Kingdom of Saudi Arabia
المملكة العربية السعودية
Ministry of Education
وزارة التعليم
Umm AlQura University
جامعة أم القرى
Adham University College
الكلية الجامعية بأضم
Computer Science Department
قسم الحاسب الآلي

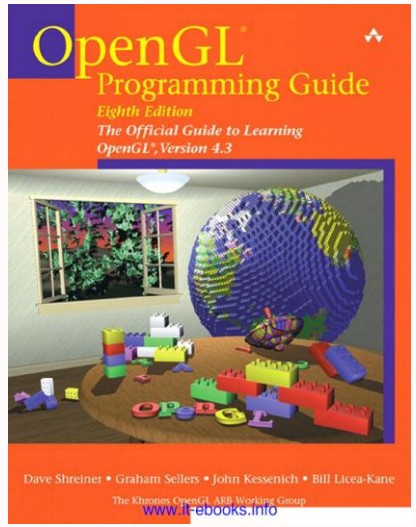
CS Department

Computer Graphics Course, 3-6803430

T.Mariah Khayat

References

- *Lab Lectures, Computer Graphics, Taif University, Faculty Of Computers And Information Technology, TA. Maha Thafar &TA. Haifa Alshehri, TA.Sohair Soliman & L.Shakila Bano.*
- *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 4.3, 8th edition, Dave Shreiner, Graham Sellers, John Kessenich, Bill Licea-Kane & The Khronos OpenGL ARB Working Group, Addison-Wesley.*



Lecture Nine

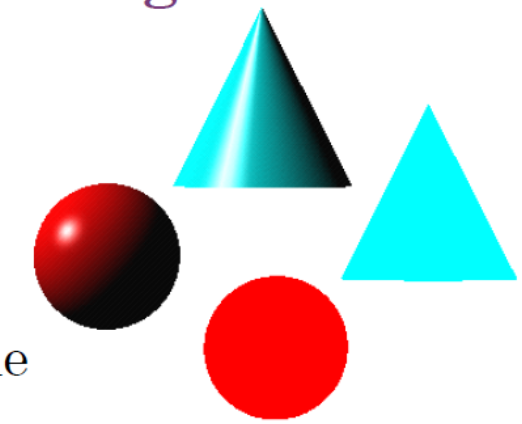
Lighting in OpenGL

content

1. Lighting Principles
2. Types of Lights
3. Light Source
4. Lights Prosperities
5. Material Prosperities
6. A Lighting Program using OpenGL
7. A Snow man program using OpenGL

Lighting Principles

- **Lighting simulates how objects reflect light**
  - material composition of object.
  - light's color and position
  - global lighting parameters
    - ambient light
    - two sided lighting
  - available in both color index and RGBA mode



- **How to apply lighting in OpenGL?**
  - **Assign** Normals to vertices.
    Normals define how a surface reflects light `glNormal3f( x,y,z).`
  - **Create**, **position** and **enable** lights.
  - **Select** a lighting model
  - **Define** the material properties of objects in the scene

Types of Lights

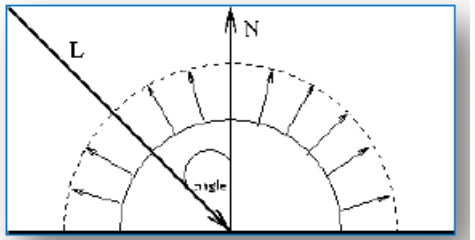
- there are 3 main types of light:

1) Ambient:

- Light scattered by environment.
- Unable to determine direction.
- Based on surfaces in scene

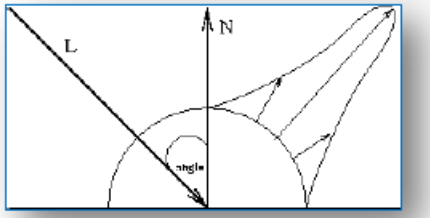
2) Diffuse:

- Coming from one direction.
- Scattered equally in all directions from surface



2) Specular:

- Coming from particular direction.
- Bounce off the surface in particular direction
- Based on law of reflection
- Specularity = shininess



Lights Source

- To Turn on the power of the light use:

```
glEnable(GL_LIGHTING);
```

and Can be **disabled** at any time.

```
glDisable( GL_LIGHTING);
```

- Enable the depth:

```
glEnable(GL_DEPTH_TEST);
```

- Enable and set properties for light sources:
  - Use `glEnable(GL_LIGHT0);` to enable light zero. You can use at most 8 light sources in OpenGL spec. (`GL_LIGHT0` → `GL_LIGHT7`)
  - Use:

```
glLight{if}[v](GLenum light, Glenum pname, TYPE value);
```

and Specify the attribute of light:

- ***light:*** can be `GL_LIGHT0 ~ GL_LIGHT7`
- ***pname :*** (Parameter name) is the characteristic (**property**) of the light
- ***value:*** is the value of pname.

