আন্তর্জাতিক ইসলামী বিশ্ববিদ্যালয় চট্টগ্রাম

الجامعة الإسلامية العالمية شيتاغونغ

International Islamic University Chittagong

Department of Computer Science & Engineering

COMPUTER GRAPHICS PROJECT

A RAILWAY STATION

WITH SEASON CHANGE

SUBMITTED BY

NAME: MAKSURA BINTE RABBANI NUHA

ID: C211224

SEMESTER:7TH

SECTION:7AF

COURSE CODE:CSE-4742

COURSE TITLE: COMPUTER GRAPHICS

SUBMITTED TO

UMME HABIBA BUSHRA

ADJUNCT LECTURER, DEPT OF CSE

INTERNATIONAL ISLAMIC UNIVERSITY CHITTAGONG

TABLE OF CONTENTS

Introduction

"A Railway Station with Season Change" project, a dynamic and immersive simulation created using C++ and OpenGL. This project brings to life a bustling railway station, complete with animated trains, buses, and boats, set against the backdrop of a constantly changing environment that transitions through all four seasons. In this animated project, main highlights are the scenario with a detailed railway station terminal, a bus stand, and a rail track where passengers wait for their trains. The project features two trains that move along the tracks, with realistic sound effects such as train horns, raindrops, and wind, enhancing the immersive experience.This project demonstrates the potential of computer graphics to create vivid, lifelike environments and offers an engaging, multifaceted experience that captures the essence of each season.

Key aspects of this project

Realistic Settings

A railway station terminal and a bus stand, where the passengers gather. The trains, buses, and boats in continuous motion.

Seasonal Changes

Summer Season: Blooming flowers and sun reflections on the river.
Rainy Season: Cloud movements and falling raindrops.
Winter Season: Falling leaves and snow.
Autumn Season: Wild sugarcane or Kans grass by the rail track.
Spring Season: Different Flowers in the tree.

Sound Effects

Train horns, raindrops, and windy effects enhanced the realism in this project.

Working Procedure

Initialization

The init() function sets up the viewing values using orthographic projection.

Main Function:

- GLUT initialization is performed.
- A window titled **"Railway Station with Season Change"** is created.
- The init() function is called to initialize the viewing settings.
- Display callback function information() is initially registered to handle window repaints.
- Keyboard callback function myKeyboard() is registered to handle keyboard input.
- Timers are set up to trigger specific actions at specified intervals for seasonal changes and sound effects.

Input Handling (myKeyboard()):

Keyboard input is captured to trigger different actions:

- Pressing 'h' displays information.
- Pressing 's', 'r', 'a', 'w', or 'q' switches the display to respective seasonal views.
- Pressing '1', '2', '3', '4', or '5' triggers different sound effects.

Timer Functions:

Timer functions are used to trigger specific actions at predefined intervals, such as cloud movement, rain, flower growth, leaf fall, and snowfall.

Overall, the program integrates OpenGL rendering with keyboard input handling, timers, and callback functions to create an interactive and dynamic simulation of a railway station environment with seasonal changes and sound effects.

Display Functions for Seasonal Views:

Functions like summer(), rainy(), autumn(), winter(), and spring() are called when the corresponding keys are pressed. These functions likely contain OpenGL rendering commands to display scenes representing each season.

Summer Season

The summer() function is responsible for rendering the scene during the summer season. Here's a breakdown of its functionality:

- **Clearing the Background**

Sets the background color to a light shade and clears the color buffer to prepare for rendering the scene.

- **Rendering Scene Elements**

Calls various functions to render different elements of the scene, including;

a. sea(), beach(), and sky(): Render the sea, beach, and sky specific to the summer season.
b. sun(): Render the sun.
c. sun_shadow(): Render the shadow of the sun.
d. cloud1(), cloud2(), cloud3(): Render clouds in the sky.
e. busStand(), road(), human_busStop(), station_grass(), road2(): Render elements related to the bus stand and roads.
f. bus1() and bus2(): Render buses.
g. boat1() and boat2(): Render boats.
h. hills(): Render hills in the background.
i. station(), rail_Line(), platform(): Render elements related to the railway station.
j. human1(), human2(), human_platform(): Render human figures.
k. tree1(), tree_leaf1(), tree_leaf2(), tree2(): Render trees and their leaves.
l. fruit() and fruit2(): Render fruits on trees.
m. train1() and train2(): Render trains.
n. Incrementing Variable: Increments the variable i by a fixed value (0.2f).
o. Rendering: Calls glFlush() to render the scene.

Rainy Season

The rainy() function is responsible for rendering the scene during the rainy season. Here's a breakdown of its functionality:

- Clearing the Background

Sets the background color to a light shade and clears the color buffer to prepare for rendering the scene.

- Rendering Scene Elements

Calls various functions to render different elements of the scene, including:

a. sea2(), beach(), and sky2(): Render the sea, beach, and sky specific to the rainy season.

b. rainCloud(), cloud1rain(), cloud2rain(), cloud3rain(): Render rain clouds and regular clouds.

c. road(), busStand(), station_grass(), road2(): Render elements related to the road, bus stand, and station.

d. bus1() and bus2(): Render buses.

e. boat2(): Render boats.

f. hills(): Render hills in the background.

g. station(), rail_Line(), platform(): Render elements related to the railway station.

h. human1(), human2(), human_platform(): Render human figures.

i. tree1(), tree_leaf1(), tree_leaf2(), tree2(): Render trees and their leaves.

j. train1() and train2(): Render trains.

k. Animating Rain: Uses translation to animate raindrops (rain()) by adjusting their position based on the variable rain.

l. Animating Rain Clouds: Uses translation to animate rain clouds (rainCloud()) by adjusting their position based on the variable RcloudP.

m. Rendering: Calls glFlush() to render the scene.

Overall, the rainy() function creates an atmospheric rainy scene with falling raindrops, moving rain clouds, and other elements characteristic of the rainy season.

Autumn Season

The autumn() function is responsible for rendering the scene during the autumn season. Here's a breakdown of its functionality:

- Clearing the Background

Sets the background color to a light shade and clears the color buffer to prepare for rendering the scene.

- Rendering Scene Elements

Calls various functions to render different elements of the scene, including:

a. sea(), beach(), and sky(): Render the sea, beach, and sky.

b. sun1(): Render the sun.

c. cloud1(), cloud2(), and cloud3(): Render clouds in the sky.

d. busStand(), road(), human_busStop(), station_grass(), road2(): Render elements related to the bus stand and roads.

e. bus1() and bus2(): Render buses.

f. boat1() and boat2(): Render boats.

g. hills(): Render hills in the background.

h. station(), rail_Line(), platform(): Render elements related to the railway station.
i. human1(), human2(), human_platform(): Render human figures.
j. tree1(), tree_leaf1(), tree_leaf2(), tree2(): Render trees and their leaves.
k. train1() and train2(): Render trains.
l. Animating Flowers: Uses translation to animate flowers (kashPhul()) by adjusting their position based on the variable flowerP.
m. Incrementing Variable: Increments the variable i by a fixed value (0.2f).
n. Rendering: Calls glFlush() to render the scene.

Overall, the autumn() function creates a picturesque autumn scene with falling leaves and blooming flowers, capturing the essence of the autumn season.

Autumn Season

The winter() function is responsible for rendering the scene during the winter season. Here's a breakdown of its functionality:

- Clearing the Background:

Sets the background color to a light shade and clears the color buffer to prepare for rendering the scene.

- Rendering Scene Elements:

Calls various functions to render different elements of the scene, including:

a. sea(), beach(), and sky(): Render the sea, beach, and sky.
b. sun1(): Render the sun.
c. cloud1(), cloud2(), and cloud3(): Render clouds in the sky.
d. busStand(), road(), human_busStop(), station_grass(), road2(): Render elements related to the bus stand and roads.
e. bus1() and bus2(): Render buses.
f. boat1() and boat2(): Render boats.
g. hills(): Render hills in the background.
h. station(), rail_Line(), platform(): Render elements related to the railway station.
i. human1(), human2(), human_platform(): Render human figures.
j. tree1(), tree_leaf2(), tree2(): Render trees and their leaves.
k. train1() and train2(): Render trains.
l. Rendering Snowflakes: Uses translation to animate snowflakes (snowball()) by adjusting their position based on the variable snowP.

m. Rendering Falling Leaves: Uses translation to simulate falling leaves by adjusting the position of circular shapes (circle()) based on variables such as treeP1, treeP2, etc.

n. Incrementing Variable: Increments the variable i by a fixed value (0.2f).

o. Rendering: Calls glFlush() to render the scene.

