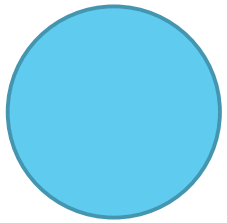
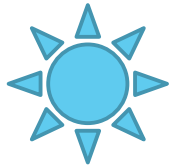
openGL – Lighting and Material

Example

(0, 0, 10)

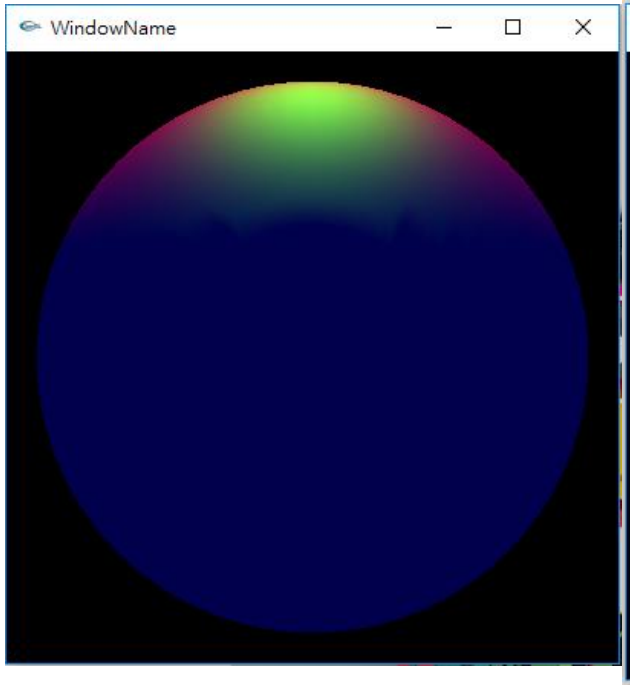


(0, 0, 0)



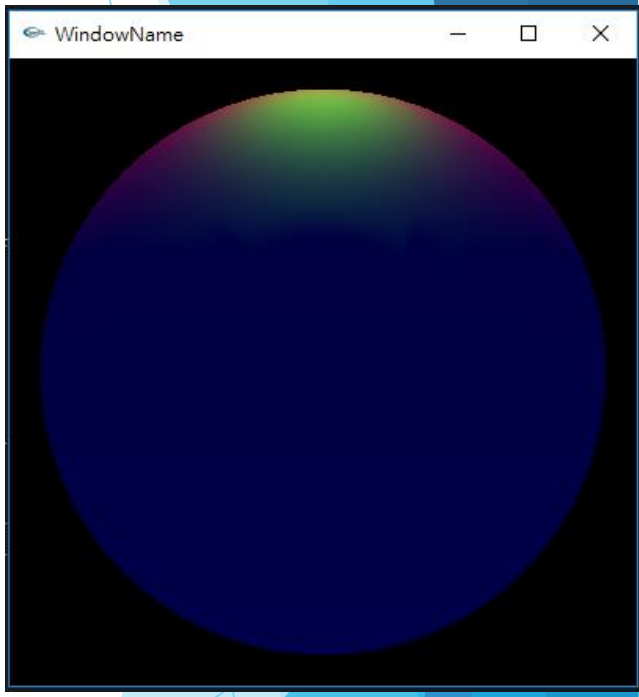
(0, 10, 0)

