Computer Graphics

(UCS505)

Terrain Side Scene

Submitted By:

Arpit Sagar	102003130
Manpreet Singh	102003171

Group No. 13

B.E. Third Year – 3COE 6

Submitted To:

Dr. Harpreet Singh

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

Patiala – 147001

April 2023

Table of Contents

1. Introduction

Computer graphics is an exciting and rapidly growing field that involves creating and manipulating visual content on a computer.

The Aim of this project is to implement a Terrain Side Scene consisting of a combination of animation of elements like translation of clouds, ship, planes, rotation of windmill and switching between scenes to demonstrate the change of day and night.

The project aims to show a graphical representation of various user defined functions.

The user is provided with several key controls like:

1. d ->To switch to day scene.
2. n ->To switch to night scene and display stars.
3. v->Erruption of volcano and sound of burst.
4. i-> Stop the moving ship
5. o->Start to move the ship
6. k->Stop the moving plane
7. l->Start the moving plane

Through visualization and simulating the components in a designated fashion, we have tried to implement the terrain side village scene while exploring the animation effects using OpenGL and GLUT library functions.

We can switch between day and night scenes , explode the volcano with sound effect and manually control the start and stop of ship,move the two boats in user defined directions and control their speed as well, name tagged plane movement and even manipulate the speed of clouds and windmill.

Overall, it was a great learning experience to dive into the knowledge of animation, frame buffers, rotation and translation effects in a scene.

2. Computer Graphics Concepts Used

Sr. No	Computer Graphics Concept	Description	Usage
1.	Coordinate Systems	A coordinate system is a reference system used to represent the positions of objects in a two-dimensional or three-dimensional space.	The glMatrixMode() function is used to define the coordinate system in many functions of the code.
2.	Shapes and Primitives	Basic geometric shapes such as points, lines, triangles, rectangles, and polygons.	Various functions such as glBegin, glEnd, and glVertex are used to draw shapes like rectangles, triangles, and lines in the code.
3.	Colors and Shading	Colors can be applied to shapes to give them a more visually appealing look. Shading is used to provide depth and realism to the shapes.	Functions like glColor3f and glBlendFunc are used in the code to set the color and shading of the shapes respectively.
4.	Transformations	Transformations involve modifying the position, orientation, and size of objects in a scene.	Used to move the cars and houses across the screen.
5.	User Interaction	This involves allowing the user to interact with the scene by responding to user inputs such as mouse clicks, key presses, or touch events.	In the given code, user interaction is implemented using the glutMouseFunc and glutKeyboardFunc functions to detect user key presses and act accordingly.

3. User Defined Functions

Our code defines several user-defined functions for the computer graphics project involving a terrain scene in OpenGL:

User Defined Functions	**Descriptions**
display()	The display() function is one of the most important functions in an OpenGL program. It is called by the graphics system whenever the window needs to be redrawn. The display() function is responsible for drawing the scene and updating the window.
update()	The update() function is designed to display the movement of objects in the scene. This user defined function includes 3 models: update for boat movement, update1 for planes and update2 for cloud movement.
move_right()	This user defined function is used for translation effect of objects such as clouds and windmill in a particular direction or orientation. This function can be called automatically called or by using user prompt to display the sync in animation.
void plane()	This function is a description of all the components required to draw a plane including base, window, flags and this function shows two planes when called from the main function.
void boat()	The prominent ship shown in the scene is defined in this boat() function in various components. This user defined function, when called in the main function displays the boat on the screen.

windmill() , cloud()	This scene showing the animation of clouds and movement of windmill is displayed using the definition of cloud and windmill structure in these two user defined functions.
Hill() , fire(), sound()	The implementation of volcano eruption shown in the scene comprising of hill, fire and the eruption sound is described using these functions. Using a variety of functions like GL_LINES, GL_POLYGON the design of all these object has been made.