Overall, the winter() function creates a serene winter scene with falling snowflakes and leaves, along with other elements such as trees, trains, and human figures, portraying the essence of the winter season.

Spring Season

The spring() function appears to be responsible for rendering the scene during the spring season. Here's a breakdown of what it does:

- Clearing the Background:

Sets the background color to a light shade and clears the color buffer to prepare for rendering the scene.

- Rendering Scene Elements:

Calls various functions to render different elements of the scene, including:

a. sea(), beach(), and sky(): Render the sea, beach, and sky.

b. sun1(): Render the sun.

c. cloud1(), cloud2(), and cloud3(): Render clouds in the sky.

d. busStand(), road(), human_busStop(), station_grass(), road2(): Render elements related to the bus stand and roads.

e. bus1() and bus2(): Render buses.

f. boat1() and boat2(): Render boats.

g. hills(): Render hills in the background.

h. station(), rail_Line(), platform(): Render elements related to the railway station.

i. human1(), human2(), human_platform(): Render human figures.

j. tree1(), tree_leaf1(), tree_leaf2(), tree2(): Render trees and their leaves.

k. train1() and train2(): Render trains.

l. tree_flower() and tree_flower2(): Render flowers on trees.

m. Animating Flowers: Uses translation to animate the flowers (flower()) by adjusting their position based on the variable flowerP1.

n. Incrementing Variable: Increments the variable i by a fixed value (0.2f).

o. Rendering: Calls glFlush() to render the scene.

Overall, the spring() function combines various OpenGL rendering commands and function calls to create a vibrant and dynamic scene representing the spring season, complete with natural elements, human figures, vehicles, and animated flowers.

There are also some other functions for, Rail tracks, Bus, Train, Railway Station Terminal, Bus stand, Rain, Cloud, trees, flowers etc.

In the next segment, a detailed explanation about the vehicles motion function is given, focusing on each command:

Boundary Checks and Position Resets

The first part of the function checks if any of the vehicles or objects have moved out of a specified boundary and resets their positions accordingly:

Cars

If position_car1x (the x-coordinate of car 1) is less than -1.5, reset it to 1.5. This means if car 1 moves too far left, it reappears on the right.

If position_car1y (the y-coordinate of car 1) is greater than 0.2, reset it to -0.1. This means if car 1 moves too far up, it reappears lower.

Buses

```
 if(position_bus1x < -1.5)
   position_bus1x = 1.5f;
if(position_bus2x > 1.5)
   position_bus2x = -1.5f;
```

Similar checks are applied to bus 1 and bus 2 for both x and y coordinates. If they move out of the bounds, they are reset to the opposite side.

Trains

```
 if(position_train1x < -1.5)
   position_train1x = 1.5f;
if(position_train2x > 1.5)
   position_train2x = -1.5f;
```

If train 1 moves too far left, it reappears on the right, and if train 2 moves too far right, it reappears on the left.

Clouds

```
 if(position_cloud1x < -1.5)
   position_cloud1x = 1.5f;
if(position_cloud2x > 1.5)
```

```
    position_cloud2x = -1.5f;
```

Similar logic for clouds 1, 2, and 3.

Boats

```
  if(position_boat1x < -1.5)
    position_boat1x = 1.5f;
  if(position_boat2x > 1.5)
    position_boat2x = -1.5f;
```

Similar logic for boats 1 and 2.

Updating Positions

The next part of the function updates the positions of the vehicles and objects based on their respective speeds:

- position_car1x -= speed;: Moves car 1 to the left by subtracting speed from its x position.
- position_bus1x -= speed;: Moves bus 1 to the left by subtracting speed.
- position_bus2x += speed;: Moves bus 2 to the right by adding speed.
- position_car1y += speed1;: Moves car 1 up by adding speed1 to its y position.
- position_bus1y += speed1;: Moves bus 1 up by adding speed1.
- position_bus2y -= speed1;: Moves bus 2 down by subtracting speed1.
- position_train1x -= speedTrain;: Moves train 1 to the left by subtracting speedTrain.
- position_train2x += speedTrain;: Moves train 2 to the right by adding speedTrain.
- position_cloud1x -= speed2;: Moves cloud 1 to the left by subtracting speed2.
- position_cloud2x += speed2;: Moves cloud 2 to the right by adding speed2.
- position_cloud3x += speed2;: Moves cloud 3 to the right by adding speed2.
- position_boat1x -= speedBoat;: Moves boat 1 to the left by subtracting speedBoat.
- position_boat2x += speedBoat;: Moves boat 2 to the right by adding speedBoat.

Redisplay and Timer

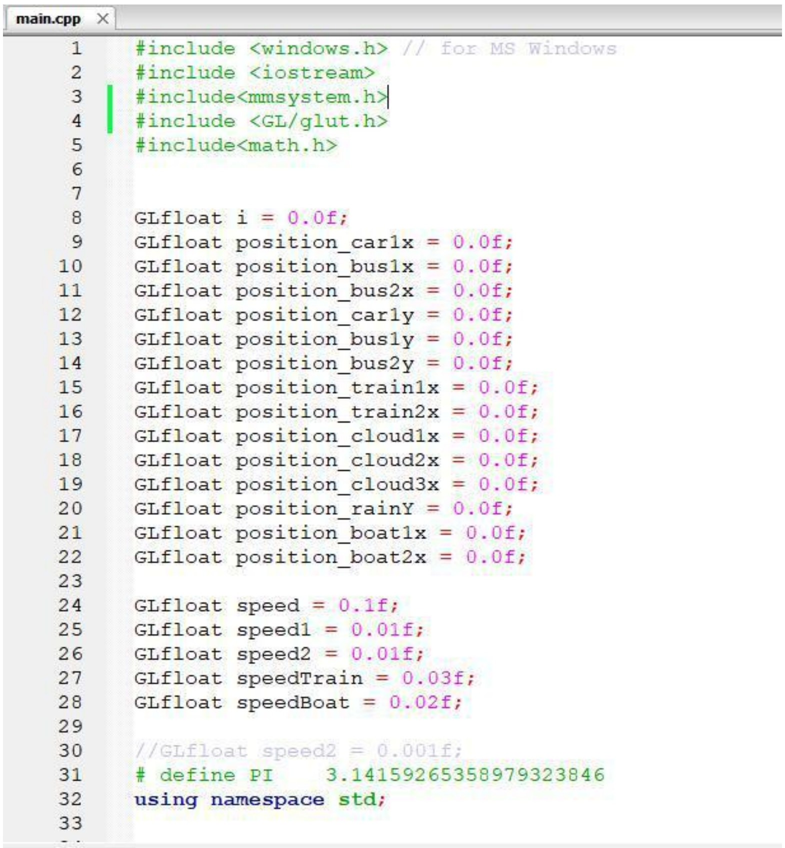
Finally, the function sets up the display to be redrawn and sets a timer for the next update:

- glutPostRedisplay();: Marks the current window as needing to be redisplayed. This triggers the display callback to update the visual representation.
- glutTimerFunc(100, motion_vehicle, 0);: Sets a timer to call motion_vehicle again after 100 milliseconds, creating a loop for continuous motion.

This function effectively creates an animation loop where various vehicles and objects move and reappear at the opposite edge when they go out of bounds, creating a seamless motion effect.