  - *Example:*

```
GLfloat  sourceLight[] = { 0.25, 0.25, 0.25, 1.0 };
glLightfv(GL_LIGHT0,GL_AMBIENT,sourceLight);
```

Lights Prosperities

Color:

- The default values listed for **GL_DIFFUSE** and **GL_SPECULAR** apply only to **GL_LIGHT0.**
- For other lights, the default value is **(0.0, 0.0, 0.0, 1.0)** for both **GL_DIFFUSE** and **GL_SPECULAR.**
- Ambient, diffuse = color of material.
- Specular = usually white.
- Emissive color = light coming from object.

Position: (x, y, z, w)

- **OpenGL supports two types of Lights:**
  1. Local (Point) light sources
  2. Infinite (Directional) light sources

Type of light controlled by w coordinate:

1. W is ZERO, **Directional light,** (x,y,z) specify its direction.
2. W is NONZERO, **Positional light,** (x,y,z) specify the location of the light source

Material Properties

- **Material :** Describe the percentages of the incoming red, green, blue light it reflects on a surface.
- **Material** has different **ambient**, **diffuse** and **specular** colors.
- **Define the material properties for objects in the scene:**
  - `glMaterialfv( face, property, value );`
    - ***face:*** can be : `GL_FRONT, GL_BACK, GL_FRONT_AND_BACK.`
    - ***property(pname):*** identifier of **property** of the light.
    - ***value:*** is the value of pname.

Pname	Def. Value	Meaning
GL_AMBIENT	(0.2, 0.2, 0.2, 1.0)	ambient color of material
GL_DIFFUSE	(0.8, 0.8, 0.8, 1.0)	diffuse color of material
GL_SPECULAR	(0.0, 0.0, 0.0, 1.0)	specular color of material
GL_SHININESS	0.0	specular exponent
GL_EMISSION	(0.0, 0.0, 0.0, 1.0)	emissive color of material

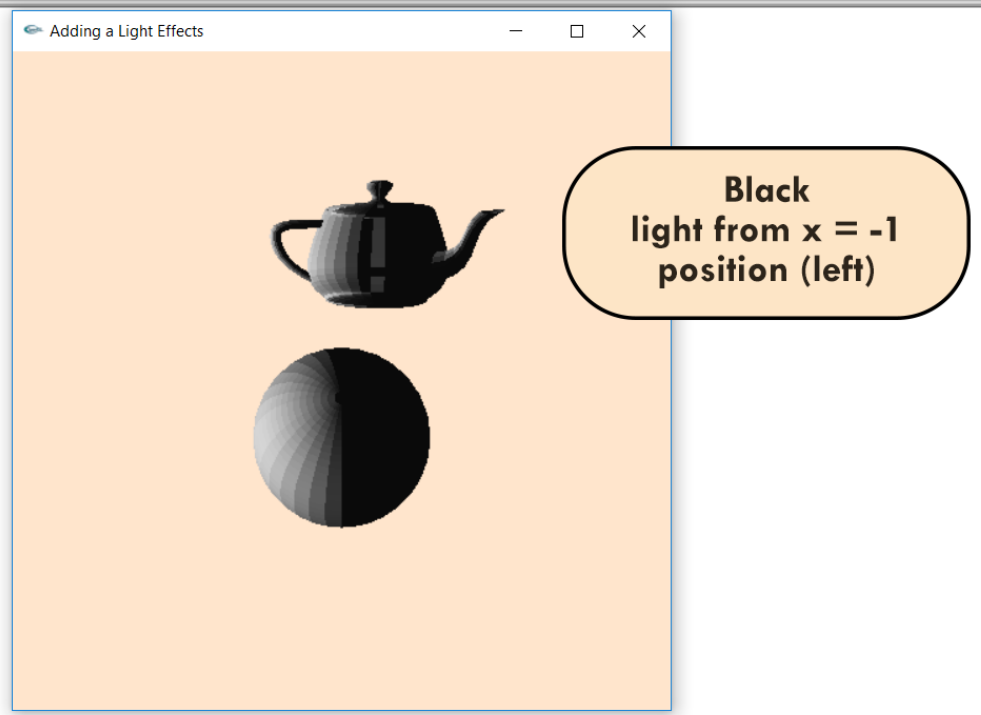
- ***To enable the color of object:*** `glEnable(GL_COLOR_MATERIAL);`

A Lighting Program Using OpenGL

```
#include<windows.h>
#include<GL/glut.h>
void Draw() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    gluLookAt(0, 3, 5, 0, 1, 0, 0, 1, 2);
    //Lighting Part
    GLfloat mat_specular[] = {0, 0, 0, 0};
    GLfloat light_position[] = {-1, 0, 0, 0};
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    //Ending of the Lighting Part
    glColor3f(1.0, 1.0, 1.0);
    glutSolidSphere(1.5, 30, 30);
    glTranslatef(0.5, 3, 0.0);
    glColor3f(0,0,0);
    glutSolidTeapot(1);
    glFlush(); }
void init() {
    glClearColor(1, 0.9, 0.8, 0.0);
    glShadeModel(GL_FLAT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glFrustum(-1, 1, -1, 1, 1, 15);
    glMatrixMode(GL_MODELVIEW); }
int main() {
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition(0,0);
    glutCreateWindow("Adding a Light Effects");
    init();
    glutDisplayFunc(Draw);
    glutMainLoop();
    return 0; }
```