4. Code

```
//-->FINAL EVAL PROJECT<--/
// ARPIT SAGAR(102003130)
// MANPREET SINGH (102003171)
// 3CO6

#include <windows.h>
#include <mmsystem.h>
#include <GL\glut.h>
#include <GL/glu.h>
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include<cmath>
//#include <FTGL/ftgl.h>

//FTGLPixmapFont font("arial.ttf");
//font.FaceSize(24);
//#define PI 3.1416
char name[35]="Arpit";
char name2[100]="Manpreet";
char tag[100]="--THE SHIP--";
GLint i, j, k, x = 0, y = 0, speed = 0, alt = 0, n1 = 1000, n2 = 1100, s1 = 0, s2 = 1, s3 = 1;
GLfloat sun_spin = 0, sun_x = 0, sun_y = 0, reduce = 10;
GLfloat ax = 0, bx = 0, cx = 0, dx = 0, str = 500.0, mn = 500.0;
GLfloat sr = 0.0, sg = 0.749, sb = 1.0;
GLfloat spin = 0.0;

bool condition = false;
GLfloat position = 0.0f;
GLfloat _move = 5.0f;
GLfloat position1 = 0.0f;
GLfloat _move1 = 3.0f;
GLfloat position2 = 900.0f;
GLfloat _move2 = 3.0f;

void init(void)
{
    glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    glClearColor(.40, .110, 1.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0, 1000.0, 0.0, 700.0);
}

float r(int a)
{

    return x = 4, y = 8;
}

float r(int a, int b)
{

    return x = 0, y = 0;
}



void updatey(int value)
{

    y = 8.0f;

    glutPostRedisplay();
    glutTimerFunc(100, updatey, 0);
}
```

```
void updatex(int value)
{

    x = 4.0f;

    glutPostRedisplay();
    glutTimerFunc(100, updatex, 0);
}
void brown_hill()
{
    glColor3f(0.50196, 0.25098, 0.0);

    glPushMatrix();
    glTranslatef(0, -220, 0);
    glScaled(0.8, 0.76, 0);
    glBegin(GL_POLYGON);

    glVertex3i(600, 600, 0);
    glVertex3i(800, 900, 0);
    glVertex3i(900, 650, 0);
    // glVertex3i(600, 600, 0);

    glColor3f(0.50196, 0.25098, 0.0);
    glBegin(GL_POLYGON);

    // glVertex3i(600, 600, 0);
    glVertex3i(900, 650, 0);
    glVertex3i(1000, 800, 0);
    glVertex3i(1100, 620, 0);

    glColor3f(0.50196, 0.25098, 0.0);
    glBegin(GL_POLYGON);

    glVertex3i(1050, 620, 0);
    glVertex3i(1200, 800, 0);
    glVertex3i(1200, 600, 0);
    // glVertex3i(1100, 620, 0);

    glPopMatrix();

    glEnd();
}

///=============================================================================================================///

///=================///
///* All_Model *///
///=================///

void sound()
{
    sndPlaySound("fire.wav", SND_ASYNC);
    // PlaySound(TEXT("fire.wav"), NULL, SND_ASYNC|SND_FILENAME);
}

///* Circle_Model***///

void circle(GLdouble rad)
{
    glBegin(GL_POLYGON);
    {
        for (int i = 0; i < 50; i++)
        {

            float pi = 3.1416;
            float A = (i * 2 * pi) / 50;
           // float A = 45;
            float r = rad;
            float x = r * cos(A);

            float y = r * sin(A);
```

```
            glVertex2f(x, y);
        }
    }
    glEnd();
}

void circle1(GLdouble rad)
{

    glBegin(GL_POLYGON);
    {
        for (int i = 0; i < 200; i++)
        {

            float pi = 3.1416;
            float A = (i * 2 * pi) / 200;
            float r = rad;
            float x = r * cos(A);

            float y = r + 10 * sin(A);
            glVertex2f(x, y);
        }
    }
    glEnd();
}

void fire()
{
