# CS 457/557-- Winter Quarter 2016

# Shaders Final Project

Instructor: Prof. Mike Bailey
Xuhao Zhou

**1. What I Did**

(1) This project draws different colors on the surface of the Canton Tower object.

(2) The program uses mix() function to blend the different colors and make it like a rainbow on the surface of the tower.

(3) The program offers sliders for adjusting the angle of the sections of color.

(4) The program implements lighting in the scene and offers sliders for adjusting the parameters of lighting.

(5) The program draws the scene in an earth Cubemap.

**2. Why It Worked**

(1) The tower has been sectioned into different sections according to the different height of the tower. Different section has different mixed color. We need to determinate that a vertex whether is in a section. If it is, draw the section's color. We use the math of Point-Plane Distance in adjusting the angle of the sections of color.

Below is an example with explanations.

For example:

```
vec3 pos = vec3(vMCposition.x,vMCposition.y,vMCposition.z); // position (vertex).


// NAngle is a normal vector of the sections, we use it to determinate the angle of the
sections of color

vec3 NAngle = vec3(uAngleX, uAngleY, uAngleZ);


// It is the section of pink color if the height (Z value) is less than 35.

float PinkHeight    = 35.;

vec3 PinkCenter    = vec3(0., 0., PinkHeight);


// PinkVector is a vector that from position (vertex) to PinkCenter

vec3 PinkVector    = PinkCenter - pos;


// If the result (PinkResult) of the dot product of PinkVector and NAngle is larger than
1, it means that the vertex are in the section of pink color. Otherwise, it is not.

float PinkResult = dot(normalize(PinkVector), normalize(NAngle));


// We use the math of Point-Plane Distance to calculate the distance of a vertex and a
plane. We can think that there is a plane named PlanePink that intercepts the tower.
```

Below this plane, it is the section of pink color. We can see NAngle as the normal vector of the plane PlanePink. –PinkVector is the vector from PinkCenter to pos. We can project the vector –PinkVector onto the normal vector NAngle to get the distance of a vertex and the plane PlanePink. Here, dPink is the distance. We use this dPink in the mix function.

```
float dPink    = abs(dot(NAngle, -PinkVector))/length(NAngle);


//if pos (points) is in the section pink ( PinkResult >= 0. ), mix two colors.

//When dPink = 0, it means it is on the plane PlanePink, it is RED color.

//When dPink = PinkHeight, it means it is far away the plane PlanePink, it is PINK color.

if ( PinkResult >= 0. )
{
       t = (PinkHeight - dPink) / PinkHeight;
       vColor = mix(PINK, RED,t );
}
```

(2) This program used below code to create an Earth Cubemap.

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

Texture2D  6  posx.bmp
QuadYZ 250. 250. 300 300
Texture2D  6  negx.bmp
QuadYZ -250. 250. 300 300
Texture2D  6  posy.bmp
QuadXZ 250. 250. 300 300
Texture2D  6  negy.bmp
QuadXZ -250. 250. 300 300
Texture2D  6  posz.bmp
QuadXY 250. 250. 300 300
Texture2D  6  negz.bmp
QuadXY -250. 250. 300 300

CubeMap 6 posx.bmp negx.bmp  posy.bmp negy.bmp   posz.bmp negz.bmp
```

(3) This program uses below code to implement the lighting.

```
vec3 Normal = normalize(vNs);
vec3 Light = normalize(vLs);
vec3 Eye = normalize(vEs);
vec4 ambient = uKa * vColor;
float d = max( dot(Normal,Light), 0. );

vec4 diffuse = uKd * d * vColor;
float s = 0.;
if( dot(Normal,Light) > 0. ) // only do specular if the light can see the point
{
       vec3 ref = normalize( 2. * Normal * dot(Normal,Light) - Light );
       s = pow( max( dot(Eye,ref),0. ), uShininess );
}
vec4 specular = uKs * s * WHITE;
gl_FragColor = vec4( ambient.rgb + diffuse.rgb + specular.rgb, 1. );
```

**3. Source Listings**

(1) CantonTower.glib

```
##OpenGL GLIB
Perspective 70
LookAt  0 70 150  0 75 0  0 1 0

Vertex   texture.vert
Fragment texture.frag
Program  Texture  TexUnit 6

Texture2D  6  posx.bmp
QuadYZ 250. 250. 300 300
Texture2D  6  negx.bmp
QuadYZ -250. 250. 300 300
Texture2D  6  posy.bmp
QuadXZ 250. 250. 300 300
Texture2D  6  negy.bmp
QuadXZ -250. 250. 300 300
Texture2D  6  posz.bmp
QuadXY 250. 250. 300 300
Texture2D  6  negz.bmp
QuadXY -250. 250. 300 300

CubeMap 6 posx.bmp negx.bmp  posy.bmp negy.bmp   posz.bmp negz.bmp

Vertex   CantonTower.vert
Fragment CantonTower.frag
Program  CantonTower       \
                     uAngleX <-0.7  0.  0.7>        \
                     uAngleY <-0.7 0.7 0.7>         \
                     uAngleZ < 0.0  1.  1.>         \
                     uLightX <-250. 50. 250.>       \
                      uLightY <-250. 50. 250.>       \
                      uLightZ <-250. 250. 250.>      \
                     uKa <0. 0.4 1.0>               \
                     uKd <0. 0.7 1.0>               \
                     uKs <0. 0.7 1.0>               \
                     uShininess <1. 100. 100.>

