CSE 358

Computer Graphics Lab

Title: Van Gogh Walking Under Starry Sky

Submitted by:

1. Arpa Bhowmik (*0432220005101102)*
2. Fahima Abida Chowdhury(0432220005101135)
3. Yeasmin Kabir Keya (0432220005101107)
4. Sawda Akter (0432220005101125)

Institution: University of Information and Technology Sciences(UITS)

Semester : Spring 2025

Batch : 52

Section : 6B1

Supervisor: Dhrubo Barua(Lecturer) ,Department of CSE ,UITS

Md. Azharul Karim Chowdhury Anik(Lecturer) , Department of CSE ,UITS

Date of Submission: 04.06.2025

1. Introduction

This project titled **"Van Gogh Walking Under Starry Sky"** is a computer graphics animation implemented using OpenGL and GLUT. It aims to simulate a day-night environment with interactive character movement. The purpose is to demonstrate core concepts of animation, real-time rendering, and user interaction in 2D graphical systems.

Problem Definition: How can we create an engaging 2D animated environment with dynamic natural elements and real-time user interaction for educational or storytelling purposes?

Scope & Objectives:

- Implement a toggle between day and night visual themes
- Simulate natural elements like sun/moon, stars, clouds, buildings, trees, and ground
- Enable character movement using keyboard arrow keys
- Illustrate the principles of real-time animation and scene management using OpenGL

Real-World Applications:

- Educational tools in visual computing
- Animation in interactive storytelling and gaming
- Prototypes for 2D game environments and simulations