A Lighting Program Using OpenGL

```
#include<windows.h>
#include<GL/glut.h>
void Draw() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    gluLookAt(0, 3, 5, 0, 1, 0, 0, 1, 2);
    //Lighting Part
    GLfloat mat_specular[] = {0, 0, 0, 0};
    GLfloat light_position[] = {-1, 0, 0, 0};
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    //Ending of the Lighting Part
    glColor3f(1.0, 1.0, 1.0);
    glutSolidSphere(1.5, 30, 30);
    glTranslatef(0.5, 3, 0.0);
    glColor3f(0,0,0);
    glutSolidTeapot(1);
    glFlush(); }
void init() {
    glClearColor(1, 0.9, 0.8, 0.0);
    glShadeModel(GL_FLAT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glFrustum(-1, 1, -1, 1, 1, 15);
    glMatrixMode(GL_MODELVIEW); }
int main() {
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition(0,0);
    glutCreateWindow("Adding a Light Effects");
    init();
    glutDisplayFunc(Draw);
    glutMainLoop();
    return 0; }
```



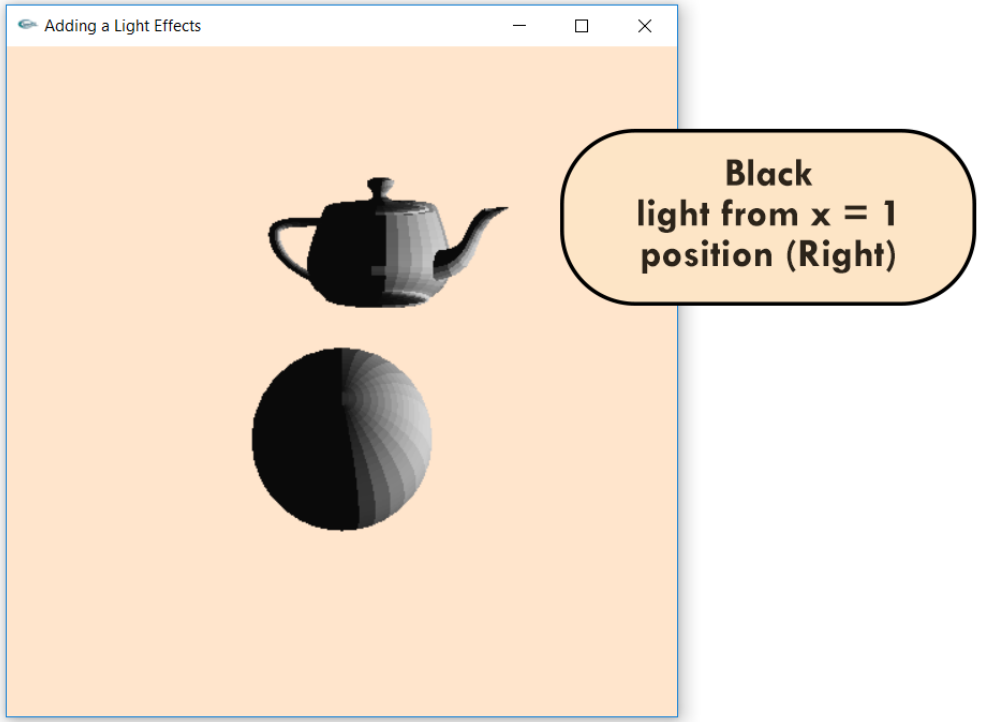
A Lighting Program Using OpenGL

Code::Blocks 17.12

Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

Lighting.cpp

```
#include<windows.h>
#include<GL/glut.h>
void Draw() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    gluLookAt(0, 3, 5, 0, 1, 0, 0, 1, 2);
    //Lighting Part
    GLfloat mat_specular[] = {0, 0, 0, 0};
    GLfloat light_position[] = {1, 0, 0, 0};
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    //Ending of the Lighting Part
    glColor3f(1.0, 1.0, 1.0);
    glutSolidSphere(1.5, 30, 30);
    glTranslatef(0.5, 3, 0.0);
    glColor3f(0,0,0);
    glutSolidTeapot(1);
    glFlush(); }
void init() {
    glClearColor(1, 0.9, 0.8, 0.0);
    glShadeModel(GL_FLAT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glFrustum(-1, 1, -1, 1, 1, 15);
    glMatrixMode(GL_MODELVIEW); }
int main() {
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition(0,0);
    glutCreateWindow("Adding a Light Effects");
    init();
    glutDisplayFunc(Draw);
    glutMainLoop();
    return 0; }
```



