NATURE SCENERY WITH COLLEGE BUILDINGS CG MINI PROJECT

```
#include <windows.h>
#include <GL/glut.h>
#include <math.h>

// Window dimensions
const int WIDTH = 800;
const int HEIGHT = 600;

// Menu options
const int MENU_SUNRISE = 1;
const int MENU_SUNSET = 2;
const int MENU_EXIT = 3;

// Current state
int state = MENU_SUNRISE;

typedef struct
{
float x;
float y;
}CIRCLE;

class Color{
public :float x,y,z;
public :Color(float a,float b, float c){
    x=a;
    y=b;
    z=c;
 }
 void setcolor(float a,float b, float c){
```

```
    x=a;
    y=b;
    z=c;
 }
};
Color color=Color(1,1,1);
CIRCLE circle;

void glutSolidSphere1 (int k, int r, int h) {
   glColor3f(color.x,color.y,color.z);
   glBegin(GL_POLYGON);
   for (int i = 0; i < 180; i++)
   {
   circle.x = r * cos(i) - h;
   circle.y = r * sin(i) + k;
   glVertex3f(circle.x + k,circle.y - h,0);

   circle.x = r * cos(i + 0.1) - h;
   circle.y = r * sin(i + 0.1) + k;
   glVertex3f(circle.x + k,circle.y - h,0);
   }
   glEnd();
}

void drawText(char* string,int x, int y){
    glColor3f(0.3,0.0,0.1);
   glBegin(GL_POLYGON);
   glVertex2f(x,y);
   glVertex2f(x+70,y);
   glVertex2f(x+70,y+30);
   glVertex2f(x,y+30);
```

```
    glEnd();
    glColor3f(1,1,1);

    int len, i;
     glRasterPos2f(x,y);
     len=(int) strlen(string);
     for(i = 0; i < len; i++)
       glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,string[i]);

    glFlush();

}

void drawGrass() {
glBegin(GL_QUADS);
    glColor3f(0.2f, 0.8f, 0.0f); // green color
    glVertex2f(0.0f * WIDTH / 0.5f, HEIGHT / 3.0f);
    glVertex2f(3.0f * WIDTH / 0.5f, HEIGHT / 3.0f);
    glVertex2f(3.0f * WIDTH / 0.5f, 0.0f * HEIGHT / 3.0f);
    glVertex2f(0.0f * WIDTH / 0.5f, 0.0f * HEIGHT / 3.0f);
    glEnd();
}

void drawClouds() {
    color.setcolor(1,1,1);// White color

    // First cloud
    glPushMatrix();
    glTranslatef(100.0f, 400.0f, 0.0f);
    glutSolidSphere1(40.0f, 20, 20);
    glPopMatrix();
```

```
glPushMatrix();
glTranslatef(120.0f, 400.0f, 0.0f);
glutSolidSphere1(40.0f, 20, 20);
glPopMatrix();

// Second cloud
glPushMatrix();
glTranslatef(300.0f, 450.0f, 0.0f);
glutSolidSphere1(60.0f, 20, 20);
glPopMatrix();

glPushMatrix();
glTranslatef(280.0f, 450.0f, 0.0f);
glutSolidSphere1(60.0f, 20, 20);
glPopMatrix();

 glPushMatrix();
glTranslatef(260.0f, 450.0f, 0.0f);
glutSolidSphere1(60.0f, 20, 20);
glPopMatrix();

// Third cloud
glPushMatrix();
glTranslatef(500.0f, 400.0f, 0.0f);
glutSolidSphere1(50.0f, 20, 20);
glPopMatrix();

glPushMatrix();
glTranslatef(475.0f, 400.0f, 0.0f);
glutSolidSphere1(50.0f, 20, 20);
```

```
glPopMatrix();

//fourth cloud
glPushMatrix();
glTranslatef(700.0f, 450.0f, 0.0f);
glutSolidSphere1(50.0f, 20, 20);
glPopMatrix();

glPushMatrix();
glTranslatef(725.0f, 450.0f, 0.0f);
glutSolidSphere1(50.0f, 20, 20);
glPopMatrix();

glPushMatrix();
glTranslatef(750.0f, 450.0f, 0.0f);
glutSolidSphere1(50.0f, 20, 20);
glPopMatrix();

//fifth cloud
glPushMatrix();
glTranslatef(900.0f, 400.0f, 0.0f);
glutSolidSphere1(50.0f, 20, 20);
glPopMatrix();

glPushMatrix();
glTranslatef(925.0f, 400.0f, 0.0f);
glutSolidSphere1(50.0f, 20, 20);
glPopMatrix();



