CS557 PROJECT #5

Ritesh Sharma

MS Student, Computer Science,
Oregon State University

GLSL is such a powerful shading language which can generate image in a fly. It's been fun generating images by writing very few lines of code. The project was a bit difficult to do, still I managed to get the results what was required by this project.

For this project, I have used hints given in project description and some codes provided during the class lecture. The main equation which generate the pleats is as follows:

$$Z = uK * (Y_0 - Y) * \sin\left(2 * \pi * \frac{X}{uP}\right)$$

Where uK is a constant that controls amplitude of the pleat fold, Y_0 is the top of the curtain where there is no z displacement, and uP is the period of the sine wave. X, Y are the original coordinate and Z is the new coordinate generated after using the above formula. This Z gets multiplied to Model-View-Projection matrix and is used further.

Above formula is implemented by the following lines in the code,

```
vec4 V = aVertex;
V.z = uK * (Y0-V.y) * sin( 2.*PI*V.x/uP );
```

Next we find two vector along x and y axis and then compute cross product to get the vector normal to the surface i.e along z axis. The code which accomplish this is given below:

```
float dzdx = uK * (Y0-V.y) * (2.*PI/uP) * cos( 2.*PI*V.x/uP );
vec3 Tx = vec3( 1., 0., dzdx );
float dzdy = -uK * sin( 2.*PI*V.x/uP );
vec3 Ty = vec3( 0., 1., dzdy );
vec3 newNormal = normalize(  cross( Tx, Ty )  );
```

In the above code, the function cross(Tx,Ty) find the vector along z axis. We normalize this vector and then use it for lighting.

Now as I have vectors along x, y and z axis, I perturbed the normal by using following noise function which is readily available in glman. First I find noise vector using the noise function given in the code below. Next we find the angx, angy and rotate them using the function ***RotateNormal(float, float, vec3)*** given in the project description. Then we multiple the resultant with the normal matrix and normalize it. The code given below accomplish this.

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

float angx = nvx.r + nvx.g + nvx.b + nvx.a;        //  1. -> 3.
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( uNormalMatrix * Normal );
```

For lighting, I find Normal, Light position and Eye position first, then normalize them. Then I find the ambient light, diffuse light and Specular light by using the following code given below:

```
vNormalfrag = newNormal;
vLightfrag = eyePositionforLight - ECposition.xyz;
vEyefrag = vec3( 0., 0., 0. ) - ECposition.xyz;

Normal = normalize(vNormalfrag);
Light =  normalize(vLightfrag);
Eye =    normalize(vEyefrag);

