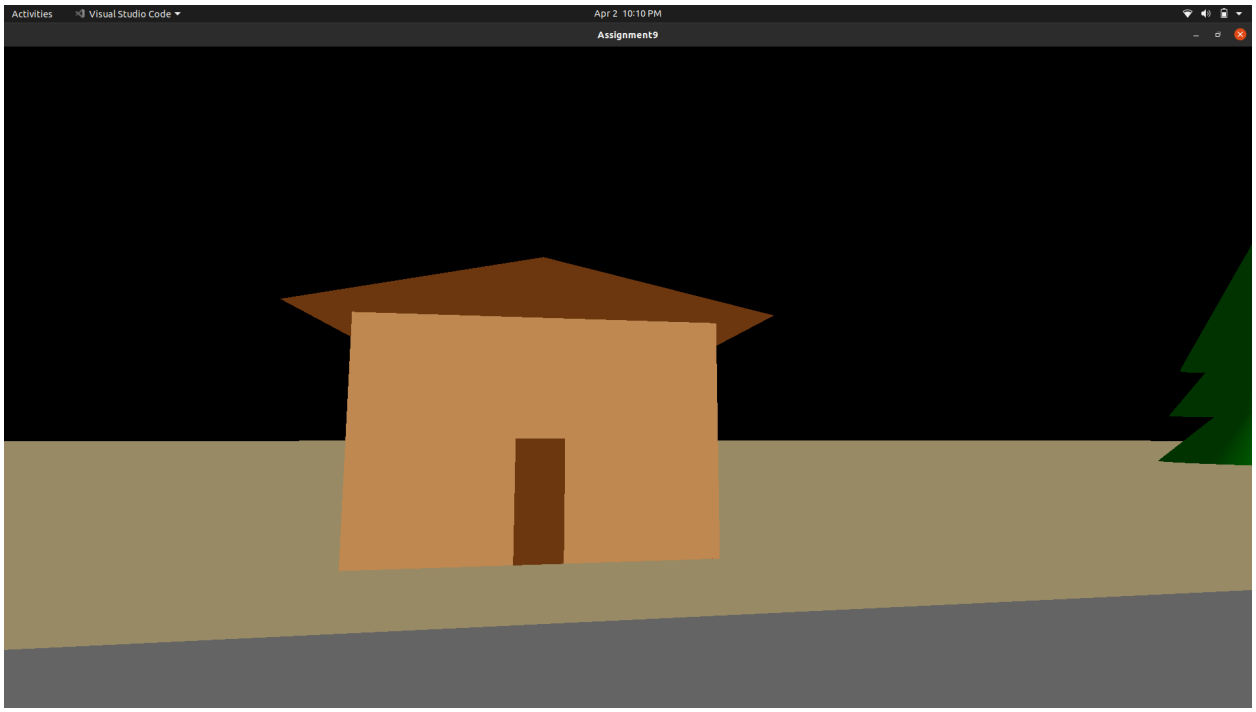
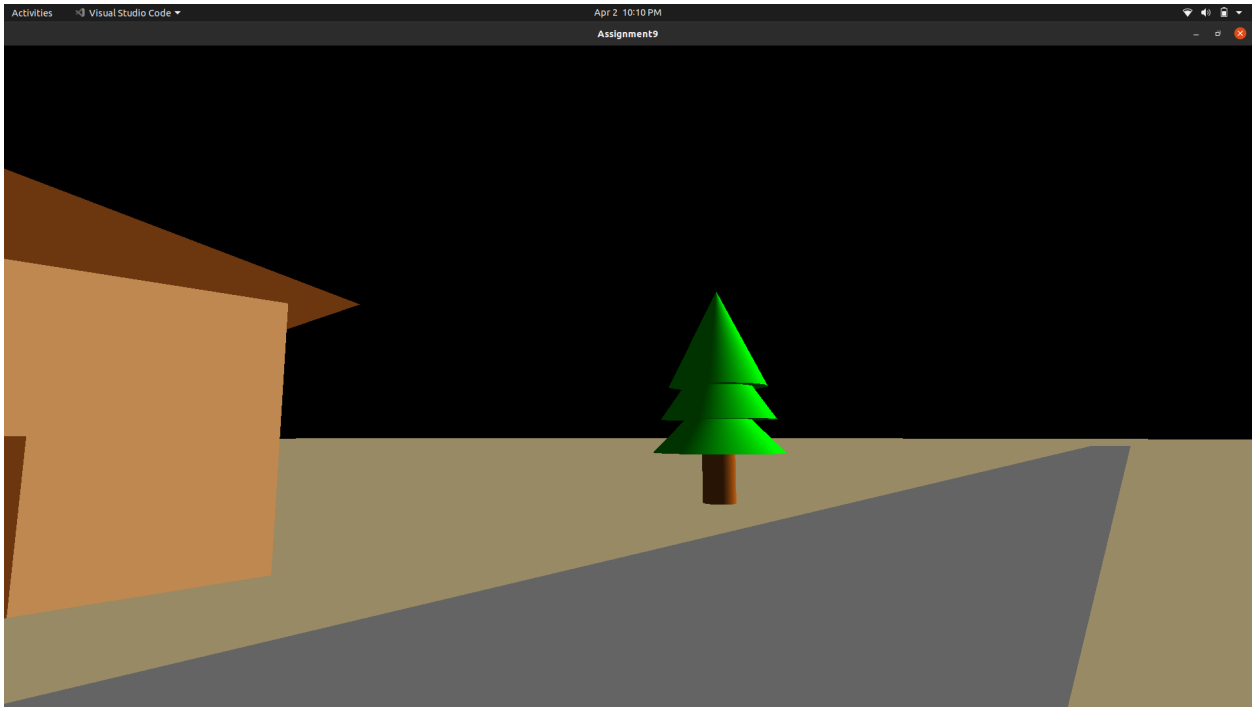
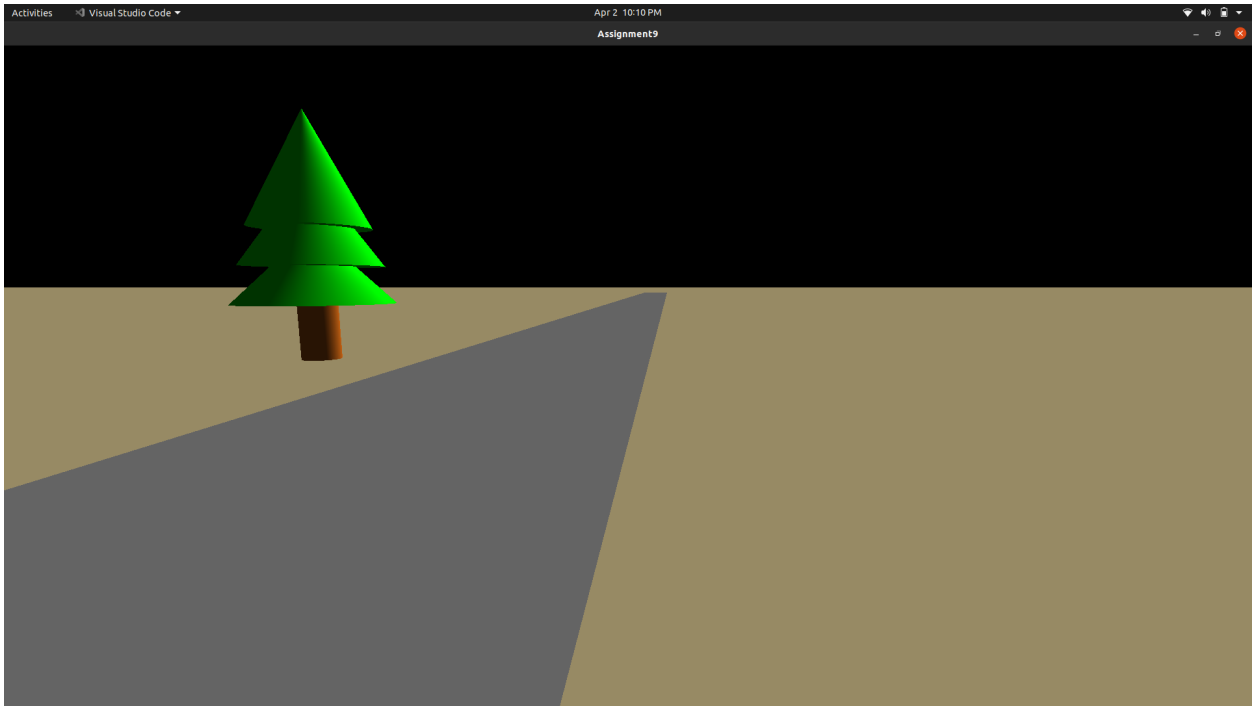
190001016
Garvit Galgat
Assignment 9








```
/**
 * @file Assignment 9
 * @author Garvit Galgat (190001016)
 * @date 2022-04-02
 */

#include <bits/stdc++.h>
#include <GL/glut.h>
using namespace std;

#define PI 3.14
#define WIDTH 500
#define HEIGHT 500

float reference_point = 0.0;
int light_intensity_ = 0;
float width_of_house_ = 6.0;
float height_of_house_ = 4.0;
float length_of_house = -4.0;
```

```
float width_of_roof = width_of_house_ + 1.5;
float height_of_roof = 8.0;
float extra_depth = 0.01;
float width_of_door = 0.8;
float height_of_door = 4.0;
float size_of_window_ = 0.8;

float house_r_val = 245.0 / 255, house_g_val = 175.0 / 255,
house_b_val = 103.0 / 255;
float door_r_val = 255 / 255, door_g_val = 255 / 255, door_b_val = 0 /
255;
float window_r_val = 160 / 255.0, window_g_val = 82 / 255.0,
window_b_val = 45 / 255.0;
float roof_r_val = 139 / 255.0, roof_g_val = 69 / 255.0, roof_b_val = 19
/ 255.0;

int start_position_x = -1;
int start_position_y = -1;

float reduce_intensity = 0.0;

float angle_theta = PI / 2;
float phi_angle = 0;
float radius = 25;
float front_back = 0;
float left_right = 0;
int flip = 0;
int window1 = 0;
int window2 = 0;

bool is_mouse_pressed_bool = false;
```

```
float delta_change_unit = (2 * PI * radius) / 1000;

float camera[3] = {0, 0, 25};

void fill_color_RGB_VALS(float x, float y, float z) {
    glColor3f(x / 255, y / 255, z / 255);
}

void change_size_of_window(int width, int height) {
    if (height == 0)
        height = 1;
    float aspectRatio = (width * 1.0) / height;
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glViewport(0, 0, width, height);
    gluPerspective(60.0, aspectRatio, 0.1, 1000.0);
    glMatrixMode(GL_MODELVIEW);
}

void wall_with_window_at_position(int X) {
    glBegin(GL_POLYGON);
    glVertex3f(X, height_of_house_, width_of_house_);
    glVertex3f(X, size_of_window_, width_of_house_);
    glVertex3f(X, size_of_window_, -width_of_house_);
    glVertex3f(X, height_of_house_, -width_of_house_);
    glEnd();

    glBegin(GL_POLYGON);
    glVertex3f(X, -height_of_house_, width_of_house_);
    glVertex3f(X, -size_of_window_, width_of_house_);
    glVertex3f(X, -size_of_window_, -width_of_house_);
    glVertex3f(X, -height_of_house_, -width_of_house_);
```

```
    glEnd();

    glBegin(GL_POLYGON);
