Computer Graphics (UCS505)

Project on:

Country Side Paradise using OpenGL

Submitted By

Aditya Bhushan	102153013
Jai Dalmotra	102103716
Sheetal	102103700

COE-25

B.E. Third Year – COE

Submitted To: Ms. Archana Kumari

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

Patiala – 147001

Table of Contents

Introduction

The Country Side Paradise, inspired by the captivating world of cellular automata, is a visual exploration crafted using OpenGL. Unlike Conway's Game of Life, which operates on a grid of cells following a set of rules, Country Side Paradise harnesses the power of OpenGL to breathe life into a sprawling countryside paradise. This project transcends a static image; it's an ever-evolving world waiting to be discovered.

Imagine rolling hills and vibrant meadows teeming with life, all generated dynamically through an intricate dance of algorithms. Witness the gentle sway of wildflowers in the breeze, the glistening surface of a winding river, or the majestic flight of a bird across a boundless sky. With each interaction, Country Side Paradise unfolds its secrets, offering a glimpse into a world where the lines between programmer and painter blur, and the canvas is as vast as your imagination.

Rules

1. 0 for Day mode 1for night mode
2. 7 for Rain 6 to stop Rain
3. P to move the plane O to stop the plane
4. T to move the train R to stop the train
5. B to move the boat V to stop the boat

Computer Graphics concepts used

Geometric Primitives:

The world of Countryside Paradise is constructed from basic geometric shapes. We utilize OpenGL's built-in primitives like GL_QUAD (squares), GL_CIRCLE (circles), GL_POLYGON (arbitrary shapes with straight edges), GL_POINT (individual dots), and GL_LINE (straight lines) to form various objects within the scene.

Polygon Filling:

To bring these objects to life, we employ a technique called polygon filling. This process assigns colour to every pixel enclosed within the defined boundaries of a geometric shape (e.g., the train, boat, and plane). This creates the illusion of solid, coloured objects within the virtual world.

Translation and Scaling:

We utilize translation and scaling to position and size our geometric shapes, ensuring a cohesive and realistic world. This allows for objects of varying sizes (like trees vs. hills) to coexist naturally within the scene.

Interactive Control:

Countryside Paradise is not just a static image; it allows for user interaction. By providing keyboard input, users can control the movement of certain elements within the scene, such as starting or stopping the train, boat, or plane. This interactive aspect adds a layer of dynamism and engagement to the overall experience.

User Defined Functions

- **Circle(rx, ry, x, y):** Draws a circle at position (x, y) with radius rx (width) and ry (height).
- **Orange():** Sets drawing colour to orange (likely for drawing something orange).
- **orangeFall():** Animates the falling of an orange object (possibly using orange colour).
- **initRain():** Initializes rain simulation (likely creates raindrop objects).
- **drawRainNight():** Draws rain on the screen in night scene.
- **DrawCircle(cx, cy, r, num_segments):** Draws a circle with centre (cx, cy), radius r, and specified number of segments.
- **DrawLine(x1, y1, x2, y2):** Draws a line from point (x1, y1) to point (x2, y2).
- **DrawPetal(x, y, radius, numSegments):** Draws a petal shape at position (x, y) with specified radius and number of segments.
- **petalFall():** Animates the falling of a petal object (likely using flower petal shape).
- **tree():** Draws a tree on the screen.
- **treenight():** Draws a tree specific to the night scene.
- **city():** Draws a city on the screen.
- **sky():** Draws the sky in the daytime scene.
- **skynight():** Draws the sky in the night scene.
- **sun():** Draws the sun on the screen.
- **river():** Draws a river on the screen.
- **rivernight():** Draws a river specific to the night scene.
- **field():** Draws a field on the screen.
- **fieldnight():** Draws a field specific to the night scene.
- **drawTrainBody():** Draws the body of a train.
- **drawTrainWindows():** Draws windows on the train body.
- **drawTrainWheels():** Draws wheels for the train.
- **bridge():** Draws a bridge on the screen.
- **train(x):** Controls the train animation (likely including movement based on x).
- **DrawRectangle(x, y, width, height):** Draws a rectangle at position (x, y) with specified width and height.
- **house(void):** Draws a house on the screen.

- **drawBoy():** Draws a boy on the screen.
- **boat(int x):** Controls the boat animation (likely including movement based on x).
- **moveboat():** Animates the movement of a boat.
- **plane(int x):** Controls the plane animation (likely including movement based on x).
- **moveplane():** Animates the movement of a plane.
- **cloud(int x):** Draws a cloud at position x.
- **nightcloud(int x):** Draws a cloud specific to the night scene at position x.
- **movecloud():** Animates the movement of clouds.
- **keyboard(key, x, y):** Handles keyboard input (likely for user interaction).
- **moveTrain():** Animates the movement of the train.
- **timer(value):** Handles timer events (likely for animation).
- **initWaterDrops():** Initializes water drops simulation (likely creates water drop objects).
- **moveWaterDrops():** Animates the movement of water drops.
- drawWaterDrops(): Draws water drops on the screen.
- **myDisplay(void):** The main function that controls what gets drawn on the screen.