A Lighting Program Using OpenGL

ode::Blocks 17.12

Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

Debug

Lighting.cpp

```
#include<windows.h>
#include<GL/glut.h>
void Draw() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    gluLookAt(0, 3, 5, 0, 1, 0, 0, 1, 2);
    //Lighting Part
    GLfloat mat_specular[] = {1, 0.5, 0.5, 0};
    GLfloat light_position[] = {1, 1, 0, 0};
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    //Ending of the Lighting Part
    glColor3f(1.0, 1.0, 1.0);
    glutSolidSphere(1.5, 30, 30);
    glTranslatef(0.5, 3, 0.0);
    glColor3f(0,0,0);
    glutSolidTeapot(1);
    glFlush(); }
void init() {
    glClearColor(1, 0.9, 0.8, 0.0);
    glShadeModel(GL_FLAT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glFrustum(-1, 1, -1, 1, 1, 15);
    glMatrixMode(GL_MODELVIEW); }
int main() {
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition(0,0);
    glutCreateWindow("Adding a Light Effects");
    init();
    glutDisplayFunc(Draw);
    glutMainLoop();
    return 0; }
```

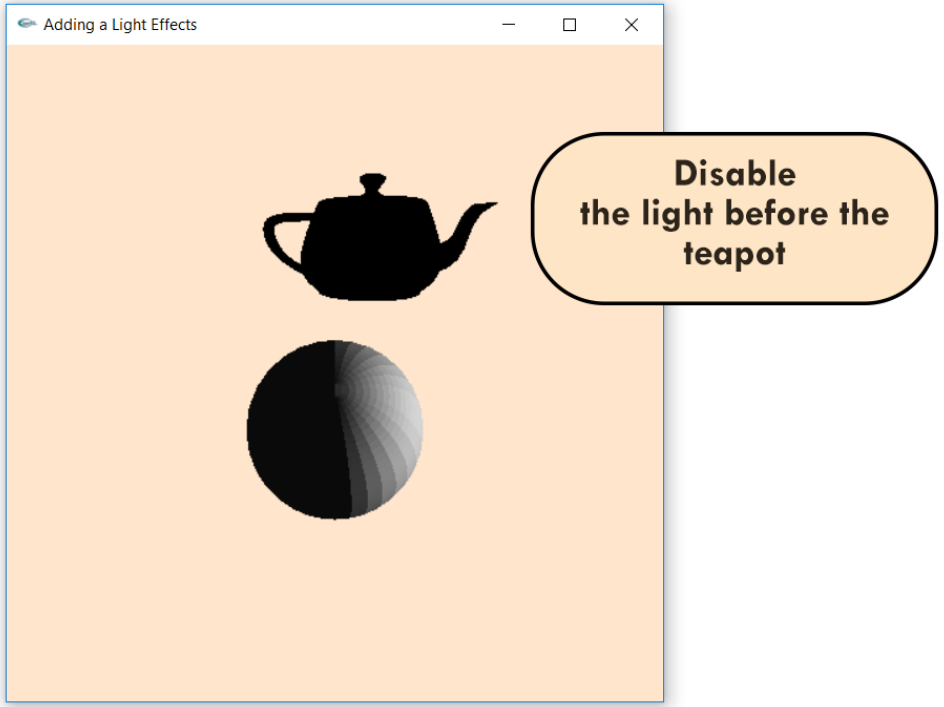
A Lighting Program Using OpenGL

Code::Blocks 17.12

Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

Lighting.cpp

```
1  #include<windows.h>
2  #include<GL/glut.h>
3  void Draw() {
4      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
5      gluLookAt(0, 3, 5, 0, 1, 0, 0, 1, 2);
6      //Lighting Part
7      GLfloat mat_specular[] = {0, 0, 0, 0};
8      GLfloat light_position[] = {1, 0, 0, 0};
9      glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
10     glLightfv(GL_LIGHT0, GL_POSITION, light_position);
11     glEnable(GL_LIGHTING);
12     glEnable(GL_LIGHT0);
13     //Ending of the Lighting Part
14     glColor3f(1.0, 1.0, 1.0);
15     glutSolidSphere(1.5, 30, 30);
16     glDisable(GL_LIGHTING);
17     glTranslatef(0.5, 3, 0.0);
18     glColor3f(0,0,0);
19     glutSolidTeapot(1);
20     glFlush(); }
21 void init() {
22     glClearColor(1, 0.9, 0.8, 0.0);
23     glShadeModel(GL_FLAT);
24     glMatrixMode(GL_PROJECTION);
25     glLoadIdentity();
26     glFrustum(-1, 1, -1, 1, 1, 15);
27     glMatrixMode(GL_MODELVIEW); }
28 int main() {
29     glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
30     glutInitWindowSize (500, 500);
31     glutInitWindowPosition(0,0);
32     glutCreateWindow("Adding a Light Effects");
33     init();
34     glutDisplayFunc(Draw);
35     glutMainLoop();
36     return 0; }
37
```