    glVertex3f(X, size_of_window_, width_of_house_);
    glVertex3f(X, size_of_window_, size_of_window_);
    glVertex3f(X, -size_of_window_, size_of_window_);
    glVertex3f(X, -size_of_window_, width_of_house_);
    glEnd();

    glBegin(GL_POLYGON);
    glVertex3f(X, size_of_window_, -width_of_house_);
    glVertex3f(X, size_of_window_, -size_of_window_);
    glVertex3f(X, -size_of_window_, -size_of_window_);
    glVertex3f(X, -size_of_window_, -width_of_house_);
    glEnd();
}

void wall_with_door(int z) {
    glBegin(GL_POLYGON);
    glVertex3f(width_of_house_, height_of_house_, z);
    glVertex3f(width_of_house_, -height_of_house_ + height_of_door,
z);
    glVertex3f(-width_of_house_, -height_of_house_ + height_of_door,
z);
    glVertex3f(-width_of_house_, height_of_house_, z);
    glEnd();

    glBegin(GL_POLYGON);
    glVertex3f(width_of_house_, -height_of_house_ + height_of_door,
z);
    glVertex3f(width_of_house_, -height_of_house_, z);
    glVertex3f(width_of_door, -height_of_house_, z);
```

```
    glVertex3f(width_of_door, -height_of_house_ + height_of_door, z);
    glEnd();

    glBegin(GL_POLYGON);
    glVertex3f(-width_of_house_, -height_of_house_ + height_of_door,
z);
    glVertex3f(-width_of_house_, -height_of_house_, z);
    glVertex3f(-width_of_door, -height_of_house_, z);
    glVertex3f(-width_of_door, -height_of_house_ + height_of_door, z);
    glEnd();
}

void drawCuboidAt(float x, float y, float z, float X, float Y, float Z) {
    glBegin(GL_POLYGON);
    glVertex3f(-x + X, Y + y, z + Z);
    glVertex3f(x + X, Y + y, z + Z);
    glVertex3f(x + X, Y - y, z + Z);
    glVertex3f(-x + X, Y - y, z + Z);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex3f(-x + X, Y + y, -z + Z);
    glVertex3f(x + X, Y + y, -z + Z);
    glVertex3f(x + X, Y - y, -z + Z);
    glVertex3f(-x + X, Y - y, -z + Z);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex3f(-x + X, Y + y, z + Z);
    glVertex3f(-x + X, Y + y, -z + Z);
    glVertex3f(-x + X, Y - y, -z + Z);
    glVertex3f(-x + X, Y - y, z + Z);
    glEnd();
    glBegin(GL_POLYGON);
```

```
    glVertex3f(x + X, Y + y, z + Z);
    glVertex3f(x + X, Y + y, -z + Z);
    glVertex3f(x + X, Y - y, -z + Z);
    glVertex3f(x + X, Y - y, z + Z);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex3f(-x + X, Y + y, z + Z);
    glVertex3f(-x + X, Y + y, -z + Z);
    glVertex3f(x + X, Y + y, -z + Z);