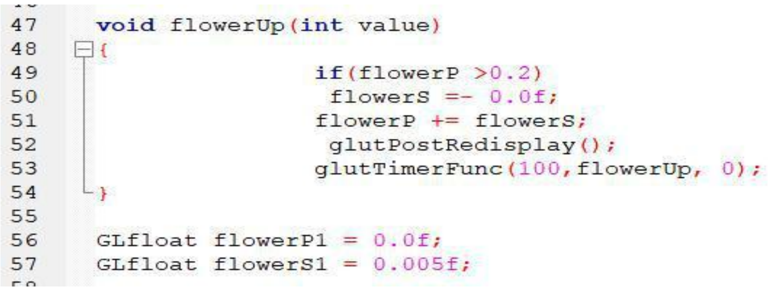
Code Segment

Here, the code segment starts , with initialization of variables;



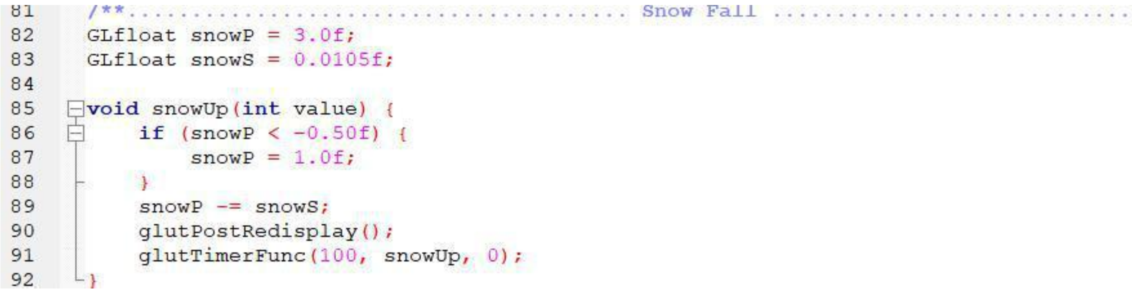
```
#include <windows.h> // for MS Windows
#include <iostream>
#include<mmsystem.h>
#include <GL/glut.h>
#include<math.h>


GLfloat i = 0.0f;
GLfloat position_car1x = 0.0f;
GLfloat position_bus1x = 0.0f;
GLfloat position_bus2x = 0.0f;
GLfloat position_car1y = 0.0f;
GLfloat position_bus1y = 0.0f;
GLfloat position_bus2y = 0.0f;
GLfloat position_train1x = 0.0f;
GLfloat position_train2x = 0.0f;
GLfloat position_cloud1x = 0.0f;
GLfloat position_cloud2x = 0.0f;
GLfloat position_cloud3x = 0.0f;
GLfloat position_rainY = 0.0f;
GLfloat position_boat1x = 0.0f;
GLfloat position_boat2x = 0.0f;

GLfloat speed = 0.1f;
GLfloat speed1 = 0.01f;
GLfloat speed2 = 0.01f;
GLfloat speedTrain = 0.03f;
GLfloat speedBoat = 0.02f;

//GLfloat speed2 = 0.001f;
# define PI    3.14159265358979323846
using namespace std;

```

For flower



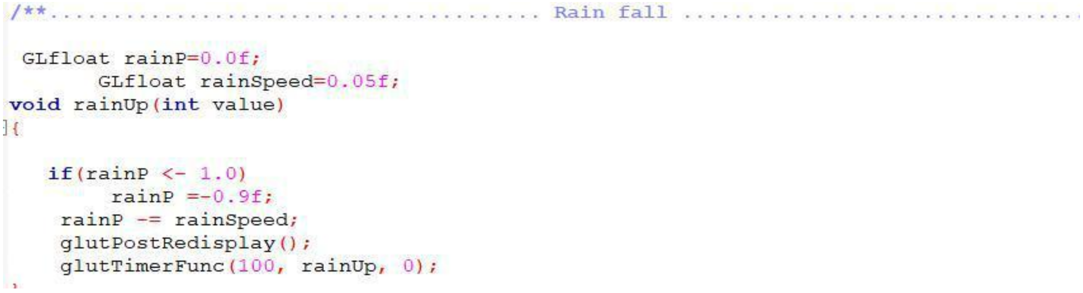
```
void flowerUp(int value)
{
                if(flowerP >0.2)
                 flowerS =- 0.0f;
                flowerP += flowerS;
                 glutPostRedisplay();
                glutTimerFunc(100,flowerUp, 0);
}

GLfloat flowerP1 = 0.0f;
GLfloat flowerS1 = 0.005f;
```

For Snow Fall



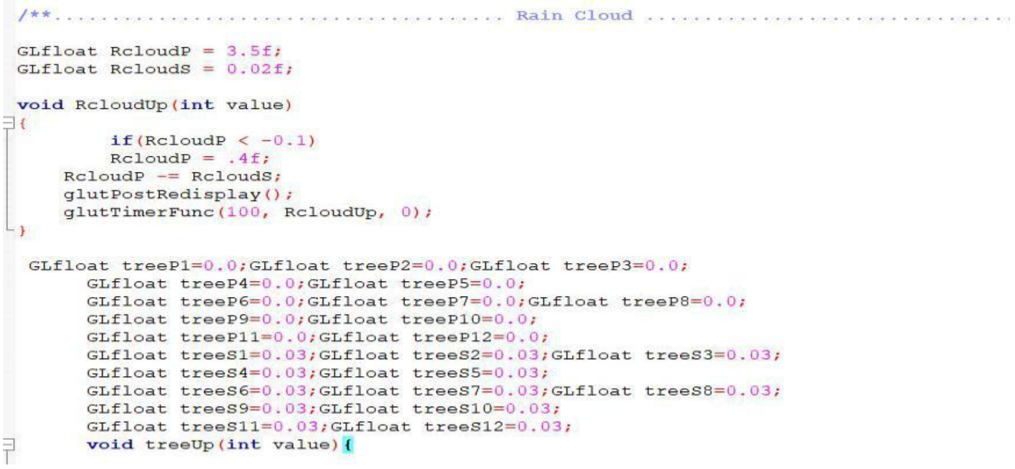
```
/**.................................... Snow Fall ............................
GLfloat snowP = 3.0f;
GLfloat snowS = 0.0105f;

void snowUp(int value) {
    if (snowP < -0.50f) {
        snowP = 1.0f;
    }
    snowP -= snowS;
    glutPostRedisplay();
    glutTimerFunc(100, snowUp, 0);
}
```

For Rain Fall



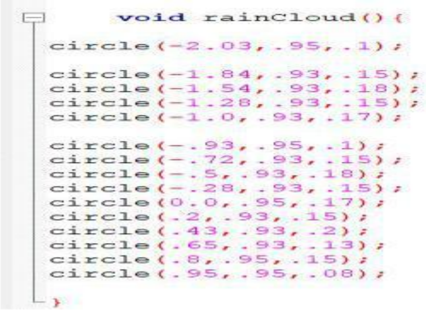
```
/**..................................... Rain fall ................................

 GLfloat rainP=0.0f;
       GLfloat rainSpeed=0.05f;
void rainUp(int value)
{

   if(rainP <- 1.0)
        rainP =-0.9f;
    rainP -= rainSpeed;
    glutPostRedisplay();
    glutTimerFunc(100, rainUp, 0);
```

For Rain Clouds



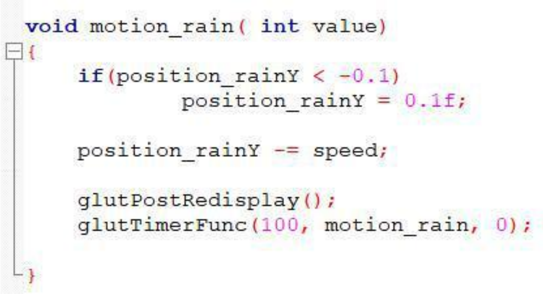
```
/**..................................... Rain Cloud ................................

GLfloat RcloudP = 3.5f;
GLfloat RcloudS = 0.02f;

void RcloudUp(int value)
{
        if(RcloudP < -0.1)
        RcloudP = .4f;
    RcloudP -= RcloudS;
    glutPostRedisplay();
    glutTimerFunc(100, RcloudUp, 0);
}

 GLfloat treeP1=0.0;GLfloat treeP2=0.0;GLfloat treeP3=0.0;
      GLfloat treeP4=0.0;GLfloat treeP5=0.0;
      GLfloat treeP6=0.0;GLfloat treeP7=0.0;GLfloat treeP8=0.0;
      GLfloat treeP9=0.0;GLfloat treeP10=0.0;
      GLfloat treeP11=0.0;GLfloat treeP12=0.0;
      GLfloat treeS1=0.03;GLfloat treeS2=0.03;GLfloat treeS3=0.03;
      GLfloat treeS4=0.03;GLfloat treeS5=0.03;
      GLfloat treeS6=0.03;GLfloat treeS7=0.03;GLfloat treeS8=0.03;
      GLfloat treeS9=0.03;GLfloat treeS10=0.03;
      GLfloat treeS11=0.03;GLfloat treeS12=0.03;
      void treeUp(int value){
```