Point light



Directional light

Spotlight



Example

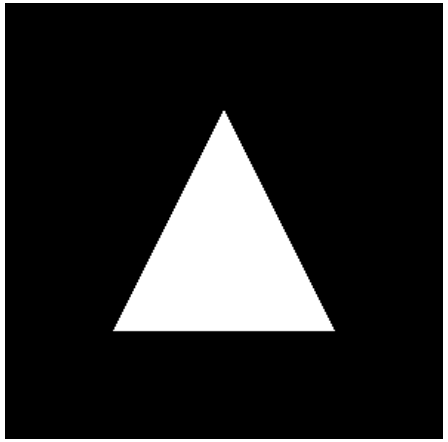
```
17  int cutoff = 0, exponent = 0;
18  int lighting_mode = 0;  //0:point light, 1:direcitonal light, 2: spotlight
```

```
40  void display()
41  {
42      //ModelView Matrix
43      glMatrixMode(GL_MODELVIEW);
44      glLoadIdentity();
45      gluLookAt(0.0f, 0.0f, 10.0f, 0.0f, 0.0f, 0.0f, 0.0f, 1.0f, 0.0f);
46      //Projection Matrix
47      glMatrixMode(GL_PROJECTION);
48      glLoadIdentity();
49      gluPerspective(45, width / (GLfloat)height, 0.1, 1000);
50      //Viewport Matrix
51      glViewport(0, 0, width, height);
52
53      //clear
54      glClear(GL_COLOR_BUFFER_BIT);
55      glClear(GL_DEPTH_BUFFER_BIT);
56
57      glMatrixMode(GL_MODELVIEW);
58      lighting(lighting_mode);
59
60      float red[] = { 1.0f, 0.0f, 0.0f, 1.0f };
61      float green[] = { 0.0f, 1.0f, 0.0f, 1.0f };
62      float blue [] = { 0.0f, 0.0f, 1.0f, 1.0f };
63
64      glPushMatrix();
65      //glRotatef(degree, 0, 1, 0);
66      glMaterialfv(GL_FRONT, GL_DIFFUSE, red);
67      glMaterialfv(GL_FRONT, GL_SPECULAR, green);
68      glMaterialfv(GL_FRONT, GL_AMBIENT, blue);
69      glMaterialf(GL_FRONT, GL_SHININESS, 10);
70      glutSolidSphere(3.5, 72, 36);
71      glPopMatrix();
72
73      glutSwapBuffers();
74  }
```

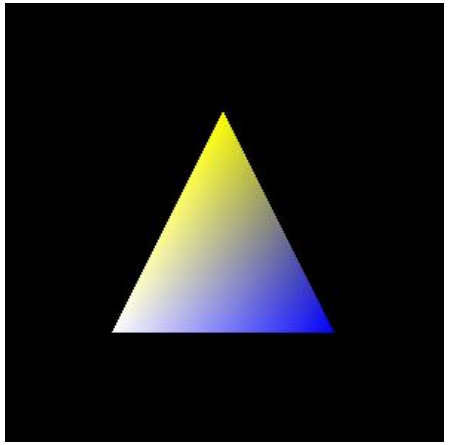
```
127 void lighting(int mode) {
128     float position[] = { 0.0f, 10.0f, 0.0f, 1.0f };
129     if (mode == 1) {    //directional light
130         position[3] = 0.0f;
131     }
132     float specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };
133     float diffuse[] = { 1.0f, 1.0f, 1.0f, 1.0f };
134     float ambient[] = { 0.1f, 0.1f, 0.1f, 1.0f };
135     float none[] = { 0.0f, 0.0f, 0.0f, 1.0f };
136
137     glEnable(GL_LIGHT0);
138     glLightfv(GL_LIGHT0, GL_POSITION, position);
139     glLightfv(GL_LIGHT0, GL_SPECULAR, specular);
140     glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
141     glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);
142
143     if (mode == 2) {    //spot light
144         float direction[] = { 0.0f, -1.0f, 0.0f, 1.0f };
145         glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, direction);
146         glLighti(GL_LIGHT0, GL_SPOT_EXPONENT, exponent);
147         glLighti(GL_LIGHT0, GL_SPOT_CUTOFF, cutoff);
148     }
149     else {
150         glLighti(GL_LIGHT0, GL_SPOT_CUTOFF, 180);   //close spotlight
151     }
152 }
153
```

Shading model

- void glShadeModel(GLenum mode);
  - mode: GL_FLAT / GL_SMOOTH (default)