    glVertex3f(x + X, Y + y, z + Z);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex3f(-x + X, Y - y, z + Z);
    glVertex3f(-x + X, Y - y, -z + Z);
    glVertex3f(x + X, Y - y, -z + Z);
    glVertex3f(x + X, Y - y, z + Z);
    glEnd();
}

void drawTableAt(float X, float Y, float Z) {
    drawCuboidAt(1.8, 0.15, 1.8, X, Y + 1.5, Z);
    drawCuboidAt(0.25, 1.5, 0.25, X - 1.5, Y, Z - 1.5);
    drawCuboidAt(0.25, 1.5, 0.25, X + 1.5, Y, Z - 1.5);
    drawCuboidAt(0.25, 1.5, 0.25, X - 1.5, Y, Z + 1.5);
    drawCuboidAt(0.25, 1.5, 0.25, X + 1.5, Y, Z + 1.5);
}

void drawBedAt(float X, float Y, float Z) {
    drawCuboidAt(1.5, 0.7, 0.15, X, Y + 1, Z + 3);
    drawCuboidAt(1.5, 0.1, 3.2, X, Y + 0.6, Z);
    drawCuboidAt(0.15, 0.6, 0.15, X - 1.4, Y, Z - 3.0);
    drawCuboidAt(0.15, 0.6, 0.15, X + 1.4, Y, Z - 3.0);
```

```
    drawCuboidAt(0.15, 0.6, 0.15, X - 1.4, Y, Z + 3.0);
    drawCuboidAt(0.15, 0.6, 0.15, X + 1.4, Y, Z + 3.0);
}

void drawDoorAtPosition() {
    if (flip) {
        float x = -width_of_door;
        float z = width_of_house_;
        glBegin(GL_POLYGON);
        glVertex3f(x, -height_of_house_ + height_of_door, z);
        glVertex3f(x, -height_of_house_ + height_of_door, z + 2 *
width_of_door);
        glVertex3f(x, -height_of_house_, z + 2 * width_of_door);
        glVertex3f(x, -height_of_house_, z);
        glEnd();
    } else {
        float z = width_of_house_;
        glBegin(GL_POLYGON);
        glVertex3f(-width_of_door, -height_of_house_ + height_of_door,
z);
        glVertex3f(width_of_door, -height_of_house_ + height_of_door,
z);
        glVertex3f(width_of_door, -height_of_house_, z);
        glVertex3f(-width_of_door, -height_of_house_, z);
        glEnd();
    }
}

void window() {
    if (window1) {
        float x = -width_of_house_;
        float z = -size_of_window_;
```

```
    glBegin(GL_POLYGON);
    glVertex3f(x, size_of_window_, z);
    glVertex3f(x - size_of_window_ * 2, size_of_window_, z);
    glVertex3f(x - size_of_window_ * 2, -size_of_window_, z);
    glVertex3f(x, -size_of_window_, z);
    glEnd();
  } else {
    float x = -width_of_house_;
    glBegin(GL_POLYGON);
    glVertex3f(x, size_of_window_, size_of_window_);
    glVertex3f(x, size_of_window_, -size_of_window_);
    glVertex3f(x, -size_of_window_, -size_of_window_);
    glVertex3f(x, -size_of_window_, size_of_window_);
    glEnd();
  }
  if (window2) {
    float x = width_of_house_;
    float z = -size_of_window_;