    glBegin(GL_POLYGON);

    glVertex2f(13.5, 0.0);
    glVertex2f(8, 10);
    glVertex2f(15, 4);

    glVertex2f(17, 10);
    glVertex2f(19, 4);
    glVertex2f(26.0, 10);
    glVertex2f(20.5, 0.0);

    glEnd();
}

/// * Sun_Model **///
void Sun_Model()
{

    glPushMatrix();
    glTranslatef(600, 1100, 0);
    circle(33);
    glPopMatrix();
}

void update2(int value)
{

    //position2 -= 5;
    if (position2 < -1.3)
    {
        position2 -= _move2;
    }
    glutPostRedisplay(); // Notify GLUT that the display has changed

    glutTimerFunc(200, update2, 0); // Notify GLUT to call update again in 25 milliseconds
}

///* Cloud_Model***///
void cloud_model_one()
{

    glColor3f(1.25, 0.924, 0.930);
```

```
/// Top_Left

glPushMatrix();
glTranslatef(320, 210, 0);
circle(15);
glPopMatrix();

/// Top

glPushMatrix();
glTranslatef(340, 225, 0);
circle(16);
glPopMatrix();

/// Right

glPushMatrix();
glTranslatef(360, 210, 0);
circle(16);
glPopMatrix();

/// middle_Fill
glPushMatrix();
glTranslatef(355, 210, 0);
circle(16);
glPopMatrix();

glPushMatrix();
glTranslatef(350, 210, 0);
circle(16);
glPopMatrix();

glPushMatrix();
glTranslatef(345, 204, 0);
circle(10);
glPopMatrix();

glPushMatrix();
glTranslatef(340, 204, 0);
circle(10);
glPopMatrix();

glPushMatrix();
glTranslatef(335, 204, 0);
circle(10);
glPopMatrix();

glPushMatrix();
glTranslatef(330, 204, 0);
circle(10);
glPopMatrix();

glPushMatrix();
glTranslatef(325, 204, 0);
circle(10);
glPopMatrix();

glPushMatrix();
glTranslatef(320, 204, 0);
circle(10);
glPopMatrix();

glPushMatrix();
glTranslatef(315, 204, 0);
circle(10);
glPopMatrix();

glPushMatrix();
glTranslatef(310, 204, 0);
circle(10);
glPopMatrix();
```

```
    glPushMatrix();
    glTranslatef(305, 204, 0);
    circle(10);
    glPopMatrix();

    ///*Fill End*
}

void cloud_model_Two()
{
    glColor3f(1.25, 0.924, 0.930);

    /// Left_Part
    glPushMatrix();
    glTranslatef(305, 205, 0);
    circle(10);
    glPopMatrix();

    /// Top

    glPushMatrix();
    glTranslatef(320, 210, 0);
    circle(15);
    glPopMatrix();

    /// Right_Part
    glPushMatrix();
    glTranslatef(334, 207, 0);
    circle(10);
    glPopMatrix();

    /// Bottom_Part
    glPushMatrix();
    glTranslatef(320, 207, 0);
    circle(10);
    glPopMatrix();
}

void cloud_model_Three()
{
    glColor3f(1.25, 0.924, 0.930);

    /// Left_Part
    glPushMatrix();
    glTranslatef(300, 200, 0);
    circle(15);
    glPopMatrix();

    /// Top_Left

    glPushMatrix();
    glTranslatef(320, 210, 0);
    circle(15);
    glPopMatrix();

    /// Top
    glPushMatrix();
    glTranslatef(340, 220, 0);
    circle(16);
    glPopMatrix();

    /// Top_Right
    glPushMatrix();
    glTranslatef(360, 210, 0);
    circle(15);
    glPopMatrix();

    /// Right_Part
    glPushMatrix();
    glTranslatef(380, 200, 0);
```

```
    circle(15);
    glPopMatrix();

    /// Bottom_Right
    glPushMatrix();
    glTranslatef(360, 190, 0);
    circle(20);
    glPopMatrix();

    /// Bottom_Left
    glPushMatrix();
    glTranslatef(320, 190, 0);
    circle(20);
    glPopMatrix();

    /// Bottom
    glPushMatrix();
    glTranslatef(340, 190, 0);
