Daffodil International University

Course: Computer Graphic Lab [CSE-422]
Project Report

Submitted To

Name: Deawan Rakin Ahamed Remal

Designation: Lecturer

Department: CSE

Date of Submission: Monday, November 27, 2023.

Team Members:

Name	ID
Tasnim Umaer Tisha	203-15-14520
Golam Kibria	203-15-14522
Tanven Arefin Sazid	203-15-14526

Contents

Title: Countryside Canvas

Project Introduction:

In out computer graphics project, we created a colorful world using OpenGL and GLUT. Picture a cozy house on hills with lots of trees around. It can change magically – sometimes it's a quiet night with a big moon other time, it's raining with droplets falling down. There's also a road, a car moving around, and everything is bright when the sun's out. It's like a little story told with pictures, showing how computer graphics can make different scenes, from daytime to nighttime and even in the rain.

Contents:

Here's a brief description of each function we used in our project:

- **Circle** Function: Draws a circle with a specified radius.
- **Quads** Function: Draws a quadrilateral (four-sided polygon).
- **Line** Function: Draws a straight line between two points.
- **Polygon** Function: Draws a polygon with a specified number of sides.
- **Triangle** Function: Draws a triangle using three specified points.
- **PushMatrix** and **PopMatrix**:

  - `PushMatrix`: Saves the current transformation state.

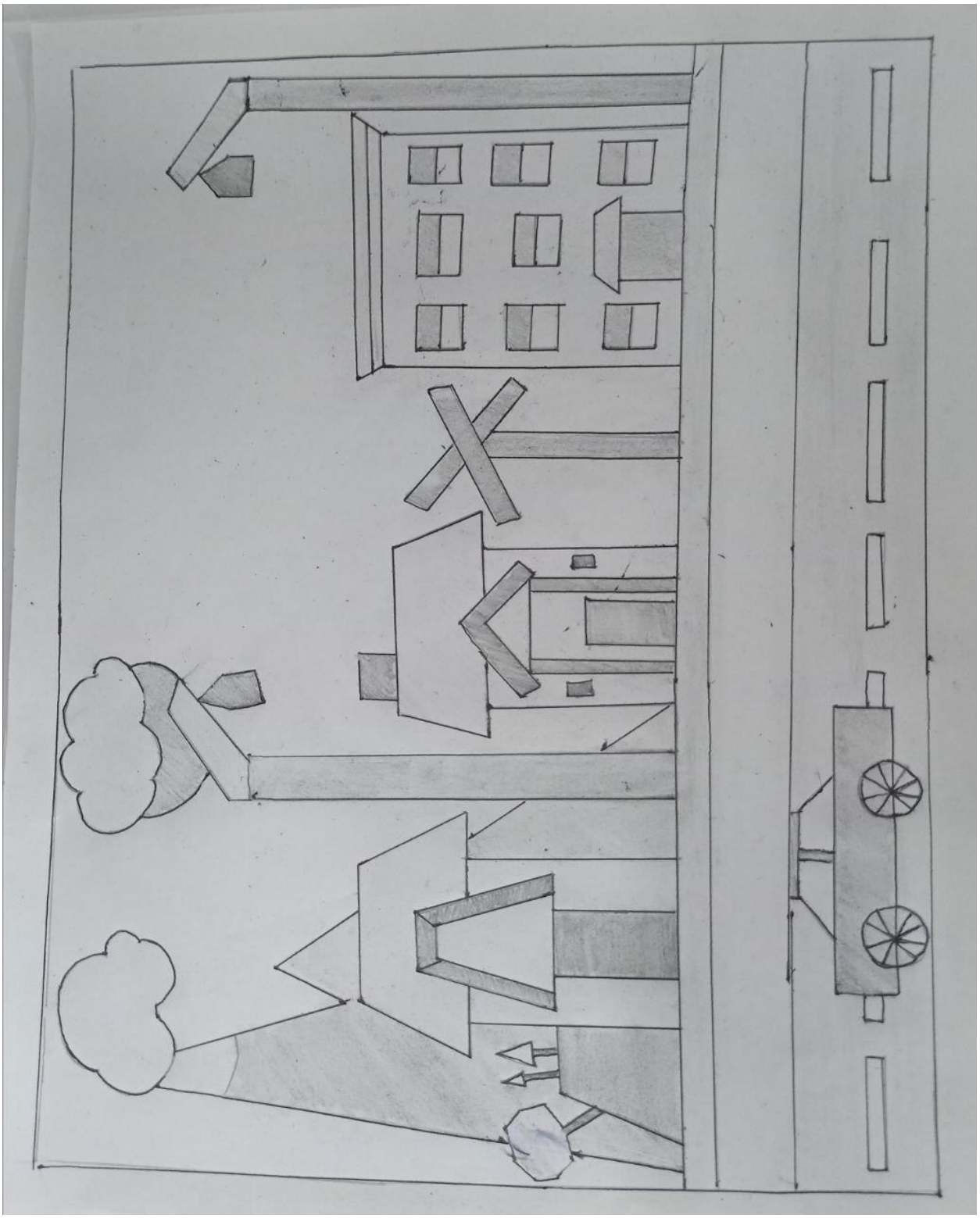
  - `PopMatrix`: Restores the transformation state saved by `PushMatrix`.