    glBegin(GL_POLYGON);
    glVertex3f(x, size_of_window_, z);
    glVertex3f(x + size_of_window_ * 2, size_of_window_, z);
    glVertex3f(x + size_of_window_ * 2, -size_of_window_, z);
    glVertex3f(x, -size_of_window_, z);
    glEnd();
  } else {
    float x = width_of_house_;
    glBegin(GL_POLYGON);
    glVertex3f(x, size_of_window_, size_of_window_);
    glVertex3f(x, size_of_window_, -size_of_window_);
    glVertex3f(x, -size_of_window_, -size_of_window_);
    glVertex3f(x, -size_of_window_, size_of_window_);
    glEnd();
```

```
    }
}

void drawTreeAt(float X, float Y, float Z) {
    fill_color_RGB_VALS(0, 255, 0);
    glPushMatrix();
    glTranslated(X, Y + 15, Z);
    glRotated(90, -1.0, 0.0, 0.0);
    glutSolidCone(3, 6, 50, 50);
    glPopMatrix();
    glPushMatrix();
    glTranslated(X, Y + 13, Z);
    glRotated(90, -1.0, 0.0, 0.0);
    glutSolidCone(3.5, 5, 50, 50);
    glPopMatrix();
    glPushMatrix();
    glTranslated(X, Y + 11, Z);
    glRotated(90, -1.0, 0.0, 0.0);
    glutSolidCone(4, 4, 50, 50);
    glPopMatrix();
    glPushMatrix();
    fill_color_RGB_VALS(200, 100, 20);
    glTranslated(X, Y + 7, Z);
    GLUquadricObj *quadratic;
    quadratic = gluNewQuadric();
    glRotatef(90.0f, -1.0f, 0.0f, 0.0f);
    gluCylinder(quadratic, 1.0, 1.0, 6.0, 32, 32);
    glPopMatrix();
}

void flatSurface(float X) {
    fill_color_RGB_VALS(194, 178, 128);
```

```
    glBegin(GL_POLYGON);
    glVertex3f(-X, -height_of_house_, -X);
    glVertex3f(X, -height_of_house_, -X);
    glVertex3f(X, -height_of_house_, X);
    glVertex3f(-X, -height_of_house_, X);
    glEnd();
}

void drawRoadAt(float l) {
    fill_color_RGB_VALS(128, 128, 128);
    glBegin(GL_POLYGON);
    glVertex3f(-l, -height_of_house_ + 0.01, 3 * height_of_house_);
    glVertex3f(l, -height_of_house_ + 0.01, 3 * height_of_house_);
    glVertex3f(l, -height_of_house_ + 0.01, 6 * height_of_house_);
    glVertex3f(-l, -height_of_house_ + 0.01, 6 * height_of_house_);
    glEnd();
}

void drawHouse() {
    fill_color_RGB_VALS(255, 255, 255);
    glBegin(GL_POLYGON);

    glVertex3f(reference_point + width_of_house_, reference_point -
height_of_house_, reference_point - width_of_house_);
    glVertex3f(reference_point + width_of_house_, reference_point +
height_of_house_, reference_point - width_of_house_);
    glVertex3f(reference_point - width_of_house_, reference_point +
height_of_house_, reference_point - width_of_house_);