vec4 ambientLight = uKa * uColor;
float d = max( dot(Normal,Light), 0. );
vec4 diffuseLight = uKd * d * 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 specularLight = uKs * s * uSpecularColor;
vPerVertexfrag = ambientLight.rgb + diffuseLight.rgb + specularLight.rgb;
```

Some of the results are shown below:
(Note: Parameters used to generate these results are shown in the figures)

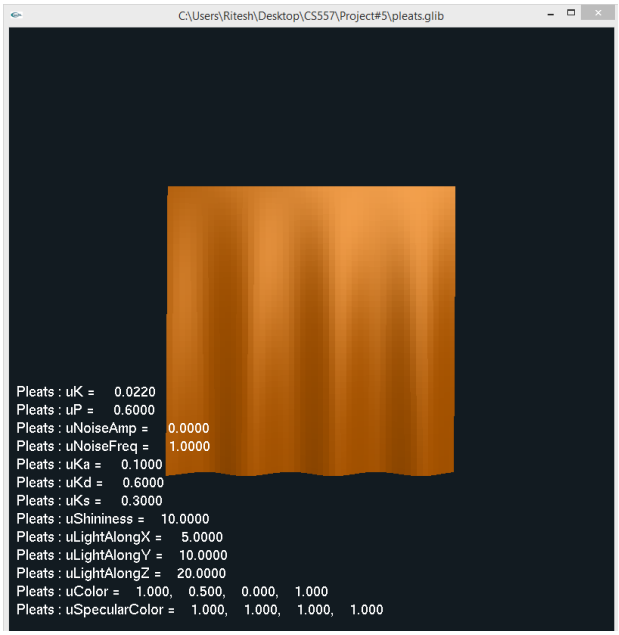

Figure 1: uK=0.0220

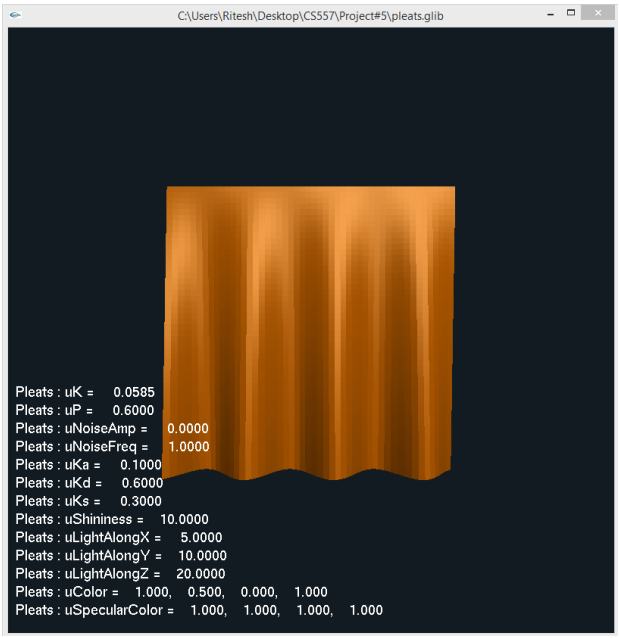

Figure 2: uK=0.0585

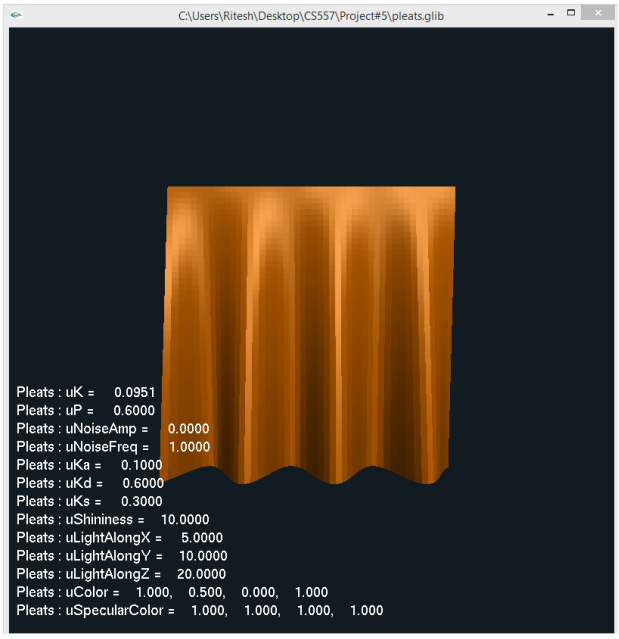

Figure 3: uK=0.0951

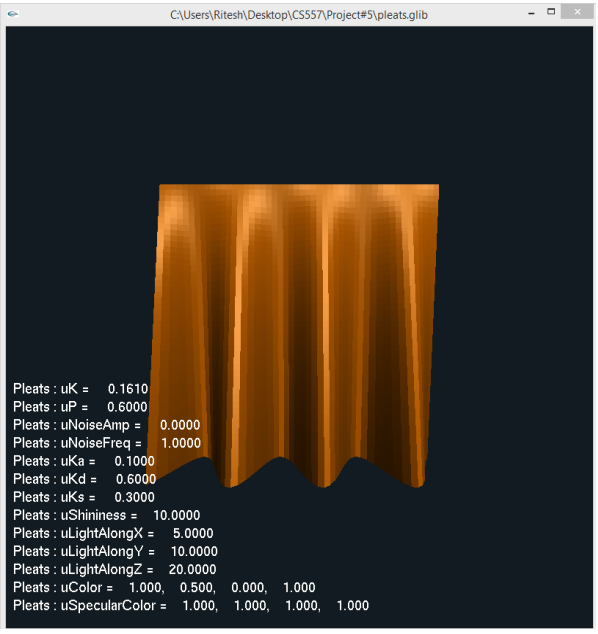