For Size of Rain Clouds



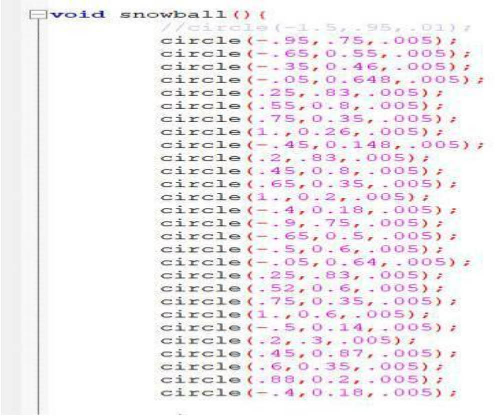
```
        void rainCloud(){
circle(-2.03,.95,.1);

circle(-1.84,.93,.15);
circle(-1.54,.93,.18);
circle(-1.28,.93,.15);
circle(-1.0,.93,.17);

circle(-.93,.95,.1);
circle(-.72,.93,.15);
circle(-.5,.93,.18);
circle(-.28,.93,.15);
circle(0.0,.95,.17);
circle(.2,.93,.15);
circle(.43,.93,.2);
circle(.65,.93,.13);
circle(.8,.95,.15);
circle(.95,.95,.08);

}
```

For Motion of The Rain Clouds



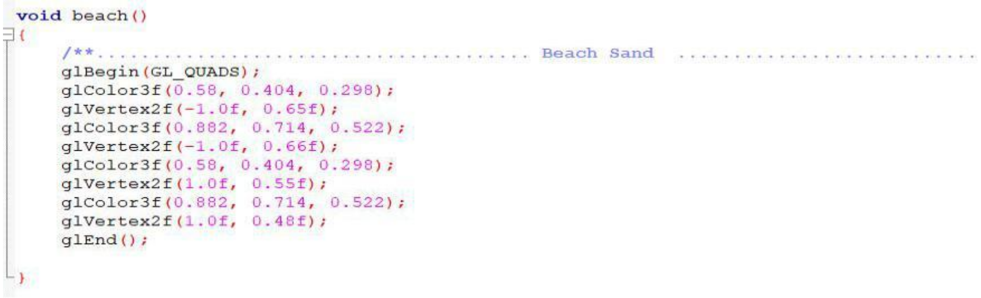
```
void motion_rain( int value)
{
    if(position_rainY < -0.1)
            position_rainY = 0.1f;

    position_rainY -= speed;

    glutPostRedisplay();
    glutTimerFunc(100, motion_rain, 0);

}
```

For Snow Ball Sizes



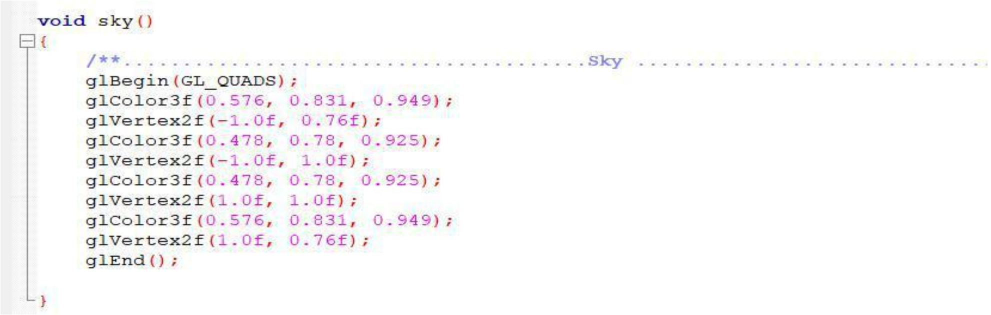
```
void snowball(){
            //circle(-1.5,.95,.01);
            circle(-.95,.75,.005);
            circle(-.65,0.55,.005);
            circle(-.35,0.46,.005);
            circle(-.05,0.648,.005);
            circle(.25,.83,.005);
            circle(.55,0.8,.005);
            circle(.75,0.35,.005);
            circle(1.,0.26,.005);
            circle(-.45,0.148,.005);
            circle(.2,.83,.005);
            circle(.45,0.8,.005);
            circle(.65,0.35,.005);
            circle(1.,0.2,.005);
            circle(-.4,0.18,.005);
            circle(-.9,.75,.005);
            circle(-.65,0.5,.005);
            circle(-.5,0.6,.005);
            circle(-.05,0.64,.005);
            circle(.25,.83,.005);
            circle(.52,0.6,.005);
            circle(.75,0.35,.005);
            circle(1.,0.6,.005);
            circle(-.5,0.14,.005);
            circle(.2,.3,.005);
            circle(.45,0.87,.005);
            circle(.6,0.35,.005);
            circle(.88,0.2,.005);
            circle(-.4,0.18,.005);
```

For The Beach Part



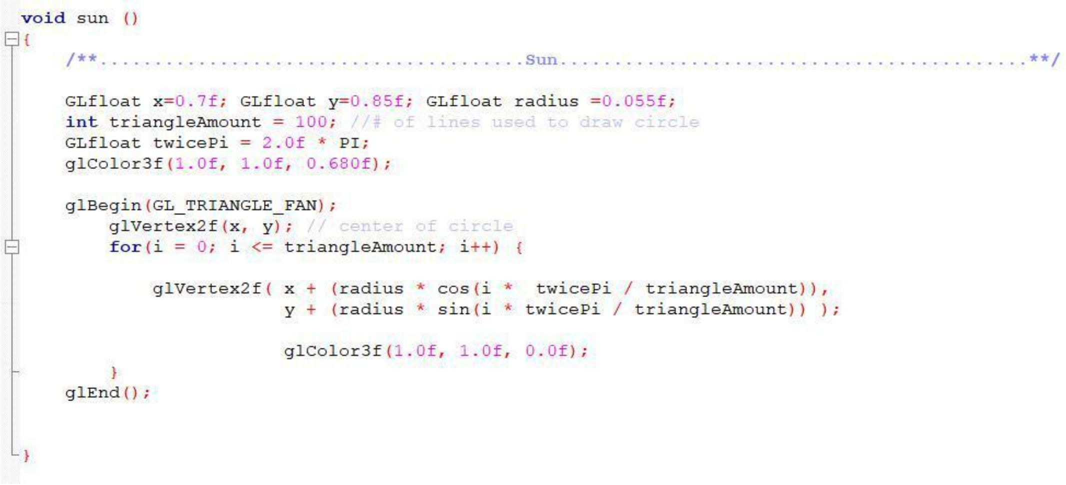
```
void beach()
{
    /**........................................ Beach Sand  ..........................
    glBegin(GL_QUADS);
    glColor3f(0.58, 0.404, 0.298);
    glVertex2f(-1.0f, 0.65f);
    glColor3f(0.882, 0.714, 0.522);
    glVertex2f(-1.0f, 0.66f);
    glColor3f(0.58, 0.404, 0.298);
    glVertex2f(1.0f, 0.55f);
    glColor3f(0.882, 0.714, 0.522);
    glVertex2f(1.0f, 0.48f);
    glEnd();

}
```

For The Clear Sky Part



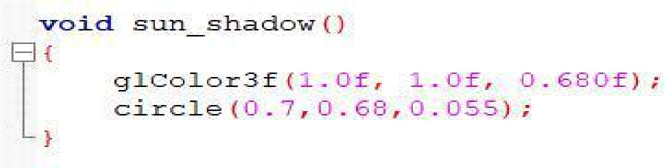
```
void sky()
{
    /**.......................................Sky ..............................
    glBegin(GL_QUADS);
    glColor3f(0.576, 0.831, 0.949);
    glVertex2f(-1.0f, 0.76f);
    glColor3f(0.478, 0.78, 0.925);
    glVertex2f(-1.0f, 1.0f);
    glColor3f(0.478, 0.78, 0.925);
    glVertex2f(1.0f, 1.0f);
    glColor3f(0.576, 0.831, 0.949);
    glVertex2f(1.0f, 0.76f);
    glEnd();

}
```