    glVertex3f(reference_point - width_of_house_, reference_point -
height_of_house_, reference_point - width_of_house_);
    glEnd();
```

```
    glColor3f(house_r_val, house_g_val, house_b_val);

    wall_with_door(width_of_house_);

    wall_with_window_at_position(width_of_house_);

    wall_with_window_at_position(-width_of_house_);

    fill_color_RGB_VALS(255, 0, 0);

    glBegin(GL_POLYGON);
    glVertex3f(reference_point + width_of_house_, reference_point -
height_of_house_ + 0.01, reference_point + width_of_house_);
    glVertex3f(reference_point + width_of_house_, reference_point -
height_of_house_ + 0.01, reference_point - width_of_house_);
    glVertex3f(reference_point - width_of_house_, reference_point -
height_of_house_ + 0.01, reference_point - width_of_house_);
    glVertex3f(reference_point - width_of_house_, reference_point -
height_of_house_ + 0.01, reference_point + width_of_house_);
    glEnd();

    glColor3f(roof_r_val, roof_g_val, roof_b_val);
    glBegin(GL_POLYGON);
    glVertex3f(reference_point + width_of_roof, reference_point +
height_of_house_, reference_point + width_of_roof);
    glVertex3f(reference_point + width_of_roof, reference_point +
height_of_house_, reference_point - width_of_roof);
    glVertex3f(reference_point - width_of_roof, reference_point +
height_of_house_, reference_point - width_of_roof);
    glVertex3f(reference_point - width_of_roof, reference_point +
height_of_house_, reference_point + width_of_roof);
    glEnd();
```

```
    glBegin(GL_POLYGON);
    glVertex3f(reference_point + width_of_roof, reference_point +
height_of_house_, reference_point + width_of_roof);
    glVertex3f(reference_point + width_of_roof, reference_point +
height_of_house_, reference_point - width_of_roof);
    glVertex3f(reference_point, height_of_roof, reference_point);
    glEnd();

    glBegin(GL_POLYGON);
    glVertex3f(reference_point + width_of_roof, reference_point +
height_of_house_, reference_point - width_of_roof);
    glVertex3f(reference_point - width_of_roof, reference_point +
height_of_house_, reference_point - width_of_roof);
    glVertex3f(reference_point, height_of_roof, reference_point);
    glEnd();

    glBegin(GL_POLYGON);
    glVertex3f(reference_point, height_of_roof, reference_point);
    glVertex3f(reference_point - width_of_roof, reference_point +
height_of_house_, reference_point - width_of_roof);