CODE

```
#include <stdio.h>
#include <iostream>
#include <GL/glut.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>

using namespace std;

int n = 0;
int w = 0;

int rainStatus = 0;
int boatstatus = 0;
int planestatus = 0;
int cloudStatus = 1;
int trainstatus = 0;

float boatX = 0;
float boatY = 0;

float trainX = 0;
float trainY = 0;

float planeX = 0;
float planeY = 0;

float cloudX = 0;
float cloudY = 0;

int petalX = 1500;
int petalY = 650;
int petalFallSpeed = 5.2;

const int minX = 1350;
const int maxX = 1650;

#define MAX_RAIN_DROPS 1000
#define RAIN_SPEED 5

struct RainDrop {
	float x;
	float y;
	bool active;
};

void circle(GLfloat rx, GLfloat ry, GLfloat x, GLfloat y)
{
	int i = 0;
	float angle;
	GLfloat PI = 2.0f * 3.1416;
	glBegin(GL_TRIANGLE_FAN);
	glVertex2f(x, y);
	for (i = 0; i < 100; i++)
	{
		angle = 2 * PI * i / 100;
		glVertex2f(x + (cos(angle) * rx), y + (sin(angle) * ry));
	}
	glEnd();
}


static float orange1 = 0;
static float orange2 = 1;
static float orange3 = 1;
static float orange4 = 0;
static float orange5 = 1;
static float orange6 = 1;

void Orange()
{
	orange1 -= .5;
	if (orange1 < -29) {
		orange1 = 0;
	}
	orange2 -= .5;
	if (orange2 < -30) {
		orange2 = 0;
	}
	orange3 -= .5;
	if (orange3 < -30) {
		orange3 = 0;
	}orange4 -= .5;
	if (orange4 < -30) {
		orange4 = 0;
	}
	orange5 -= .3;
	if (orange5 < -30) {
		orange5 = 0;
	}
	orange6 -= .3;
	if (orange6 < -30) {
		orange6 = 0;
	}
	glutPostRedisplay();
}
void orangeFall()
{
	glColor3ub(255, 165, 0);
	Orange();
	glPushMatrix();
	glTranslated(0, orange1, 0);
	circle(.5, 1, 33, 37);
	glPopMatrix();
	glPushMatrix();
	glTranslated(0, orange2, 0);
	circle(.5, 1, 35, 39);
	glPopMatrix();

	glPushMatrix();
	glTranslated(0, orange3, 0);
	circle(.5, 1, 38, 41);
	//circle(.5,1,41,39);
	glPopMatrix();
	glPushMatrix();
	glTranslated(0, orange4, 0);
	circle(.5, 1, 41, 39);
	glPopMatrix();
```

```
    glPushMatrix();
    glTranslated(0, orange5, 0);
    circle(.5, 1, 43, 41);
    glPopMatrix();

    glPushMatrix();
    glTranslated(0, orange6, 0);
    circle(.5, 1, 45, 39);
    glPopMatrix();
}

RainDrop rainDrops[MAX_RAIN_DROPS];

void initRain() {
    srand(time(NULL));
    for (int i = 0; i < MAX_RAIN_DROPS; ++i) {
        rainDrops[i].x = rand() % 2000;
        rainDrops[i].y = rand() % 970;
        rainDrops[i].active = true;
    }
}

void drawRainNight() {
    if (rainStatus && (n == 1 || n == 0)) { // Check if it's night time
        glLineWidth(1.0);
        glColor3f(0.0, 0.0, 1.0); // Blue color for rain
        glBegin(GL_LINES);
        for (int i = 0; i < MAX_RAIN_DROPS; ++i) {
            if (rainDrops[i].active) {
                glVertex2f(rainDrops[i].x, rainDrops[i].y);
                glVertex2f(rainDrops[i].x, rainDrops[i].y - 10);
                rainDrops[i].y -= RAIN_SPEED;
                if (rainDrops[i].y < 0) {
                    rainDrops[i].y = 970;
                }
            }
        }
        glEnd();
    }
}

void DrawCircle(float cx, float cy, float r, int num_segments)
{
    glBegin(GL_TRIANGLE_FAN);
    for (int i = 0; i < num_segments; i++)
    {
        float theta = 2.0f * 3.1415926f * float(i) / float(num_segments);//get the current angle

        float x = r * cosf(theta);//calculate x
        float y = r * sinf(theta);//calculate y

        glVertex2f(x + cx, y + cy);//output vertex
    }
    glEnd();
}

void DrawLine(float x1, float y1, float x2, float y2) {
    glBegin(GL_LINES);
    glVertex2f(x1, y1); // Starting point of the line
    glVertex2f(x2, y2); // Ending point of the line
    glEnd();
```

```
}


void DrawPetal(int x, int y, int radius, int numSegments) {
	glColor3f(1.0, 0.5, 0.5); // Pink color for petals
	DrawCircle(x, y, radius, numSegments);
}

void petalFall() {
	petalY -= petalFallSpeed; // Move petal upward
	if (petalY < 0) { // Reset petal position if it reaches top
		petalX = minX + rand() % (maxX - minX + 1); // Random X position
		petalY = 650; // Reset Y position
	}
}


void tree()
{
	if (n == 0)
	{
		for (int xOffset = -700; xOffset < 900; xOffset += 200)
		{
			// Tree body dark
			glBegin(GL_POLYGON);
			glColor3f(.616, .333, .208);
			glVertex2i(1490 - 22 + xOffset, 505 - 50);
			glVertex2i(1500 - 22 + xOffset, 500 - 50);
			glVertex2i(1500 - 22 + xOffset, 400 - 50);
			glVertex2i(1495 - 22 + xOffset, 350 - 50);
			glVertex2i(1490 - 22 + xOffset, 200 - 50);
			glVertex2i(1480 - 22 + xOffset, 50);

			glVertex2i(1530 + 2 + xOffset, 50);
			glVertex2i(1520 + 2 + xOffset, 200 - 50);
			glVertex2i(1515 + 2 + xOffset, 350 - 50);
			glVertex2i(1510 + 2 + xOffset, 400 - 50);
			glVertex2i(1510 + 2 + xOffset, 500 - 50);
			glVertex2i(1520 + 2 + xOffset, 505 - 50);
			glEnd();

			// Tree body
			glBegin(GL_POLYGON);
			glColor3f(.8, .537, .365);
			glVertex2i(1490 - 20 + xOffset, 505 - 50);
			glVertex2i(1500 - 20 + xOffset, 500 - 50);
			glVertex2i(1500 - 20 + xOffset, 400 - 50);
			glVertex2i(1495 - 20 + xOffset, 350 - 50);
			glVertex2i(1490 - 20 + xOffset, 200 - 50);
			glVertex2i(1480 - 20 + xOffset, 50);

			glColor3f(.616, .333, .208);
			glVertex2i(1530 + xOffset, 50);
			glVertex2i(1520 + xOffset, 200 - 50);
			glVertex2i(1515 + xOffset, 350 - 50);
			glVertex2i(1510 + xOffset, 400 - 50);
			glVertex2i(1510 + xOffset, 500 - 50);
			glVertex2i(1520 + xOffset, 505 - 50);
			glEnd();

			// Tree body left side dark
```

```
glBegin(GL_POLYGON);
glColor3f(.616, .333, .208);
glVertex2i(1410 - 2 + xOffset, 490 - 50);
glVertex2i(1420 - 2 + xOffset, 450 - 50);
glVertex2i(1440 - 2 + xOffset, 410 - 50);
glVertex2i(1470 - 2 + xOffset, 380 - 50);
glVertex2i(1470 - 2 + xOffset, 350 - 50);
glVertex2i(1480 - 2 + xOffset, 340 - 50);

glVertex2i(1480 + xOffset, 360 - 50);
glVertex2i(1470 + xOffset, 372 - 50);
glVertex2i(1470 + xOffset, 412 - 50);
glVertex2i(1442 + xOffset, 432 - 50);
glVertex2i(1422 + xOffset, 490 - 50);
glVertex2i(1422 + xOffset, 510 - 50);
glEnd();

// Tree body left side
glBegin(GL_POLYGON);
glColor3f(.8, .537, .365);
glVertex2i(1410 + xOffset, 490 - 50);
glVertex2i(1420 + xOffset, 450 - 50);
glVertex2i(1440 + xOffset, 410 - 50);
glVertex2i(1470 + xOffset, 380 - 50);
glVertex2i(1470 + xOffset, 350 - 50);
glVertex2i(1480 + xOffset, 340 - 50);

glVertex2i(1480 + xOffset, 360 - 50);
glVertex2i(1470 + xOffset, 370 - 50);
glVertex2i(1470 + xOffset, 410 - 50);
glVertex2i(1440 + xOffset, 430 - 50);
glVertex2i(1420 + xOffset, 490 - 50);
glVertex2i(1420 + xOffset, 510 - 50);
glEnd();

// Tree body Right side dark
glBegin(GL_POLYGON);
glColor3f(.616, .333, .208);
glVertex2i(1511 + xOffset, 230 - 3);
glVertex2i(1531 + xOffset, 270 - 3);
glVertex2i(1541 + xOffset, 290 - 3);
glVertex2i(1551 + xOffset, 320 - 3);
glVertex2i(1566 + xOffset, 380 - 3);
glVertex2i(1581 + xOffset, 420 - 3);
glVertex2i(1591 + xOffset, 440 - 3);
glVertex2i(1591 + xOffset, 450 - 3);

glVertex2i(1585 + xOffset, 470 + 3);
glVertex2i(1590 + xOffset, 440 + 23);
glVertex2i(1580 + xOffset, 420 + 23);
glVertex2i(1560 + xOffset, 380 + 23);
glVertex2i(1552 + xOffset, 330 + 23);
glVertex2i(1542 + xOffset, 290 + 45);
glVertex2i(1530 + xOffset, 270 + 43);
glVertex2i(1510 + xOffset, 250 + 40);
glEnd();

// Tree body Right side
glBegin(GL_POLYGON);
glColor3f(.616, .333, .208);
glVertex2i(1510 + xOffset, 230);
```

```
        glVertex2i(1530 + xOffset, 270);