For The Sun Part



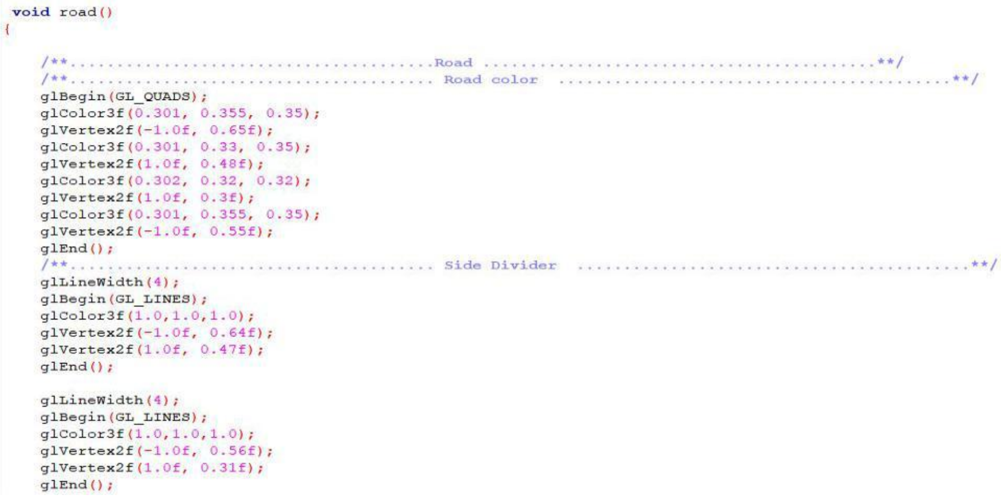
```
void sun ()
{
    /**....................................Sun.........................................**/

    GLfloat x=0.7f; GLfloat y=0.85f; GLfloat radius =0.055f;
    int triangleAmount = 100; //# of lines used to draw circle
    GLfloat twicePi = 2.0f * PI;
    glColor3f(1.0f, 1.0f, 0.680f);

    glBegin(GL_TRIANGLE_FAN);
        glVertex2f(x, y); // center of circle
        for(i = 0; i <= triangleAmount; i++) {

            glVertex2f( x + (radius * cos(i *  twicePi / triangleAmount)),
                        y + (radius * sin(i * twicePi / triangleAmount)) );

                        glColor3f(1.0f, 1.0f, 0.0f);

        }
    glEnd();


}
```

For The Sun Reflection On the river(Summer Season)



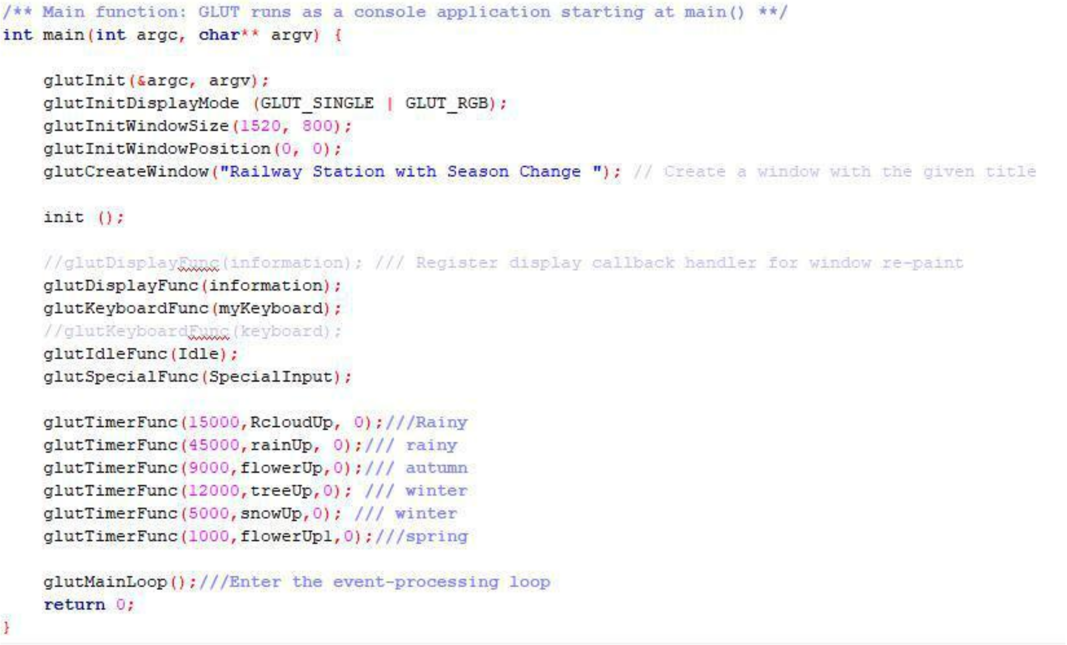
```
void sun_shadow()
{
    glColor3f(1.0f, 1.0f, 0.680f);
    circle(0.7,0.68,0.055);
}
```

For The Road



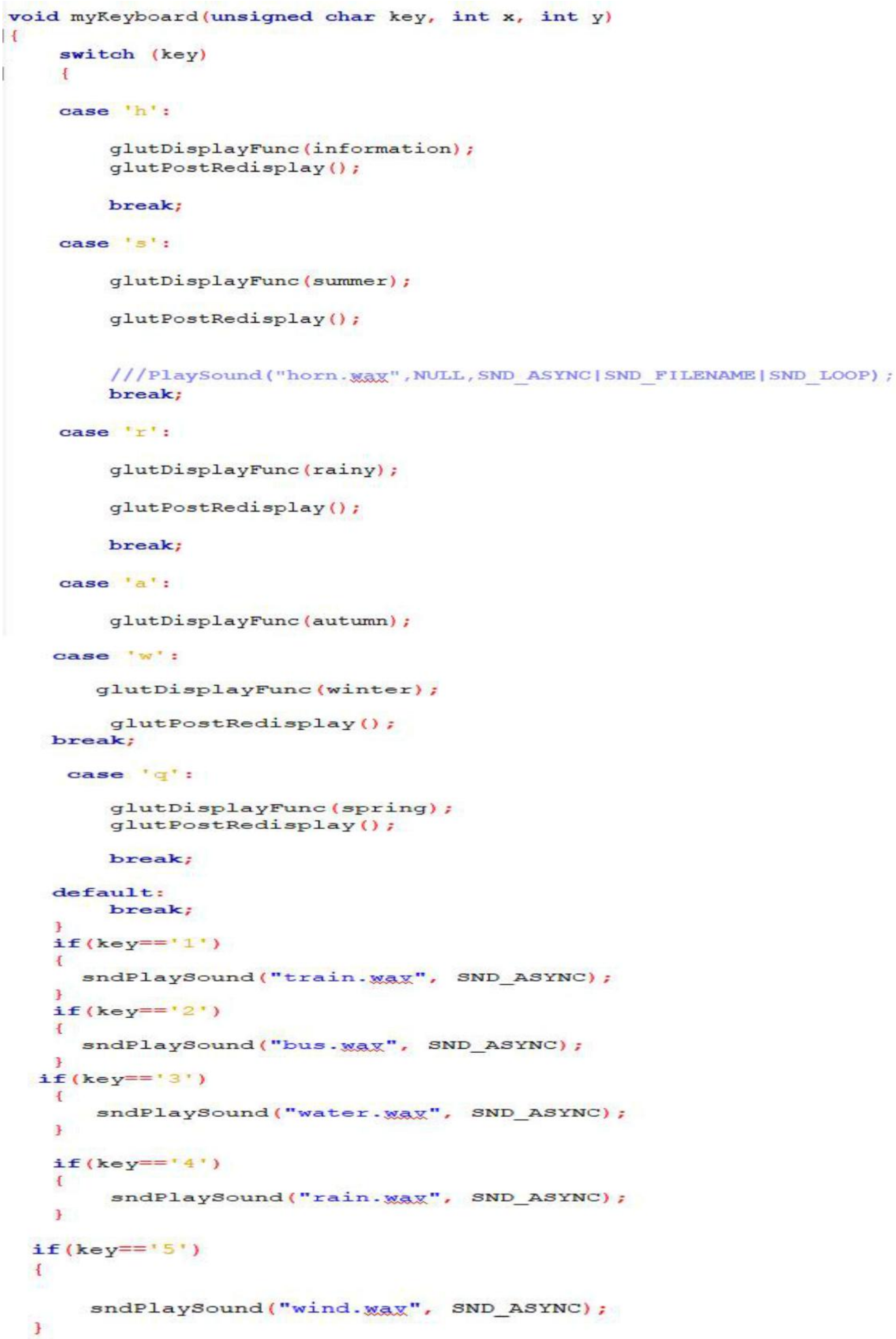
```
void road()
{

    /**......................................Road .........................................**/
    /**...................................... Road color  .........................................**/
    glBegin(GL_QUADS);
    glColor3f(0.301, 0.355, 0.35);
    glVertex2f(-1.0f, 0.65f);
    glColor3f(0.301, 0.33, 0.35);
    glVertex2f(1.0f, 0.48f);
    glColor3f(0.302, 0.32, 0.32);
    glVertex2f(1.0f, 0.3f);
    glColor3f(0.301, 0.355, 0.35);
    glVertex2f(-1.0f, 0.55f);
    glEnd();
    /**...................................... Side Divider  .........................................**/
    glLineWidth(4);
    glBegin(GL_LINES);
    glColor3f(1.0,1.0,1.0);
    glVertex2f(-1.0f, 0.64f);
    glVertex2f(1.0f, 0.47f);
    glEnd();

    glLineWidth(4);
    glBegin(GL_LINES);
    glColor3f(1.0,1.0,1.0);
    glVertex2f(-1.0f, 0.56f);
    glVertex2f(1.0f, 0.31f);
    glEnd();
```