Rotate 90 0 -1 0
Rotate 90 -1 0 0
Obj CantonTower.obj
```

(2) CantonTower.vert

```
#version 400 compatibility
out vec3 vNs;
out vec3 vLs;
out vec3 vEs;
out vec3 vMCposition;
uniform float uLightX, uLightY, uLightZ, uA, uB, uT;
vec3 eyeLightPosition = vec3( uLightX, uLightY, uLightZ );

void main( )
{
```

```
	vec3 ECposition = vec3( gl_ModelViewMatrix * gl_Vertex );
	vNs = normalize( gl_NormalMatrix * gl_Normal );
	vLs = eyeLightPosition - ECposition.xyz;
	vEs = vec3( 0., 0., 0. ) - ECposition.xyz;
	gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;

	vMCposition = gl_Vertex.xyz;
}
```

(3) CantonTower.frag

```
#version 400 compatibility
in vec3 vNs;
in vec3 vLs;
in vec3 vEs;
in vec3 vMCposition;
const vec4 PINK      = vec4( 0.804, 0.416, 0.925, 1. );
const vec4 LPURPLE   = vec4( 0.765, 0.451, 0.906, 1. );
const vec4 PURPLE    = vec4( 0.525, 0.180, 0.753, 1. );
const vec4 RED       = vec4( 1.000, 0.000, 0.000, 1. );
const vec4 ORANGE    = vec4( 1.000, 0.500, 0.000, 1. );
const vec4 YELLOW    = vec4( 1.000, 1.000, 0.000, 1. );
const vec4 GREEN     = vec4( 0.000, 1.000, 0.000, 1. );
const vec4 LBLUE     = vec4( 0.000, 1.000, 1.000, 1. );
const vec4 BLUE      = vec4( 0.000, 0.000, 1.000, 1. );
const vec4 DBLUE     = vec4( 0.400, 0.247, 0.788, 1. );
const vec4 WHITE     = vec4( 1.000, 1.000, 1.000, 1. );
uniform float uKa, uKd, uKs, uShininess, uAngleX, uAngleY, uAngleZ;

void main( )
{
	float t;
	vec4 vColor;
	vec3 pos = vec3(vMCposition.x,vMCposition.y,vMCposition.z);
	vec3 NAngle = vec3(uAngleX, uAngleY, uAngleZ);
	float PinkHeight     = 35.;
	float RedHeight      = 37.;
	float OrangeHeight   = 57.;
	float YellowHeight   = 71.;
	float GreenHeight    = 75.;
	float LblueHeight    = 90.;
	float BlueHeight     = 100.;
	float DblueHeight    = 110.;
	float LpurpleHeight  = 130.;
	float PurpleHeight   = 180.;
	vec3 PinkCenter      = vec3(0., 0., PinkHeight);
	vec3 RedCenter       = vec3(0., 0., RedHeight);
	vec3 OrangeCenter    = vec3(0., 0., OrangeHeight);
	vec3 YellowCenter    = vec3(0., 0., YellowHeight);
	vec3 GreenCenter     = vec3(0., 0., GreenHeight);
	vec3 LblueCenter     = vec3(0., 0., LblueHeight);
	vec3 BlueCenter      = vec3(0., 0., BlueHeight);
	vec3 DblueCenter     = vec3(0., 0., DblueHeight);
	vec3 LpurpleCenter   = vec3(0., 0., LpurpleHeight);
	vec3 PurpleCenter    = vec3(0., 0., PurpleHeight);
	vec3 PinkVector      = PinkCenter - pos;
```