    glVertex3f(reference_point - width_of_roof, reference_point +
height_of_house_, reference_point + width_of_roof);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex3f(reference_point + width_of_roof, reference_point +
height_of_house_, reference_point + width_of_roof);
    glVertex3f(reference_point, height_of_roof, reference_point);
    glVertex3f(reference_point - width_of_roof, reference_point +
height_of_house_, reference_point + width_of_roof);
    glEnd();
```

```
    drawTableAt(-width_of_house_ + 2, -height_of_house_ + 1.5,
-width_of_house_ + 2);
    drawBedAt(0.5 * width_of_house_, -height_of_house_ + 0.6, 0);
    drawDoorAtPosition();
    window();
}

void lightSourceAt(float X, float Y, float Z) {
    GLfloat positionOfLight[] = {X, Y, Z, 0.0};
    GLfloat ambienceLight[] = {0.0, 0.0, 0.0, 1.0};
    GLfloat specularLight[] = {1.0, 1.0, 1.0, 1.0};
    GLfloat diffuseLight[] = {1.0, 1.0, 1.0, 1.0};

    GLfloat specular_material_properties[] = {0, 0, 0, 1};
    GLfloat emission_material_properties[] = {0, 0, 0, 1};

    glEnable(GL_LIGHTING);
    glEnable(GL_COLOR_MATERIAL);
    glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);

    glMaterialfv(GL_FRONT, GL_SPECULAR,
specular_material_properties);
    glMaterialfv(GL_FRONT, GL_EMISSION,
emission_material_properties);

    if (light_intensity_ == 1) {
        glEnable(GL_LIGHT0);
        glLightfv(GL_LIGHT0, GL_POSITION, positionOfLight);
        glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseLight);
        glLightfv(GL_LIGHT0, GL_AMBIENT, ambienceLight);
        glLightfv(GL_LIGHT0, GL_SPECULAR, specularLight);
    } else {
```

```
    glDisable(GL_LIGHT0);
}
if (light_intensity_ == 2) {
    glEnable(GL_LIGHT1);
    glLightfv(GL_LIGHT1, GL_POSITION, positionOfLight);
    glLightfv(GL_LIGHT1, GL_DIFFUSE, diffuseLight);
    glLightfv(GL_LIGHT1, GL_AMBIENT, ambienceLight);
    glLightfv(GL_LIGHT1, GL_SPECULAR, specularLight);
}
if (light_intensity_ == 3) {
    glEnable(GL_LIGHT2);
    glLightfv(GL_LIGHT2, GL_POSITION, positionOfLight);
    glLightfv(GL_LIGHT2, GL_DIFFUSE, diffuseLight);
    glLightfv(GL_LIGHT2, GL_AMBIENT, ambienceLight);
    glLightfv(GL_LIGHT2, GL_SPECULAR, specularLight);
}
if (light_intensity_ == 4) {
    glEnable(GL_LIGHT3);
    glLightfv(GL_LIGHT3, GL_POSITION, positionOfLight);
    glLightfv(GL_LIGHT3, GL_DIFFUSE, diffuseLight);
    glLightfv(GL_LIGHT3, GL_AMBIENT, ambienceLight);