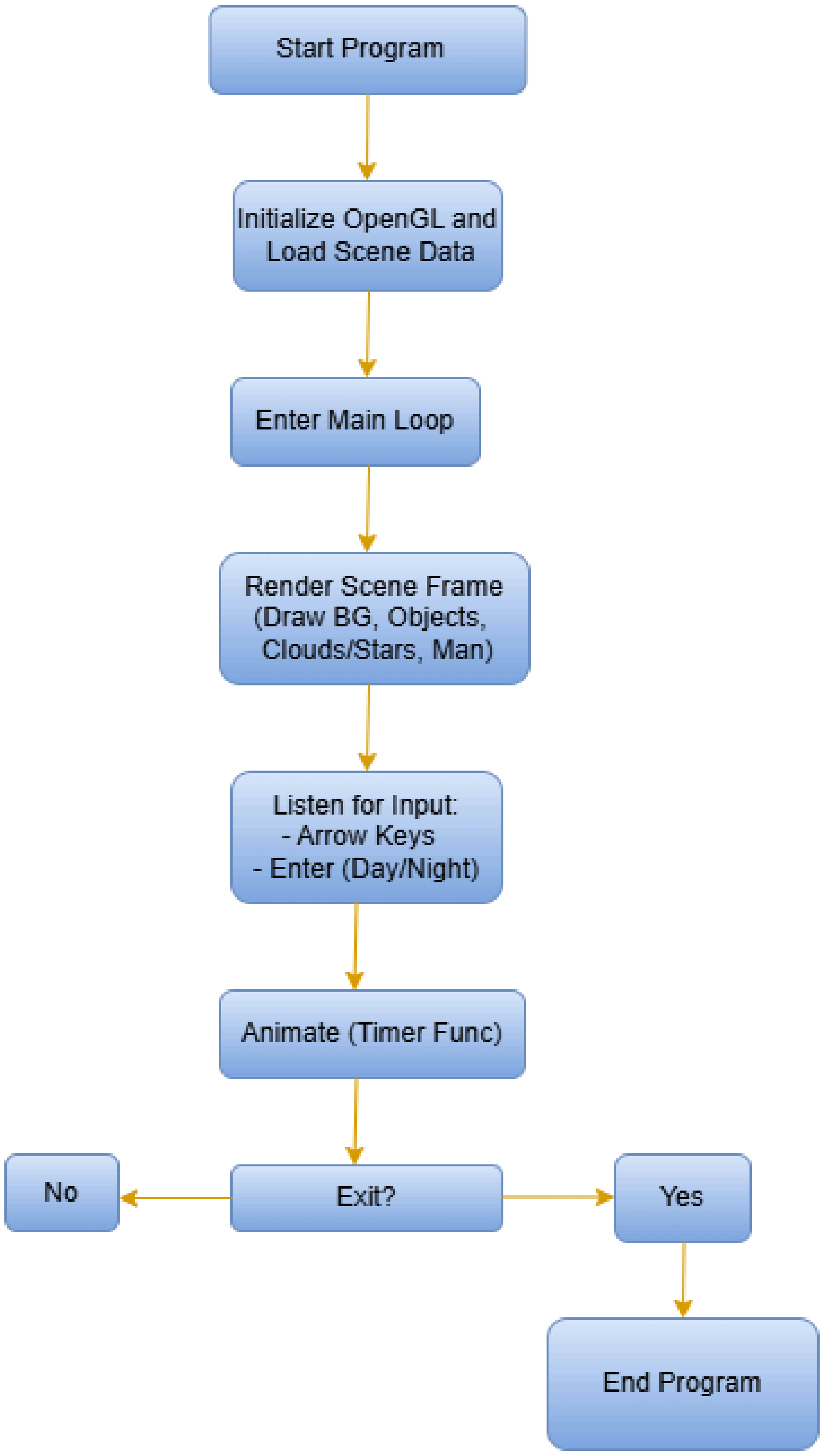
2. Methodology

This section describes the approach used to implement the project.

The project was developed in **C++** using the **OpenGL Utility Toolkit (GLUT)** for graphical rendering and user interaction. The development process includes:

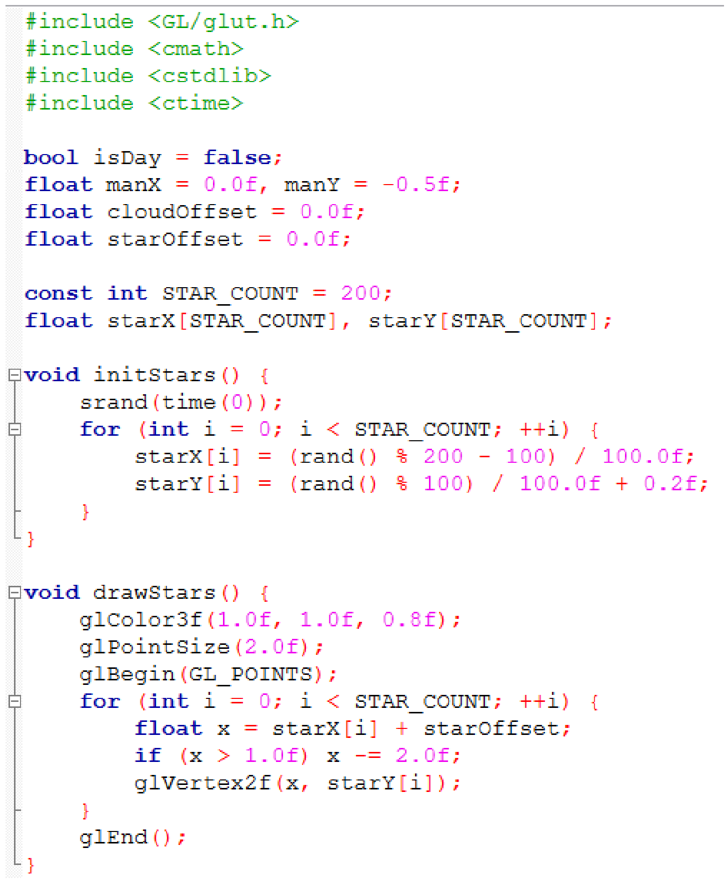
- Initializing the rendering context and coordinate system
- Drawing and layering 2D elements such as hills, buildings, stars, trees, and grass
- Rendering dynamic components (clouds, stars) with animation through timer functions
- Handling keyboard input to move the character and toggle between day/night modes

Flowchart:

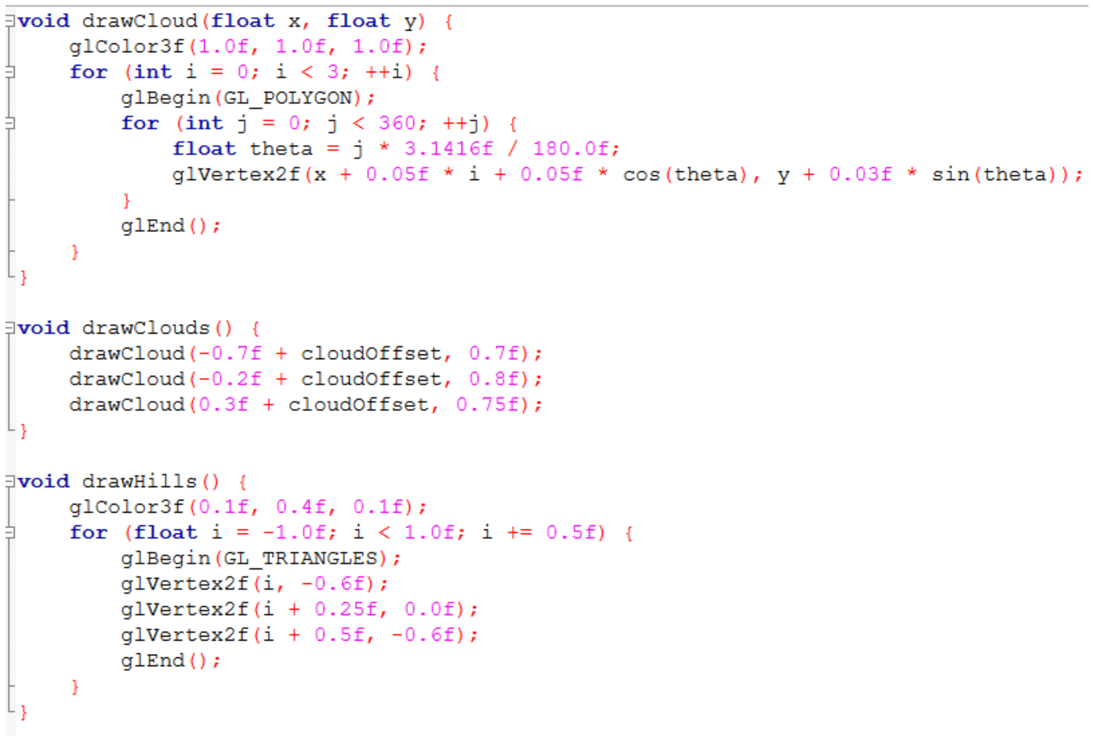

Start Program
Initialize OpenGL and Load Scene Data
Enter Main Loop
Render Scene Frame (Draw BG, Objects, Clouds/Stars, Man)
Listen for Input: - Arrow Keys - Enter (Day/Night)
Animate (Timer Func)
Exit?
No
Yes
End Program

3. Result & Discussion

Project Code:



```cpp
#include <GL/glut.h>
#include <cmath>
#include <cstdlib>
#include <ctime>

bool isDay = false;
float manX = 0.0f, manY = -0.5f;
float cloudOffset = 0.0f;
float starOffset = 0.0f;

const int STAR_COUNT = 200;
float starX[STAR_COUNT], starY[STAR_COUNT];

void initStars() {
    srand(time(0));
    for (int i = 0; i < STAR_COUNT; ++i) {
        starX[i] = (rand() % 200 - 100) / 100.0f;
        starY[i] = (rand() % 100) / 100.0f + 0.2f;
    }
}

void drawStars() {
    glColor3f(1.0f, 1.0f, 0.8f);
    glPointSize(2.0f);
    glBegin(GL_POINTS);
    for (int i = 0; i < STAR_COUNT; ++i) {
        float x = starX[i] + starOffset;
        if (x > 1.0f) x -= 2.0f;
        glVertex2f(x, starY[i]);
    }
    glEnd();
}
```