```
	vec3 RedVector       = RedCenter - pos;
	vec3 OrangeVector    = OrangeCenter - pos;
	vec3 YellowVector    = YellowCenter - pos;
	vec3 GreenVector     = GreenCenter - pos;
	vec3 LblueVector     = LblueCenter - pos;
	vec3 BlueVector      = BlueCenter - pos;
	vec3 DblueVector     = DblueCenter - pos;
	vec3 LpurpleVector   = LpurpleCenter - pos;
	vec3 PurpleVector    = PurpleCenter - pos;
	float dPink          = abs(dot(NAngle, -PinkVector))/length(NAngle);
	float dRed           = abs(dot(NAngle, -RedVector))/length(NAngle);
	float dOrange        = abs(dot(NAngle, -OrangeVector))/length(NAngle);
	float dYellow        = abs(dot(NAngle, -YellowVector))/length(NAngle);
	float dGreen         = abs(dot(NAngle, -GreenVector))/length(NAngle);
	float dLblue         = abs(dot(NAngle, -LblueVector))/length(NAngle);
	float dBlue          = abs(dot(NAngle, -BlueVector))/length(NAngle);
	float dDblue         = abs(dot(NAngle, -DblueVector))/length(NAngle);
	float dLpurple       = abs(dot(NAngle, -LpurpleVector))/length(NAngle);
	float dPurple        = abs(dot(NAngle, -PurpleVector))/length(NAngle);
	float PinkResult     = dot(normalize(PinkVector), normalize(NAngle));
	float RedResult      = dot(normalize(RedVector), normalize(NAngle));
	float OrangeResult   = dot(normalize(OrangeVector), normalize(NAngle));
	float YellowResult   = dot(normalize(YellowVector), normalize(NAngle));
	float GreenResult    = dot(normalize(GreenVector), normalize(NAngle));
	float LblueResult    = dot(normalize(LblueVector), normalize(NAngle));
	float BlueResult     = dot(normalize(BlueVector), normalize(NAngle));
	float DblueResult    = dot(normalize(DblueVector), normalize(NAngle));
	float LpurpleResult  = dot(normalize(LpurpleVector), normalize(NAngle));
	float PurpleResult   = dot(normalize(PurpleVector), normalize(NAngle));
	if ( PinkResult >= 0. )
	{
		t = (PinkHeight - dPink) / PinkHeight;
		vColor = mix(PINK, RED,t );
	}
	else if ( RedResult >= 0. )
	{
		t = ((RedHeight - PinkHeight ) - dRed) / ( RedHeight - PinkHeight);
		vColor = mix(RED, ORANGE,t );
	}
	else if( OrangeResult >= 0. )
	{
		t = ((OrangeHeight - RedHeight) - dOrange) / (OrangeHeight - RedHeight);
		vColor = mix(ORANGE, YELLOW, t );
	}
	else if( YellowResult >= 0. )
	{
		t = ((YellowHeight - OrangeHeight) - dYellow) / (YellowHeight -
OrangeHeight);
		vColor = mix(YELLOW, GREEN, t );
	}
	else if( GreenResult >= 0. )
	{
		t = ((GreenHeight - YellowHeight) - dGreen) / (GreenHeight - YellowHeight);
		vColor = mix(GREEN, LBLUE, t );
	}
	else if( LblueResult >= 0. )
	{
		t = ((LblueHeight - GreenHeight) - dLblue ) / (LblueHeight - GreenHeight);
```