//sixth cloud
```

```
    glPushMatrix();
    glTranslatef(1200.0f, 450.0f, 0.0f);
    glutSolidSphere1(50.0f, 20, 20);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(1225.0f, 450.0f, 0.0f);
    glutSolidSphere1(50.0f, 20, 20);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(1175.0f, 450.0f, 0.0f);
    glutSolidSphere1(50.0f, 20, 20);
    glPopMatrix();

    drawText("NIEIT",900,200);
}

void drawBirds() {
    glColor3f(0.0f, 0.0f, 0.0f); // Black color

    // First bird
    glPushMatrix();
    glTranslatef(600.0f, 600.0f, 0.0f);
    glBegin(GL_TRIANGLES);
    glVertex2f(0.0f, 10.0f);
    glVertex2f(-5.0f, 0.0f);
    glVertex2f(5.0f, 0.0f);
    glEnd();
    glPopMatrix();
```

```
    // Second bird
    glPushMatrix();
    glTranslatef(400.0f, 450.0f, 0.0f);
    glBegin(GL_TRIANGLES);
    glVertex2f(0.0f, 10.0f);
    glVertex2f(-5.0f, 0.0f);
    glVertex2f(5.0f, 0.0f);
    glEnd();
    glPopMatrix();

    // Third bird
    glPushMatrix();
    glTranslatef(600.0f, 450.0f, 0.0f);
    glBegin(GL_TRIANGLES);
    glVertex2f(0.0f, 10.0f);
    glVertex2f(-5.0f, 0.0f);
    glVertex2f(5.0f, 0.0f);
    glEnd();
    glPopMatrix();

    // fourth bird
    glPushMatrix();
    glTranslatef(650.0f, 450.0f, 0.0f);
    glBegin(GL_TRIANGLES);
    glVertex2f(0.0f, 10.0f);
    glVertex2f(-5.0f, 0.0f);
    glVertex2f(5.0f, 0.0f);
    glEnd();
    glPopMatrix();
}
```

```
void drawSun() {
    color.setcolor(1,1,0); // Yellow color
    glPushMatrix();
    glTranslatef(WIDTH / 2.0f, HEIGHT - 100.0f, 0.0f);
    glutSolidSphere1(50.0f, 40, 40);
    glPopMatrix();
}

void drawsunset() {
    color.setcolor(1,0.0,0.0); // red color
    glPushMatrix();
    glTranslatef(WIDTH / 0.75f, HEIGHT - 175.0f, 0.0f);
    glutSolidSphere1(0.0f, 60, 60);
    glPopMatrix();



}

void drawCollageBuilding() {
    // First building
    glBegin(GL_QUADS);
    glColor3f(0.7f, 0.7f, 0.7f); // Gray color
    glVertex2f(WIDTH / 5.0f, HEIGHT / 3.0f);
    glVertex2f(2.0f * WIDTH / 5.0f, HEIGHT / 3.0f);
    glVertex2f(2.0f * WIDTH / 5.0f, 2.0f * HEIGHT / 3.0f);
    glVertex2f(WIDTH / 5.0f, 2.0f * HEIGHT / 3.0f);
    glEnd();

    // First building windows
    glColor3f(0.2f, 0.2f, 0.2f); // Dark gray color
    glBegin(GL_QUADS);
```

```
glVertex2f(WIDTH / 5.0f + 20.0f, HEIGHT / 3.0f + 20.0f);
glVertex2f(WIDTH / 5.0f + 80.0f, HEIGHT / 3.0f + 20.0f);
glVertex2f(WIDTH / 5.0f + 80.0f, HEIGHT / 3.0f + 80.0f);
glVertex2f(WIDTH / 5.0f + 20.0f, HEIGHT / 3.0f + 80.0f);
glEnd();

glBegin(GL_QUADS);
glVertex2f(WIDTH / 5.0f + 20.0f, HEIGHT / 3.0f + 100.0f);
glVertex2f(WIDTH / 5.0f + 80.0f, HEIGHT / 3.0f + 100.0f);
glVertex2f(WIDTH / 5.0f + 80.0f, HEIGHT / 3.0f + 160.0f);
glVertex2f(WIDTH / 5.0f + 20.0f, HEIGHT / 3.0f + 160.0f);
glEnd();

glBegin(GL_QUADS);
glVertex2f(WIDTH / 6.0f + 120.0f, HEIGHT / 3.0f + 20.0f);
glVertex2f(WIDTH / 6.0f + 180.0f, HEIGHT / 3.0f + 20.0f);
glVertex2f(WIDTH / 6.0f + 180.0f, HEIGHT / 3.0f + 80.0f);
glVertex2f(WIDTH / 6.0f + 120.0f, HEIGHT / 3.0f + 80.0f);
glEnd();

glBegin(GL_QUADS);
glVertex2f(WIDTH / 6.0f + 120.0f, HEIGHT / 3.0f + 100.0f);
glVertex2f(WIDTH / 6.0f + 180.0f, HEIGHT / 3.0f + 100.0f);