These functions are fundamental in creating diverse shapes and managing transformations within your computer graphics project, enabling you to build a rich and dynamic visual environment.

We also use this user define function in our project which helps us to maintain and track our project easily.

- big_hill(): Renders a large hill in the scene.
- house_one(): Creates the first type of house.
- house_two(): Generates the second type of house.
- house_three(): Displays the third type of house.
- first_small_square(): Draws the first small square shape.
- second_small_square(): Draws the second small square shape.
- hill_house_one_left(): Integrates a house on the left side of the hill.
- road(): Renders a road in the scene.
- moon(): Illustrates the moon in the sky.
- purple_moon(): Presents a stylized representation of the moon in a different appearance.
- clouds(): Generates clouds in the sky.
- car(): Renders a car, contributing to the dynamic aspect of the scene.
- lamp_post1(): Creates the first type of lamp post.
- lamp_post2(): Creates the second type of lamp post.

Initial Sketch:



Code: https://ideone.com/CnoegV

```
#include<cstdio>
#include <windows.h>
#include<math.h>
#include <vector>
#include <cstdlib>
# define PI 3.14159265358979323846
#include <GL/gl.h>
#include <GL/glut.h>
#include<MMSystem.h>
//////////////////////////////////////////test
#include <GL/glut.h>
#include <vector>
#include <cstdlib>
#include <ctime>


// Define a structure to represent a raindrop
struct Raindrop {
    float x, y;


    Raindrop(float _x, float _y) : x(_x), y(_y) {}
```

```
};

// Constants
const int screenWidth = 2000;
const int screenHeight = 1000;
const int numRaindrops = 12000;

// Vector to store raindrops
std::vector<Raindrop> raindrops;

// Function to draw a raindrop at (x, y)
void drawRaindrop(float x, float y) {
    glBegin(GL_LINES);
    glVertex2f(x, y);
    glVertex2f(x, y - 3); // Adjust length of raindrop
    glEnd();
}
////////////////////////////////////////////

float x=0;
float y=0;
```

```
void init(void)
{
   glClearColor(0.686, 0.886, 0.961, 1.0);
   glMatrixMode(GL_PROJECTION);
   gluOrtho2D(0.0, 100.0, 0.0, 100.0);
}

void init2(void)
{
   glClearColor(0.0, 0.0, 0.0, 1.0);
   glMatrixMode(GL_PROJECTION);
   gluOrtho2D(0.0, 100.0, 0.0, 100.0);
}

void init4(void)
{
   glClearColor(0.4, 0.082, 0.439, 1.0);
   glMatrixMode(GL_PROJECTION);
   gluOrtho2D(0.0, 100.0, 0.0, 100.0);
}
```

```
//=====================================
void big_hill(void)
{
   glBegin(GL_TRIANGLES);
   glColor3f(0.408, 0.424, 0.361);
   glVertex2i(6, 35);
   glVertex2i(21, 75);
   glVertex2i(37, 35);
   glEnd();

   glBegin(GL_TRIANGLES);
   glColor3f(0.408, 0.424, 0.361);
   glVertex2i(12, 82);
   glVertex2i(1, 35);
   glVertex2i(25, 35);
   glEnd();

   //big hill white part
   glBegin(GL_POLYGON);
   glColor3f(1, 1, 1);
```

```
    glVertex2i(12, 74);
    glVertex2f(11.5, 75);
    glVertex2f(10.3, 75);
    glVertex2i(12, 82);
    glVertex2i(14, 75);
    glVertex2i(13, 74);
    glEnd();
}
void house_one(void)
{
    glBegin(GL_QUADS);
    glColor3f(0.745, 0.753, 0.757);
    glVertex2i(15, 55);
    glVertex2i(15, 35);
    glVertex2i(30, 35);
    glVertex2i(30, 55);
    glEnd();

    //chall
    glBegin(GL_QUADS);
    glColor3f(0.831, 0.188, 0.314);
```

```
glVertex2i(17, 63);
glVertex2i(13, 53);
glVertex2i(32, 53);
glVertex2i(28, 63);
glEnd();

//door
glBegin(GL_QUADS);
glColor3f(0.588, 0.416, 0.251);
glVertex2i(20, 45);
glVertex2i(20, 35);
glVertex2i(25, 35);
glVertex2i(25, 45);
glEnd();

//tribuj-left
glBegin(GL_QUADS);
glColor3f(1, 1, 1);
glVertex2i(20, 57);
glVertex2i(16, 45);
glVertex2i(18, 45);
```

```
glVertex2i(21, 55);
glEnd();

//tribuj-right
glBegin(GL_QUADS);
glColor3f(1, 1, 1);
glVertex2i(24, 55);
glVertex2i(27, 45);
glVertex2i(29, 45);
glVertex2i(25, 57);
glEnd();

//tribuj-middle
glBegin(GL_QUADS);
glColor3f(0.125, 0, 1);
glVertex2i(20, 57);
glVertex2i(21, 55);
glVertex2i(24, 55);
glVertex2i(25, 57);
glEnd();
```

```
    //lines
    glBegin(GL_LINES);
    glColor3f(1, 0, 1);
    glVertex2i(16, 45);
    glVertex2i(29, 45);
    glEnd();

    //middle-pentagon
    glBegin(GL_QUADS);
    glColor3f(0.871, 0.878, 0.886);
    glVertex2f(20.4, 53);
    glVertex2i(18, 45);
    glVertex2i(27, 45);
    glVertex2f(24.6, 53);
    glEnd();
}
////////////////////////
void hill_house_one_left(void)
{
    glBegin(GL_QUADS);
    glColor3f(0.694, 1, 0);
```

```
glVertex2i(8, 45);
glVertex2i(4, 35);
glVertex2i(15, 35);
glVertex2i(15, 45);
glEnd();

//left-hill-tree-01
glLineWidth(3.0);
glBegin(GL_LINES);
glColor3f(0, 0, 0);
glVertex2i(10, 50);