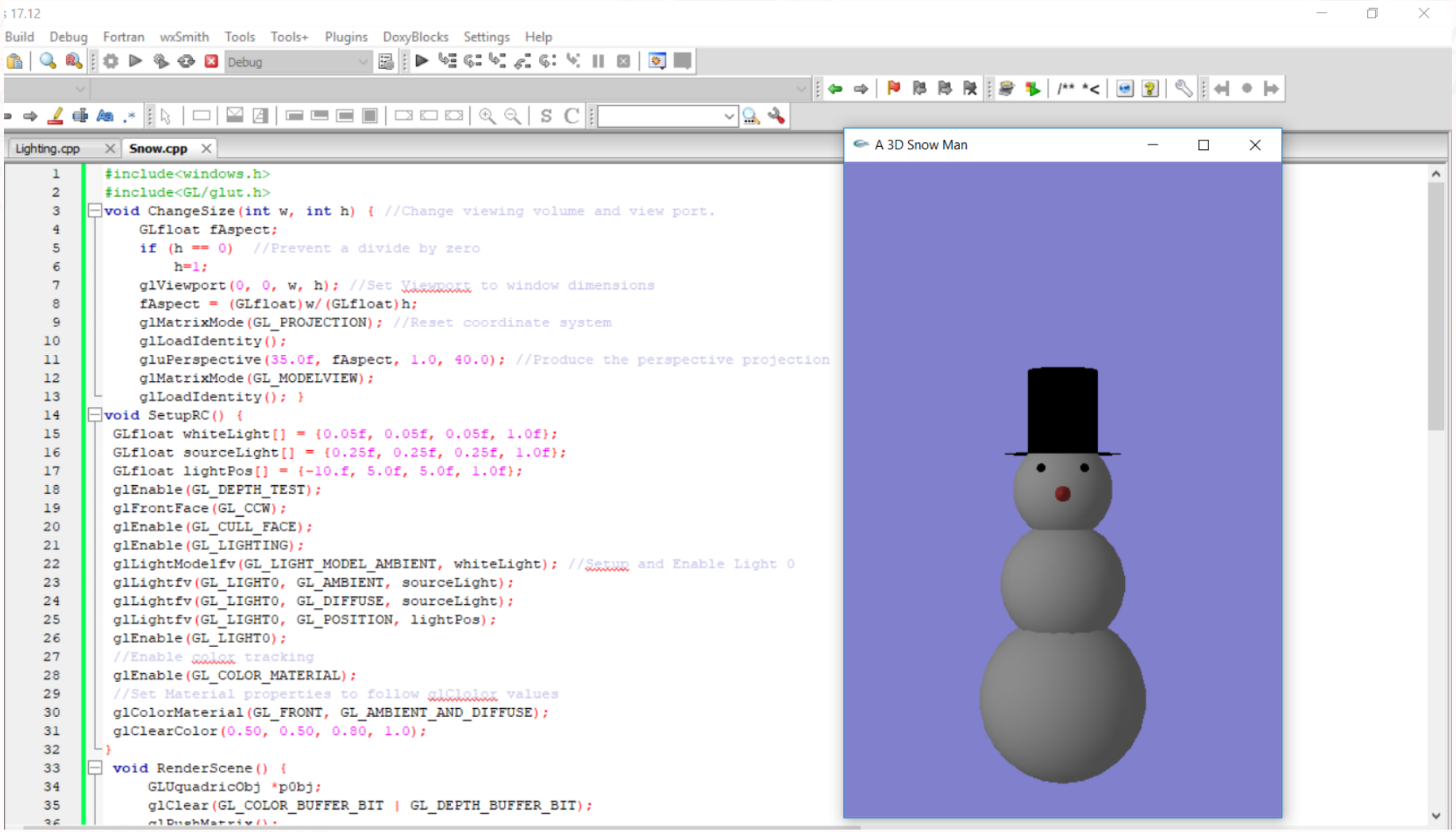
A Snow Man Program Using OpenGL

```
#include<windows.h>
#include<GL/glut.h>
void ChangeSize(int w, int h) { //Change viewing volume and view port.
    GLfloat fAspect;
    if (h == 0)  //Prevent a divide by zero
        h=1;
    glViewport(0, 0, w, h); //Set Viewport to window dimensions
    fAspect = (GLfloat)w/(GLfloat)h;
    glMatrixMode(GL_PROJECTION); //Reset coordinate system
    glLoadIdentity();
    gluPerspective(35.0f, fAspect, 1.0, 40.0); //Produce the perspective projection
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity(); }
void SetupRC() {
 GLfloat whiteLight[] = {0.05f, 0.05f, 0.05f, 1.0f};
 GLfloat sourceLight[] = {0.25f, 0.25f, 0.25f, 1.0f};
 GLfloat lightPos[] = {-10.f, 5.0f, 5.0f, 1.0f};
 glEnable(GL_DEPTH_TEST);
 glFrontFace(GL_CCW);
 glEnable(GL_CULL_FACE);
 glEnable(GL_LIGHTING);
 glLightModelfv(GL_LIGHT_MODEL_AMBIENT, whiteLight); //Setup and Enable Light 0
 glLightfv(GL_LIGHT0, GL_AMBIENT, sourceLight);
 glLightfv(GL_LIGHT0, GL_DIFFUSE, sourceLight);
 glLightfv(GL_LIGHT0, GL_POSITION, lightPos);
 glEnable(GL_LIGHT0);
 //Enable color tracking
 glEnable(GL_COLOR_MATERIAL);
 //Set Material properties to follow glClolor values
 glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
 glClearColor(0.50, 0.50, 0.80, 1.0);
}
```