glVertex2f(WIDTH / 6.0f + 180.0f, HEIGHT / 3.0f + 160.0f);
glVertex2f(WIDTH / 6.0f + 120.0f, HEIGHT / 3.0f + 160.0f);
glEnd();

// Second building
glBegin(GL_QUADS);
glColor3f(0.6f, 0.6f, 0.6f); // Light gray color
glVertex2f(3.0f * WIDTH / 5.0f, HEIGHT / 3.0f);
```

```
glVertex2f(4.0f * WIDTH / 5.0f, HEIGHT / 3.0f);
glVertex2f(4.0f * WIDTH / 5.0f, 2.0f * HEIGHT / 3.0f);
glVertex2f(3.0f * WIDTH / 5.0f, 2.0f * HEIGHT / 3.0f);
glEnd();

// Second building windows
glColor3f(0.2f, 0.2f, 0.2f); // Dark gray color
glBegin(GL_QUADS);
glVertex2f(3.0f * WIDTH / 5.0f + 20.0f, HEIGHT / 3.0f + 20.0f);
glVertex2f(3.0f * WIDTH / 5.0f + 80.0f, HEIGHT / 3.0f + 20.0f);
glVertex2f(3.0f * WIDTH / 5.0f + 80.0f, HEIGHT / 3.0f + 80.0f);
glVertex2f(3.0f * WIDTH / 5.0f + 20.0f, HEIGHT / 3.0f + 80.0f);
glEnd();

glBegin(GL_QUADS);
glVertex2f(3.0f * WIDTH / 5.0f + 20.0f, HEIGHT / 3.0f + 100.0f);
glVertex2f(3.0f * WIDTH / 5.0f + 80.0f, HEIGHT / 3.0f + 100.0f);
glVertex2f(3.0f * WIDTH / 5.0f + 80.0f, HEIGHT / 3.0f + 160.0f);
glVertex2f(3.0f * WIDTH / 5.0f + 20.0f, HEIGHT / 3.0f + 160.0f);
glEnd();

glBegin(GL_QUADS);
glVertex2f(3.0f * WIDTH / 5.0f + 95.0f, HEIGHT / 3.0f + 20.0f);
glVertex2f(3.0f * WIDTH / 5.0f + 153.0f, HEIGHT / 3.0f + 20.0f);
glVertex2f(3.0f * WIDTH / 5.0f + 153.0f, HEIGHT / 3.0f + 80.0f);
glVertex2f(3.0f * WIDTH / 5.0f + 95.0f, HEIGHT / 3.0f + 80.0f);
glEnd();

glBegin(GL_QUADS);
glVertex2f(3.0f * WIDTH / 5.0f + 95.0f, HEIGHT / 3.0f + 100.0f);
glVertex2f(3.0f * WIDTH / 5.0f + 153.0f, HEIGHT / 3.0f + 100.0f);
```

```
    glVertex2f(3.0f * WIDTH / 5.0f + 153.0f, HEIGHT / 3.0f + 160.0f);
    glVertex2f(3.0f * WIDTH / 5.0f + 95.0f, HEIGHT / 3.0f + 160.0f);
    glEnd();

}

void display() {

    glClear(GL_COLOR_BUFFER_BIT);

    if (state == MENU_SUNRISE) {
        glClearColor(0.0f, 0.5f, 1.0f, 1.0f); // Sky blue color
        drawSun();
    } else if (state == MENU_SUNSET) {
        glClearColor(1.0f, 0.5f, 0.0f, 1.0f); // Orange color
        drawsunset();
    }

    drawGrass();
    drawClouds();
    drawBirds();
    drawCollageBuilding();

    glutSwapBuffers();
}

void menu(int choice) {
    switch (choice) {
        case MENU_SUNRISE:
            state = MENU_SUNRISE;
            break;
```

```
        case MENU_SUNSET:
            state = MENU_SUNSET;
            break;
        case MENU_EXIT:
            exit(0);
    }

    glutPostRedisplay();
}

void createMenu() {
    glutCreateMenu(menu);
    glutAddMenuEntry("Sunrise", MENU_SUNRISE);
    glutAddMenuEntry("Sunset", MENU_SUNSET);
    glutAddMenuEntry("Exit", MENU_EXIT);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
}

void reshape(int width, int height) {
    glViewport(0, 0, width, height);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0, width, 0, height);
    glMatrixMode(GL_MODELVIEW);
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(WIDTH, HEIGHT);
    glutInitWindowPosition(100, 100); // Adjust window position as desired
```

```
    glutCreateWindow("Nature Scenery with Collage Buildings");

    createMenu();

    glutDisplayFunc(display);
    glutReshapeFunc(reshape);

    glutMainLoop();

    return 0;
}
```