glVertex2i(10, 45);
glEnd();

//tree-01-leat
glLineWidth(3.0);
glBegin(GL_TRIANGLES);
glColor3f(0.239, 0.678, 0.208);
glVertex2i(10, 51);
glVertex2i(9, 48);
glVertex2i(11, 48);
```

```
glEnd();

//left-hill-tree-02
glLineWidth(3.0);
glBegin(GL_LINES);
glColor3f(0, 0, 0);
glVertex2i(13, 45);
glVertex2i(13, 47);
glEnd();

//tree-02-leat
glLineWidth(3.0);
glBegin(GL_TRIANGLES);
glColor3f(0.239, 0.678, 0.208);
glVertex2i(13, 52);
glVertex2i(12, 47);
glVertex2i(14, 47);
glEnd();

//left-hill-tree-03-polygon
glLineWidth(10.0);
```

```
    glBegin(GL_LINES);
    glColor3f(0, 0, 0);
    glVertex2i(4, 42);
    glVertex2i(5, 38);
    glEnd();
    glBegin(GL_POLYGON);
    glColor3f(0.239, 0.678, 0.208);
    glVertex2i(3, 41);
    glVertex2i(2, 42);
    glVertex2i(2, 44);
    glVertex2i(3, 45);
    glVertex2i(5, 45);
    glVertex2i(6, 44);
    glVertex2i(6, 42);
    glVertex2i(5, 41);
    glVertex2i(4, 41);
    glEnd();
}
void house_two(void)
{
    //body
```

```
glBegin(GL_POLYGON);
glColor3f(0, 0.196, 0.235);
glVertex2i(35,50);
glVertex2i(37, 59);
glVertex2i(51, 59);
glVertex2i(54, 50);
glEnd();

//body
glBegin(GL_POLYGON);
glColor3f(0.984, 0.706, 0.38);
glVertex2i(37,50);
glVertex2i(52,50);
glVertex2i(52,35);
glVertex2i(37,35);
glEnd();

//uporer danda
glBegin(GL_POLYGON);
glColor3f(0.114, 0.596, 0.635);
glVertex2i(39, 46);
```

```
glVertex2i(38, 47);
glVertex2i(44, 53);
glVertex2i(45.07, 52.08);
glEnd();

//uporer danda
glBegin(GL_POLYGON);
glColor3f(0.114, 0.596, 0.635);
glVertex2i(45.07, 52.08);
glVertex2i(51, 47);
glVertex2i(50, 46);
glVertex2i(44,51);
glEnd();

//lomba danda
glBegin(GL_POLYGON);
glColor3f(0.749, 0.757, 0.761);
glVertex2i(40,47);
glVertex2i(41, 47);
glVertex2i(41, 35);
glVertex2i(40, 35);
```

```
glEnd();

//lomba danda
glBegin(GL_POLYGON);
glColor3f(0.749, 0.757, 0.761);
glVertex2i(48,47);
glVertex2i(49, 47);
glVertex2i(49, 35);
glVertex2i(48, 35);
glEnd();

//
glBegin(GL_POLYGON);
glColor3f(0.871, 0.878, 0.886);
glVertex2i(44,51);
glVertex2i(40,47);
glVertex2i(49,47);
glEnd();

//dorja
glBegin(GL_POLYGON);
```

```
glColor3f(0.459, 0.239, 0.169);
glVertex2i(43,43);
glVertex2i(46, 43);
glVertex2i(46,35);
glVertex2i(43,35);
glEnd();

//janala
glBegin(GL_POLYGON);
glColor3f(0.278, 0.643, 0.804);
glVertex2i(38, 43);
glVertex2i(39, 43);
glVertex2i(39,40);
glVertex2i(38, 40);
glEnd();

//janla
glBegin(GL_POLYGON);
glColor3f(0.278, 0.643, 0.804);
glVertex2i(50, 43);
glVertex2i(51, 43);
```

```
    glVertex2i(51,40);
    glVertex2i(50, 40);
    glEnd();

    //chimney
    glBegin(GL_POLYGON);
    glColor3f(0.929, 0.498, 0.345);
    glVertex2i(39, 63);
    glVertex2i(42, 63);
    glVertex2i(42, 59);
    glVertex2i(39,59);
    glEnd();
}

void house_three(void)
{
    glBegin(GL_QUADS);
    glColor3f(0.824, 0.573, 0.471);
    glVertex2i(70, 35);
    glVertex2i(70, 62);
    glVertex2i(95, 62);
```

```
glVertex2i(95, 35);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0, 0, 1);
glVertex2i(72, 38);
glVertex2i(76, 38);
glVertex2i(76, 41);
glVertex2i(72, 41);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.976, 0.949, 0.584);
glVertex2i(76, 41);
glVertex2i(72, 41);
glVertex2i(72, 44);
glVertex2i(76, 44);
glEnd();
```

```
glBegin(GL_POLYGON);
glColor3f(0, 0, 1);
glVertex2i(76, 46);
glVertex2i(72, 46);
glVertex2i(72, 49);
glVertex2i(76, 49);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.976, 0.949, 0.584);
glVertex2i(76, 49);
glVertex2i(72, 49);
glVertex2i(72, 52);
glVertex2i(76, 52);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0, 0, 1);
glVertex2i(76, 54);
glVertex2i(72, 54);
glVertex2i(72, 57);
```

```
glVertex2i(76, 57);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.976, 0.949, 0.584);
glVertex2i(76, 57);
glVertex2i(72, 57);
glVertex2i(72, 60);
glVertex2i(76, 60);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0, 0, 1);
glVertex2i(89, 38);
glVertex2i(93, 38);
glVertex2i(93, 41);
glVertex2i(89, 41);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.976, 0.949, 0.584);
```

```
glVertex2i(89, 41);
glVertex2i(93, 41);
glVertex2i(93, 44);
glVertex2i(89, 44);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0, 0, 1);
glVertex2i(89, 46);
glVertex2i(93, 46);
glVertex2i(93, 49);
glVertex2i(89, 49);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.976, 0.949, 0.584);
glVertex2i(89, 49);
glVertex2i(93, 49);
glVertex2i(93, 52);
glVertex2i(89, 52);
glEnd();
```

```
glBegin(GL_POLYGON);
glColor3f(0, 0, 1);
glVertex2i(89, 54);
glVertex2i(93, 54);
glVertex2i(93, 57);
glVertex2i(89, 57);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.976, 0.949, 0.584);
glVertex2i(89, 57);