    circle(20);
    glPopMatrix();

    ///*Fill End*
}
///* Hill_Model***///
void hill_big1()
{

    glColor3ub(231, 76, 60);
    glPushMatrix();
    glTranslatef(200, 150, 0);
    circle1(34);
    glPopMatrix();
    /*
     glColor3f(1, 0, 0);
        glPushMatrix();
        glTranslatef(63,187,0);
    glScaled(8.0,8.0,0);
        fire();
        glPopMatrix();*/
    /// Hill_BODY
    glBegin(GL_POLYGON);
    glColor3ub(151.0, 154.0, 154.0);
    glVertex2i(330, 70);

    glVertex2i(200, 170);
    glVertex2i(230, 190);
    glVertex2i(220, 180);
    glVertex2i(200, 190);
    glVertex2i(190, 180);
    glVertex2i(170, 190);
    glVertex2i(70, 70);
    glEnd();
}
void hill_big2()
{

    /// Hill
    glBegin(GL_POLYGON);
    glColor3f(0.38, 0.41, 0.36);
    glVertex2i(70, 70);
    glVertex2i(200, 225);
    glVertex2i(330, 70);

    glEnd();

    /// Hill_Snow
    glBegin(GL_POLYGON);
    glColor3f(1.25, 0.924, 0.930);

    glVertex2i(200, 225);
    glVertex2i(230, 190);
```

```
    glVertex2i(220, 180);
    glVertex2i(200, 190);
    glVertex2i(190, 180);
    glVertex2i(170, 190);

    glEnd();
}

void hill_small()
{
    /// Hill_Small
    glBegin(GL_POLYGON);
    glColor3f(0.11, 0.23, 0.36);
    glVertex2i(250, 100);
    glVertex2i(310, 175);
    glVertex2i(370, 100);

    glEnd();

    /// Hill_Small_Snow
    glBegin(GL_POLYGON);
    glColor3f(1.25, 0.924, 0.930);
    glVertex2i(290, 150);
    glVertex2i(310, 175);
    glVertex2i(330, 150);
    glVertex2i(325, 140);
    glVertex2i(310, 150);
    glVertex2i(300, 140);

    glEnd();
}
///* Tilla_Model *///
void Tilla_One_Model()
{
    /// Tilla
    glBegin(GL_POLYGON);
    glColor3ub(34.0, 153.0, 84.0);
    glVertex2i(125, 70);
    glVertex2i(150, 80);
    glVertex2i(160, 90);
    glVertex2i(170, 90);
    glVertex2i(180, 100);
    glVertex2i(190, 105);
    glVertex2i(200, 108);
    glVertex2i(300, 110);
    glVertex2i(300, 70);

    glEnd();
}

void Tilla_Two_Model()
{

    glColor3ub(34.0, 153.0, 84.0);
    /// Left_Part
    glPushMatrix();
    glTranslatef(430, 90, 0);
    circle(30);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(420, 87, 0);
    circle(30);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(410, 80, 0);
    circle(30);
    glPopMatrix();

    glPushMatrix();
```

```
    glTranslatef(400, 80, 0);
    circle(30);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(390, 70, 0);
    circle(30);
    glPopMatrix();

    /// Right_Part
    glPushMatrix();
    glTranslatef(445, 80, 0);
    circle(30);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(455, 75, 0);
    circle(30);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(465, 70, 0);
    circle(30);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(470, 65, 0);
    circle(30);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(480, 60, 0);
    circle(30);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(485, 55, 0);
    circle(20);
    glPopMatrix();
}
///* House_Model *///
void house()
{
    /// House_Roof
    glBegin(GL_POLYGON);
    glColor3ub(44.0, 62.0, 80.0);
    glVertex2i(285, 105);
    glVertex2i(285, 130);
    glVertex2i(380, 115);
    glVertex2i(380, 105);

