Oregon State University

CS_557_X001_W2022 COMPUTER GRAPHICS SHADERS

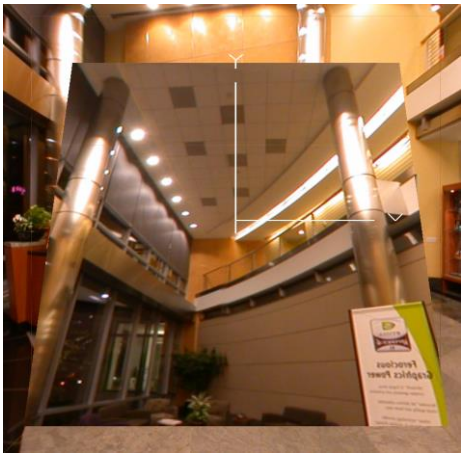
Project #4

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

Like the last assignment, the I used the custom Sinc function and DerivSinc function to achieve a sine wave effect by changing the parameters of the z-axis, and use uNoiseAmp and uNoiseFreq values to generate different noise effects as last assignment. In addition, I used cube-mapping to create a reflective and refractive display with uMix to adjust the reflection and refraction version of the scene, uEta to adjust the refractive index and uWhiteMix to adjust the visibility.

Screen Shots：
Kaltura link: https://media.oregonstate.edu/media/t/1_ayzba57c

Original



adjust uA

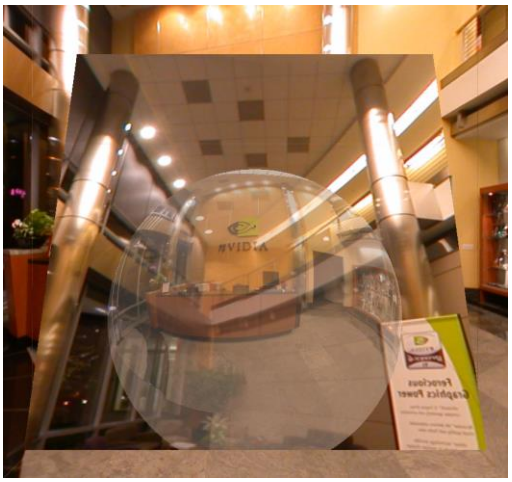
adjust uK

adjust uNoiseAmp

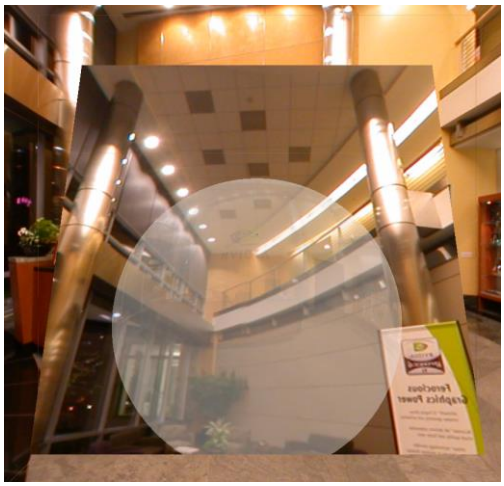
adjust uNoiseFreq

adjust uMix

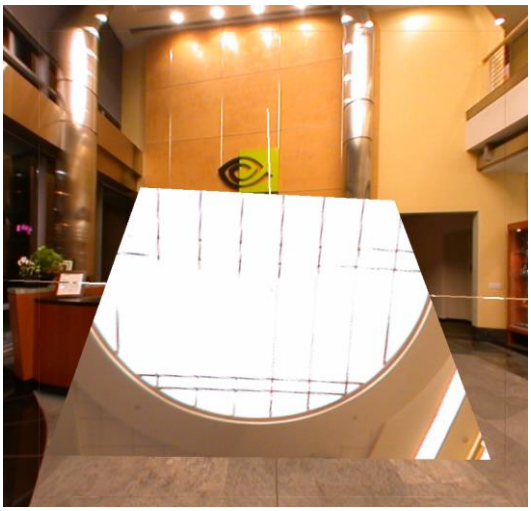
adjust uEta

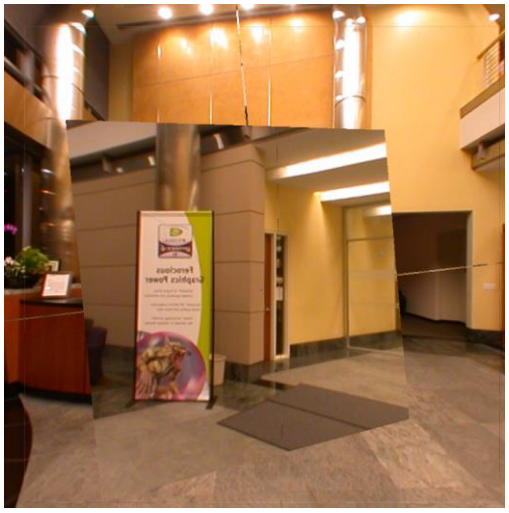


adjust uWhiteMix



Adject object





Key snippets:

Parameter predefined:

```
Vertex   sinccube.vert
Fragment sinccube.frag
Program  SincCube                \
    uReflectUnit 6               \
    uRefractUnit 7               \
    uA <0.  0.  2.>              \
    uK <0.1 3.14 20.>            \
    uNoiseAmp <0.  0.  5.>       \
    uNoiseFreq <0.  0.1 0.5>     \
    uEta <1.  1.2 4.>            \
    uMix <0.  0.  1.>            \
    uWhiteMix <0.  0.3 1.>
```

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;

vec3 Tx = vec3(1., 0., dzdx );
vec3 Ty = vec3(0., 1., dzdy );

vec3 newNormal = normalize(cross(Tx, Ty));
```

Cut quads into sub-quads:

```
QuadXY  -0.2  2.  300 300
```

cubemaps:

```
Vertex texture.vert
Fragment texture.frag
Program Texture uTexUnit 6

Texture2D  6  nvposx.bmp
QuadYZ 5. 5. 10 10

Texture2D  6  nvnegx.bmp
QuadYZ -5. 5. 10 10

Texture2D  6  nvposy.bmp
QuadXZ 5. 5. 10 10

Texture2D  6  nvnegy.bmp
QuadXZ -5. 5. 10 10

Texture2D  6  nvposz.bmp
QuadXY 5. 5. 10 10

Texture2D  6  nvnegz.bmp
QuadXY -5. 5. 10 10
```

uNoiseAmp & uNoiseFreq application:

```
vec4 nvx = texture(Noise3,uNoiseFreq*vMC);
vec4 nvy = texture( Noise3, uNoiseFreq*vec3(vMC.xy,vMC.z+0.5) );

float angx = nvx.r + nvx.g + nvx.b + nvx.a; //  1. -> 3.
angx = angx - 2.;                           // -1. -> 1.
angx *= uNoiseAmp;

float angy = nvy.r + nvy.g + nvy.b + nvy.a; //  1. -> 3.
angy = angy - 2.;                           // -1. -> 1.
angy *= uNoiseAmp;

Normal = RotateNormal(angx,angy,Normal);
Normal = normalize( gl_NormalMatrix * Normal);
```

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 );
```

reflectVector and refractVector:

```
vec3 reflectVector = reflect(Eye, Normal);
vec4 reflectColor = textureCube( uReflectUnit, reflectVector );

vec3 refractVector = refract(Eye,Normal,uEta);

vec4 refractColor;
```

Adjustment of reflectivity:

```
refractColor = texture( uRefractUnit, refractVector );
refractColor = mix( refractColor, WHITE, uWhiteMix  );
```

```
gl_FragColor = mix(reflectColor,refractColor,uMix);
```

wall decorations:

```
#version 330 compatibility

uniform sampler2D uTexUnit;

in vec2 vST;

void
main( )
{
    vec3 newColor = texture(uTexUnit,vST).rgb;
    gl_FragColor = vec4(newColor, 1. );
}
```

```
#version 330 compatibility
out vec2 vST;
void
main( )
{
    vST = gl_MultiTexCoord0.st;
    gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}
```