glVertex2i(93, 57);
glVertex2i(93, 60);
glVertex2i(89, 60);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0, 0, 1);
glVertex2i(80, 46);
glVertex2i(85, 46);
```

```
glVertex2i(85, 49);
glVertex2i(80, 49);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.976, 0.949, 0.584);
glVertex2i(80, 49);
glVertex2i(85, 49);
glVertex2i(85, 52);
glVertex2i(80, 52);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0, 0, 1);
glVertex2i(80, 54);
glVertex2i(85, 54);
glVertex2i(85, 57);
glVertex2i(80, 57);
glEnd();

glBegin(GL_POLYGON);
```

```
glColor3f(0.976, 0.949, 0.584);
glVertex2i(80, 57);
glVertex2i(85, 57);
glVertex2i(85, 60);
glVertex2i(80, 60);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0, 0, 1);
glVertex2i(80, 43);
glVertex2i(85, 43);
glVertex2i(86, 41);
glVertex2i(79, 41);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.976, 0.949, 0.584);
glVertex2i(85, 41);
glVertex2i(85, 35);
glVertex2i(80, 35);
glVertex2i(80, 41);
```

```
    glEnd();


    glBegin(GL_POLYGON);
    glColor3f(0, 0, 1);
    glVertex2i(95, 62);
    glVertex2i(70, 62);
    glVertex2i(69, 63);
    glVertex2i(96, 63);
    glEnd();


    glBegin(GL_POLYGON);
    glColor3f(0.976, 0.949, 0.584);
    glVertex2i(69, 63);
    glVertex2i(96, 63);
    glVertex2i(97, 64);
    glVertex2i(68, 64);
    glEnd();


}
```

```
//**************************
void first_small_square(void)
{
   glBegin(GL_QUADS);
   glColor3f(0.271, 0.929, 0.455);
   glVertex2i(0, 35);
   glVertex2i(0, 30);
   glVertex2i(100, 30);
   glVertex2i(100, 35);
   glEnd();
}
void second_small_square(void)
{
   glBegin(GL_QUADS);
   glColor3f(0.694, 1, 0);
   glVertex2i(0, 30);
   glVertex2i(0, 20);
   glVertex2i(100, 20);
   glVertex2i(100, 30);
   glEnd();
}
```

```
void road(void)
{
   //road black part
   glBegin(GL_QUADS);
   glColor3f(0, 0, 0);
   glVertex2i(0, 20);
   glVertex2i(0, 0);
   glVertex2i(100, 0);
   glVertex2i(100, 20);
   glEnd();

   //road white part-1
   glBegin(GL_QUADS);
   glColor3f(1, 1, 1);
   glVertex2i(2, 11);
   glVertex2i(2, 9);
   glVertex2i(10, 9);
   glVertex2i(10, 11);
   glEnd();

   //road white part-2
```

```
glBegin(GL_QUADS);
glColor3f(1, 1, 1);
glVertex2i(12, 11);
glVertex2i(12, 9);
glVertex2i(20, 9);
glVertex2i(20, 11);
glEnd();

//road white part-3
glBegin(GL_QUADS);
glColor3f(1, 1, 1);
glVertex2i(22, 11);
glVertex2i(22, 9);
glVertex2i(30, 9);
glVertex2i(30, 11);
glEnd();

//road white part-4
glBegin(GL_QUADS);
glColor3f(1, 1, 1);
glVertex2i(32, 11);
```

```
glVertex2i(32, 9);
glVertex2i(40, 9);
glVertex2i(40, 11);
glEnd();

//road white part-5
glBegin(GL_QUADS);
glColor3f(1, 1, 1);
glVertex2i(42, 11);
glVertex2i(42, 9);
glVertex2i(50, 9);
glVertex2i(50, 11);
glEnd();

//road white part-6
glBegin(GL_QUADS);
glColor3f(1, 1, 1);
glVertex2i(52, 11);
glVertex2i(52, 9);
glVertex2i(60, 9);
glVertex2i(60, 11);
```

```
glEnd();

//road white part-7
glBegin(GL_QUADS);
glColor3f(1, 1, 1);
glVertex2i(62, 11);
glVertex2i(62, 9);
glVertex2i(70, 9);
glVertex2i(70, 11);
glEnd();

//road white part-8
glBegin(GL_QUADS);
glColor3f(1, 1, 1);
glVertex2i(72, 11);
glVertex2i(72, 9);
glVertex2i(80, 9);
glVertex2i(80, 11);
glEnd();

//road white part-9
```

```
    glBegin(GL_QUADS);
    glColor3f(1, 1, 1);
    glVertex2i(82, 11);
    glVertex2i(82, 9);
    glVertex2i(90, 9);
    glVertex2i(90, 11);
    glEnd();

    //road white part-10
    glBegin(GL_QUADS);
    glColor3f(1, 1, 1);
    glVertex2i(92, 11);
    glVertex2i(92, 9);
    glVertex2i(99, 9);
    glVertex2i(99, 11);
    glEnd();
}
//==============================
void lamp_post1(void) {

//main big line
```

```
glLineWidth(10.0);
glBegin(GL_LINES);
glColor3f(0.027, 0.125, 0.361);
glVertex2i(33,35);
glVertex2i(33,65);
glEnd();

//front quad
glBegin(GL_POLYGON);
glColor3f(0.976, 0.949, 0.584);
glVertex2i(30,65);
glVertex2i(28,71);
glVertex2i(38,71);
glVertex2i(36,65);
glEnd();

//side quad
glBegin(GL_QUADS);
glColor3f(0.976, 0.949, 0.584);
glVertex2i(28,71);
glVertex2i(27,70);
```

```
glVertex2i(29,65);
glVertex2i(30,65);
glEnd();

glBegin(GL_LINES);
glColor3f(0.0,0.0,0.0);
glVertex2i(28,71);
glVertex2i(30,65);
glEnd();

glBegin(GL_LINES);
glColor3f(0.0,0.0,0.0);
glVertex2i(38,71);
glVertex2i(36,65);
glEnd();

glBegin(GL_LINES);
glColor3f(0.0,0.0,0.0);
glVertex2i(28,71);
glVertex2i(38,71);
glEnd();
```

```
glBegin(GL_LINES);
glColor3f(0.0,0.0,0.0);
glVertex2i(29,65);
glVertex2i(36,65);
glEnd();

glBegin(GL_LINES);
glColor3f(0.0,0.0,0.0);
glVertex2i(28,71);
glVertex2i(27,70);
glEnd();

glBegin(GL_LINES);
glColor3f(0.0,0.0,0.0);
glVertex2i(27,70);
glVertex2i(29,65);
glEnd();

//uporer quad
glBegin(GL_QUADS);
```