The Int_Main() Function part where all Functions are called :



```
/** Main function: GLUT runs as a console application starting at main() **/
int main(int argc, char** argv) {

    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(1520, 800);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("Railway Station with Season Change "); // Create a window with the given title

    init ();

    //glutDisplayFunc(information); /// Register display callback handler for window re-paint
    glutDisplayFunc(information);
    glutKeyboardFunc(myKeyboard);
    //glutKeyboardFunc(keyboard);
    glutIdleFunc(Idle);
    glutSpecialFunc(SpecialInput);

    glutTimerFunc(15000,RcloudUp, 0);///Rainy
    glutTimerFunc(45000,rainUp, 0);/// rainy
    glutTimerFunc(9000,flowerUp,0);/// autumn
    glutTimerFunc(12000,treeUp,0); /// winter
    glutTimerFunc(5000,snowUp,0); /// winter
    glutTimerFunc(1000,flowerUp1,0);///spring

    glutMainLoop();///Enter the event-processing loop
    return 0;
}
```

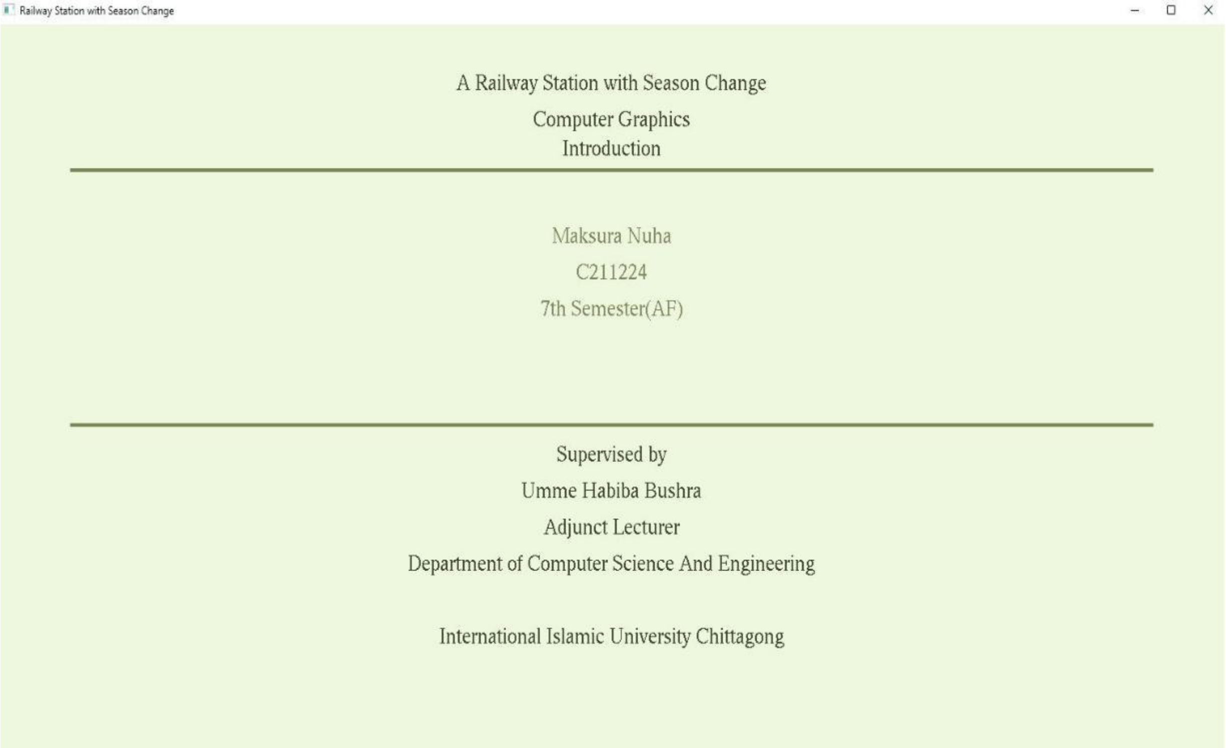
Key System for Commanding function Movements :



```
void myKeyboard(unsigned char key, int x, int y)
{
    switch (key)
    {

    case 'h':

        glutDisplayFunc(information);
        glutPostRedisplay();

        break;

    case 's':

        glutDisplayFunc(summer);

        glutPostRedisplay();


        ///PlaySound("horn.wav",NULL,SND_ASYNC|SND_FILENAME|SND_LOOP);
        break;

    case 'r':

        glutDisplayFunc(rainy);

        glutPostRedisplay();

        break;

    case 'a':

        glutDisplayFunc(autumn);
    case 'w':

       glutDisplayFunc(winter);

        glutPostRedisplay();
    break;

     case 'q':

        glutDisplayFunc(spring);
        glutPostRedisplay();

        break;

    default:
        break;
    }
    if(key=='1')
    {
      sndPlaySound("train.wav", SND_ASYNC);
    }
    if(key=='2')
    {
      sndPlaySound("bus.wav", SND_ASYNC);
    }
  if(key=='3')
    {
       sndPlaySound("water.wav", SND_ASYNC);
    }

    if(key=='4')
    {
        sndPlaySound("rain.wav", SND_ASYNC);
    }

  if(key=='5')
  {

      sndPlaySound("wind.wav", SND_ASYNC);
  }
```

Output(Snapshots)

Introduction Part



Railway Station with Season Change

A Railway Station with Season Change

Computer Graphics

Introduction

Maksura Nuha

C211224

7th Semester(AF)