A Snow Man Program Using OpenGL

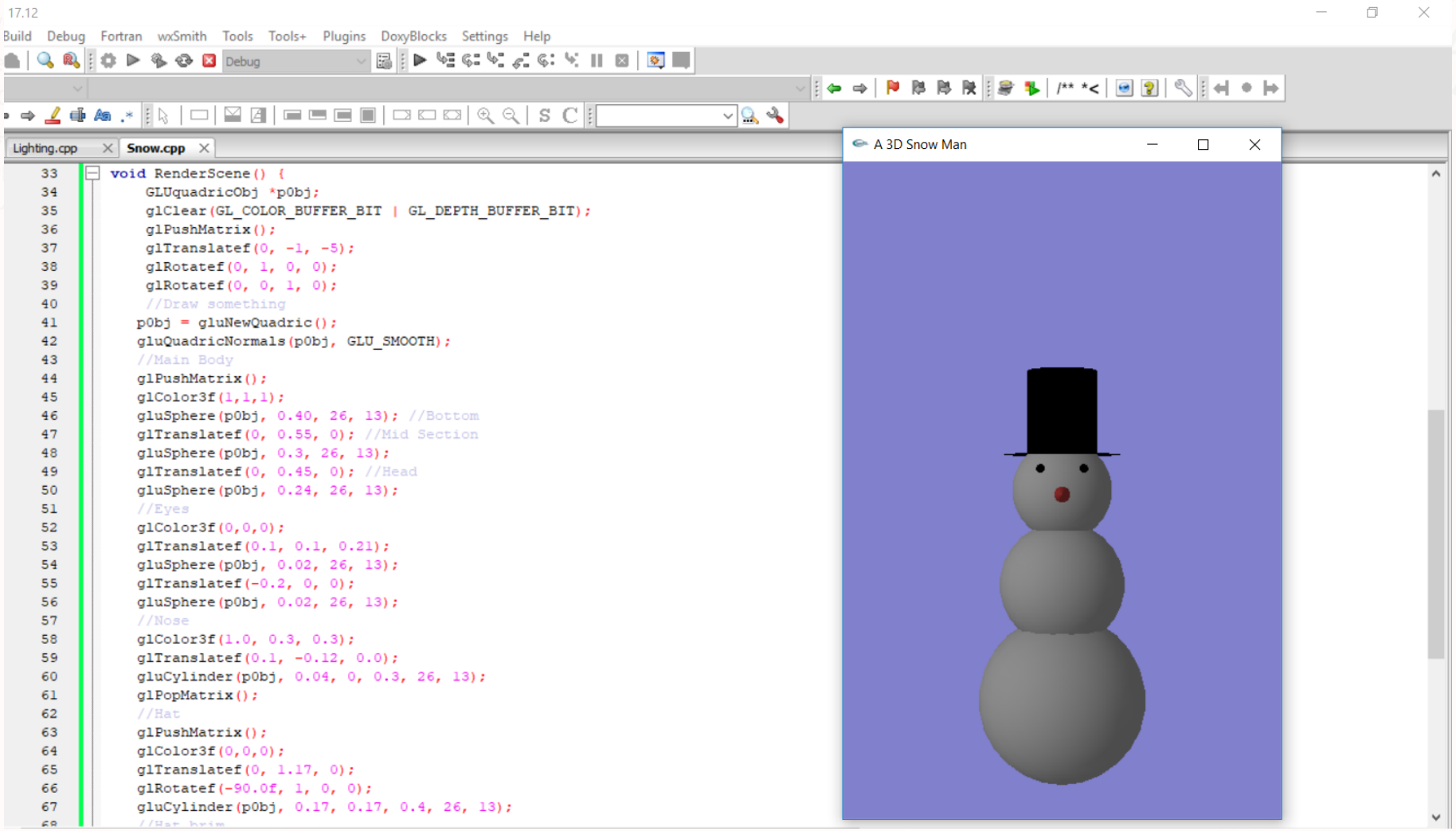
```
void RenderScene() {
     GLUquadricObj *p0bj;
     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
     glPushMatrix();
     glTranslatef(0, -1, -5);
     glRotatef(0, 1, 0, 0);
     glRotatef(0, 0, 1, 0);
     //Draw something
    p0bj = gluNewQuadric();
    gluQuadricNormals(p0bj, GLU_SMOOTH);
    //Main Body
    glPushMatrix();
    glColor3f(1,1,1);
    gluSphere(p0bj, 0.40, 26, 13); //Bottom
    glTranslatef(0, 0.55, 0); //Mid Section
    gluSphere(p0bj, 0.3, 26, 13);
    glTranslatef(0, 0.45, 0); //Head
    gluSphere(p0bj, 0.24, 26, 13);
    //Eyes
    glColor3f(0,0,0);
    glTranslatef(0.1, 0.1, 0.21);
    gluSphere(p0bj, 0.02, 26, 13);
    glTranslatef(-0.2, 0, 0);
    gluSphere(p0bj, 0.02, 26, 13);
    //Nose
    glColor3f(1.0, 0.3, 0.3);
    glTranslatef(0.1, -0.12, 0.0);
    gluCylinder(p0bj, 0.04, 0, 0.3, 26, 13);
    glPopMatrix();
    //Hat
    glPushMatrix();
    glColor3f(0,0,0);
    glTranslatef(0, 1.17, 0);
    glRotatef(-90.0f, 1, 0, 0);
    gluCylinder(p0bj, 0.17, 0.17, 0.4, 26, 13);
    //Hat brim
    glDisable(GL_CULL_FACE);
    gluDisk(p0bj, 0.17f, 0.28, 26, 13);
    glEnable(GL_CULL_FACE);
    glTranslatef(0.0f, 0.0f, 0.40f);
    gluDisk(p0bj, 0.0f, 0.17f, 26, 13);
    glPopMatrix();
    //Restore the Matrix State
    glPopMatrix();
    //Buffer swap
    glutSwapBuffers(); }
int main() {
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB| GLUT_DEPTH);
    glutInitWindowSize(400, 600);
    glutInitWindowPosition(0,0);
    glutCreateWindow("A 3D Snow Man");
    glutReshapeFunc(ChangeSize);
    glutDisplayFunc(RenderScene);
    SetupRC();
    glutMainLoop();
return 0; }
```