        glVertex2i(1540 + xOffset, 290);
        glVertex2i(1550 + xOffset, 320);
        glVertex2i(1565 + xOffset, 380);
        glVertex2i(1580 + xOffset, 420);
        glVertex2i(1590 + xOffset, 440);
        glVertex2i(1590 + xOffset, 450);

        glColor3f(.8, .537, .365);
        glVertex2i(1585 + xOffset, 470);
        glVertex2i(1590 + xOffset, 440 + 20);
        glVertex2i(1580 + xOffset, 420 + 20);
        glVertex2i(1560 + xOffset, 380 + 20);
        glVertex2i(1550 + xOffset, 330 + 20);
        glVertex2i(1540 + xOffset, 290 + 40);
        glVertex2i(1530 + xOffset, 270 + 40);
        glVertex2i(1510 + xOffset, 250 + 40);
        glEnd();

        // dark circle
        glColor3f(0.3137, 0.5137, 0.1412);
        DrawCircle(1590 + xOffset, 500, 83, 2000);
        glColor3f(0.3137, 0.5137, 0.1412);
        DrawCircle(1590 + xOffset, 600, 73, 2000);
        glColor3f(0.3137, 0.5137, 0.1412);
        DrawCircle(1410 + xOffset, 510, 83, 2000);
        glColor3f(0.3137, 0.5137, 0.1412);
        DrawCircle(1410 + xOffset, 600, 73, 2000);
        glColor3f(0.3137, 0.5137, 0.1412);
        DrawCircle(1495 + xOffset, 530, 83, 2000);
        glColor3f(0.3137, 0.5137, 0.1412);
        DrawCircle(1495 + xOffset, 630, 103, 2000);

        // down right
        glColor3f(1.0, 0.753, 0.796);
        DrawCircle(1590 + xOffset, 500, 80, 2000);

        // top right
        glColor3f(1.0, 0.753, 0.796);
        DrawCircle(1590 + xOffset, 600, 70, 2000);

        // down left
        glColor3f(1.0, 0.753, 0.796);
        DrawCircle(1410 + xOffset, 510, 80, 2000);

        // top left
        glColor3f(1.0, 0.753, 0.796);
        DrawCircle(1410 + xOffset, 600, 70, 2000);

        // middle top
        glColor3f(1.0, 0.753, 0.796);
        DrawCircle(1495 + xOffset, 530, 80, 2000);

        // middle
        glColor3f(1.0, 0.753, 0.796);
        DrawCircle(1495 + xOffset, 630, 100, 2000);

        DrawPetal(petalX + xOffset, petalY, 10, 20); // Adjust size and segments as needed

}
petalFall();
```

```
                petalFall();
                petalFall();
                petalFall();


        }
}

void treenight()
{
        if (n == 1)
        {
                for (int xOffset = -700; xOffset < 900; xOffset += 200)
                {
                        // Tree body dark
                        glBegin(GL_POLYGON);
                        glColor3f(.616, .333, .208);
                        glVertex2i(1490 - 22 + xOffset, 505 - 50);
                        glVertex2i(1500 - 22 + xOffset, 500 - 50);
                        glVertex2i(1500 - 22 + xOffset, 400 - 50);
                        glVertex2i(1495 - 22 + xOffset, 350 - 50);
                        glVertex2i(1490 - 22 + xOffset, 200 - 50);
                        glVertex2i(1480 - 22 + xOffset, 50);

                        glVertex2i(1530 + 2 + xOffset, 50);
                        glVertex2i(1520 + 2 + xOffset, 200 - 50);
                        glVertex2i(1515 + 2 + xOffset, 350 - 50);
                        glVertex2i(1510 + 2 + xOffset, 400 - 50);
                        glVertex2i(1510 + 2 + xOffset, 500 - 50);
                        glVertex2i(1520 + 2 + xOffset, 505 - 50);
                        glEnd();

                        // Tree body
                        glBegin(GL_POLYGON);
                        glColor3f(.8, .537, .365);
                        glVertex2i(1490 - 20 + xOffset, 505 - 50);
                        glVertex2i(1500 - 20 + xOffset, 500 - 50);
                        glVertex2i(1500 - 20 + xOffset, 400 - 50);
                        glVertex2i(1495 - 20 + xOffset, 350 - 50);
                        glVertex2i(1490 - 20 + xOffset, 200 - 50);
                        glVertex2i(1480 - 20 + xOffset, 50);

                        glColor3f(.616, .333, .208);
                        glVertex2i(1530 + xOffset, 50);
                        glVertex2i(1520 + xOffset, 200 - 50);
                        glVertex2i(1515 + xOffset, 350 - 50);
                        glVertex2i(1510 + xOffset, 400 - 50);
                        glVertex2i(1510 + xOffset, 500 - 50);
                        glVertex2i(1520 + xOffset, 505 - 50);
                        glEnd();

                        // Tree body left side dark
                        glBegin(GL_POLYGON);
                        glColor3f(.616, .333, .208);
                        glVertex2i(1410 - 2 + xOffset, 490 - 50);
                        glVertex2i(1420 - 2 + xOffset, 450 - 50);
                        glVertex2i(1440 - 2 + xOffset, 410 - 50);
                        glVertex2i(1470 - 2 + xOffset, 380 - 50);
                        glVertex2i(1470 - 2 + xOffset, 350 - 50);
                        glVertex2i(1480 - 2 + xOffset, 340 - 50);
```

```
glVertex2i(1480 + xOffset, 360 - 50);
glVertex2i(1470 + xOffset, 372 - 50);
glVertex2i(1470 + xOffset, 412 - 50);
glVertex2i(1442 + xOffset, 432 - 50);
glVertex2i(1422 + xOffset, 490 - 50);
glVertex2i(1422 + xOffset, 510 - 50);
glEnd();

// Tree body left side
glBegin(GL_POLYGON);
glColor3f(.8, .537, .365);
glVertex2i(1410 + xOffset, 490 - 50);
glVertex2i(1420 + xOffset, 450 - 50);
glVertex2i(1440 + xOffset, 410 - 50);
glVertex2i(1470 + xOffset, 380 - 50);
glVertex2i(1470 + xOffset, 350 - 50);
glVertex2i(1480 + xOffset, 340 - 50);

glVertex2i(1480 + xOffset, 360 - 50);
glVertex2i(1470 + xOffset, 370 - 50);
glVertex2i(1470 + xOffset, 410 - 50);
glVertex2i(1440 + xOffset, 430 - 50);
glVertex2i(1420 + xOffset, 490 - 50);
glVertex2i(1420 + xOffset, 510 - 50);
glEnd();

// Tree body Right side dark
glBegin(GL_POLYGON);
glColor3f(.616, .333, .208);
glVertex2i(1511 + xOffset, 230 - 3);
glVertex2i(1531 + xOffset, 270 - 3);
glVertex2i(1541 + xOffset, 290 - 3);
glVertex2i(1551 + xOffset, 320 - 3);
glVertex2i(1566 + xOffset, 380 - 3);
glVertex2i(1581 + xOffset, 420 - 3);
glVertex2i(1591 + xOffset, 440 - 3);
glVertex2i(1591 + xOffset, 450 - 3);

glVertex2i(1585 + xOffset, 470 + 3);
glVertex2i(1590 + xOffset, 440 + 23);
glVertex2i(1580 + xOffset, 420 + 23);
glVertex2i(1560 + xOffset, 380 + 23);
glVertex2i(1552 + xOffset, 330 + 23);
glVertex2i(1542 + xOffset, 290 + 45);
glVertex2i(1530 + xOffset, 270 + 43);
glVertex2i(1510 + xOffset, 250 + 40);
glEnd();

// Tree body Right side
glBegin(GL_POLYGON);
glColor3f(.616, .333, .208);
glVertex2i(1510 + xOffset, 230);
glVertex2i(1530 + xOffset, 270);
glVertex2i(1540 + xOffset, 290);
glVertex2i(1550 + xOffset, 320);
glVertex2i(1565 + xOffset, 380);
glVertex2i(1580 + xOffset, 420);
glVertex2i(1590 + xOffset, 440);
glVertex2i(1590 + xOffset, 450);

glColor3f(.8, .537, .365);
```

```
                        glVertex2i(1585 + xOffset, 470);
                        glVertex2i(1590 + xOffset, 440 + 20);
                        glVertex2i(1580 + xOffset, 420 + 20);
                        glVertex2i(1560 + xOffset, 380 + 20);
                        glVertex2i(1550 + xOffset, 330 + 20);
                        glVertex2i(1540 + xOffset, 290 + 40);
                        glVertex2i(1530 + xOffset, 270 + 40);
                        glVertex2i(1510 + xOffset, 250 + 40);
                        glEnd();

                        // dark circle
                        glColor3f(0.3137, 0.5137, 0.1412);
                        DrawCircle(1590 + xOffset, 500, 83, 2000);