Figure 4: uK=0.1610

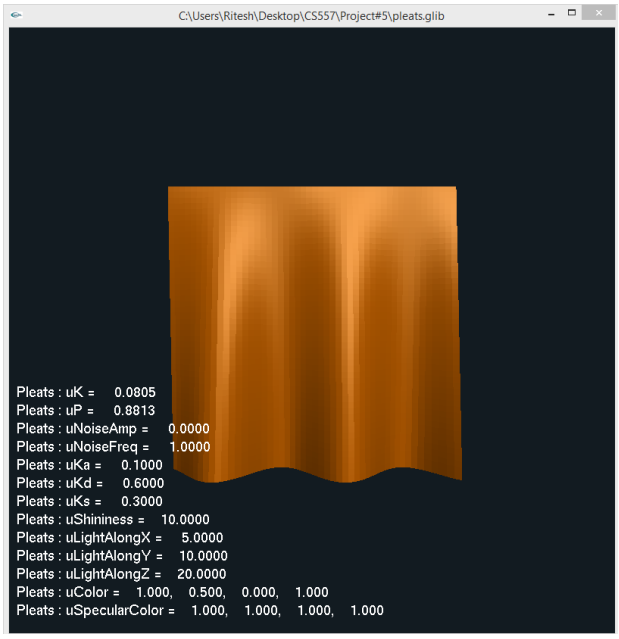

Figure 5: uP=0.8813

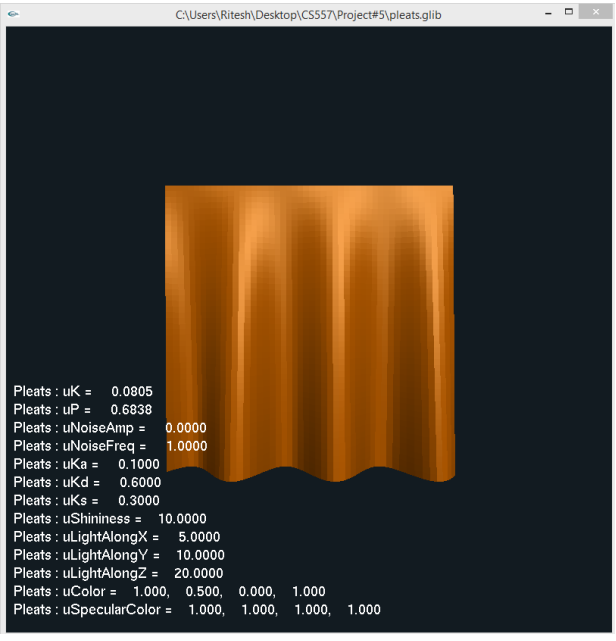

Figure 6: uP=0.6838

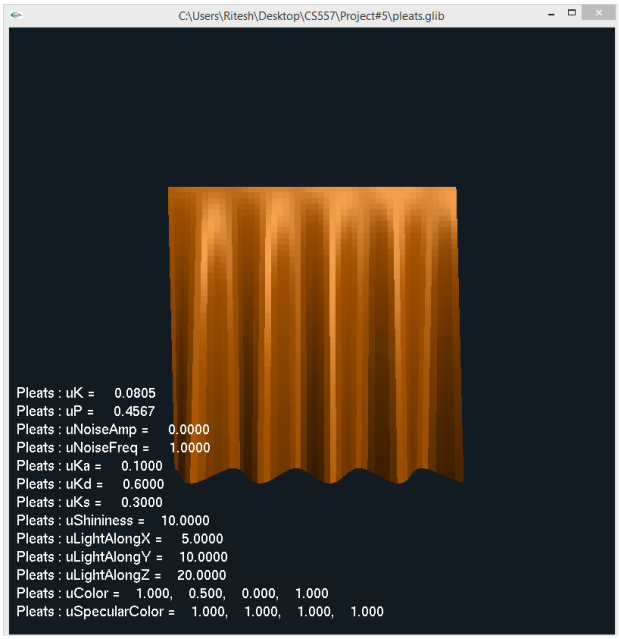

Figure 7: uP=0.4567

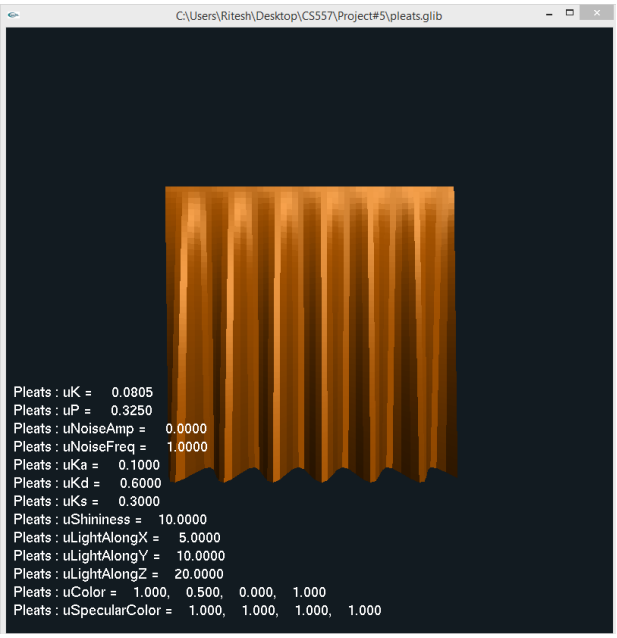

Figure 8: uP=0.3250

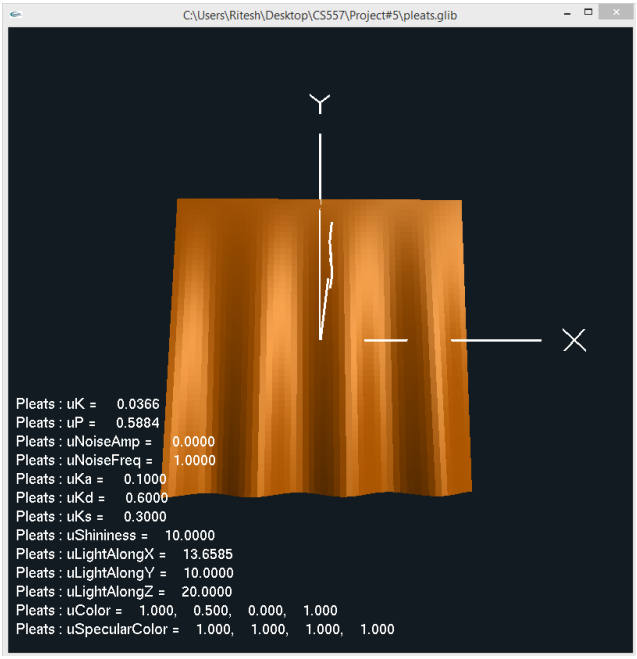