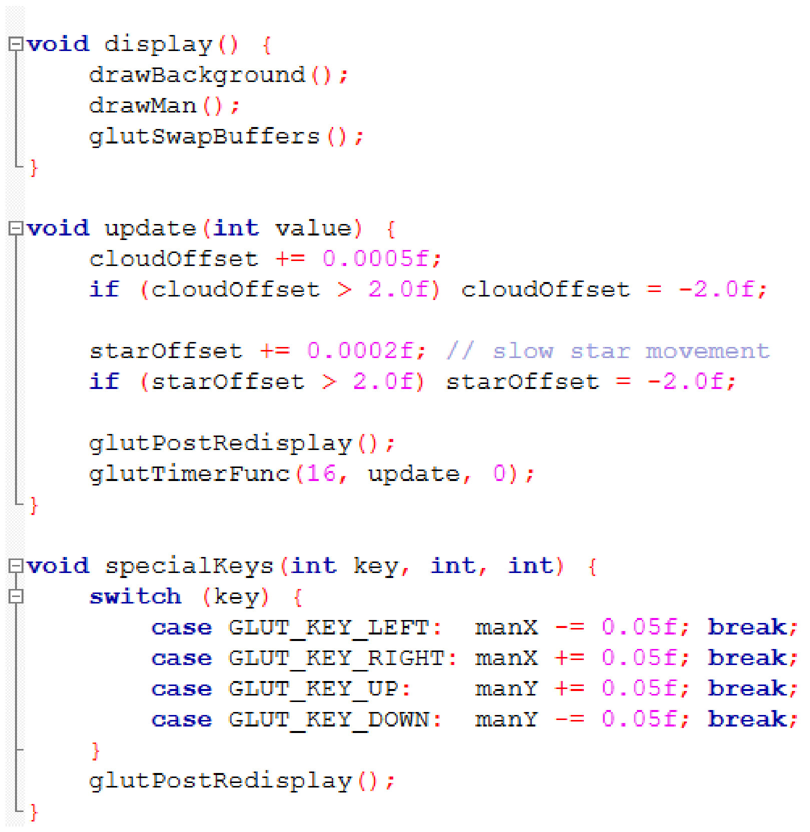


```cpp
void drawCloud(float x, float y) {
    glColor3f(1.0f, 1.0f, 1.0f);
    for (int i = 0; i < 3; ++i) {
        glBegin(GL_POLYGON);
        for (int j = 0; j < 360; ++j) {
            float theta = j * 3.1416f / 180.0f;
            glVertex2f(x + 0.05f * i + 0.05f * cos(theta), y + 0.03f * sin(theta));
        }
        glEnd();
    }
}

void drawClouds() {
    drawCloud(-0.7f + cloudOffset, 0.7f);
    drawCloud(-0.2f + cloudOffset, 0.8f);
    drawCloud(0.3f + cloudOffset, 0.75f);
}

void drawHills() {
    glColor3f(0.1f, 0.4f, 0.1f);
    for (float i = -1.0f; i < 1.0f; i += 0.5f) {
        glBegin(GL_TRIANGLES);
        glVertex2f(i, -0.6f);
        glVertex2f(i + 0.25f, 0.0f);
        glVertex2f(i + 0.5f, -0.6f);
        glEnd();
    }
}
```