                        glColor3f(0.3137, 0.5137, 0.1412);
                        DrawCircle(1590 + xOffset, 600, 73, 2000);
                        glColor3f(0.3137, 0.5137, 0.1412);
                        DrawCircle(1410 + xOffset, 510, 83, 2000);
                        glColor3f(0.3137, 0.5137, 0.1412);
                        DrawCircle(1410 + xOffset, 600, 73, 2000);
                        glColor3f(0.3137, 0.5137, 0.1412);
                        DrawCircle(1495 + xOffset, 530, 83, 2000);
                        glColor3f(0.3137, 0.5137, 0.1412);
                        DrawCircle(1495 + xOffset, 630, 103, 2000);

                        // down right
                        glColor3f(1.0, 0.753, 0.796);
                        DrawCircle(1590 + xOffset, 500, 80, 2000);

                        // top right
                        glColor3f(1.0, 0.753, 0.796);
                        DrawCircle(1590 + xOffset, 600, 70, 2000);

                        // down left
                        glColor3f(1.0, 0.753, 0.796);
                        DrawCircle(1410 + xOffset, 510, 80, 2000);

                        // top left
                        glColor3f(1.0, 0.753, 0.796);
                        DrawCircle(1410 + xOffset, 600, 70, 2000);

                        // middle top
                        glColor3f(1.0, 0.753, 0.796);
                        DrawCircle(1495 + xOffset, 530, 80, 2000);

                        // middle
                        glColor3f(1.0, 0.753, 0.796);
                        DrawCircle(1495 + xOffset, 630, 100, 2000);

                        DrawPetal(petalX + xOffset, petalY, 10, 20); // Adjust size and segments as needed

                }
                petalFall();
                petalFall();
                petalFall();
                petalFall();


        }
}

void city()
```

```
{
        glBegin(GL_POLYGON);
        glColor3f(0.298, .561, .619);
        glVertex2i(2000, 950);
        glVertex2i(2000, 1050);
        glVertex2i(1950, 1050);
        glVertex2i(1950, 1060);
        glVertex2i(1940, 1060);
        glVertex2i(1940, 1000);
        glVertex2i(1900, 1000);
        glVertex2i(1900, 1050);
        glVertex2i(1875, 1075);
        glVertex2i(1850, 1050);
        glVertex2i(1850, 950);
        glEnd();

        glBegin(GL_POLYGON);
        glColor3f(0.298, .561, .619);
        glVertex2i(1920, 950);
        glVertex2i(1920, 1140);
        glVertex2i(1890, 1140);
        glVertex2i(1890, 950);
        glEnd();

        glBegin(GL_POLYGON);
        glColor3f(0.298, .561, .619);
        glVertex2i(1850, 950);
        glVertex2i(1850, 1050);
        glVertex2i(1840, 1050);
        glVertex2i(1840, 1030);
        glVertex2i(1800, 1030);
        glVertex2i(1800, 950);
        glEnd();

        glBegin(GL_POLYGON);
        glColor3f(0.298, .561, .619);
        glVertex2i(1800, 950);
        glVertex2i(1800, 970);
        glVertex2i(1700, 970);
        glVertex2i(1700, 950);
        glEnd();

        glBegin(GL_POLYGON);
        glColor3f(0.298, .561, .619);
        glVertex2i(1700, 950);
        glVertex2i(1700, 1010);
        glVertex2i(1650, 1010);
        glVertex2i(1650, 950);
        glEnd();

        glBegin(GL_TRIANGLES);
        glColor3f(0.298, .561, .619);
        glVertex2i(1710, 1010);
        glVertex2i(1675, 1030);
        glVertex2i(1640, 1010);
        glEnd();

        glBegin(GL_POLYGON);
        glColor3f(0.298, .561, .619);
        glVertex2i(1650, 950);
        glVertex2i(1650, 1060);
```

```
glVertex2i(1655, 1065);
glVertex2i(1605, 1065);
glVertex2i(1610, 1060);
glVertex2i(1610, 950);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.298, .561, .619);
glVertex2i(1610, 950);
glVertex2i(1610, 980);
glVertex2i(1580, 980);
glVertex2i(1580, 950);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.298, .561, .619);
glVertex2i(0, 950);
glVertex2i(80, 950);
glVertex2i(80, 1050);
glVertex2i(0, 1090);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.298, .561, .619);
glVertex2i(80, 950);
glVertex2i(110, 950);
glVertex2i(110, 970);
glVertex2i(80, 970);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.298, .561, .619);
glVertex2i(110, 1100);
glVertex2i(110, 950);
glVertex2i(180, 950);
glVertex2i(180, 1100);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.298, .561, .619);
glVertex2i(180, 1000);
glVertex2i(180, 950);
glVertex2i(230, 950);
glVertex2i(230, 1000);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.298, .561, .619);
glVertex2i(230, 1100);
glVertex2i(220, 950);
glVertex2i(290, 950);
glVertex2i(280, 1100);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.298, .561, .619);
glVertex2i(300, 950);
glVertex2i(400, 950);
glVertex2i(400, 1180);
glVertex2i(300, 1180);
glEnd();
```

```
glBegin(GL_POLYGON);
glColor3f(0.298, .561, .619);
glVertex2i(400, 1050);
glVertex2i(400, 950);
glVertex2i(480, 950);
glVertex2i(480, 1050);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.298, .561, .619);
glVertex2i(480, 950);
glVertex2i(530, 950);
glVertex2i(530, 1080);
glVertex2i(535, 1090);
glVertex2i(475, 1090);
glVertex2i(480, 1080);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.298, .561, .619);
glVertex2i(530, 950);
glVertex2i(630, 950);
glVertex2i(630, 1000);
glVertex2i(530, 1000);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.298, .561, .619);
glVertex2i(630, 950);
glVertex2i(1000, 950);
glVertex2i(1000, 1010);
glVertex2i(980, 1010);
glVertex2i(980, 1110);
glVertex2i(900, 1110);
glVertex2i(900, 1050);
glVertex2i(870, 1070);
glVertex2i(840, 1050);
glVertex2i(840, 990);
glVertex2i(810, 990);
glVertex2i(810, 1090);
glVertex2i(770, 1090);
glVertex2i(770, 1040);
glVertex2i(730, 1040);
glVertex2i(630, 1020);
glEnd();

glBegin(GL_TRIANGLES);
glColor3f(0.298, .561, .619);
glVertex2i(730, 1040);
glVertex2i(760, 1040);
glVertex2i(745, 1100);
glEnd();

glBegin(GL_TRIANGLES);
glColor3f(0.298, .561, .619);
glVertex2i(650, 1000);
glVertex2i(680, 1000);
glVertex2i(665, 1100);
glEnd();
```

```
glBegin(GL_POLYGON);
glColor3f(0.298, .561, .619);
glVertex2i(1650 - 550, 950);
glVertex2i(1650 - 550, 1060);
glVertex2i(1655 - 550, 1065);
glVertex2i(1605 - 550, 1065);
glVertex2i(1610 - 550, 1060);
glVertex2i(1610 - 550, 950);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.298, .561, .619);
glVertex2i(1650 - 700, 950 + 60);
glVertex2i(1650 - 700, 1060 + 60);
glVertex2i(1655 - 700, 1065 + 60);
glVertex2i(1605 - 700, 1065 + 60);
glVertex2i(1610 - 700, 1060 + 60);
glVertex2i(1610 - 700, 950 + 60);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.298, .561, .619);
glVertex2i(1700 - 550, 950);
glVertex2i(1700 - 550, 1010);
glVertex2i(1650 - 550, 1010);
glVertex2i(1650 - 550, 950);
glEnd();

glBegin(GL_TRIANGLES);
glColor3f(0.298, .561, .619);
glVertex2i(1710 - 550, 1010);
glVertex2i(1675 - 550, 1030);
glVertex2i(1640 - 550, 1010);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.298, .561, .619);
glVertex2i(1200, 1095);
glVertex2i(1200, 950);
glVertex2i(1320, 950);
glVertex2i(1320, 1095);
glVertex2i(1330, 1105);
glVertex2i(1190, 1105);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.298, .561, .619);
glVertex2i(1430, 1020);
glVertex2i(1380, 950);
glVertex2i(1520, 950);
glVertex2i(1470, 1020);
glVertex2i(1455, 1100);
glVertex2i(1450, 1120);
glVertex2i(1445, 1100);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.298, .561, .619);