GL_FLAT



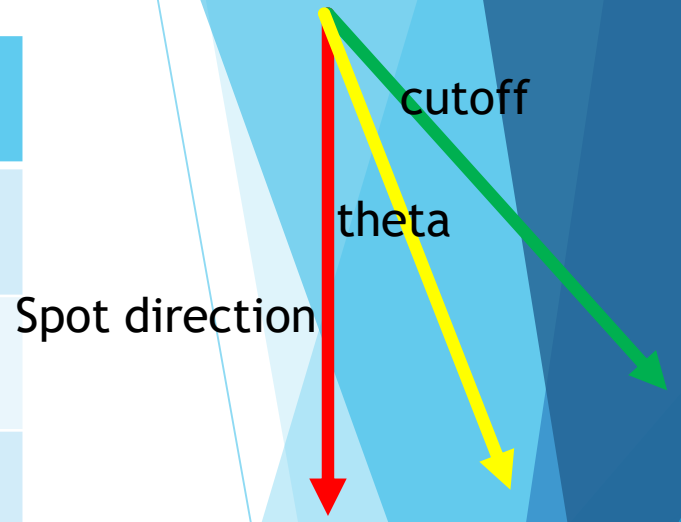
GL_SMOOTH

Lighting

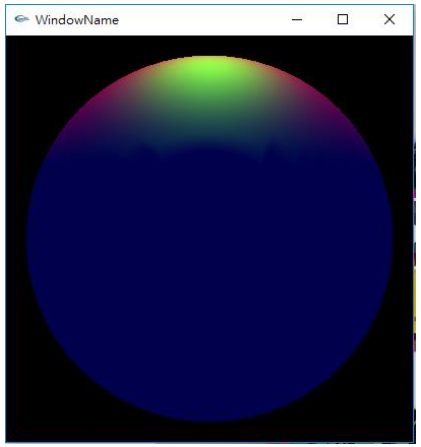
- glEnable(GL_LIGHTING);
- glEnable(GL_LIGHTi);
  - i = 0 - GL_MAX_LIGHT-1
- void glLight[fi](GLenum light, GLenum pname, GLfloat param)
  - light: specifies a light. There are at least eight lights are supported in OpenGL.
  - pname: specifies a light source parameter for light.
    - GL_SPOT_EXPONENT, GL_SPOT_CUTOFF,
    - GL_CONSTANT_ATTENUATION, GL_LINEAR_ATTENUATION, GL_QUADRATIC_ATTENUATION
  - param: specifies the value that parameter pname of light source light will be set to.
- void glLight[fi][v](GLenum light, GLenum pname, const GLfloat* param)
  - pname: specifies a light source parameter for light.
    - GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_POSITION,
    - GL_SPOT_DIRECTION

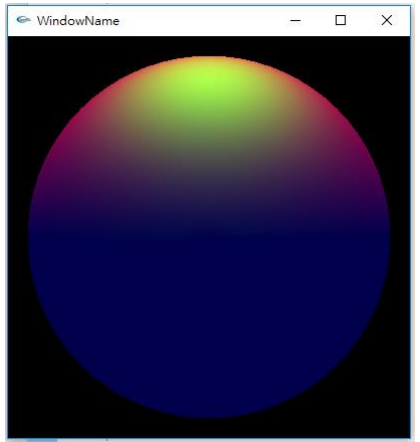
Light type

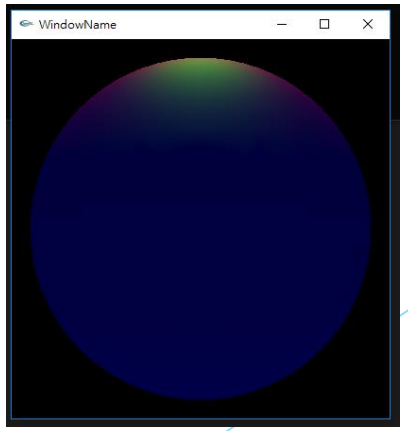
	Point light	Directional light	Spotlight
GL_POSITION	(x, y, z, w)	(x, y, z, 0)	(x, y, z, w)
Distance	Position to vertex	Ignore attenuation	Position to vertex
Direction	Position to vertex	(x, y, z) to origin	GL_SPOT_DIRECTION
			GL_SPOT_EXPONENT GL_SPOT_CUTOFF



Cos(theta)^exponent







Attenuation

- d: distance between the light's position and the vertex
- K_c = GL_CONSTANT_ATTENUATION
- K_l = GL_LINEAR_ATTENUATION
- K_q = GL_QUADRATIC_ATTENUATION
- If light is directional light, the attenuation is always 1