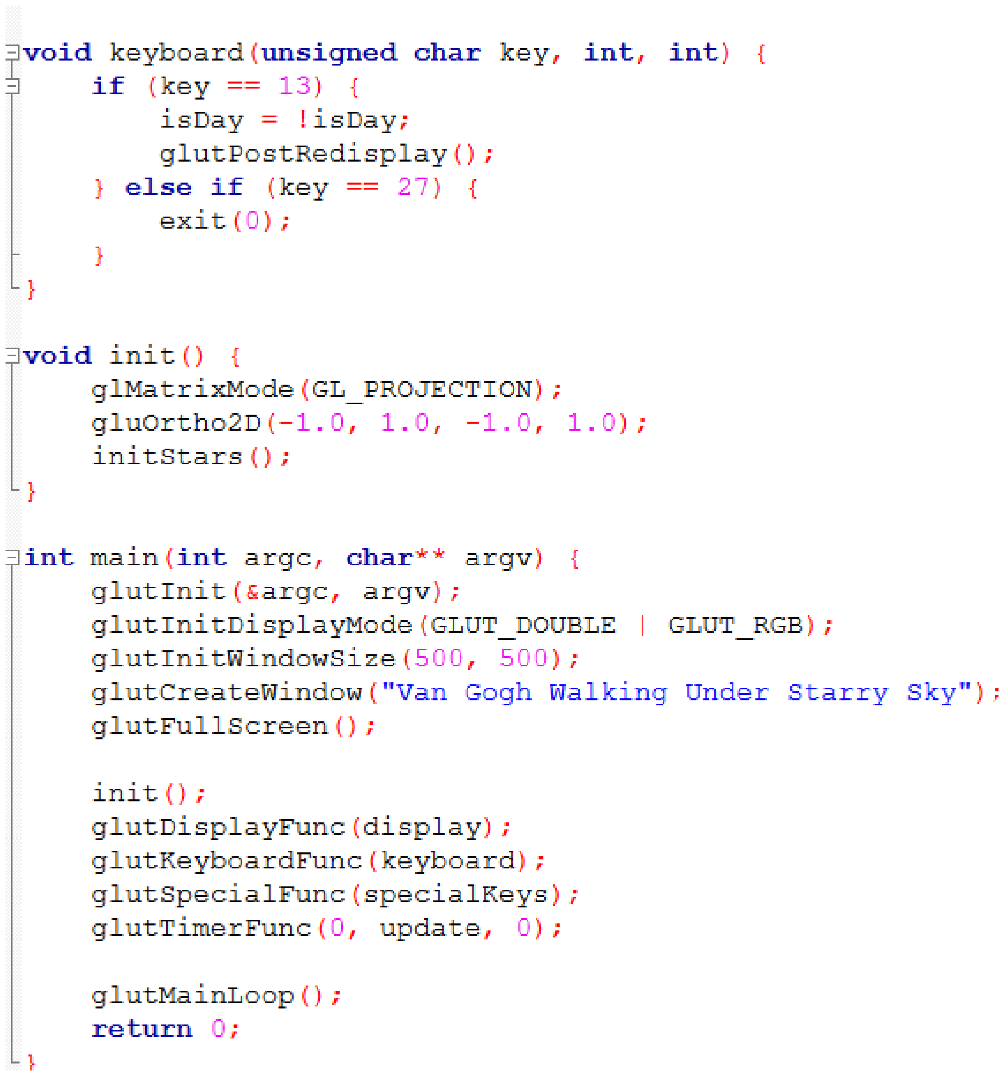
```
void drawSunOrMoon() {
    if (isDay)
        glColor3f(1.0f, 0.9f, 0.0f);
    else
        glColor3f(0.9f, 0.9f, 1.0f);

    glBegin(GL_POLYGON);
    for (int i = 0; i < 360; i++) {
        float theta = i * 3.1416f / 180.0f;
        glVertex2f(0.8f + 0.1f * cos(theta), 0.8f + 0.1f * sin(theta));
    }
    glEnd();
}
void drawBuilding(float x) {
    glColor3f(0.3f, 0.3f, 0.5f);
    float w = 0.2f, h = 0.3f;
    float y = -0.6f;

    glBegin(GL_QUADS);
    glVertex2f(x, y);
    glVertex2f(x + w, y);
    glVertex2f(x + w, y + h);
    glVertex2f(x, y + h);
    glEnd();
    for (int row = 0; row < 3; ++row) {
        for (int col = 0; col < 2; ++col) {
            float winX = x + 0.03f + col * 0.07f;
            float winY = y + 0.05f + row * 0.08f;
            glColor3f(1.0f, 1.0f, isDay ? 0.6f : 0.9f);
            glBegin(GL_QUADS);
            glVertex2f(winX, winY);
            glVertex2f(winX + 0.04f, winY);
            glVertex2f(winX + 0.04f, winY + 0.05f);
            glVertex2f(winX, winY + 0.05f);
            glEnd();
        }

void drawBuildings() {
    for (float i = -0.9f; i < 1.0f; i += 0.3f) {
        drawBuilding(i);
    }
}

void drawTree(float x, float y) {
    glColor3f(0.4f, 0.2f, 0.1f);
    glBegin(GL_QUADS);
    glVertex2f(x - 0.01f, y);
    glVertex2f(x + 0.01f, y);
    glVertex2f(x + 0.01f, y + 0.1f);
    glVertex2f(x - 0.01f, y + 0.1f);
    glEnd();

    glColor3f(0.0f, 0.6f, 0.2f);
    glBegin(GL_TRIANGLES);
    glVertex2f(x - 0.05f, y + 0.1f);
    glVertex2f(x + 0.05f, y + 0.1f);
    glVertex2f(x, y + 0.2f);
    glEnd();
}

void drawGrass() {
    glColor3f(0.0f, 0.6f, 0.0f);
    for (float i = -1.0f; i < 1.0f; i += 0.02f) {
        glBegin(GL_LINES);
        glVertex2f(i, -0.6f);
        glVertex2f(i + 0.005f, -0.58f + ((rand() % 10) / 1000.0f));
        glEnd();
    }
}
```

```
void drawBackground() {
    if (isDay)
        glClearColor(0.53f, 0.81f, 0.98f, 1.0f); // Day
    else
        glClearColor(0.05f, 0.05f, 0.2f, 1.0f);  // Night

    glClear(GL_COLOR_BUFFER_BIT);