glVertex2i(280, 950);
glVertex2i(1600, 950);
glVertex2i(1600, 970);
```

```
    glVertex2i(280, 970);
    glEnd();
}

void sky()
{
    if (n == 0)
    {
        glBegin(GL_POLYGON);
        glColor3f(0.34, 0.808, 1.0);
        glVertex2i(2000, 1500);
        glVertex2i(0, 1500);
        glColor3f(2.49, 1.87, 1.0);
        glVertex2i(0, 970);
        glVertex2i(2000, 970);
        glEnd();
    }
}

void skynight()
{
    if (n == 1)
    {
        glBegin(GL_POLYGON);
        glColor3f(0.04705, 0.078431, 0.50588);
        glVertex2i(2000, 1500);
        glVertex2i(0, 1500);
        glColor3f(0.403921, 0.44705, 0.94509);
        glVertex2i(0, 970);
        glVertex2i(2000, 970);
        glEnd();
    }
}

void sun()
{
    if (n == 0)
    {
        glColor3f(1.0, 1.0, 0.792156);
        DrawCircle(1400, 1300, 105, 2000);

        glColor3f(1.0, 0.9176470, 0.63529411);
        DrawCircle(1400, 1300, 100, 2000);

        glColor3f(0.99647843, 0.87058823, 0.43921568);
        DrawCircle(1400, 1300, 90, 2000);

        glColor3f(0.99647843, 0.84313725, 0.30588235);
        DrawCircle(1400, 1300, 85, 2000);

        glColor3f(0.9490196, 0.6745098, 0.1098039);
        DrawCircle(1400, 1300, 75, 2000);
    }
}

void moon()
{
    if (n == 1)
    {
        glColor3f(1.0, 1.0, 0.843137);
        DrawCircle(1450, 1300, 75, 2000);
```

```
                glColor3f(0.95686, 0.90980, 0.701960);
                DrawCircle(1450, 1300, 65, 2000);
        }
}
void river()
{
        if (n == 0)
        {
                glBegin(GL_POLYGON);
                glColor3f(.106, .69, .918);
                glVertex2i(0, 200);
                glColor3f(1.4, .8, .949);
                glVertex2i(1000, 200);
                glColor3f(0.106, 1.69, .918);
                glVertex2i(2000, 200);
                glColor3f(0.733, 0.886, .953);
                glVertex2i(2000, 970);
                glColor3f(0.733, 0.886, .953);
                glVertex2i(0, 970);
                glEnd();
        }
}


void rivernight()
{
        if (n == 1)
        {
                glBegin(GL_POLYGON);
                glColor3f(0.0, 0.58431, 0.866666667);
                glVertex2i(0, 200);
                glColor3f(0.0666667, 0.670588, 0.8);
                glVertex2i(2000, 200);
                glColor3f(0.0, 0.486274, 0.72549);
                glVertex2i(2000, 970);
                glColor3f(0.0, 0.486274, 0.72549);
                glVertex2i(0, 970);
                glEnd();
        }
}

void field()
{
        if (n == 0)
        {
                //Middle ground
                glBegin(GL_POLYGON);
                glColor3f(0.545, .671, .0313);
                glVertex2i(0, 0);
                glVertex2i(2000, 0);
                glVertex2i(2000, 350);
                glVertex2i(1900, 350);
                glVertex2i(1800, 320);
                glVertex2i(1700, 350);
                glVertex2i(1600, 370);
                glVertex2i(1500, 375);
                glVertex2i(1350, 365);
                glVertex2i(1200, 390);
                glVertex2i(1000, 410);
                glVertex2i(1700 - 700, 350 + 60);
```

```
			glVertex2i(1600 - 700, 370 + 50);
			glVertex2i(1500 - 700, 375 + 50);
			glVertex2i(1350 - 700, 365 + 50);
			glVertex2i(1200 - 700, 390 + 60);
			glVertex2i(1000 - 700, 410 + 50);
			glVertex2i(200, 455);
			glVertex2i(100, 465);
			glVertex2i(0, 455);
			glEnd();

			//Middle ground
			glBegin(GL_POLYGON);
			glColor3f(.537, 1.776, .239);
			glVertex2i(0, 0);
			glVertex2i(2000, 0);

			glColor3f(0.6549, .780, .1098);
			glVertex2i(2000, 350 - 5);
			glVertex2i(1900, 350 - 5);
			glVertex2i(1800, 320 - 5);
			glVertex2i(1700, 350 - 5);
			glVertex2i(1600, 370 - 5);
			glVertex2i(1500, 375 - 5);
			glVertex2i(1350, 365 - 5);
			glVertex2i(1200, 390 - 5);
			glVertex2i(1000, 410 - 5);
			glVertex2i(1700 - 700, 350 + 55);
			glVertex2i(1600 - 700, 370 + 45);
			glVertex2i(1500 - 700, 375 + 45);
			glVertex2i(1350 - 700, 365 + 45);
			glVertex2i(1200 - 700, 390 + 55);
			glVertex2i(1000 - 700, 410 + 45);
			glVertex2i(200, 455 - 5);
			glVertex2i(100, 465 - 5);
			glVertex2i(0, 455 - 5);
			glEnd();
	}
}

void fieldnight()
{
	if (n == 1)
	{
			//Middle ground
			glBegin(GL_POLYGON);
			glColor3f(0.545, 0.671, 0.0313);
			glVertex2i(0, 0);
			glVertex2i(2000, 0);
			glVertex2i(2000, 350);
			glVertex2i(1900, 350);
			glVertex2i(1800, 320);
			glVertex2i(1700, 350);
			glVertex2i(1600, 370);
			glVertex2i(1500, 375);
			glVertex2i(1350, 365);
			glVertex2i(1200, 390);
			glVertex2i(1000, 410);
			glVertex2i(1700 - 700, 350 + 60);
			glVertex2i(1600 - 700, 370 + 50);
			glVertex2i(1500 - 700, 375 + 50);
			glVertex2i(1350 - 700, 365 + 50);
```

```
                glVertex2i(1200 - 700, 390 + 60);
                glVertex2i(1000 - 700, 410 + 50);
                glVertex2i(200, 455);
                glVertex2i(100, 465);
                glVertex2i(0, 455);
                glEnd();

                //Middle ground
                glBegin(GL_POLYGON);
                glColor3f(0.403921, 0.807843, 0.0);
                glVertex2i(0, 0);
                glVertex2i(2000, 0);

                glColor3f(0.6549, .780, .1098);
                glVertex2i(2000, 350 - 5);
                glVertex2i(1900, 350 - 5);
                glVertex2i(1800, 320 - 5);
                glVertex2i(1700, 350 - 5);
                glVertex2i(1600, 370 - 5);
                glVertex2i(1500, 375 - 5);
                glVertex2i(1350, 365 - 5);
                glVertex2i(1200, 390 - 5);
                glVertex2i(1000, 410 - 5);
                glVertex2i(1700 - 700, 350 + 55);
                glVertex2i(1600 - 700, 370 + 45);
                glVertex2i(1500 - 700, 375 + 45);
                glVertex2i(1350 - 700, 365 + 45);
                glVertex2i(1200 - 700, 390 + 55);
                glVertex2i(1000 - 700, 410 + 45);
                glVertex2i(200, 455 - 5);
                glVertex2i(100, 465 - 5);
                glVertex2i(0, 455 - 5);
                glEnd();
        }
}


float trainPosition = -2000.0f; // Initial position of the train

void drawTrainBody() {
        // Train body
        glBegin(GL_POLYGON);
        glColor3f(0.8, 0.2, 0.2); // Reddish color for the train body
        glVertex2i(10, 940);
        glVertex2i(110, 940);
        glVertex2i(110, 1010);
        glVertex2i(10, 1010);
        glEnd();
}

void drawTrainWindows() {
        // Windows
        glColor3f(0.6, 0.8, 1.0); // Light blue color for windows
        glBegin(GL_QUADS);
        glVertex2i(20, 960);
        glVertex2i(35, 960);
        glVertex2i(35, 990);
        glVertex2i(20, 990);
        glEnd();

        glBegin(GL_QUADS);
```

```
	glVertex2i(50, 960);
	glVertex2i(65, 960);
	glVertex2i(65, 990);
	glVertex2i(50, 990);
	glEnd();

	glBegin(GL_QUADS);
	glVertex2i(80, 960);
	glVertex2i(95, 960);
	glVertex2i(95, 990);
	glVertex2i(80, 990);
	glEnd();
}

void drawTrainWheels() {
	// Wheels
	glColor3f(0.1, 0.1, 0.1); // Dark gray color for wheels
	glBegin(GL_POLYGON);
	glVertex2i(20, 920);
	glVertex2i(30, 920);
	glVertex2i(30, 940);
	glVertex2i(20, 940);
	glEnd();

	glBegin(GL_POLYGON);
	glVertex2i(90, 920);
	glVertex2i(100, 920);
	glVertex2i(100, 940);
	glVertex2i(90, 940);
	glEnd();
}

void bridge()
{
	//pillars dark
	glBegin(GL_POLYGON);