    glEnd();

    /// House_Roof_Shadow
    glBegin(GL_POLYGON);
    glColor3f(0.0, 0.0, 0.0);
    glVertex2i(285, 105);
    glVertex2i(285, 120);
    glVertex2i(380, 105);
    glVertex2i(380, 105);

    glEnd();

    /// House_Fence
    glBegin(GL_POLYGON);
    glColor3f(255.0, 0.0, 0.0);
    glVertex2i(290, 70);
    glVertex2i(290, 104);
    glVertex2i(375, 104);
    glVertex2i(375, 70);
```

```
    glEnd();

    /// House_Fence_Shadow
    glBegin(GL_POLYGON);
    glColor3f(255.0, 0.0, 0.0);
    glVertex2i(310, 70);
    glVertex2i(350, 104);
    glVertex2i(375, 104);
    glVertex2i(375, 70);

    glEnd();

    /// House_Door
    glBegin(GL_POLYGON);
    glColor3f(0.38, 0.41, 0.36);
    glVertex2i(330, 70);
    glVertex2i(330, 100);
    glVertex2i(350, 100);
    glVertex2i(350, 70);

    glEnd();

    /// House_Window1
    glBegin(GL_POLYGON);
    glColor3f(0.38, 0.21, 0.26);
    glVertex2i(295, 75);
    glVertex2i(295, 90);
    glVertex2i(310, 90);
    glVertex2i(310, 75);

    glEnd();

    /// House_Window2
    glBegin(GL_POLYGON);
    glColor3f(0.38, 0.21, 0.26);
    glVertex2i(312, 75);
    glVertex2i(312, 90);
    glVertex2i(327, 90);
    glVertex2i(327, 75);

    glEnd();

    /// House_Window3
    glBegin(GL_POLYGON);
    glColor3f(0.38, 0.21, 0.26);
    glVertex2i(355, 75);
    glVertex2i(355, 90);
    glVertex2i(370, 90);
    glVertex2i(370, 75);

    glEnd();

    /// House_Small_Roof
    glBegin(GL_POLYGON);
    glColor3f(0.0, 0.0, 0.0);
    glVertex2i(250, 90);
    glVertex2i(257, 104);
    glVertex2i(290, 104);
    glVertex2i(290, 90);

    glEnd();

    /// House_Small_Fence
    glBegin(GL_POLYGON);
    glColor3ub(243.0, 156.0, 18.0);
    glVertex2i(255, 70);
    glVertex2i(255, 90);
    glVertex2i(290, 90);
    glVertex2i(290, 70);

    glEnd();
```

```
    /// House_Small_Door
    glBegin(GL_POLYGON);
    glColor3f(0.11, 0.23, 0.36);
    glVertex2i(260, 70);
    glVertex2i(260, 80);
    glVertex2i(285, 80);
    glVertex2i(285, 70);

    glEnd();
}

void house1()
{
    /// House_Roof
    glBegin(GL_POLYGON);
    glColor3ub(243.0, 156.0, 18.0);
    glVertex2i(285, 105);
    glVertex2i(285, 130);
    glVertex2i(380, 115);
    glVertex2i(380, 105);

    glEnd();

    /// House_Roof_Shadow
    glBegin(GL_POLYGON);
    glColor3ub(245.0, 176.0, 65.0);
    glVertex2i(285, 105);
    glVertex2i(285, 120);
    glVertex2i(380, 105);
    glVertex2i(380, 105);

    glEnd();

    /// House_Fence
    glBegin(GL_POLYGON);
    glColor3f(255.0, 0.0, 0.0);
    glVertex2i(290, 70);
    glVertex2i(290, 104);
    glVertex2i(375, 104);
    glVertex2i(375, 70);

    glEnd();

    /// House_Fence_Shadow
    glBegin(GL_POLYGON);
    glColor3f(255.0, 0.0, 0.0);
    glVertex2i(310, 70);
    glVertex2i(350, 104);
    glVertex2i(375, 104);
    glVertex2i(375, 70);

    glEnd();

    /// House_Door
    glBegin(GL_POLYGON);
    glColor3ub(19, 141.0, 117.0);
    glVertex2i(330, 70);
    glVertex2i(330, 100);
    glVertex2i(350, 100);
    glVertex2i(350, 70);

    glEnd();

    /// House_Window1
    glBegin(GL_POLYGON);
    glColor3f(0.38, 0.21, 0.26);
    glVertex2i(295, 75);
    glVertex2i(295, 90);
    glVertex2i(310, 90);