Figure 9: Showing lighting coming from X-axis

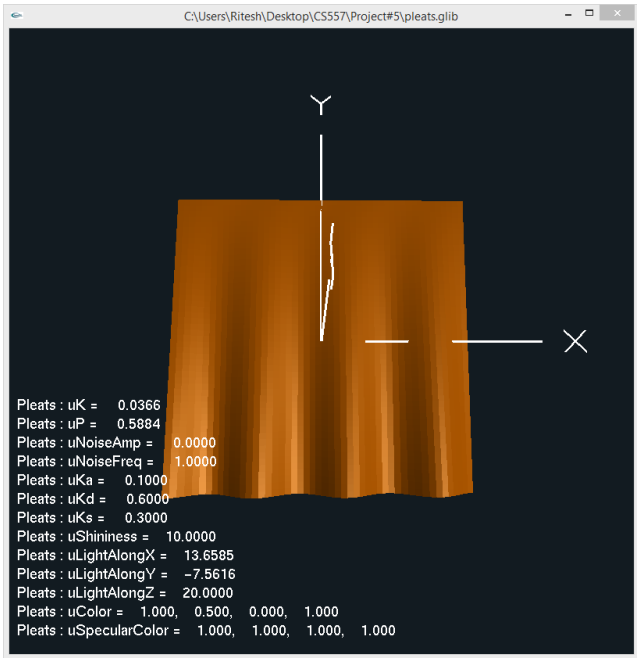

Figure 10: Showing lighting coming from Y-axis

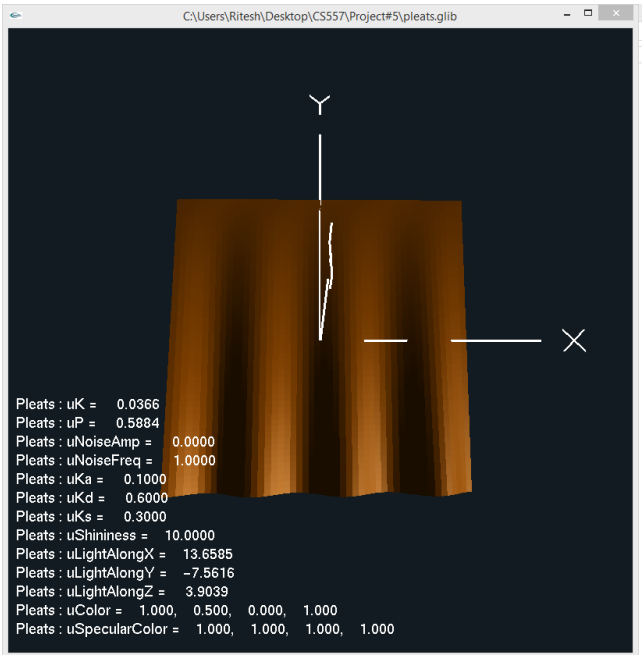

Figure 11: Showing lighting coming from Z-axis

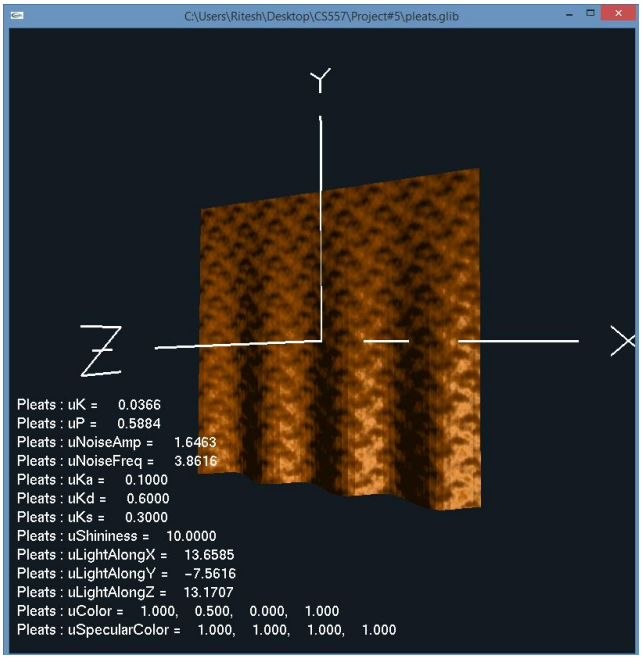

Figure 12: Showing Bump-Mapping