	glColor3f(0, 0, 0);
	glVertex2i(440 - 2, 615 - 2);
	glVertex2i(520 + 2, 615 - 2);
	glVertex2i(520 + 2, 830 + 2);
	glVertex2i(440 - 2, 830 + 2);
	glEnd();

	glBegin(GL_POLYGON);
	glColor3f(0, 0, 0);
	glVertex2i(1040 - 2, 615 - 2);
	glVertex2i(1120 + 2, 615 - 2);
	glVertex2i(1120 + 2, 830 + 2);
	glVertex2i(1040 - 2, 830 + 2);
	glEnd();

	glBegin(GL_POLYGON);
	glColor3f(0, 0, 0);
	glVertex2i(1640 - 2, 615 - 2);
	glVertex2i(1720 + 2, 615 - 2);
	glVertex2i(1720 + 2, 830 + 2);
	glVertex2i(1640 - 2, 830 + 2);
	glEnd();

	//pillars
	glBegin(GL_POLYGON);
```

```
glColor3f(0.35294, 0.35294, 0.35294);
glVertex2i(440, 615);
glVertex2i(520, 615);
glColor3f(0.5215686, 0.5215686, 0.5215686);
glVertex2i(520, 830);
glVertex2i(440, 830);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.35294, 0.35294, 0.35294);
glVertex2i(1040, 615);
glVertex2i(1120, 615);
glColor3f(0.5215686, 0.5215686, 0.5215686);
glVertex2i(1120, 830);
glVertex2i(1040, 830);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.35294, 0.35294, 0.35294);
glVertex2i(1640, 615);
glVertex2i(1720, 615);
glColor3f(0.5215686, 0.5215686, 0.5215686);
glVertex2i(1720, 830);
glVertex2i(1640, 830);
glEnd();

//body
glBegin(GL_POLYGON);
glColor3f(0, 0, 0);
glVertex2i(0, 830 + 2);
glVertex2i(2000, 830 + 2);
glVertex2i(2000, 900 + 2);
glVertex2i(0, 900 + 2);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.5215686, 0.5215686, 0.5215686);
glVertex2i(0, 830 + 4);
glVertex2i(2000, 830 + 4);
glColor3f(0.643137, 0.643137, 0.643137);
glVertex2i(2000, 900);
glVertex2i(0, 900);
glEnd();

//towers dark
glBegin(GL_POLYGON);
glColor3f(0, 0, 0);
glVertex2i(470, 900);
glVertex2i(490, 900);
glVertex2i(490, 1100);
glVertex2i(470, 1100);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0, 0, 0);
glVertex2i(1070, 900);
glVertex2i(1090, 900);
glVertex2i(1090, 1100);
glVertex2i(1070, 1100);
glEnd();
```

```
        glBegin(GL_POLYGON);
        glColor3f(0, 0, 0);
        glVertex2i(1670, 900);
        glVertex2i(1690, 900);
        glVertex2i(1690, 1100);
        glVertex2i(1670, 1100);
        glEnd();

        //towers
        glBegin(GL_POLYGON);
        glColor3f(0.643137, 0.643137, 0.643137);
        glVertex2i(470 + 2, 900 + 2);
        glVertex2i(490 - 2, 900 + 2);
        glVertex2i(490 - 2, 1100 - 2);
        glVertex2i(470 + 2, 1100 - 2);
        glEnd();

        glBegin(GL_POLYGON);
        glColor3f(0.643137, 0.643137, 0.643137);
        glVertex2i(1070 + 2, 900 + 2);
        glVertex2i(1090 - 2, 900 + 2);
        glVertex2i(1090 - 2, 1100 - 2);
        glVertex2i(1070 + 2, 1100 - 2);
        glEnd();

        glBegin(GL_POLYGON);
        glColor3f(0.643137, 0.643137, 0.643137);
        glVertex2i(1670 + 2, 900 + 2);
        glVertex2i(1690 - 2, 900 + 2);
        glVertex2i(1690 - 2, 1100 - 2);
        glVertex2i(1670 + 2, 1100 - 2);
        glEnd();


}

void train(int x)
{
        // Train body
        glBegin(GL_POLYGON);
        glColor3f(0.5, 0.2, 0.1); // Brown color for the body
        glVertex2i(50, 925);      // Half of 100, 950 + 450
        glVertex2i(50, 1025);     // Half of 100, 1150 + 450
        glVertex2i(300, 1025);    // Half of 600, 1150 + 450
        glVertex2i(300, 925);     // Half of 600, 950 + 450
        glEnd();

        // Train roof
        glBegin(GL_POLYGON);
        glColor3f(0.8, 0.4, 0.2); // Lighter brown color for the roof
        glVertex2i(25, 1025);     // Half of 50, 1150 + 450
        glVertex2i(50, 1075);     // Half of 100, 1250 + 450
        glVertex2i(300, 1075);    // Half of 600, 1250 + 450
        glVertex2i(325, 1025);    // Half of 650, 1150 + 450
        glEnd();

        // Train windows
        glColor3f(0.1, 0.5, 0.7); // Blue color for windows
        DrawCircle(75, 975, 15, 150);    // Half of 150, 1050 + 450, and 300
        DrawCircle(150, 975, 15, 150);   // Half of 300, 1050 + 450, and 300
        DrawCircle(225, 975, 15, 150);   // Half of 450, 1050 + 450, and 300
        // Train wheels
        glColor3f(0.2, 0.2, 0.2); // Dark gray color for wheels
        DrawCircle(75, 925, 25, 100);   // Half of 150, 950 + 450, and 200
        DrawCircle(150, 925, 25, 100);  // Half of 300, 950 + 450, and 200
        DrawCircle(225, 925, 25, 100);  // Half of 450, 950 + 450, and 200
}
void DrawRectangle(float x, float y, float width, float height) {
        // Calculate the coordinates of the four corners of the rectangle
        float x1 = x;
        float y1 = y;
        float x2 = x + width;
        float y2 = y;
        float x3 = x + width;
        float y3 = y + height;
        float x4 = x;
        float y4 = y + height;

        // Draw the rectangle using OpenGL commands
        glBegin(GL_POLYGON);
        glVertex2f(x1, y1);
        glVertex2f(x2, y2);
        glVertex2f(x3, y3);
        glVertex2f(x4, y4);
        glEnd();
}
void house(void)
{
        // Draw the body of the house (rectangle)
        glBegin(GL_POLYGON);
        glColor3f(0.98, 0.75, 0.6); // Brighter light brown color for the body
        glVertex2i(250, 200);       // Bottom-left (lowered by 100) and increased by 200
        glVertex2i(250, 500);       // Top-left (lowered by 100) and increased by 200
        glVertex2i(500, 500);       // Top-right (lowered by 100) and increased by 200
        glVertex2i(500, 200);       // Bottom-right (lowered by 100) and increased by 200
        glEnd();

        // Draw the roof of the house (triangle)
        glBegin(GL_POLYGON);
        glColor3f(0.95, 0.6, 0.4); // Brighter lighter brown color for the roof
        glVertex2i(225, 500);      // Left point of the roof (lowered by 100) and increased by 200
        glVertex2i(375, 650);      // Top point of the roof (lowered by 100) and increased by 200
        glVertex2i(525, 500);      // Right point of the roof (lowered by 100) and increased by 200
        glEnd();

        // Draw windows
        glColor3f(0.8, 0.95, 1.0); // Brighter light blue color for windows
        DrawRectangle(275, 325, 50, 100);  // Draw a rectangle for the window (lowered by 100) and increased by 200
        DrawRectangle(400, 325, 50, 100); // Another window (lowered by 100) and increased by 200

        // Draw doors
        glColor3f(0.7, 0.5, 0.3); // Darker brown color for the door
        DrawRectangle(325, 200, 50, 100); // Draw a rectangle for the door (lowered by 100) and increased by 200

        // Draw chimney
        glColor3f(0.5, 0.5, 0.5); // Darker gray color for chimney
        DrawRectangle(425, 550, 25, 75); // Rectangle for chimney (lowered by 100) and increased by 200
}

void drawBoy() {
        // Draw body
        glColor3f(0.0, 0.5, 1.0); // Blue color for the body
        glBegin(GL_POLYGON);
        glVertex2f(535, 395);
        glVertex2f(565, 395);
        glVertex2f(570, 335);
        glVertex2f(530, 335);
        glEnd();

        // Draw head with brown color using DrawCircle function
        glColor3f(0.78, 0.57, 0.44); // Brown color for the face
        DrawCircle(550, 410, 20, 100); // Draw a circle for the head

        // Draw eyes
        glColor3f(0.0, 0.0, 0.0); // Black color for the eyes
        glPointSize(4.0); // Set point size for eyes
        glBegin(GL_POINTS);
        glVertex2f(545, 410); // Left eye
        glVertex2f(555, 410); // Right eye
        glEnd();

        // Draw mouth