    drawSunOrMoon();
    if (!isDay) drawStars(); else drawClouds();

    drawHills();
    drawBuildings();

    drawTree(-0.85f, -0.6f);
    drawTree(-0.5f, -0.6f);
    drawTree(-0.15f, -0.6f);
    drawTree(0.2f, -0.6f);
    drawTree(0.6f, -0.6f);

    drawGrass();

    // Ground (yellowish brown)
    glColor3f(0.8f, 0.6f, 0.2f);
    glBegin(GL_QUADS);
    glVertex2f(-1.0f, -0.6f);
    glVertex2f(1.0f, -0.6f);
    glVertex2f(1.0f, -1.0f);
    glVertex2f(-1.0f, -1.0f);
    glEnd();
}
```

```
void drawMan() {
    glPushMatrix();
    glTranslatef(manX, manY, 0.0f);

    // Head (skin color)
    glColor3f(1.0f, 0.8f, 0.6f);
    glBegin(GL_POLYGON);
    for (int i = 0; i < 360; ++i) {
        float theta = i * 3.1416f / 180.0f;
        glVertex2f(0.05f * cos(theta), 0.05f * sin(theta) + 0.5f)
    }
    glEnd();

    // Cap (orange)
    glColor3f(1.0f, 0.5f, 0.0f);
    glBegin(GL_TRIANGLES);
    glVertex2f(-0.06f, 0.56f);
    glVertex2f(0.06f, 0.56f);
    glVertex2f(0.0f, 0.62f);
    glEnd();

    // Hair band or rim under cap (darker orange)
    glColor3f(0.8f, 0.3f, 0.0f);
    glBegin(GL_QUADS);
    glVertex2f(-0.06f, 0.53f);
    glVertex2f(0.06f, 0.53f);
    glVertex2f(0.06f, 0.56f);
    glVertex2f(-0.06f, 0.56f);
    glEnd();

    // Neck area (brown)
    glColor3f(0.3f, 0.2f, 0.1f);
    glBegin(GL_QUADS);
    glVertex2f(-0.06f, 0.53f);
    glVertex2f(0.06f, 0.53f);
    glVertex2f(0.06f, 0.56f);
    glVertex2f(-0.06f, 0.56f);
    glEnd();

    // Shirt (red)
    glColor3f(0.8f, 0.0f, 0.0f);
    glBegin(GL_QUADS);
    glVertex2f(-0.05f, 0.48f);
    glVertex2f(0.05f, 0.48f);
    glVertex2f(0.05f, 0.35f);
    glVertex2f(-0.05f, 0.35f);
    glEnd();

    // Arms (black lines)
    glColor3f(0.0f, 0.0f, 0.0f);
    glBegin(GL_LINES);
    glVertex2f(-0.02f, 0.35f); glVertex2f(-0.02f, 0.2f);
    glVertex2f(0.02f, 0.35f); glVertex2f(0.02f, 0.2f);
    glEnd();

    glBegin(GL_LINES);
    glVertex2f(-0.05f, 0.45f); glVertex2f(-0.08f, 0.4f);
    glVertex2f(0.05f, 0.45f); glVertex2f(0.08f, 0.4f);
    glEnd();

    glPopMatrix();
}
```