```
    glColor3f(0.027, 0.125, 0.361);
    glVertex2i(28,71);
    glVertex2i(38,71);
    glVertex2i(36,74);
    glVertex2i(30,74);
    glEnd();

}


void lamp_post2(void) {

    //main big line
    glLineWidth(10.0);
    glBegin(GL_LINES);
    glColor3f(0.027, 0.125, 0.361);
    glVertex2i(62,65);
    glVertex2i(62,35);
    glEnd();

    //front quad
```

```
glBegin(GL_QUADS);
glColor3f(0.976, 0.949, 0.584);
glVertex2i(59, 65);
glVertex2i(57,71);
glVertex2i(67,71);
glVertex2i(65,65);
glEnd();

//side quads
glBegin(GL_QUADS);
glColor3f(0.976, 0.949, 0.584);
glVertex2i(56,70);
glVertex2i(57,71);
glVertex2i(59,65);
glVertex2i(58,65);
glEnd();

//upper quad
glBegin(GL_QUADS);
glColor3f(0.027, 0.125, 0.361);
glVertex2i(57,71);
```

```
glVertex2i(59,74);
glVertex2i(65,74);
glVertex2i(67,71);
glEnd();

glBegin(GL_LINES);
glColor3f(0.0, 0.0, 0.0);
glVertex2i(57,71);
glVertex2i(67,71);
glEnd();

glBegin(GL_LINES);
glColor3f(0.0, 0.0, 0.0);
glVertex2i(57,71);
glVertex2i(59,65);
glEnd();

glBegin(GL_LINES);
glColor3f(0.0, 0.0, 0.0);
glVertex2i(56,70);
glVertex2i(58,65);
```

```
    glEnd();

    glBegin(GL_LINES);
    glColor3f(0.0, 0.0, 0.0);
    glVertex2i(56,70);
    glVertex2i(57,71);
    glEnd();

    glBegin(GL_LINES);
    glColor3f(0.0, 0.0, 0.0);
    glVertex2i(58,65);
    glVertex2i(65,65);
    glEnd();

    glBegin(GL_LINES);
    glColor3f(0.0, 0.0, 0.0);
    glVertex2i(65,65);
    glVertex2i(67,71);
    glEnd();
}
//=========================
```

```
void circle(GLfloat rx, GLfloat ry, GLfloat cx, GLfloat cy)
{
   glBegin(GL_POLYGON);
   glVertex2f(cx, cy);
   for (int i = 0; i <= 360; i++)
   {
      float angle = i * 3.1416 / 180;
      float x = rx * cos(angle);
      float y = ry * sin(angle);
      glVertex2f((x + cx), (y + cy));
   }
   glEnd();
}
void circle2(GLfloat rx, GLfloat ry, GLfloat cx, GLfloat cy)
{
   glBegin(GL_POLYGON);
   glVertex2f(cx, cy);
   for (int i = 0; i <= 360; i++)
   {
      float angle = i * 3.1416 / 180;
      float x = rx * cos(angle);
```

```
        float y = ry * sin(angle);
        glVertex2f((x + cx), (y + cy));
    }
    glEnd();
}
void circle3(GLfloat rx, GLfloat ry, GLfloat cx, GLfloat cy)
{
    glBegin(GL_POLYGON);
    glVertex2f(cx, cy);
    for (int i = 0; i <= 360; i++)
    {
        float angle = i * 3.1416 / 180;
        float x = rx * cos(angle);
        float y = ry * sin(angle);
        glVertex2f((x + cx), (y + cy));
    }
    glEnd();
}
float bx = 10;
float ax = 10;
void surjo(void)
```

```
{
   //sun design

   glColor3f(0.949, 0.537, 0);
   circle(5, 8, 35, 85);

}
void chad(void)
{

   glColor3f(0.961, 0.91, 0.62);
   circle2(5, 8, 20, 85);
}

void chadupor(void)
{

   glColor3f(0, 0, 0);
   circle2(5, 8, 22, 86);
}
```

```
void purple_moon(void)
{

    glColor3f(0.4, 0.082, 0.439);
    circle2(5, 8, 22, 86);
}

void clouds()
{
    glPushMatrix();
    glTranslatef(bx, 0, 0);
    // 1st cloud
    glColor3f(1, 1, 1);
    circle(3, 5, 60, 85);
    circle(2, 4, 62, 83);
    circle(2, 4, 58, 83);
    circle(2, 3, 64, 82);
    circle(2, 3, 64, 84);
    circle(2, 3, 63, 85);

    // 2nd cloud
```

```
glColor3f(1, 1, 1);
circle(3, 5, 80, 90);
circle(2, 4, 83, 88);
circle(2, 4, 77, 88);
circle(2, 3, 78, 85);
circle(2, 3, 81, 85);
circle(2, 3, 84, 85);

// 3rd cloud
glColor3f(1, 1, 1);
circle(3, 5, 40, 90);
circle(2, 4, 43, 88);
circle(2, 4, 37, 88);
circle(1, 3, 43, 87);

glPopMatrix();
bx += .05;
if (bx > 0)
   bx = -20;
glutPostRedisplay();
```

```
}


void car(void)
{
   glColor3f(1.0, 1.0, 0.0);
   glBegin(GL_POLYGON);
   glVertex2i(5+x,8);
   glVertex2i(5+x,15);
   glVertex2i(8+x,15);
   glVertex2i(12+x,20);
   glVertex2i(23+x,20);
   glVertex2i(26+x,15);
   glVertex2i(30+x,15);
   glVertex2i(30+x,8);
   glVertex2i(5+x,8);
   glEnd();

   glLineWidth(4.0);
   glColor3f(0.941, 0.773, 0.165);
   glBegin(GL_LINES);
```

```
glVertex2i(8+x,15);
glVertex2i(26+x,15);
glEnd();


glColor3f(0.773, 0.953, 1);
glBegin(GL_POLYGON);
glVertex2i(8+x,15);
glVertex2i(12+x,20);
glVertex2i(17+x,20);
glVertex2i(17+x,15);
glEnd();

glColor3f(0.773, 0.953, 1);
glBegin(GL_QUADS);
glVertex2i(17+x,15);
glVertex2i(17+x,20);
glVertex2i(23+x,20);
glVertex2i(26+x,15);
glEnd();
```

```
glLineWidth(6.0);
glColor3f(0.941, 0.773, 0.165);
glBegin(GL_LINES);
glVertex2i(17+x,20);
glVertex2i(17+x,15);
glEnd();

glLineWidth(6.0);
glColor3f(0.455, 0.894, 0.988);
glBegin(GL_LINES);
glVertex2i(12+x,20);
glVertex2i(23+x,20);
glEnd();


glColor3f(0, 0, 0);
circle(2, 4, 25+x, 10);
glColor3f(0.729, 0.729, 0.729);
circle(1.5, 3, 25+x, 10);

glColor3f(0, 0, 0);
```

```
    circle(2, 4, 10+x, 10);
    glColor3f(0.729, 0.729, 0.729);