    glLightfv(GL_LIGHT3, GL_SPECULAR, specularLight);
}
if (light_intensity_ == 5) {
    glEnable(GL_LIGHT4);
    glLightfv(GL_LIGHT4, GL_POSITION, positionOfLight);
    glLightfv(GL_LIGHT4, GL_DIFFUSE, diffuseLight);
    glLightfv(GL_LIGHT4, GL_AMBIENT, ambienceLight);
    glLightfv(GL_LIGHT4, GL_SPECULAR, specularLight);
}
if (light_intensity_ == 6) {
    glEnable(GL_LIGHT5);
```

```
        glLightfv(GL_LIGHT5, GL_POSITION, positionOfLight);
        glLightfv(GL_LIGHT5, GL_DIFFUSE, diffuseLight);
        glLightfv(GL_LIGHT5, GL_AMBIENT, ambienceLight);
        glLightfv(GL_LIGHT5, GL_SPECULAR, specularLight);
    }
}

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

    glLoadIdentity();
    camera[0] += front_back * sin(phi_angle);
    camera[2] -= front_back * cos(phi_angle);
    front_back = 0;
    float lookat[] = {radius * sin(phi_angle) + camera[0], radius *
cos(angle_theta) + camera[1], -radius * cos(phi_angle) + camera[2]};
    gluLookAt(camera[0], camera[1], camera[2], lookat[0], lookat[1],
lookat[2], 0, 1, 0);

    glPushMatrix();
    lightSourceAt(30, 30, 30);
    flatSurface(1000);
    drawTreeAt(5 * width_of_house_, -3 * height_of_house_,
width_of_house_);
    drawHouse();
    drawRoadAt(300);

    glPopMatrix();

    glutSwapBuffers();
}
```

```
void mouse_press_function(int button, int state, int x, int y) {
    if (button == GLUT_LEFT_BUTTON) {
        if (state == GLUT_UP) {
            is_mouse_pressed_bool = false;
            start_position_x = -1;
            start_position_y = -1;
        } else {
            is_mouse_pressed_bool = true;
        }
    }
    if (state == GLUT_DOWN) {
        switch (button) {
            case 3:
                radius -= 0.5;
                break;
            case 4:
                radius += 0.5;
                break;
            default:
                break;
        }
    }
}

void track_mouse_movement_function(int x, int y) {
    if (start_position_x == -1)
        start_position_x = x;
    if (start_position_y == -1)
        start_position_y = y;
    if (is_mouse_pressed_bool) {
        angle_theta += (y - start_position_y) * delta_change_unit * 0.015;
```

```
        phi_angle -= (x - start_position_x) * delta_change_unit * 0.015;

        if (phi_angle > 2 * PI)
            phi_angle -= 2 * PI;
        if (phi_angle < 0)
            phi_angle += 2 * PI;
        if (angle_theta > 2 * PI)
            angle_theta -= 2 * PI;
