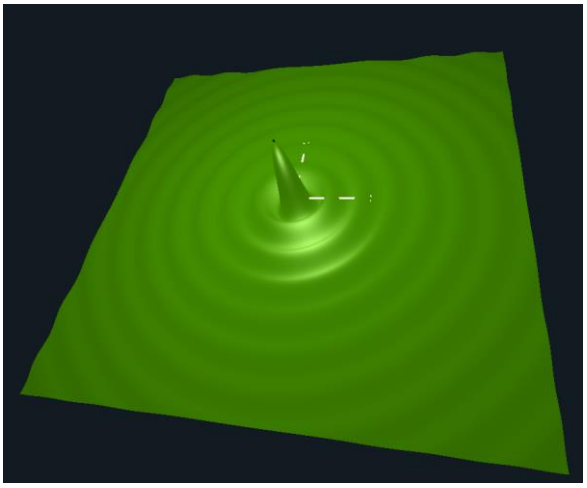
Original



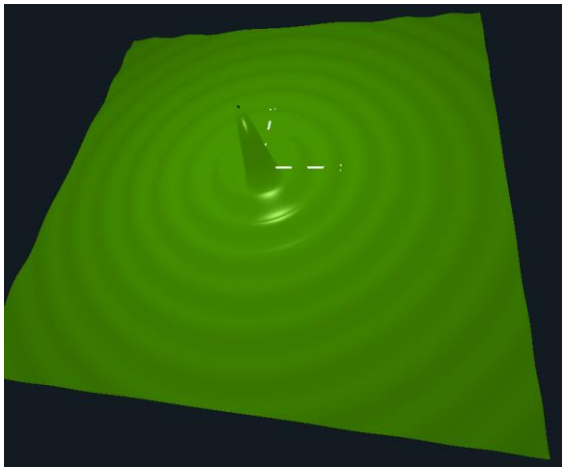
adjust uA



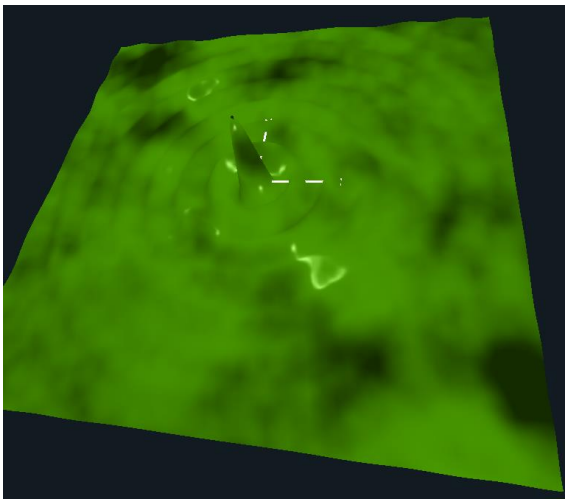
adjust uK



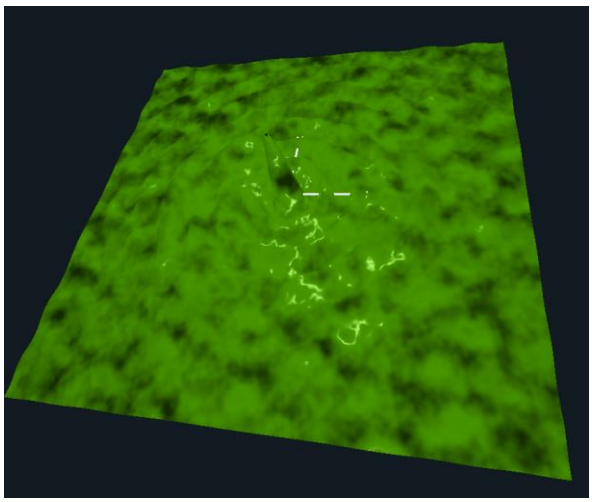
adjust uShiness



adjust uNoiseAmp



adjust uNoiseFreq



Key snippets:

Parameter predefined:

```
Vertex   sinc.vert
Fragment sinc.frag
Program  Sinc                    \
    uA <-2. 0. 2.>               \
    uK <0.. 3.14 10.>            \
    uNoiseAmp <0. 0. 5.>         \
    uNoiseFreq <0. .1 1.>        \
    uKa <0. .2 1.>               \
    uKd <0. .5 1.>               \
    uKs <0. .4 1.>               \
    uShininess <.01 10. 200.>    \
```

Sine wave function definition:

```
float
Sinc( float r, float k )
{
    if( r == 0. )
        return 1.;
    return sin(r*k) / (r*k);
}

float
DerivSinc( float r, float k )
{
    if( r == 0. )
        return 0;
    return ( r*k*cos(r*k) - sin(r*k) ) / ( r*k*r*k );
}
```

Sine wave function application:

```
    vec4 newVertex = gl_Vertex;
    float r = length( newVertex.xy );
    newVertex.z = uA * Sinc( r, uK );
```

Vector normalize:

```
float dzdr = uA * DerivSinc( r, uK );
float drdx = newVertex.x / r;
float drdy = newVertex.y / r;
float dzdx = dzdr * drdx;
float dzdy = dzdr * drdy;
```

Cut quads into sub-quads:

```
QuadXY  -0.2  5.  300  300
```

uNoiseAmp & uNoiseFreq application:

```
vec4 nvx = texture(Noise3,uNoiseFreq*vMC);
float angx = nvx.r + nvx.g + nvx.b + nvx.a  -  2.; // -1. to +1.
angx *= uNoiseAmp;

vec4 nvy = texture( Noise3, uNoiseFreq*vec3(vMC.xy,vMC.z+0.5) );
float angy = nvy.r + nvy.g + nvy.b + nvy.a  -  2.;
angy *= uNoiseAmp;
```

Rotate definition :

```
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;

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

        return normalize( n );
```

Rotate function application:

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