Supervised by

Umme Habiba Bushra

Adjunct Lecturer

Department of Computer Science And Engineering

International Islamic University Chittagong

Fig 01 : Introductory part

Summer Season

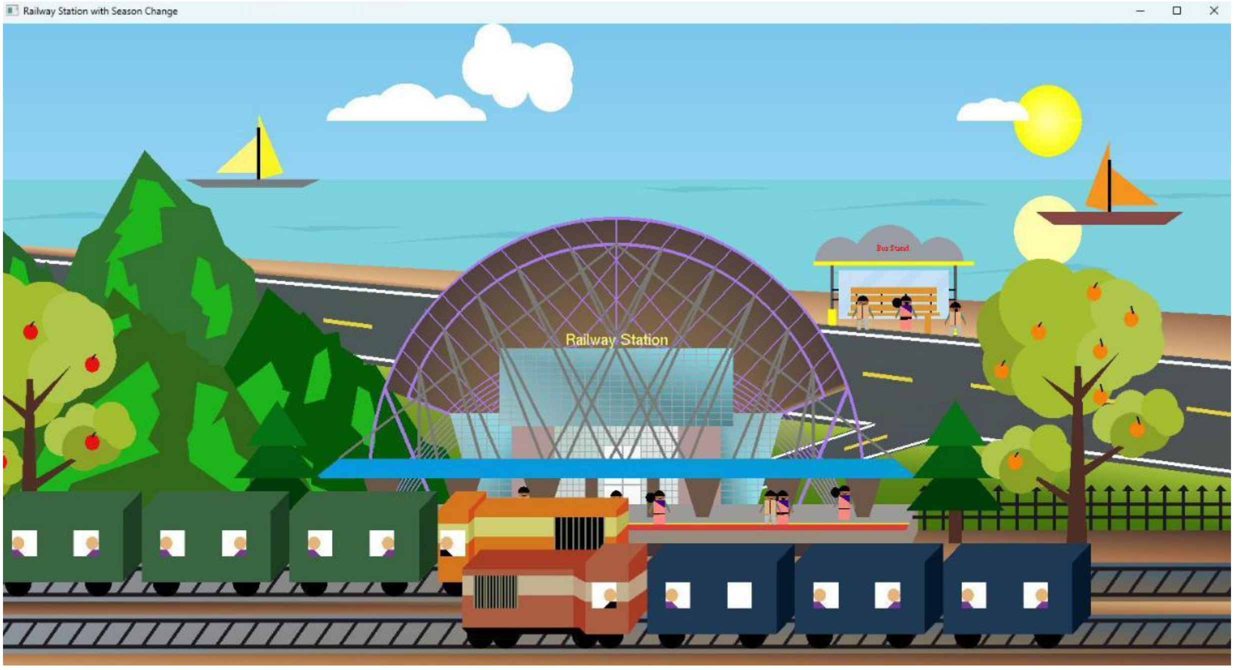


Fig 02: suns reflection with summer scenario

Winter Season

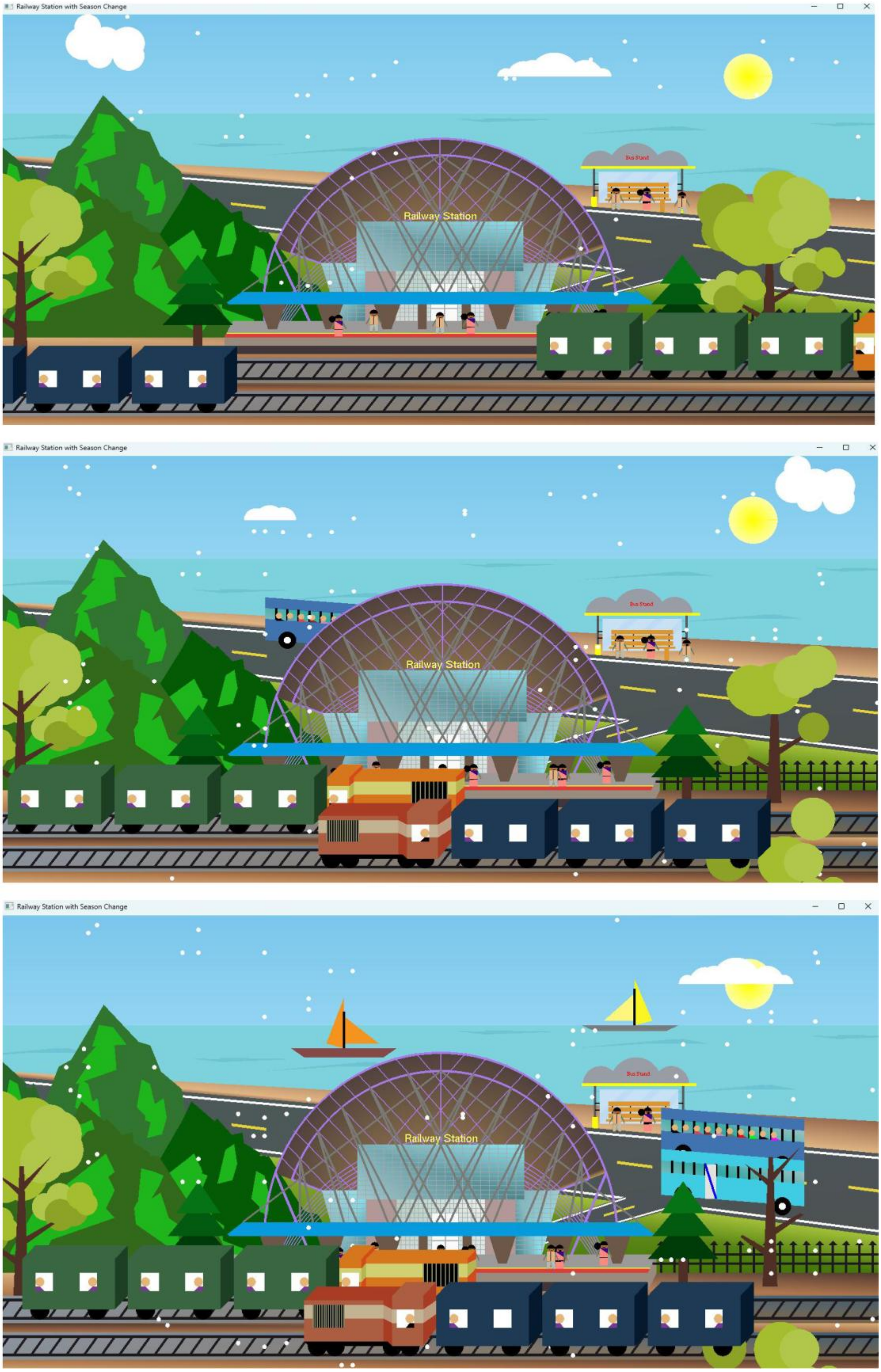

Railway Station with Season Change
Railway Station
Bus Stand
Railway Station with Season Change
Railway Station
Bus Stand
Railway Station with Season Change
Railway Station
Bus Stand