        glColor3f(0.9, 0.6, 0.6); // Light pink color for the mouth
        glBegin(GL_LINE_LOOP);
        glVertex2f(545, 405);
        glVertex2f(555, 405);
        glEnd();

        // Draw hair
        glColor3f(0.0, 0.0, 0.0); // Black color for the hair
        glBegin(GL_POLYGON);
        glVertex2f(530, 430);
        glVertex2f(570, 430);
        glVertex2f(565, 450);
        glVertex2f(535, 450);
        glEnd();

        // Draw arms
        glBegin(GL_LINES);
        glVertex2f(530, 385); // Left arm
        glVertex2f(510, 365);
        glVertex2f(570, 385); // Right arm
        glVertex2f(590, 365);
        glEnd();

        // Draw legs
        glBegin(GL_LINES);
        glVertex2f(535, 335); // Left leg
        glVertex2f(525, 295);
        glVertex2f(565, 335); // Right leg
        glVertex2f(575, 295);
        glEnd();
}
void boat(int x)
{
        //back part
        glBegin(GL_POLYGON);
        glColor3f(0, 0, 0);
        glVertex2i(118, 640);
        glVertex2i(192, 640);
        glVertex2i(192, 677);
        glVertex2i(113, 667);
```

```
glEnd();

glBegin(GL_POLYGON);
glColor3f(.4078, .275, .063);
glVertex2i(120, 640);
glVertex2i(190, 640);
glVertex2i(190, 675);
glVertex2i(115, 665);
glEnd();

//boat main
glBegin(GL_POLYGON);
glColor3f(0, 0, 0);
glVertex2i(189, 760);
glVertex2i(189, 650);
glVertex2i(282, 650);
glVertex2i(282, 760);
glEnd();

glBegin(GL_POLYGON);
glColor3f(.7647, .7647, .7647);
glVertex2i(190, 760);
glVertex2i(190, 650);
glColor3f(1, 1, 1);
glVertex2i(280, 650);
glVertex2i(280, 760);
glEnd();

//chimny
glBegin(GL_POLYGON);
glColor3f(0, 0, 0);
glVertex2i(205, 797);
glVertex2i(205, 765);
glVertex2i(230, 765);
glVertex2i(230, 797);
glEnd();

glBegin(GL_POLYGON);
glColor3f(.7647, .7647, .7647);
glVertex2i(207, 795);
glVertex2i(207, 765);
glColor3f(1, 1, 1);
glVertex2i(228, 765);
glVertex2i(228, 795);
glEnd();

//boat top base
glBegin(GL_POLYGON);
glColor3f(1.0f / 255 * 181, 1.0f / 255 * 42, 1.0f / 255 * 46);
glVertex2i(185, 770);
glVertex2i(185, 750);
glColor3f(1.0f / 255 * 253, 1.0f / 255 * 0, 1.0f / 255 * 6);
glVertex2i(285, 755);
glVertex2i(285, 775);
glEnd();

//window
glColor3f(0, 0, 0);
DrawCircle(243, 720, 17, 720);

glColor3f((1.0f / 255) * 11, (1.0f / 255) * 119, (1.0f / 255) * 136);
```

```
	DrawCircle(243, 720, 15, 720);

	//front part
	glBegin(GL_POLYGON);
	glColor3f(0, 0, 0);
	glVertex2i(262, 662);
	glVertex2i(102, 652);
	glVertex2i(122, 598);
	glVertex2i(220, 593);
	glVertex2i(322, 598);
	glVertex2i(372, 712);
	glVertex2i(282, 692);
	glEnd();

	glBegin(GL_POLYGON);
	glColor3f(.4078, .275, .063);
	glVertex2i(260, 660);
	glVertex2i(100, 650);
	glVertex2i(120, 600);
	glVertex2i(220, 595);
	glVertex2i(320, 600);

	glColor3f(.6039, .4549, .321568);
	glVertex2i(370, 710);
	glVertex2i(280, 690);
	glEnd();
}

void moveboat()
{
	if (boatstatus == 1)
	{
		boatX += 3;
		w += 1;

	}
	if (boatX > 2300)
	{
		boatX = -400;
	}
	glPushMatrix();
	glTranslatef(boatX, boatY, 0);
	boat(1);
	glPopMatrix();
}

void plane(int x)
{
	//left wing dark
	glBegin(GL_POLYGON);
	glColor3f(0.0, 0.0, 0.0);
	glVertex2i(330, 1360);
	glVertex2i(345, 1380);
	glVertex2i(390, 1380);
	glVertex2i(380, 1310);
	glEnd();

	//left wing
	glBegin(GL_POLYGON);
	glColor3f(0.756862, 0.5372549, 0.878431);
	glVertex2i(330 + 2, 1360 + 2);
```

```
glVertex2i(345 + 2, 1380 - 2);
glVertex2i(390 - 2, 1380 - 2);
glVertex2i(380 - 2, 1310 + 2);
glEnd();

//left flap dark
glBegin(GL_POLYGON);
glColor3f(0.0, 0.0, 0.0);
glVertex2i(135, 1380);
glVertex2i(160, 1380);
glVertex2i(150, 1350);
glVertex2i(130, 1350);
glEnd();

//left flap
glBegin(GL_POLYGON);
glColor3f(0.756862, 0.5372549, 0.878431);
glVertex2i(135 + 2, 1380 - 2);
glVertex2i(160 - 2, 1380 - 2);
glVertex2i(150 - 2, 1350 + 2);
glVertex2i(130 + 2, 1350 + 2);
glEnd();

//mainbody tail dark
glBegin(GL_POLYGON);
glColor3f(0.0, 0.0, 0.0);
glVertex2i(100, 1400);
glVertex2i(120, 1410);
glVertex2i(140, 1360);
glVertex2i(90, 1350);
glEnd();

//mainbody dark
glBegin(GL_POLYGON);
glColor3f(0.0, 0.0, 0.0);
glVertex2i(90, 1350);
glVertex2i(140, 1360);
glVertex2i(190, 1363);
glVertex2i(350, 1363);
glVertex2i(350, 1263);
glVertex2i(330, 1263);
glVertex2i(190, 1293);
glVertex2i(175, 1330);
glVertex2i(110, 1335);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.0, 0.0, 0.0);
glVertex2i(350, 1363);
glVertex2i(370, 1333);
glVertex2i(370, 1263);
glVertex2i(350, 1263);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.0, 0.0, 0.0);
glVertex2i(370, 1333);
glVertex2i(430, 1323);
glVertex2i(405, 1263);
glVertex2i(370, 1263);
glEnd();
```

```
glBegin(GL_POLYGON);
glColor3f(0.0, 0.0, 0.0);
glVertex2i(430, 1323);
glVertex2i(435, 1315);
glVertex2i(460, 1295);
glVertex2i(405, 1263);
glEnd();

//mainbody tail
glBegin(GL_POLYGON);
glColor3f(0.756862, 0.5372549, 0.878431);
glVertex2i(100 + 2, 1400 - 2);
glVertex2i(120 - 2, 1410 - 3);
glVertex2i(140 - 2, 1360);
glVertex2i(90 + 2, 1350);
glEnd();

//mainbody
glBegin(GL_POLYGON);
glColor3f(0.756862, 0.5372549, 0.878431);
glVertex2i(90 + 2, 1350);
glVertex2i(140 - 2, 1360);
glVertex2i(190, 1363 - 2);
glVertex2i(350, 1363 - 2);
glVertex2i(350, 1263 + 2);
glVertex2i(330, 1263 + 2);
glVertex2i(190, 1293 + 2);
glVertex2i(175, 1330 + 2);
glVertex2i(110, 1335 + 2);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.756862, 0.5372549, 0.878431);
glVertex2i(350, 1363 - 2);
glVertex2i(370, 1333 - 2);
glVertex2i(370, 1263 + 2);
glVertex2i(350, 1263 + 2);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.756862, 0.5372549, 0.878431);
glVertex2i(370, 1333 - 2);
glVertex2i(430, 1323 - 2);
glVertex2i(405, 1263 + 2);
glVertex2i(370, 1263 + 2);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.756862, 0.5372549, 0.878431);
glVertex2i(430, 1323 - 2);
glVertex2i(435, 1315 - 2);
glVertex2i(460 - 2, 1295);
glVertex2i(405, 1263 + 2);
glEnd();

//right wing dark
glBegin(GL_POLYGON);
glColor3f(0.0, 0.0, 0.0);
glVertex2i(300, 1293);
glVertex2i(280, 1200);
```

```
glVertex2i(350, 1200);
glVertex2i(390, 1293);
glEnd();

//right wing
glBegin(GL_POLYGON);
glColor3f(0.756862, 0.5372549, 0.878431);
glVertex2i(300 + 2, 1293 - 2);
glVertex2i(280 + 2, 1200 + 2);
glVertex2i(350 - 2, 1200 + 2);
glVertex2i(390 - 2, 1293 - 2);
glEnd();

//right flap dark
glBegin(GL_POLYGON);
glColor3f(0.0, 0.0, 0.0);
glVertex2i(100, 1300);
glVertex2i(115, 1340);
glVertex2i(155, 1340);
glVertex2i(125, 1300);
glEnd();

//right flap
glBegin(GL_POLYGON);
glColor3f(0.756862, 0.5372549, 0.878431);
glVertex2i(100 + 2, 1300 + 2);
glVertex2i(115 + 2, 1340 - 2);
glVertex2i(155 - 2, 1340 - 2);
glVertex2i(125 - 2, 1300 + 2);
glEnd();

//cockpit dark
glBegin(GL_POLYGON);
glColor3f(0.0, 0.0, 0.0);
glVertex2i(310, 1363);
glVertex2i(350, 1363);
glVertex2i(370, 1333);
glVertex2i(340, 1323);
glVertex2i(290, 1333);
glEnd();

//cockpit
glBegin(GL_POLYGON);
glColor3f(0.9372549, 0.9372549, 0.9372549);
glVertex2i(310 + 2, 1363 - 2);
glVertex2i(350 - 2, 1363 - 2);
glVertex2i(370 - 2, 1333);
glVertex2i(340, 1323 + 2);
glVertex2i(290 + 2, 1333 + 2);
glEnd();

//star dark
glBegin(GL_TRIANGLES);
glColor3f(0.0, 0.0, 0.0);
glVertex2i(105 - 2, 1375 + 2);
glVertex2i(125 + 2, 1375 + 2);
glVertex2i(115, 1355 - 2);
glEnd();

glBegin(GL_TRIANGLES);
glColor3f(0.0, 0.0, 0.0);
```

```
	glVertex2i(105 - 2, 1365 - 2);
	glVertex2i(125 + 2, 1365 - 2);
	glVertex2i(115, 1385 + 2);