```
		vColor = mix(LBLUE, BLUE, t );
	}
	else if( BlueResult >= 0. )
	{
		t = ((BlueHeight - LblueHeight) - dBlue) /(BlueHeight - LblueHeight);
		vColor = mix(BLUE, DBLUE, t );
	}
	else if( DblueResult >= 0. )
	{
		t = ((DblueHeight - BlueHeight) - dDblue) / (DblueHeight - BlueHeight);
		vColor = mix(DBLUE, LPURPLE, t );
	}
	else if( LpurpleResult >= 0. )
	{
		t = ((LpurpleHeight - DblueHeight) - dLpurple) / (LpurpleHeight -
DblueHeight);
		vColor = mix(LPURPLE, PURPLE, t );
	}
	else if(  PurpleResult >= 0. )
	{
		vColor = PURPLE;
	}
	vec3 Normal = normalize(vNs);
	vec3 Light = normalize(vLs);
	vec3 Eye = normalize(vEs);
	vec4 ambient = uKa * vColor;
	float d = max( dot(Normal,Light), 0. );
	vec4 diffuse = uKd * d * vColor;
	float s = 0.;
	if( dot(Normal,Light) > 0. ) // only do specular if the light can see the point
	{
		vec3 ref = normalize( 2. * Normal * dot(Normal,Light) - Light );
		s = pow( max( dot(Eye,ref),0. ), uShininess );
	}
	vec4 specular = uKs * s * WHITE;
	gl_FragColor = vec4( ambient.rgb + diffuse.rgb + specular.rgb, 1. );
}
```

(4) texture.vert

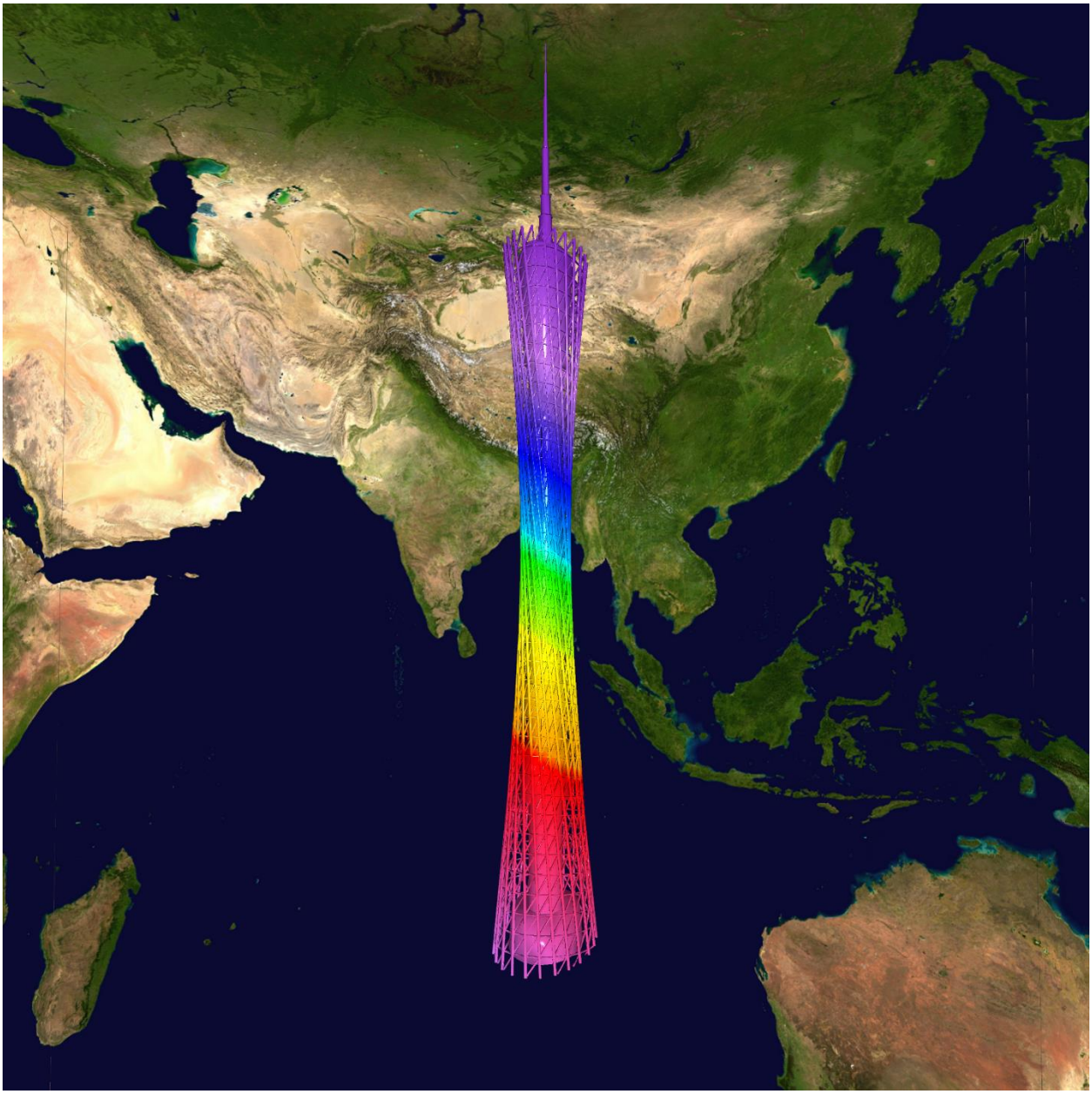
```
void main()
{
	gl_TexCoord[0] = gl_MultiTexCoord0;
	gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}
```

(5) texture.frag

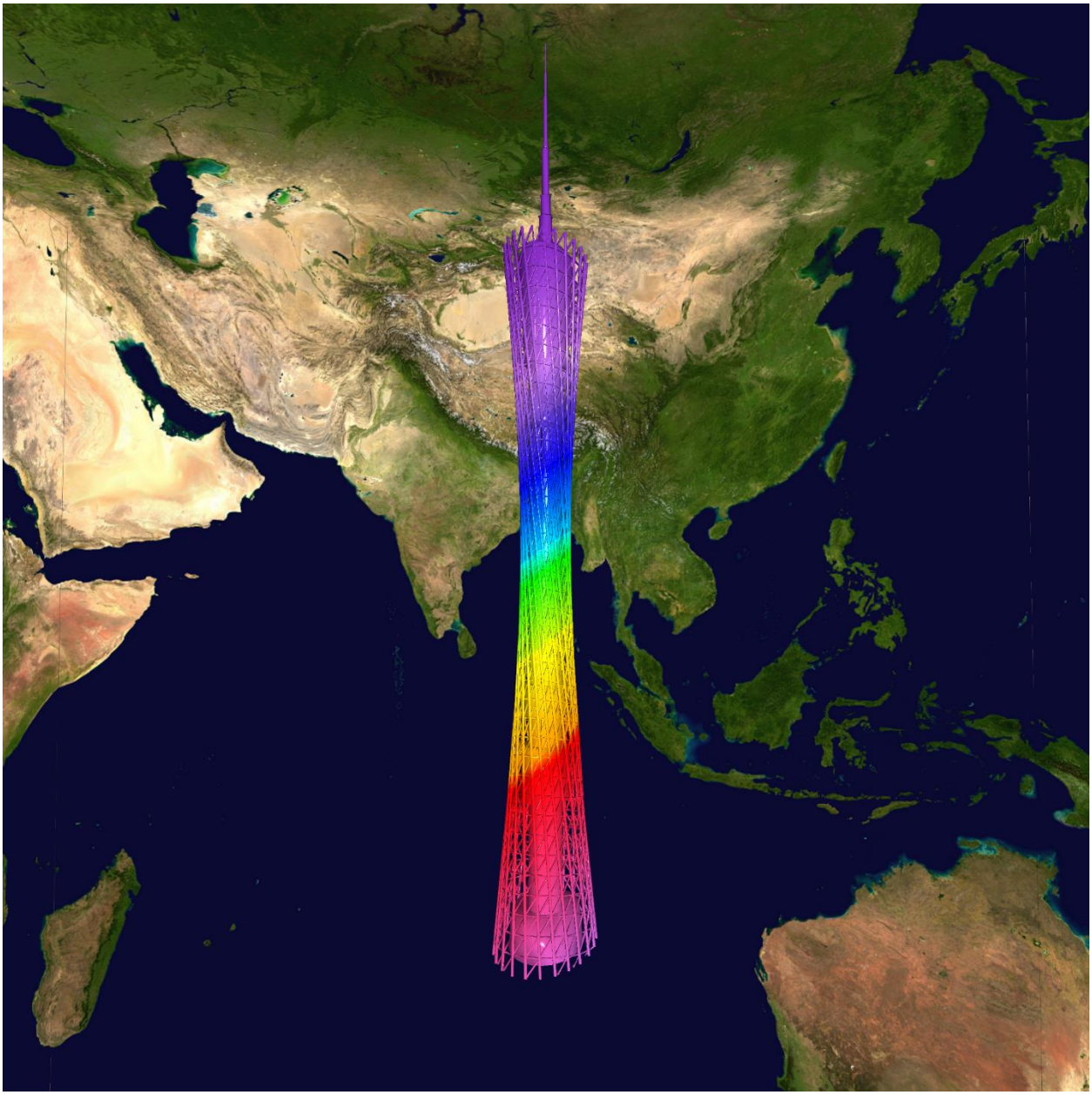
```
uniform sampler2D TexUnit;