    circle(1.5, 3, 10+x, 10);

    if(x<100 || y<100)
    {
       x+= 0.01;
       y+= 0.01;
    }
    else
    {
       x=0;
       y=0;
    }
    glutPostRedisplay();
}

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0.141, 0.161, 0.18);
```

```
    big_hill();
    house_one();
    house_two();
    house_three();
    first_small_square();
    second_small_square();
    hill_house_one_left();
    road();
    surjo();
    clouds();
    car();
    lamp_post1();
    lamp_post2();

    glFlush();
}
void display2()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0.0, 0.0, 0.0);
```

```
    big_hill();
    house_one();
    house_two();
    house_three();
    first_small_square();
    second_small_square();
    hill_house_one_left();
    road();
    chad();
    chadupor();
    clouds();
    car();
    lamp_post1();
    lamp_post2();



    glFlush();
}
void display3()
{
    glClear(GL_COLOR_BUFFER_BIT);
```

```
glColor3f(0.0, 0.0, 0.0);
big_hill();
house_one();
house_two();
house_three();
first_small_square();
second_small_square();
hill_house_one_left();
road();
chad();
chadupor();
clouds();
car();
lamp_post1();
lamp_post2();
// Draw each raindrop
glColor3f(0.784, 0.886, 0.973); // White rain
for (const auto& raindrop : raindrops) {
    drawRaindrop(raindrop.x, raindrop.y);
}
glutSwapBuffers();
```

```
    glFlush();
}

////////////////////////////
void display4()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 1.0, 0.0);
    big_hill();
    house_one();
    house_two();
    house_three();
    first_small_square();
    second_small_square();
    hill_house_one_left();
    road();
    chad();
    purple_moon();
    clouds();
```

```
    car();
    lamp_post1();
    lamp_post2();



    glFlush();
}
//=========================================test
// Function to update the raindrop positions
void update(int value) {
    for (auto& raindrop : raindrops) {
        // Move the raindrop down the screen
        raindrop.y -= 5; // Adjust the speed of raindrops


        // Reset raindrop position if it goes below the screen
        if (raindrop.y < 0) {
            raindrop.y = screenHeight;
        }
    }


    glutPostRedisplay();
```

```
    glutTimerFunc(16, update, 0); // 60 frames per second
}
//========================================
void reshape(int width, int height)
{
    glViewport(0, 0, (GLsizei)width, (GLsizei)height);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, 20.0, 0.0, 20.0);
    glMatrixMode(GL_MODELVIEW);
}

void keyboard(unsigned char key, int x, int y)
{
    switch (key)
    {
    case 'n':
        // Create a new window
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(2000, 1000);
        glutInitWindowPosition(0, 0);
```

```
glutCreateWindow("Countryside Canvas- NIGHT VIEW");
//glClearColor(1.0, 1.0, 1.0, 1.0);  // Initial background color is white
init2();
glutDisplayFunc(display2);
//glutReshapeFunc(reshape);
glutKeyboardFunc(keyboard);
break;

 case 'd':
// Create a new window
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(2000, 1000);
glutInitWindowPosition(0, 0);
glutCreateWindow("Countryside Canvas- DAY VIEW");
//glClearColor(1.0, 1.0, 1.0, 1.0);  // Initial background color is white
init();
glutDisplayFunc(display);
//glutReshapeFunc(reshape);
glutKeyboardFunc(keyboard);
break;
```

```
case 'r':
    // Create a new window
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(2000, 1000);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("Countryside Canvas- RAIN VIEW");
    //glClearColor(1.0, 1.0, 1.0, 1.0);  // Initial background color is white
    init2();
    // Seed for random initial raindrop positions
    std::srand(std::time(0));

    // Initialize raindrops with random x and y positions
    for (int i = 0; i < numRaindrops; ++i)
    {
        float x = static_cast<float>(std::rand() % screenWidth);
        float y = static_cast<float>(std::rand() % screenHeight);
        raindrops.push_back(Raindrop(x, y));
    }

    glutDisplayFunc(display3);
```

```
    glutTimerFunc(25, update, 0);
    glutDisplayFunc(display3);
    //glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);
    break;



////////////////////////////////////////////////

case 't':
    // Create a new window
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(2000, 1000);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("Countryside Canvas- PURPLE NIGHT SKY VIEW");
    //glClearColor(1.0, 1.0, 1.0, 1.0);  // Initial background color is white
    init4();
    glutDisplayFunc(display4);
    //glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);
    break;
```