	glEnd();

	//star
	glBegin(GL_TRIANGLES);
	glColor3f(0.929411, 0.109803, 0.141176);
	glVertex2i(105, 1375);
	glVertex2i(125, 1375);
	glVertex2i(115, 1355);
	glEnd();

	glBegin(GL_TRIANGLES);
	glColor3f(0.929411, 0.109803, 0.141176);
	glVertex2i(105, 1365);
	glVertex2i(125, 1365);
	glVertex2i(115, 1385);
	glEnd();
}

void moveplane()
{
	if (planestatus == 1)
	{
		planeX += 6;
		w += 1;

	}
	if (planeX > 2300)
	{
		planeX = -400;
	}
	glPushMatrix();
	glTranslatef(planeX, planeY, 0);
	plane(1);
	glPopMatrix();
}

void cloud(int x)
{
	glColor3f(0.447, 0.624, 0.812);
	DrawCircle(196 + 400, 1280 - 20, 60, 2000);//left
	DrawCircle(236 + 400, 1350 - 20, 55, 2000);//left
	DrawCircle(360 + 400, 1374 - 20, 55, 2000);//right
	DrawCircle(400 + 400, 1324 - 20, 55, 2000);//right
	DrawCircle(290 + 400, 1393, 60, 2000);//top
	DrawCircle(296 + 400, 1296, 80, 2000);//middle

	glColor3f(0.933, 0.933, 0.933);
	DrawCircle(200 + 400, 1280 - 20, 60, 2000);//1
	DrawCircle(240 + 400, 1350 - 20, 55, 2000);//2
	DrawCircle(360 + 400, 1370 - 20, 55, 2000);//3
	DrawCircle(400 + 400, 1320 - 20, 55, 2000);//4
	DrawCircle(290 + 400, 1390, 60, 2000);//4
	DrawCircle(300 + 400, 1300, 80, 2000);//middle

	glColor3f(0.447, 0.624, 0.812);//last
	DrawCircle(96 + 1400, 1280 - 20, 60, 2000);//left
	DrawCircle(136 + 1400, 1350 - 20, 55, 2000);//left
	DrawCircle(260 + 1400, 1374 - 20, 55, 2000);//right
	DrawCircle(300 + 1400, 1324 - 20, 55, 2000);//right
	DrawCircle(190 + 1400, 1393, 60, 2000);//top
	DrawCircle(196 + 1400, 1296, 80, 2000);//middle

	glColor3f(0.933, 0.933, 0.933);//last
	DrawCircle(100 + 1400, 1280 - 20, 60, 2000);//1
	DrawCircle(140 + 1400, 1350 - 20, 55, 2000);//2
	DrawCircle(260 + 1400, 1370 - 20, 55, 2000);//3
	DrawCircle(300 + 1400, 1320 - 20, 55, 2000);//4
	DrawCircle(190 + 1400, 1390, 60, 2000);//4
	DrawCircle(200 + 1400, 1300, 80, 2000);//middle
}

void nightcloud(int x)
{
	glColor3f(0.447, 0.447, 0.447);
	DrawCircle(196 + 400, 1280 - 20, 60, 2000);//left
	DrawCircle(236 + 400, 1350 - 20, 55, 2000);//left
	DrawCircle(360 + 400, 1374 - 20, 55, 2000);//right
	DrawCircle(400 + 400, 1324 - 20, 55, 2000);//right
	DrawCircle(290 + 400, 1393, 60, 2000);//top
	DrawCircle(296 + 400, 1296, 80, 2000);//middle

	glColor3f(0.733, 0.733, 0.733);
	DrawCircle(200 + 400, 1280 - 20, 60, 2000);//1
	DrawCircle(240 + 400, 1350 - 20, 55, 2000);//2
	DrawCircle(360 + 400, 1370 - 20, 55, 2000);//3
	DrawCircle(400 + 400, 1320 - 20, 55, 2000);//4
	DrawCircle(290 + 400, 1390, 60, 2000);//4
	DrawCircle(300 + 400, 1300, 80, 2000);//middle

	glColor3f(0.447, 0.447, 0.447);//last
	DrawCircle(96 + 1400, 1280 - 20, 60, 2000);//left
	DrawCircle(136 + 1400, 1350 - 20, 55, 2000);//left
	DrawCircle(260 + 1400, 1374 - 20, 55, 2000);//right
	DrawCircle(300 + 1400, 1324 - 20, 55, 2000);//right
	DrawCircle(190 + 1400, 1393, 60, 2000);//top
	DrawCircle(196 + 1400, 1296, 80, 2000);//middle

	glColor3f(0.733, 0.733, 0.733);//last
	DrawCircle(100 + 1400, 1280 - 20, 60, 2000);//1
	DrawCircle(140 + 1400, 1350 - 20, 55, 2000);//2
	DrawCircle(260 + 1400, 1370 - 20, 55, 2000);//3
	DrawCircle(300 + 1400, 1320 - 20, 55, 2000);//4
	DrawCircle(190 + 1400, 1390, 60, 2000);//4
	DrawCircle(200 + 1400, 1300, 80, 2000);//middle
}

void movecloud()
{
	if (cloudStatus == 1)
	{
		cloudX -= 1;
		w += 1;

	}
	if (cloudX < -1500)
	{
		cloudX = 2000;
	}
	glPushMatrix();
	glTranslatef(cloudX, cloudY, 0);
	if (n == 0)
	{
		cloud(1);
	}
	else if (n == 1)
	{
		nightcloud(1);
	}
	glPopMatrix();
}

void keyboard(unsigned char key, int x, int y)
{
	if (key == '1')
	{
		n = 1;
	}
	else if (key == '6')
	{
		rainStatus = 0;
	}
	else if (key == '7')
	{
		rainStatus = 1;
	}
	else if (key == '0')
	{
		n = 0;
	}
	else if (key == 'T' || key == 't')
	{
		trainstatus = 1;
	}
	else if (key == 'R' || key == 'r')
	{
		trainstatus = 0;
	}
	else if (key == 'B' || key == 'b')
	{
		boatstatus = 1;
	}
	else if (key == 'V' || key == 'v')
	{
		boatstatus = 0;
	}
	else if (key == 'P' || key == 'p')
	{
		planestatus = 1;
	}
	else if (key == 'O' || key == 'o')
	{
		planestatus = 0;
	}
}

void moveTrain()
{
	if (trainstatus == 1)
	{
		trainX += 3;
```

```
                w += 1;

        }
        if (trainX > 2300)
        {
                trainX = -400;
        }
        glPushMatrix();
        glTranslatef(trainX, trainY, 0);
        train(1);
        glPopMatrix();
}


void timer(int value) {


        glutPostRedisplay();
        glutTimerFunc(16, timer, 0); // Update every 16 milliseconds (about 60 FPS)
}

struct WaterDrop {
        float x;
        float y;
        bool active;
};

const int MAX_WATER_DROPS = 100;
WaterDrop waterDrops[MAX_WATER_DROPS];

void initWaterDrops()
{
        // Initialize water drops to cover the entire width of the river
        for (int i = 0; i < MAX_WATER_DROPS; i++)
        {
                waterDrops[i].x = i * (2000.0f / MAX_WATER_DROPS);  // Evenly distribute drops across the river
width
                waterDrops[i].y = rand() % 770 + 200;  // Random initial Y position within a range (adjust as needed)
                waterDrops[i].active = true;  // Activate the drop
        }
}


void moveWaterDrops()
{
        // Move water drops horizontally (from left to right)
        for (int i = 0; i < MAX_WATER_DROPS; i++)
        {
                if (waterDrops[i].active)
                {
                        waterDrops[i].x += 2;  // Adjust the horizontal movement speed as needed

                        // Check if the drop has moved out of bounds
                        if (waterDrops[i].x > 2000)
                        {
                                waterDrops[i].x = 0;  // Reset the drop's position to the left side
                        }
                }
        }
}
```

```
void drawWaterDrops()
{
	// Light blue color for water drops
	glColor3f(0.7f, 0.7f, 1.0f);  // Adjust RGB values for a lighter shade

	// Draw water drops as longer lines
	glBegin(GL_LINES);
	for (int i = 0; i < MAX_WATER_DROPS; i++)
	{
		if (waterDrops[i].active)
		{
			float x1 = waterDrops[i].x;
			float y1 = waterDrops[i].y;
			float x2 = x1 + 20.0f;  // Length of the line (adjust as needed)
			float y2 = y1;  // Horizontal line, same Y coordinate

			glVertex2f(x1, y1);  // Start point of the line
			glVertex2f(x2, y2);  // End point of the line
		}
	}
	glEnd();
}




void myDisplay(void) {
	sky();
	skynight();
	sun();
	moon();
	movecloud();
	river();
	rivernight();
	moveWaterDrops();
	drawWaterDrops();
	moveplane();
	city();
	bridge();
	field();
	fieldnight();
	moveboat();
	house();
	drawBoy();
	tree();
	treenight();
	drawRainNight(); // Draw rain during night
	moveTrain();
	glFlush();
	// glLoadIdentity();
	// Call the functions to simulate moving water drops


	glutPostRedisplay();
	glutSwapBuffers();
}
```

```
void myInit(void)
{
	glClearColor(0.0, 0.0, 1.0, 0.0);
	glColor3f(1.0f, 1.0f, 1.0f);
	glPointSize(0.0);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0.0, 2000.0, 0.0, 1500.0);
	initRain();
}

int main(int argc, char** argv) {
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
	glutInitWindowSize(2000, 1500);
	glutInitWindowPosition(0, 0);
	glutCreateWindow("Group-8!");
	glutKeyboardFunc(keyboard);
	glutDisplayFunc(myDisplay);
	myInit();
	glutTimerFunc(0, timer, 0);
	initWaterDrops();
	glutMainLoop();
	return 0;
}
```

Output / Screen Shots