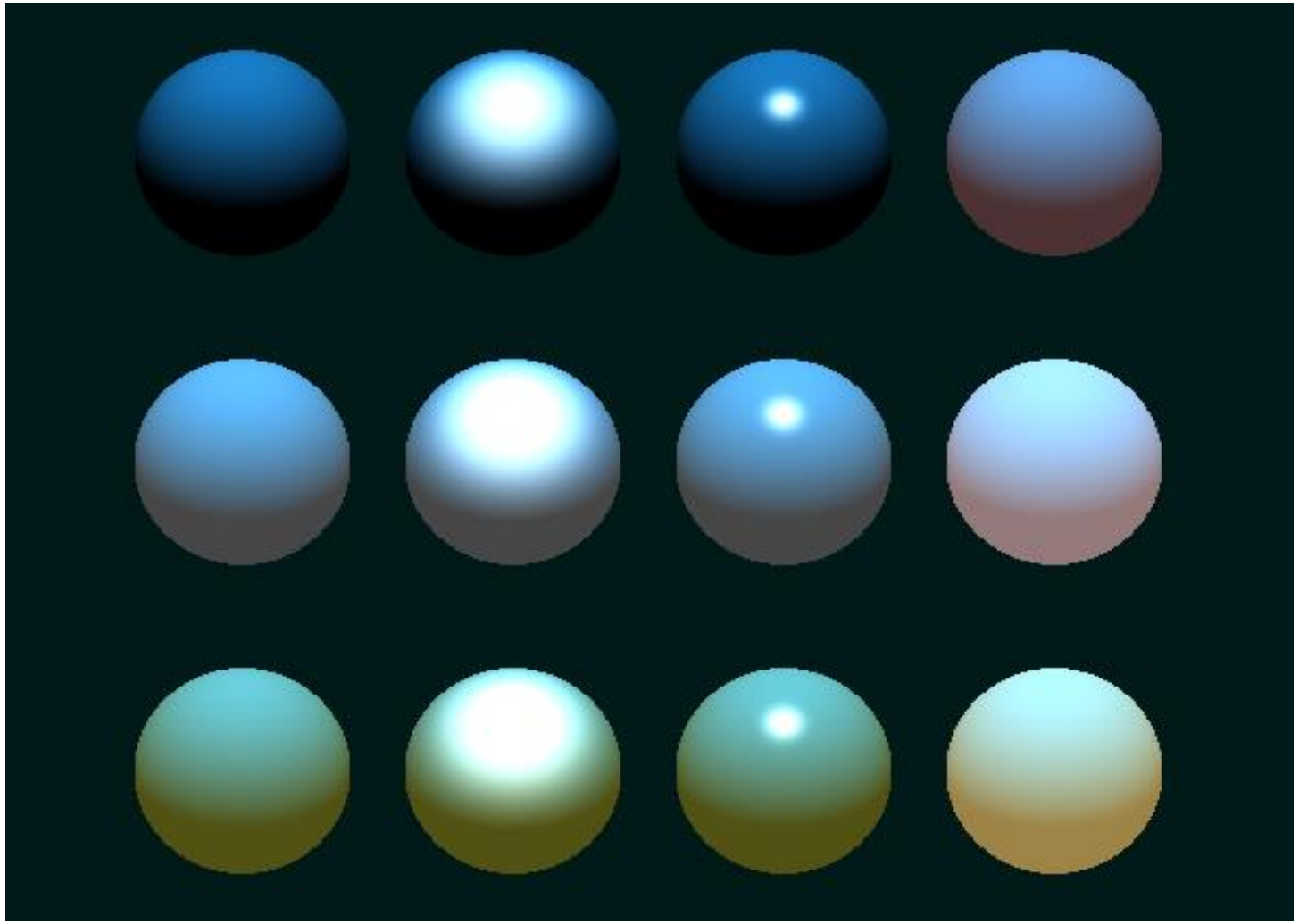
- void glLight[fi](GLenum light, GLenum pname, GLfloat param)
  - Pname: GL_CONSTANT_ATTENUATION, GL_LINEAR_ATTENUATION, GL_QUADRATIC_ATTENUATION

$$AttenuationFacror = \frac{1}{k_c + k_l d + k_q d^2}$$

Material

- void glMaterial{if}(GLenum face, GLenum pname, TYPE* param);
  - face: GL_FRONT, GL_BACK, GL_FRONT_AND_BACK
  - Pname: GL_SHININESS
- void glMaterial{if}[v](GLenum face, GLenum pname, TYPE* param);
  - face: GL_FRONT, GL_BACK, GL_FRONT_AND_BACK
  - Panme: GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_SHININESS, GL_EMISSION

Material



No Ambient

Gray Ambient

Blue Ambient

Diffuse Only

Specular

Higher Shininess

Emission

Normal

- void glNormal{34}{isfd}[v](TYPE* normal);
  - normal: normal vector of vertex
  - The normal vector should be assigned before you assign vertex
- Normal vectors must be normalize. OpenGL can automatically normalize normal vector by glEnable(GL_NORMALIZE)