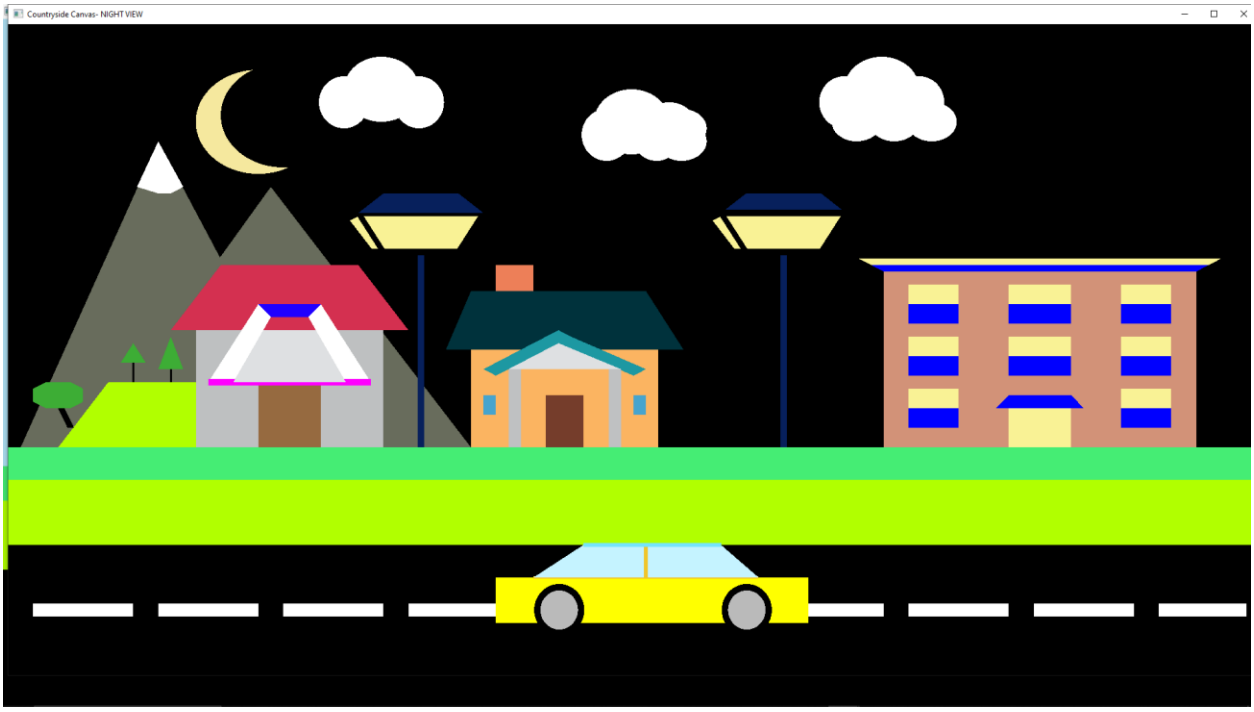
```
        case 27:  // ESC key to exit
        exit(0);
        break;


}


    }



int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    int screen_width = glutGet(GLUT_SCREEN_WIDTH);
    int screen_height = glutGet(GLUT_SCREEN_HEIGHT);
    glutInitWindowSize(screen_width, screen_height);
```

```
    glutInitWindowPosition(0, 0);
    glutCreateWindow("Countryside Canvas- DAY VIEW");

    //glClearColor(1.0, 1.0, 1.0, 1.0);  // Initial background color is white

    glutDisplayFunc(display);
    //glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);
    init();
    glutMainLoop();

    return 0;
}
```