        if (angle_theta < 0)
            angle_theta += 2 * PI;
    }
    start_position_x = x;
    start_position_y = y;
}

void keyboardFunction(unsigned char key, int x, int y) {
    switch (key) {
        case 'z':
            light_intensity_ += 1;
            break;
        case 'x':
            light_intensity_ -= 1;
            break;
    }
    switch (key) {
        case 'w':
            front_back += 1;
            break;
        case 's':
            front_back -= 1;
            break;
    }
```

```
    switch (key) {
        case 'f':
            flip = !flip;
            break;
        case 'c':
            window1 = !window1;
            break;
        case 'v':
            window2 = !window2;
            break;
    }
}

void keyboardFunction2(int key_int, int x, int y) {
    switch (key_int) {
        case GLUT_KEY_UP:
            angle_theta -= delta_change_unit * 0.3;
            break;
        case GLUT_KEY_DOWN:
            angle_theta += delta_change_unit * 0.3;
            break;
    }
    switch (key_int) {
        case GLUT_KEY_RIGHT:
            if (angle_theta >= (3 * PI / 2) || angle_theta <= (PI / 2))
                phi_angle -= delta_change_unit * 0.3;
            else
                phi_angle += delta_change_unit * 0.3;
            break;
        case GLUT_KEY_LEFT:
            if (angle_theta >= (3 * PI / 2) || angle_theta <= (PI / 2))
                phi_angle += delta_change_unit * 0.3;
```

```
            else
                phi_angle -= delta_change_unit * 0.3;
            break;
        }
        if (phi_angle > 2 * PI)
            phi_angle -= 2 * PI;
        if (phi_angle < 0)
            phi_angle += 2 * PI;
        if (angle_theta > 2 * PI)
            angle_theta -= 2 * PI;
        if (angle_theta < 0)
            angle_theta += 2 * PI;
    }

    int main(int c, char *v[]) {
        glutInit(&c, v);
        glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE |
    GLUT_RGBA);
        glutInitWindowSize(WIDTH, HEIGHT);
        glutCreateWindow("Assignment9");

        glutDisplayFunc(drawFunction);
        glutReshapeFunc(change_size_of_window);
        glutIdleFunc(drawFunction);

        glutMouseFunc(mouse_press_function);
        glutMotionFunc(track_mouse_movement_function);
        glutKeyboardFunc(keyboardFunction);
        glutSpecialFunc(keyboardFunction2);

        glEnable(GL_DEPTH_TEST);
        glutMainLoop();
```

```
    return 0;
}
```