A Snow Man Program Using OpenGL



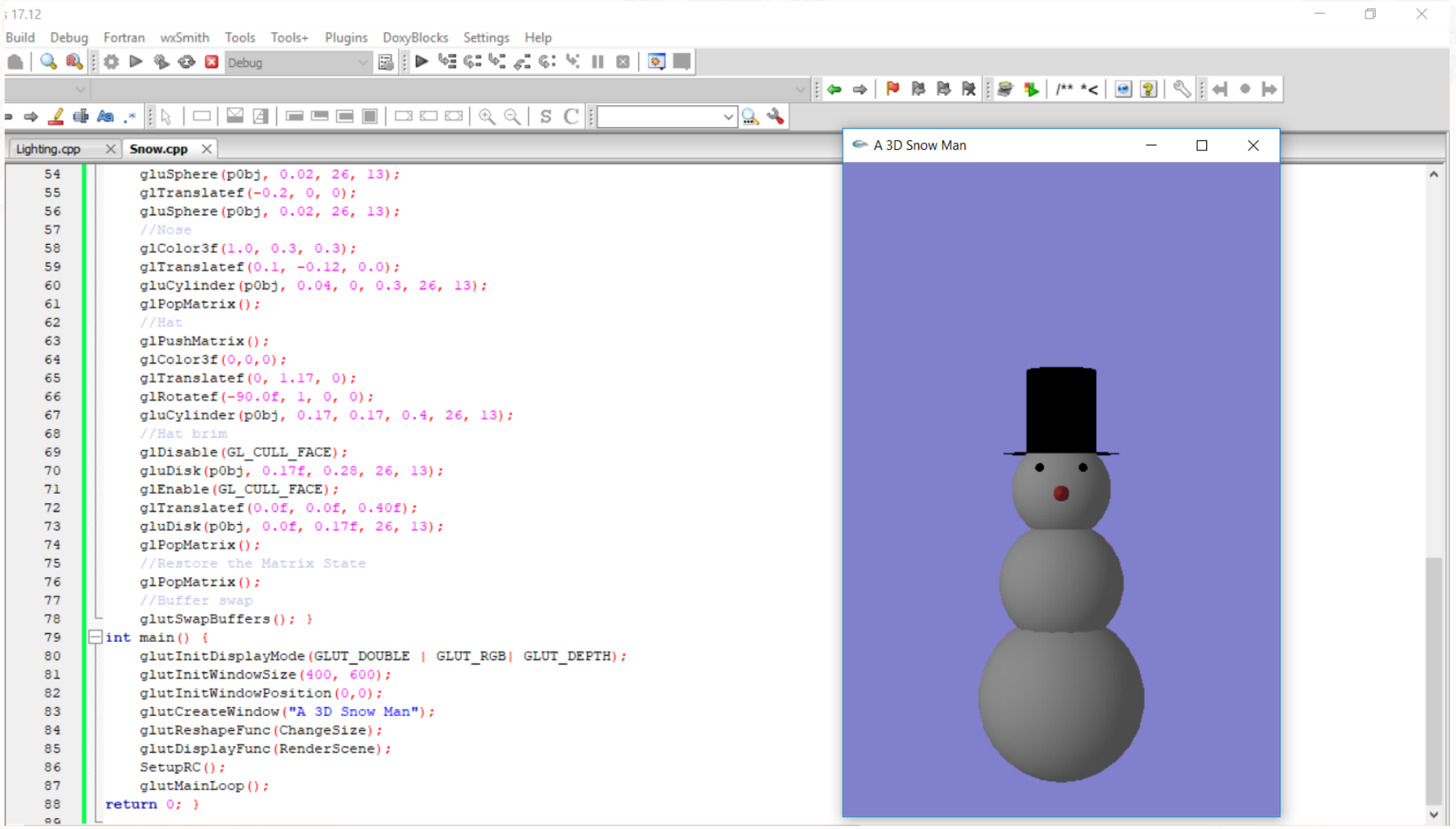

```
#include<windows.h>
#include<GL/glut.h>
void ChangeSize(int w, int h) { //Change viewing volume and view port.
    GLfloat fAspect;
    if (h == 0)  //Prevent a divide by zero
        h=1;
    glViewport(0, 0, w, h); //Set Viewport to window dimensions
    fAspect = (GLfloat)w/(GLfloat)h;
    glMatrixMode(GL_PROJECTION); //Reset coordinate system
    glLoadIdentity();
    gluPerspective(35.0f, fAspect, 1.0, 40.0); //Produce the perspective projection
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity(); }
void SetupRC() {
 GLfloat whiteLight[] = {0.05f, 0.05f, 0.05f, 1.0f};
 GLfloat sourceLight[] = {0.25f, 0.25f, 0.25f, 1.0f};
 GLfloat lightPos[] = {-10.f, 5.0f, 5.0f, 1.0f};
 glEnable(GL_DEPTH_TEST);
 glFrontFace(GL_CCW);
 glEnable(GL_CULL_FACE);
 glEnable(GL_LIGHTING);
 glLightModelfv(GL_LIGHT_MODEL_AMBIENT, whiteLight); //Setup and Enable Light 0
 glLightfv(GL_LIGHT0, GL_AMBIENT, sourceLight);
 glLightfv(GL_LIGHT0, GL_DIFFUSE, sourceLight);
 glLightfv(GL_LIGHT0, GL_POSITION, lightPos);
 glEnable(GL_LIGHT0);
 //Enable color tracking
 glEnable(GL_COLOR_MATERIAL);
 //Set Material properties to follow glColor values
 glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
 glClearColor(0.50, 0.50, 0.80, 1.0);
}
 void RenderScene() {
     GLUquadricObj *pObj;
     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