void main()

{

	vec3 newcolor = texture2D( TexUnit, gl_TexCoord[0].st ).rgb;

	gl_FragColor = vec4( newcolor.rgb, 1. );

}
```

**4. Images**

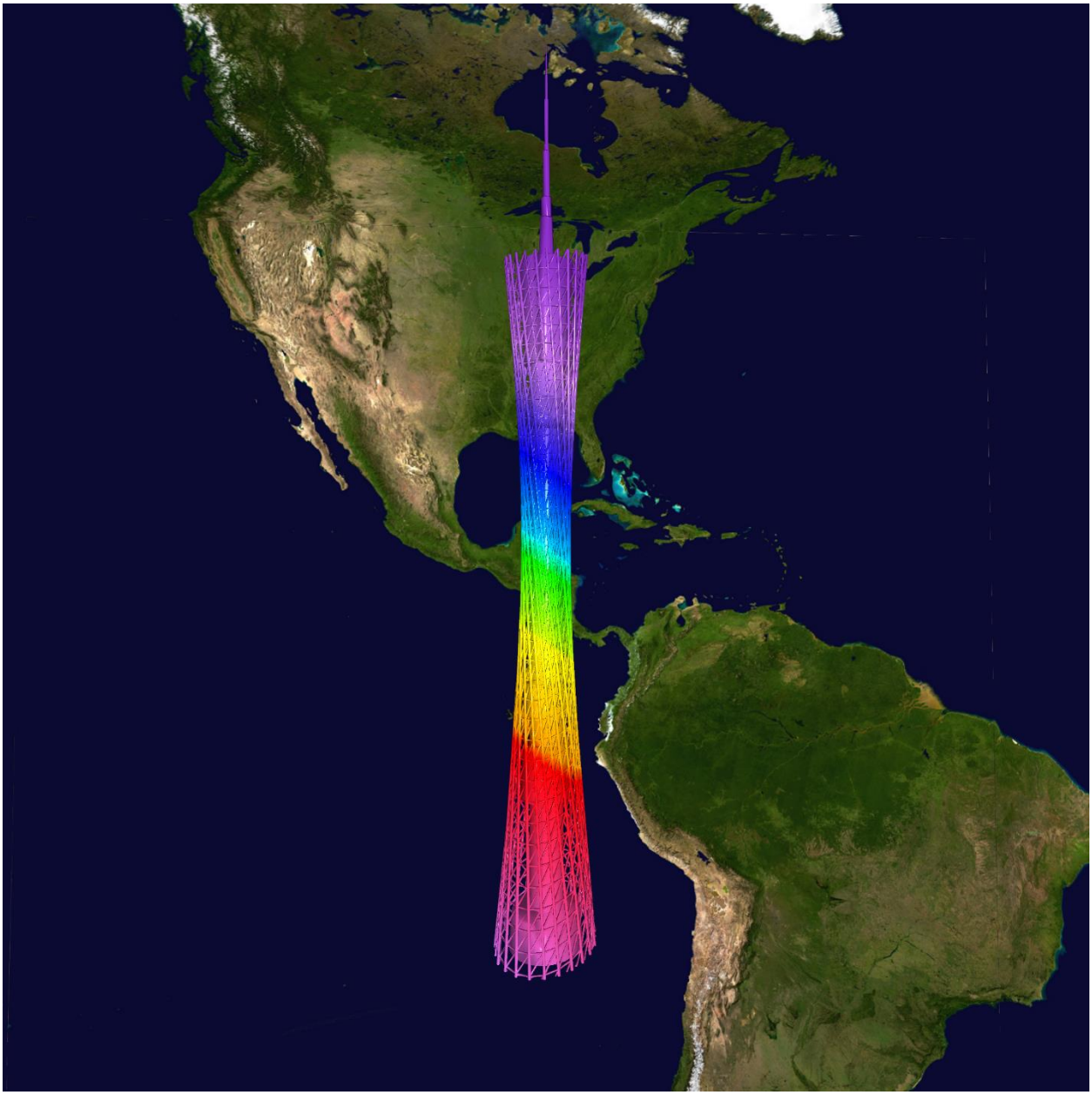
(1) Negative slope



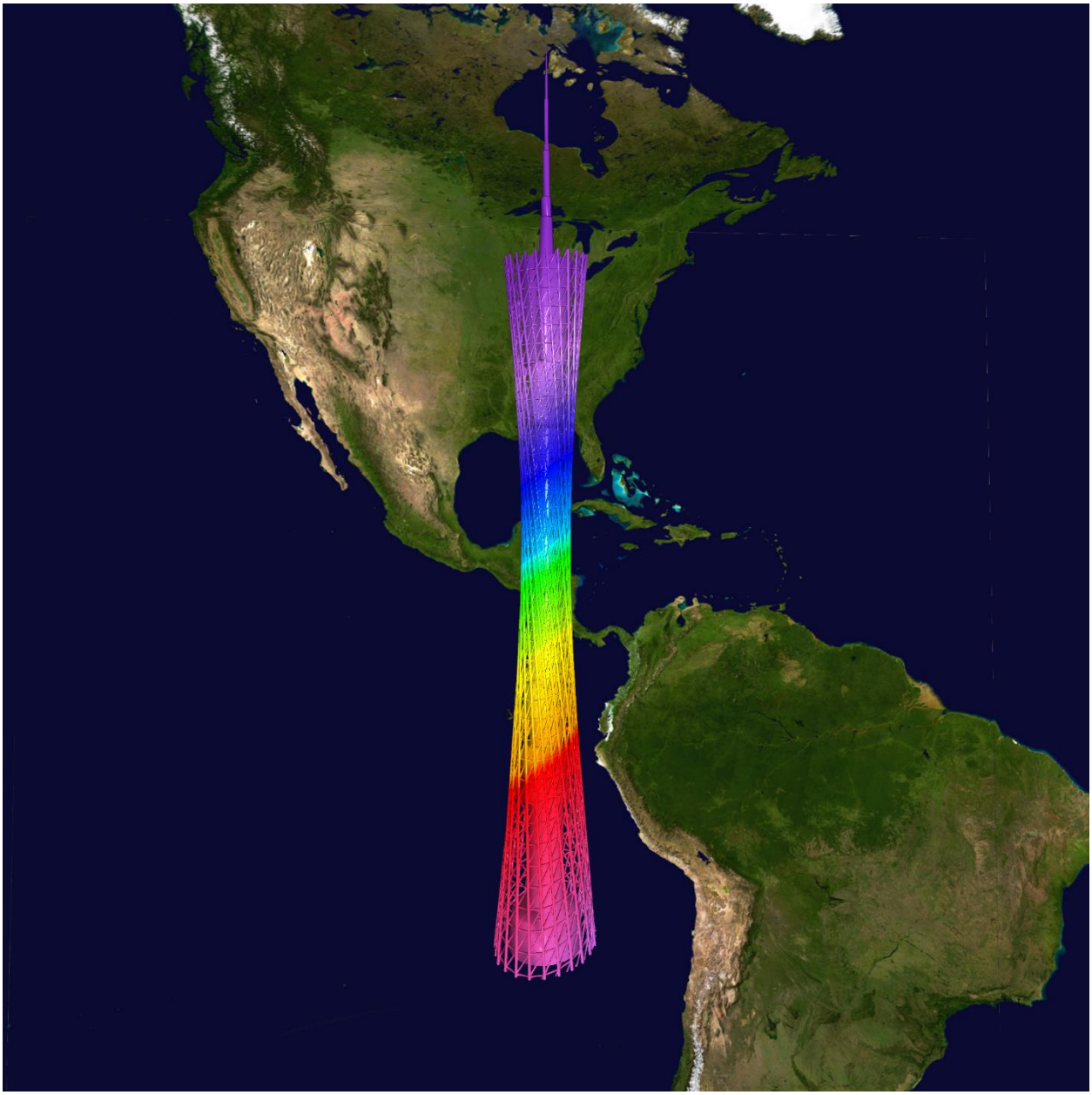
(2) Positive slope



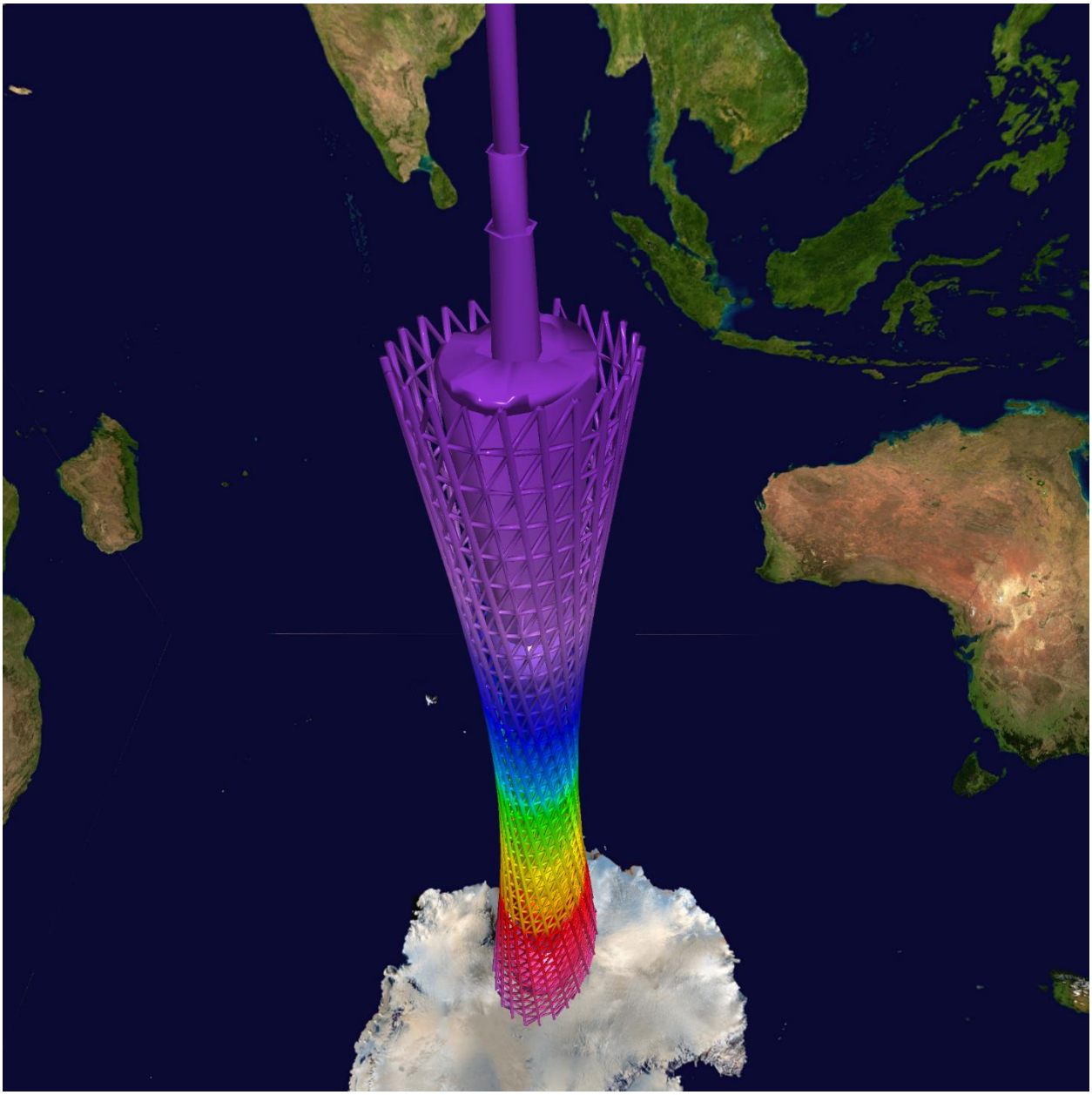