    glVertex2i(310, 75);
```

```
  glEnd();

  /// House_Window2
  glBegin(GL_POLYGON);
  glColor3f(0.38, 0.21, 0.26);
  glVertex2i(312, 75);
  glVertex2i(312, 90);
  glVertex2i(327, 90);
  glVertex2i(327, 75);

  glEnd();

  /// House_Window3
  glBegin(GL_POLYGON);
  glColor3f(0.38, 0.21, 0.26);
  glVertex2i(355, 75);
  glVertex2i(355, 90);
  glVertex2i(370, 90);
  glVertex2i(370, 75);

  glEnd();

  /*   ///House_Small_Roof
     glBegin(GL_POLYGON);
     glColor3f(1.0, 0.0, 0.0);
     glVertex2i(250, 90);
     glVertex2i(257, 104);
     glVertex2i(290, 104);
     glVertex2i(290, 90);

     glEnd();
*/
  /// House_Small_Fence
  glBegin(GL_POLYGON);
  glColor3ub(160.0, 64.0, 0.0);
  glVertex2i(290, 70);
  glVertex2i(290, 140);
  glVertex2i(270, 140);

  glVertex2i(270, 70);

  glEnd();

  /// House_somewall side
  glBegin(GL_POLYGON);
  glColor3ub(160.0, 64.0, 0.0);
  glVertex2i(265, 90);
  glVertex2i(265, 160);
  glVertex2i(270, 140);
  glVertex2i(270, 70);

  glEnd();

  /// House_somewall top
  /// House_Small_Roof
  glBegin(GL_POLYGON);
  glColor3f(0.0, 0.0, 0.0);
  glVertex2i(265, 160);
  glVertex2i(270, 140);
  glVertex2i(290, 140);
  glVertex2i(285, 160);
  glEnd();

  ///*House_Small_Door
     glBegin(GL_POLYGON);
     glColor3f(0.11, 0.23, 0.36);
     glVertex2i(260, 70);
     glVertex2i(260, 80);
     glVertex2i(285, 80);
     glVertex2i(285, 70);
```

```
      glEnd();

}

void house2()
{

    glPushMatrix();
    glTranslatef(690, 250, 0);
    glScalef(0.18, 0.35, 0);

    glBegin(GL_QUADS);                // Each set of 4 vertices form a quad
    glColor3ub(0.0f, 128.0f, 128.0f); // Red
    glVertex2f(150.0f, 30.0f);        // x, y
    glVertex2f(150.0f, 200.0f);       // x, y
    glVertex2f(450.0f, 200.0f);
    glVertex2f(450.0f, 30.0f);
    glEnd();

    glBegin(GL_POLYGON);              // Each set of 4 vertices form a quad
    glColor3ub(165.0f, 42.0f, 42.0f); // Red
    glVertex2f(100.0f, 200.0f);       // x, y
    glVertex2f(300.0f, 300.0f);       // x, y
    glVertex2f(500.0f, 200.0f);

    glEnd();

    glBegin(GL_QUADS);              // Each set of 4 vertices form a quad
    glColor3ub(128.0f, 0.0f, 0.0f); // Red
    glVertex2f(250.0f, 30.0f);      // x, y
    glVertex2f(250.0f, 120.0f);     // x, y
    glVertex2f(320.0f, 120.0f);
    glVertex2f(320.0f, 30.0f);
    glEnd();

    glBegin(GL_QUADS);              // Each set of 4 vertices form a quad
    glColor3ub(128.0f, 0.0f, 0.0f); // Red
    glVertex2f(350.0f, 120.0f);     // x, y
    glVertex2f(350.0f, 145.0f);     // x, y
    glVertex2f(395.0f, 145.0f);
    glVertex2f(395.0f, 120.0f);
    glEnd();

    glBegin(GL_QUADS);              // Each set of 4 vertices form a quad
    glColor3ub(128.0f, 0.0f, 0.0f); // Red
    glVertex2f(175.0f, 120.0f);     // x, y
    glVertex2f(175.0f, 145.0f);     // x, y
    glVertex2f(220.0f, 145.0f);
    glVertex2f(220.0f, 120.0f);
    glEnd();

    glBegin(GL_QUADS);                // Each set of 4 vertices form a quad
    glColor3ub(185.0f, 119.0f, 14.0f); // Red
    glVertex2f(450.0f, 30.0f);        // x, y
    glVertex2f(450.0f, 150.0f);       // x, y
    glVertex2f(800.0f, 150.0f);