Actual Project Output:

Day View:

Night View:



Rain View:

Purple Night Sky View:

Discussion:

In the canvas of our computer graphics project, the `big_hill()` function emerges as a commanding force, shaping a substantial hill that dominates the visual narrative. As we delve into the architectural tapestry, functions like `house_one()`, `house_two()`, and `house_three()` weave distinct residences, each contributing its unique character to the evolving landscape. Introducing finer details are `first_small_square()` and `second_small_square()`, which delicately embellish the scene with intricate elements.

The deliberate placement of `hill_house_one_left()` strategically situates a dwelling on the hill's left flank, enriching the contextual depth of our graphical environment. The interplay of these structures is seamlessly connected by the artfully crafted `road()`, fostering a sense of visual continuity. Celestial entities, brought to life by `moon()` and `purple_moon()`, infuse the atmosphere with artistic nuances, while dynamic cloud formations conjured by `clouds()` lend a touch of realism.

The addition of a moving `car()` imparts vitality and kinetic energy, elevating the scene to a realm of dynamic realism. Illuminating the virtual world are `lamp_post1()` and `lamp_post2()`, casting artificial light that adds depth and ambiance. Collectively, these meticulously designed functions coalesce to form a visually stunning and multifaceted computer graphics landscape, where each element plays a vital role in the immersive and aesthetically pleasing experience.

[The End]