(3) Negative slope



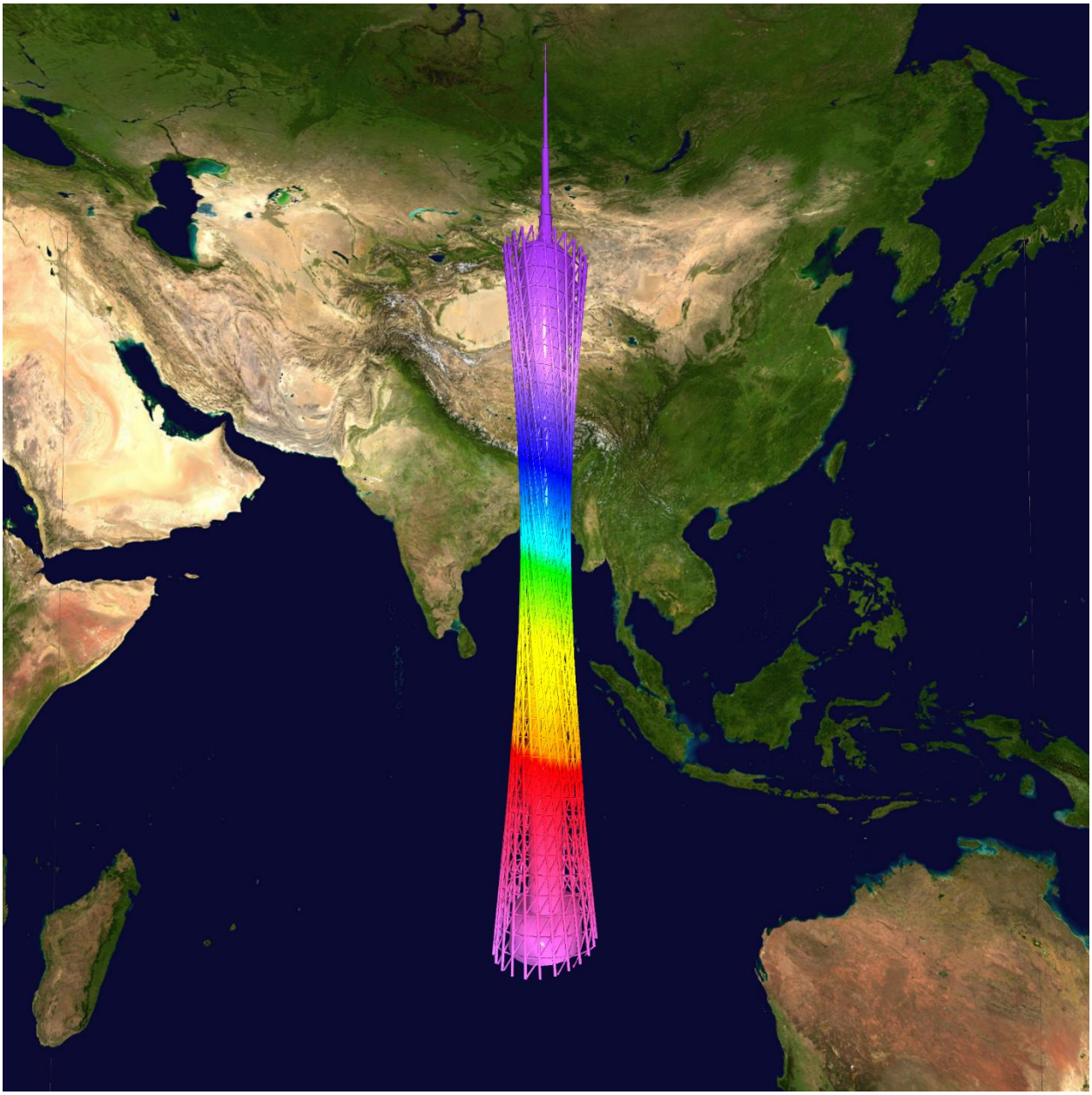
(4) Positive slope



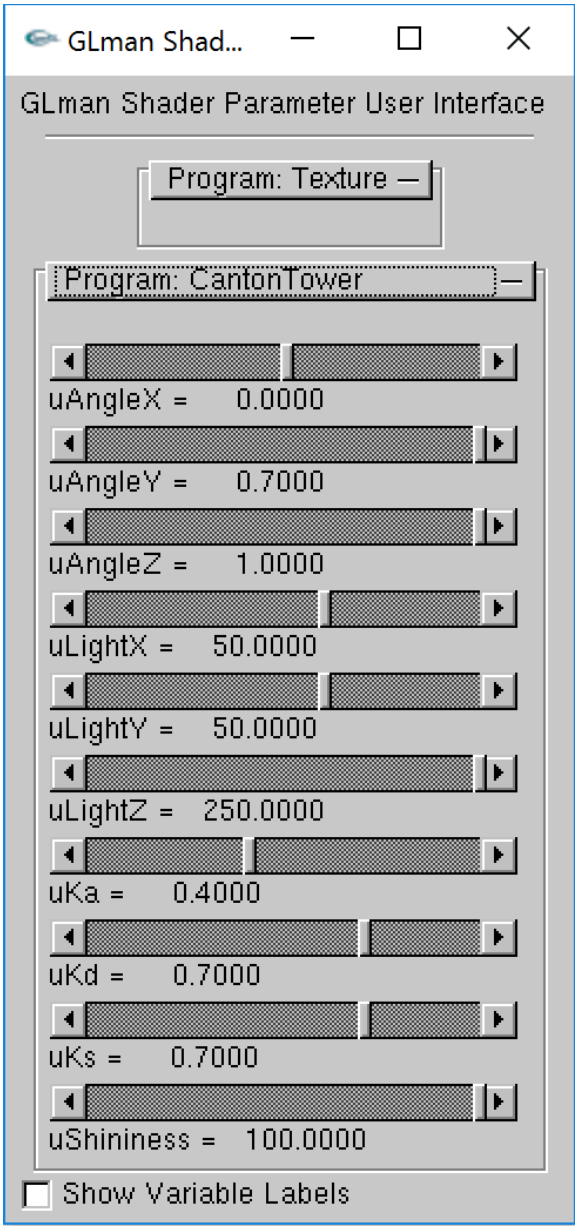
(5)



(6)



(7) Sliders


GLman Shad...
GLman Shader Parameter User Interface
Program: Texture
Program: CantonTower
uAngleX = 0.0000
uAngleY = 0.7000
uAngleZ = 1.0000
uLightX = 50.0000
uLightY = 50.0000
uLightZ = 250.0000
uKa = 0.4000
uKd = 0.7000
uKs = 0.7000
uShininess = 100.0000
Show Variable Labels

Acknowledgements

Thanks for the assistance from Prof. Mike Bailey and the TA Chris Schultz because they provided precious suggestions and led me on the right track.