    glVertex2f(800.0f, 30.0f);
    glEnd();

    glBegin(GL_QUADS);              // Each set of 4 vertices form a quad
    glColor3ub(128.0f, 0.0f, 0.0f); // Red
    glVertex2f(500.0f, 80.0f);      // x, y
    glVertex2f(500.0f, 110.0f);     // x, y
    glVertex2f(570.0f, 110.0f);
    glVertex2f(570.0f, 80.0f);
    glEnd();

    glBegin(GL_QUADS);              // Each set of 4 vertices form a quad
    glColor3ub(128.0f, 0.0f, 0.0f); // Red
    glVertex2f(600.0f, 80.0f);      // x, y
```

```
    glVertex2f(600.0f, 110.0f);     // x, y
    glVertex2f(670.0f, 110.0f);
    glVertex2f(670.0f, 80.0f);
    glEnd();

    glBegin(GL_QUADS);              // Each set of 4 vertices form a quad
    glColor3ub(128.0f, 0.0f, 0.0f); // Red
    glVertex2f(700.0f, 80.0f);      // x, y
    glVertex2f(700.0f, 110.0f);     // x, y
    glVertex2f(770.0f, 110.0f);
    glVertex2f(770.0f, 80.0f);
    glEnd();
    glPopMatrix();
}

///* Field_Model *///
void field()
{
    /// Field

    glBegin(GL_POLYGON);
    glColor3ub(90, 153, 51);
    glVertex2i(0, 250);
    glVertex2i(0, 270);
    glVertex2i(1000, 270);
    glVertex2i(1000, 250);

    glEnd();

    /// Field_Shadow
    glBegin(GL_POLYGON);
    glColor3ub(0, 153, 51);
    glVertex2i(0, 230);
    glVertex2i(0, 250);
    glVertex2i(1000, 250);
    glVertex2i(1000, 200);

    glEnd();

    /// river0.7/0.4/0.4

    glBegin(GL_POLYGON);
    glColor3ub(0, 143, 179);
    glVertex2i(0, 0);
    glVertex2i(0, 230);
    glVertex2i(600, 225);
    glVertex2i(600, 0);

    glEnd();

    /*/// river curve
        glPushMatrix();
        //glPushMatrix();
        float w=0;
        glTranslated(w,0,0.0);
        glColor3f(0.1,0.5,1.0);
        glBegin(GL_POLYGON);
        glVertex2f(0,0);
        glVertex2f(0,100);
        glVertex2f(10,96);
        glVertex2f(25,98);
        glVertex2f(39,94);
        glVertex2f(50,92);
        glVertex2f(70,98);
        glVertex2f(85,95);
        glVertex2f(95,96);
        glVertex2f(110,99);
        glVertex2f(128,97);
        glVertex2f(139,95);
        glVertex2f(145,97);
        glVertex2f(155,99);
```

```
        glVertex2f(172,95);
        glVertex2f(195,96);
        glVertex2f(212,95);
        glVertex2f(254,92);
        glVertex2f(284,96);
        glVertex2f(344,98);
        glVertex2f(360,93);
        glVertex2f(390,94);
        glVertex2f(410,99);
        glVertex2f(450,94);
        glVertex2f(485,100);
        glVertex2f(502,92);
        glVertex2f(552,92);
        glVertex2f(592,96);
        glVertex2f(630,105);
        glVertex2f(680,93);
        glVertex2f(720,97);
        glVertex2f(750,93);
        glVertex2f(800,95);
        glVertex2f(850,97);
        glVertex2f(880,108);
        glVertex2f(900,96);
        glVertex2f(920,93);
        glVertex2f(950,99);
        glVertex2f(980,92);
        glVertex2f(1000,99);
        glVertex2f(1000,0);
        glVertex2f(1600,10);
        glVertex2f(3000,280);
        glEnd();
        glPopMatrix();*/

    glBegin(GL_POLYGON);
    glColor3ub(0, 143, 179);
    glVertex2i(600, 0);
    glVertex2i(600, 225);
    glVertex2i(1000, 245);
    glVertex2i(1000, 0);

    glEnd();

    /// Field_2
    glBegin(GL_POLYGON);
