```cpp
#include "../gwm.h"
#define _USE_MATH_DEFINES 1
#include <math.h>

#include "res.h"

GLfloat dTheta = 2.0 * M_PI ;
GLfloat dPhi = M_PI / 16.0;

GLuint texture_moon;
GLUquadric *quadric = NULL;
bool bLight = false;

GLfloat LightAmbient[] = { 0.0f, 0.0f, 0.0f, 1.0f };
GLfloat LightDiffuse[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat LightSpecular[] = { 0.5f, 0.5f, 0.5f, 1.0f };
GLfloat LightPosition[] = { 0.0f, 0.0f, 0.0f, 1.0f };

GLfloat MaterialAmbient[] = { 0.0f, 0.0f, 0.0f, 1.0f };
GLfloat MaterialDiffuse[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat MaterialSpecular[] = { 0.5f, 0.5f, 0.5f, 1.0f };
GLfloat MaterialShininess[] = { 128.0f };

void main (void)
{
    // declarations
    void initFunc(void);
    void keyboardFunc(unsigned int);
    void displayFunc(void);
    void reshapeFunc(int, int);
    void updateFunc(void);
    void uninitFunc(void);

    // code
    gwmInitializeCallback(initFunc);
    gwmKeyboardCallback(keyboardFunc);
    gwmDisplayCallback(displayFunc);
    gwmReshapeCallback(reshapeFunc);
    gwmUpdateCallback(updateFunc);
    gwmUninitializeCallback(uninitFunc);

    gwmCreateWindow("Moon", 100, 100, 800, 600);
    gwmEventLoop();
}

void initFunc(void)
{
    // Texture
    glEnable(GL_TEXTURE_2D);
    gwmLoadTexture(&texture_moon, MAKEINTRESOURCE(IDBITMAP_MOON));

    // Light
    glLightfv(GL_LIGHT0, GL_AMBIENT, LightAmbient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, LightDiffuse);
    glLightfv(GL_LIGHT0, GL_POSITION, LightPosition);
    glLightfv(GL_LIGHT0, GL_SPECULAR, LightSpecular);
    glEnable(GL_LIGHT0);

    glMaterialfv(GL_FRONT, GL_AMBIENT, MaterialAmbient);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, MaterialDiffuse);
    glMaterialfv(GL_FRONT, GL_SPECULAR, MaterialSpecular);
    glMaterialfv(GL_FRONT, GL_SHININESS, MaterialShininess);
}

void keyboardFunc(unsigned int key)
{
    switch(key)
    {
        // VK_ESCAPE
        case 0x1B:
            gwmExitEventLoop();
            break;

        // L
        case 0x4C:
            if (bLight == false)
            {
                bLight = true;
                glEnable(GL_LIGHTING);
            }
            else
            {
                bLight = false;
                glDisable(GL_LIGHTING);
            }
            break;
    }
}

void reshapeFunc(int width, int height)
{
    if (height == 0)
    {
        height = 1;
    }

    glViewport(0, 0, (GLsizei)width, (GLsizei)height);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    gluPerspective(45.0, (GLfloat)width / (GLfloat)height, 0.1f, 100.0f);

}
```

```cpp

void displayFunc(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    /* Load ModelView Matrix */
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();       /* Reset to Identity Matrix | center */

    /* Multiply it by Translation Matrix */
    glTranslatef(0.0f, 0.0f, -0.70f);

    //glPushMatrix();
    glColor3f(0, 0, 0);
    glRotatef(90.0f, 0.0f, 1.0f, 0.0f);
    glRotatef(180.0f, 1.0f, 0.0f, 0.0f);
    glRotatef(-90.0f, 0.0f, 0.0f, 1.0f);

    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);

    quadric = gluNewQuadric();
    gluQuadricTexture(quadric, true);

    glBindTexture(GL_TEXTURE_2D, texture_moon);
    gluSphere(quadric, 0.2f, 160, 160);

    gwmSwapBuffers();
}

void updateFunc(void)
{
    if (dTheta > 0.0)
    {
        dTheta -= 0.0015;
    }
    else
    {
        dTheta = 2 * M_PI;
    }

    GLfloat _cosTheta = 4.0*cos(dTheta);
    GLfloat _sinTheta = 4.0*sin(dTheta);

    GLfloat _cosPhi = cos(dPhi);
    GLfloat _sinPhi = sin(dPhi);

    LightPosition[0] = _sinTheta * _sinPhi;
    LightPosition[1] = _cosTheta;
    LightPosition[2] = _sinTheta * _cosPhi;

    glLightfv(GL_LIGHT0, GL_POSITION, LightPosition);
}
```

```

void uninitFunc(void)
{
    if (quadric)
    {
        gluDeleteQuadric(quadric);
        quadric = NULL;
    }
}

```