Fig 03: Snowfall, train movements, and falling tree leaves

Rainy Season

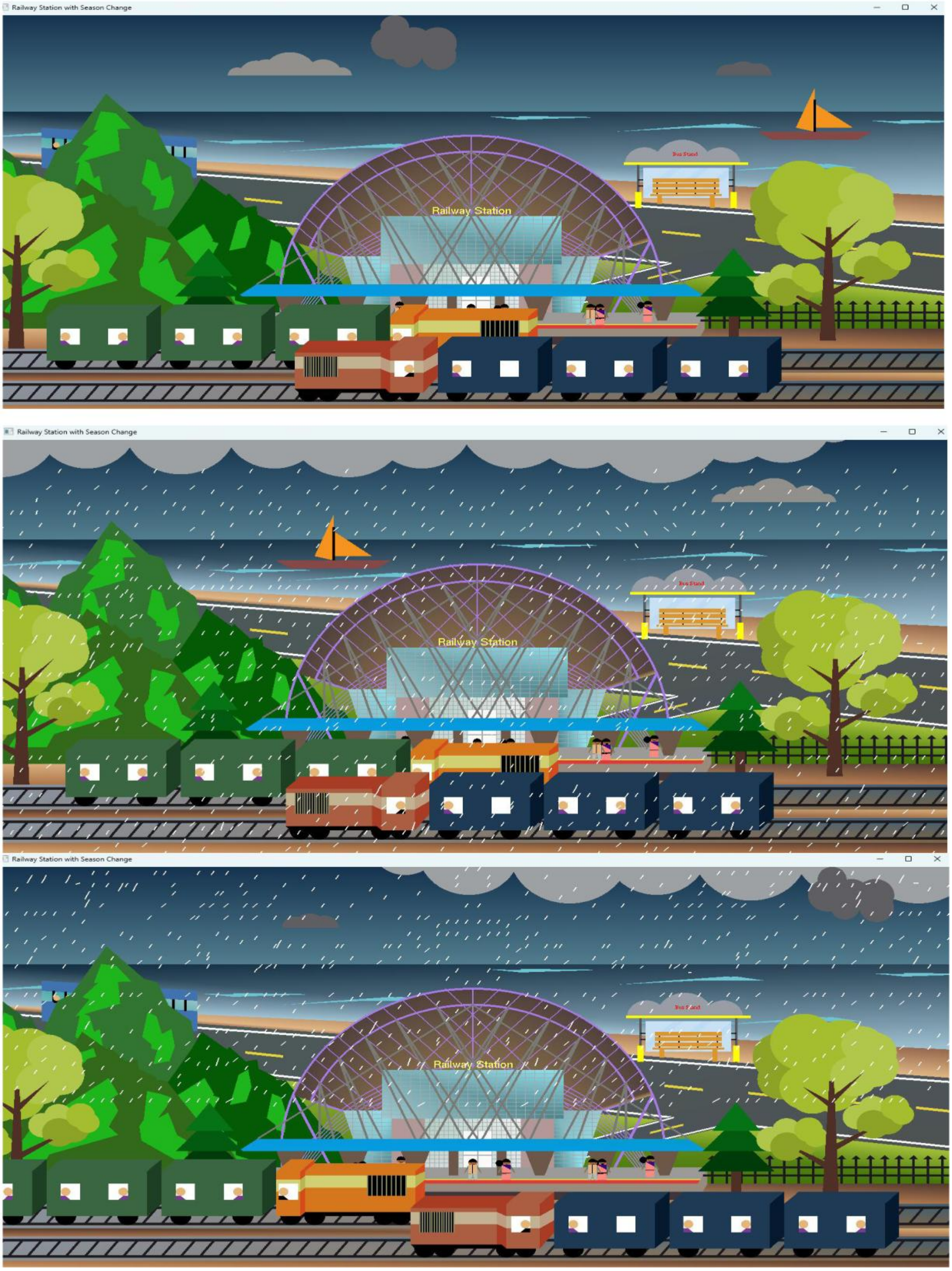

Railway Station with Season Change
Railway Station
Bus Stand
Railway Station with Season Change
Railway Station
Bus Stand
Railway Station with Season Change
Railway Station
Bus Stand

Fig 04: Rain drops, train movements, and a Cloudy scenario

Autumns Season

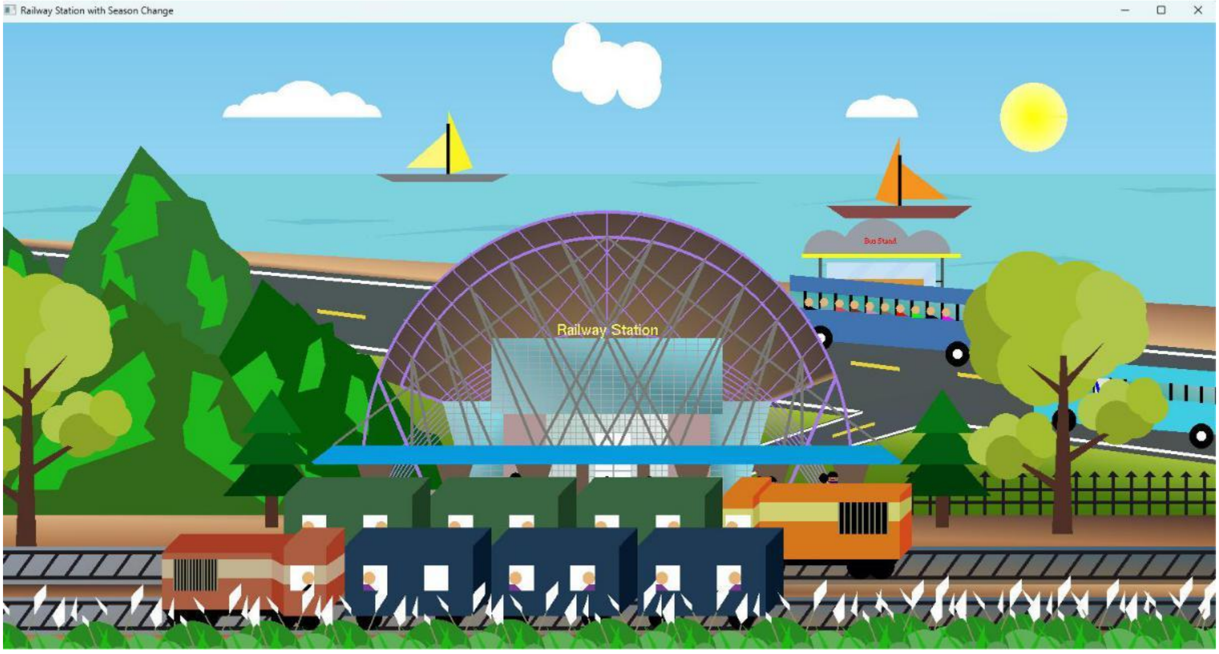


Fig 05: Autumn scene with Kans grass beside the rail track

Spring Season

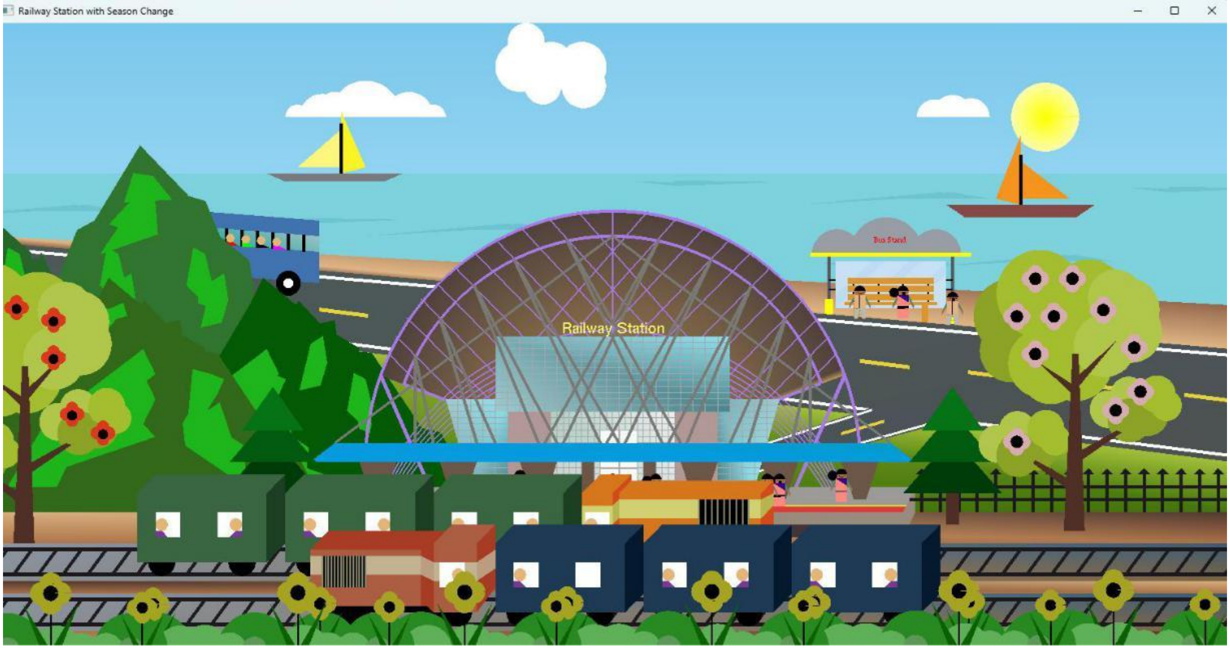


Fig 06: A Spring scenario with flowers everywhere

Learning Outcomes

Developing this project using C++ and OpenGL has been an enlightening experience, offering numerous benefits and learning opportunities

Animation and Graphics Coding: This project provided hands-on experience in coding animations and understanding the nuances of creating lifelike simulations.

Computer Science Skills: It enhanced my skills, problem-solving abilities, and understanding of computer graphics. The project fostered creativity in designing and implementing dynamic scenes and realistic interactions.

Technical Proficiency: Improved proficiency in using C++ and OpenGL, which are powerful tools in the field of graphics programming.

Overall, this project has been a pivotal milestone in my exploration of computer science, offering profound insights into graphics programming and the transformative capabilities of technology in building immersive environments. It has broadened my horizons within the field and has been an invaluable learning journey throughout my graphics course.

Conclusion

In conclusion, "A Railway Station with Season Change" project showcases the capabilities of computer graphics through C++ and OpenGL. It meticulously animates a bustling railway station, complete with trains, buses, and boats, against the backdrop of changing seasons. With realistic sound effects enhancing the experience, this project not only serves as a practical application of graphics concepts but also underscores the potential of technology to create captivating virtual environments.