A Snow Man Program Using OpenGL




```
33  void RenderScene() {
34      GLUquadricObj *p0bj;
35      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
36      glPushMatrix();
37      glTranslatef(0, -1, -5);
38      glRotatef(0, 1, 0, 0);
39      glRotatef(0, 0, 1, 0);
40      //Draw something
41     p0bj = gluNewQuadric();
42     gluQuadricNormals(p0bj, GLU_SMOOTH);
43     //Main Body
44     glPushMatrix();
45     glColor3f(1,1,1);
46     gluSphere(p0bj, 0.40, 26, 13); //Bottom
47     glTranslatef(0, 0.55, 0); //Mid Section
48     gluSphere(p0bj, 0.3, 26, 13);
49     glTranslatef(0, 0.45, 0); //Head
50     gluSphere(p0bj, 0.24, 26, 13);
51     //Eyes
52     glColor3f(0,0,0);
53     glTranslatef(0.1, 0.1, 0.21);
54     gluSphere(p0bj, 0.02, 26, 13);
55     glTranslatef(-0.2, 0, 0);
56     gluSphere(p0bj, 0.02, 26, 13);
57     //Nose
58     glColor3f(1.0, 0.3, 0.3);
59     glTranslatef(0.1, -0.12, 0.0);
60     gluCylinder(p0bj, 0.04, 0, 0.3, 26, 13);
61     glPopMatrix();
62     //Hat
63     glPushMatrix();
64     glColor3f(0,0,0);
65     glTranslatef(0, 1.17, 0);
66     glRotatef(-90.0f, 1, 0, 0);
67     gluCylinder(p0bj, 0.17, 0.17, 0.4, 26, 13);
```

A Snow Man Program Using OpenGL



وصلى الله وبارك على نبينا محمد

The End Summary of Lecture Nine

T.Mariah Khayat
الأستاذة/ مارية خياط
Adham University College
الكلية الجامعية بأضم

mskhayat@uqu.edu.sa