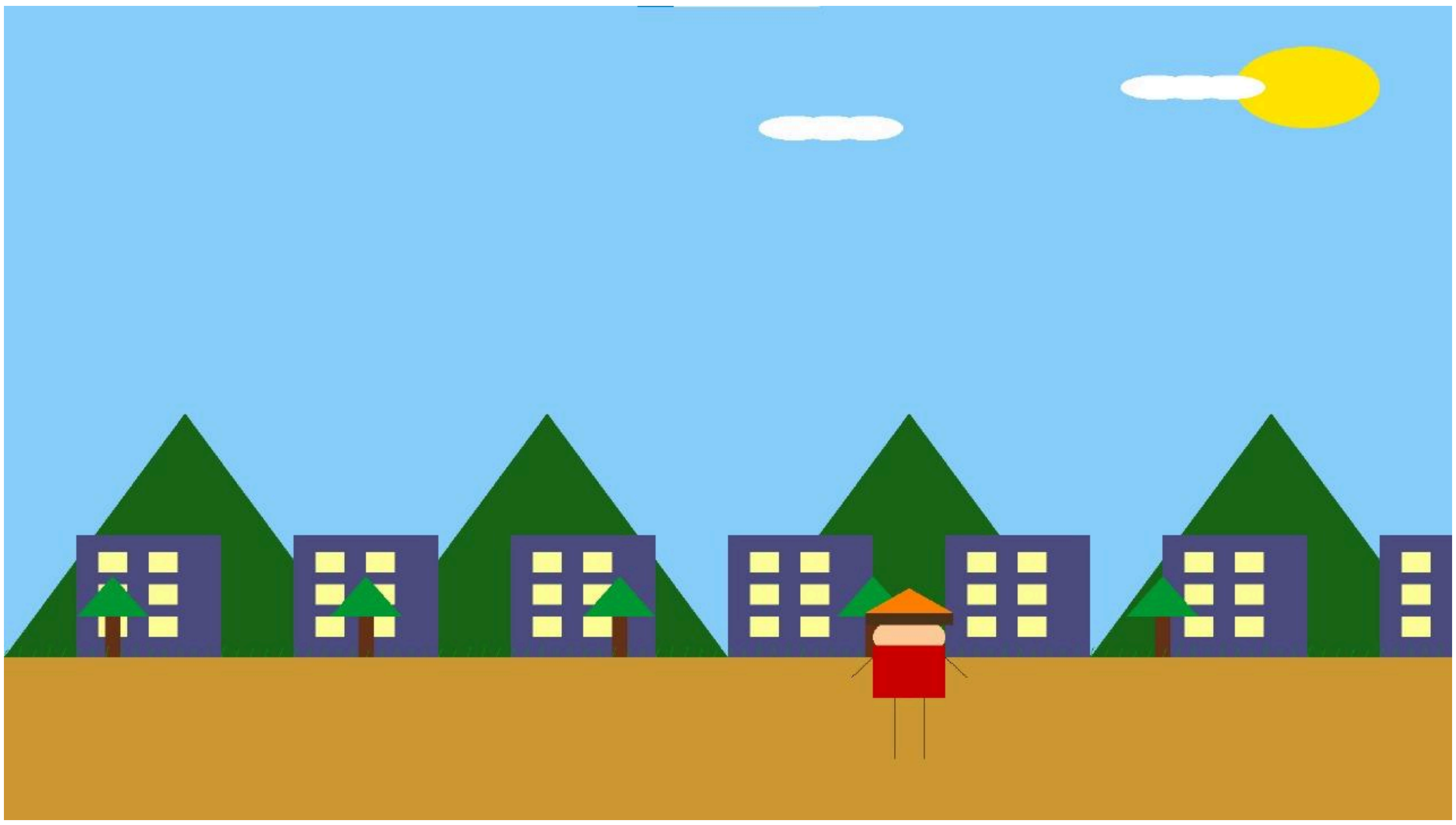


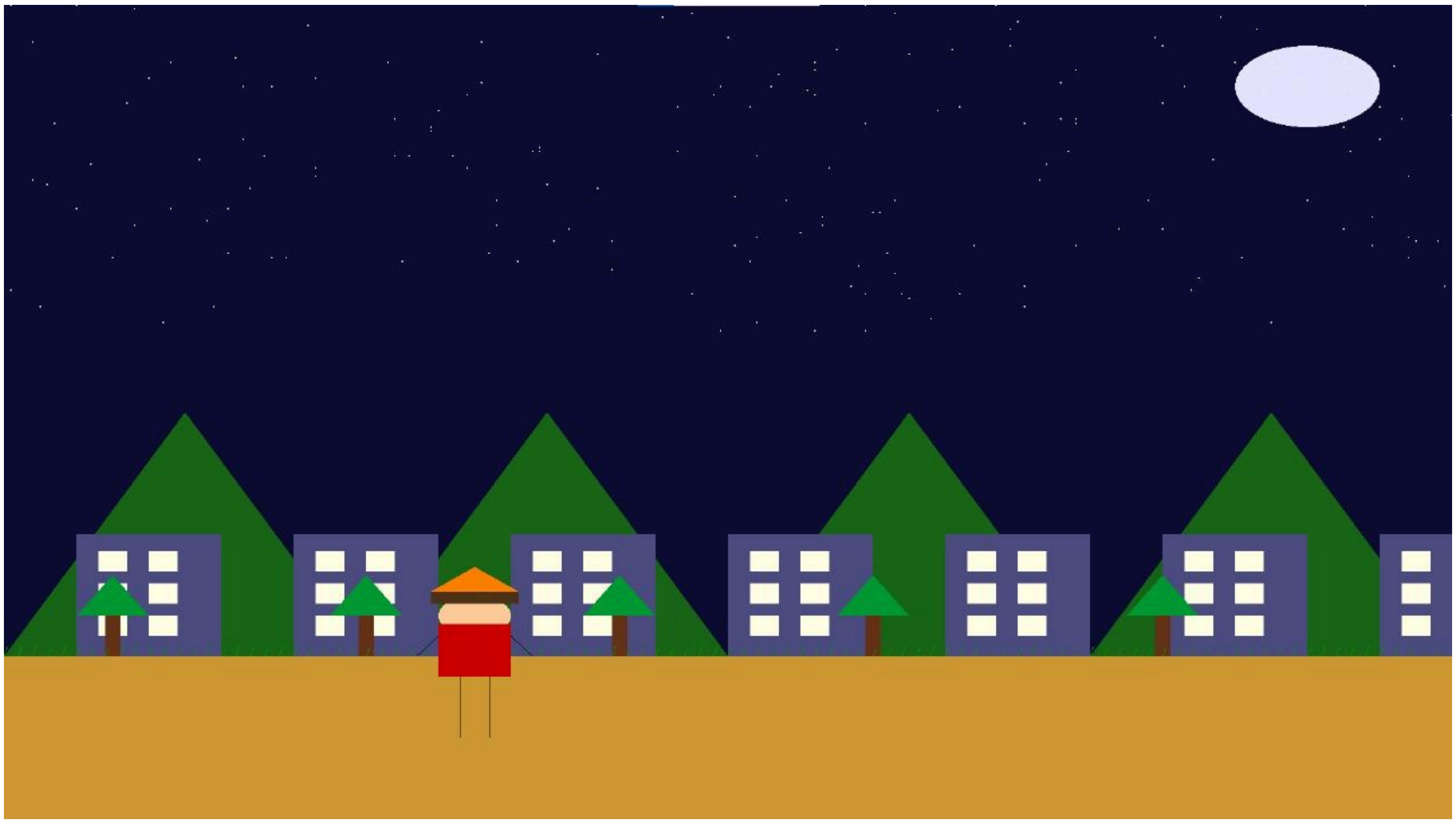
```
void display() {
    drawBackground();
    drawMan();
    glutSwapBuffers();
}

void update(int value) {
    cloudOffset += 0.0005f;
    if (cloudOffset > 2.0f) cloudOffset = -2.0f;

    starOffset += 0.0002f; // slow star movement
    if (starOffset > 2.0f) starOffset = -2.0f;

    glutPostRedisplay();
    glutTimerFunc(16, update, 0);
}

void specialKeys(int key, int, int) {
    switch (key) {
        case GLUT_KEY_LEFT:  manX -= 0.05f; break;
        case GLUT_KEY_RIGHT: manX += 0.05f; break;
        case GLUT_KEY_UP:    manY += 0.05f; break;
        case GLUT_KEY_DOWN:  manY -= 0.05f; break;
    }
    glutPostRedisplay();
}
```



```
void keyboard(unsigned char key, int, int) {
    if (key == 13) {
        isDay = !isDay;
        glutPostRedisplay();
    } else if (key == 27) {
        exit(0);
    }
}

void init() {
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(-1.0, 1.0, -1.0, 1.0);
    initStars();
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Van Gogh Walking Under Starry Sky");
    glutFullScreen();

    init();
    glutDisplayFunc(display);
    glutKeyboardFunc(keyboard);
    glutSpecialFunc(specialKeys);
    glutTimerFunc(0, update, 0);

    glutMainLoop();
    return 0;
}
```

Output Samples:





Implementation Outcomes:

- Dynamic visual transition between day and night (triggered with Enter key)
- Continuous movement of clouds (day) and stars (night) using time-based animation
- Complex yet consistent static environment: hills, trees, buildings, and grass
- Responsive character movement via arrow keys in all four directions

Objective Fulfillment: The project meets all stated objectives, providing a real-time, animated environment that reacts to user input while simulating natural cycles.

Challenges Encountered:

- Frame synchronization across different systems
- Drawing order to preserve scene realism (depth perception in 2D)
- Handling character movement boundaries within the window

4. Conclusion

This section summarizes project achievements, key learnings, and possible improvements.

The "Van Gogh Walking Under Starry Sky" project successfully demonstrates the power of OpenGL in creating interactive and visually dynamic 2D scenes. By combining basic primitives and animation techniques, it delivers a complete experience from scene construction to real-time interaction.

Key Learnings:

- Real-time rendering and animation principles using OpenGL
- Keyboard input handling for object control
- Layering of visual elements and managing object depth
- Using color theory to distinguish time-based scene changes

Future Enhancements:

- Implement parallax background scrolling for depth simulation
- Add sound effects or music synchronized with time-of-day
- Expand character animations (e.g., walking motion, jumping)

5. References

1. Angel, Edward & Shreiner, Dave. *Interactive Computer Graphics*. Pearson.
2. OpenGL Programming Guide (The Red Book)
3. https://learnopengl.com/
4. https://www.geeksforgeeks.org/computer-graphics-tutorials/
5. OpenGL GLUT documentation
6. Class lectures and lab instructions

.