    glColor3ub(0, 153, 51);
    glVertex2i(0, 00);
    glVertex2i(0, 80);
    glVertex2i(600, 50);
    glVertex2i(640, 0);

    glEnd();

    /// Field_Shadow2
    glBegin(GL_POLYGON);
    glColor3ub(0, 153, 51);
    glVertex2i(0, 00);
    glVertex2i(0, 50);
    glVertex2i(600, 50);
    glVertex2i(600, 0);

    glEnd();

    /// Field_Shadow3
    glBegin(GL_POLYGON);
    glColor3ub(0, 153, 51);
    glVertex2i(600, 0);
    glVertex2i(600, 50);
    glVertex2i(1000, 30);
    glVertex2i(1000, 0);

    glEnd();
}
```

```
void Drawarc(float sa,float ea,float cx,float cy,float rd)
{
     float PI = 3.14;
     float step=1.0;
     float angle,x=0,y=0,centerX=cx,centerY=cy,radius=rd;

     glBegin(GL_POLYGON);
     for(angle=sa;angle<ea; angle+=step)
     {
     float rad ;
     rad = PI*angle/180;
     x  = centerX+radius*cos(rad);
     y  = centerY+radius*sin(rad);
     glVertex2f(x,y);
     }
     glEnd();
     glFlush();
}
void cloud(int m, int n)
{
    float c,p,col=0;
    for(c=p=0;c<31;c+=10,p-=1)
     {
     glColor3f(0.5,0.5,0.5);
     Drawarc(0,360,m+c,n,10+p);
     }
}


/// Boat/////////////*******
void Boat()
{
    glPushMatrix();
    glTranslatef(position, 0.0f, 0.0f);
    glTranslatef(-70, 40.0f, 0.0f);
    //glEnable(GL_BLEND);
    //glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    float c,p,col=0;
    //base
        glColor4f(0.2+col,0.2+col,0.2+col,0.5);
        glBegin(GL_POLYGON);
        glVertex2f(10,119);
        glVertex2f(10,110);
        glVertex2f(41,70);
        glColor4f(0.3+col,0.3+col,0.8+col,0.5);
        glVertex2f(219,42);
        glVertex2f(292,98);
        glVertex2f(300,110);
        glEnd();



    //p1
        glColor3f(1.0+col,1.0+col,1.0+col);
        glBegin(GL_POLYGON);
        glVertex2f(35,118);
        glVertex2f(35,128);
        glColor3f(0.5+col,0.5+col,0.5+col);
        glVertex2f(239,131);
        glVertex2f(239,111);
        glVertex2f(35,119);
        glEnd();
        //side
        glBegin(GL_POLYGON);
        glColor3f(0.8+col,0.8+col,0.8+col);
        glVertex2f(239,131);
        glVertex2f(239,111);
        glVertex2f(257,110);
        glVertex2f(257,127);
        glEnd();
```

```
    //p2
        glColor3f(0.0+col,0.0+col,0.5+col);
        glBegin(GL_POLYGON);
        glVertex2f(45,129);
        glVertex2f(45,140);
        glVertex2f(233,149);
        glVertex2f(233,131);
        glEnd();
        //side
        glBegin(GL_POLYGON);
        glColor3f(0.1+col,0.1+col,0.8+col);
        glVertex2f(233,149);
        glVertex2f(233,131);
        glVertex2f(254,128);
        glVertex2f(254,145);
        glEnd();

    //p3
        glColor3f(0.2+col,0.5+col,0.2+col);
        glBegin(GL_POLYGON);
        glVertex2f(51,151);
        glVertex2f(51,140);
        glVertex2f(221,149);
        glColor3f(0.9+col,0.6+col,0.3+col);
        glVertex2f(221,165);
        glVertex2f(51,151);
        glEnd();
        //side
        glBegin(GL_POLYGON);
        glColor3f(0.1+col,0.4+col,0.1+col);
        glVertex2f(221,164);
        glVertex2f(221,149);
        glVertex2f(247,147);
        glVertex2f(247,162);
        glEnd();

    //p4
        //pipe1
        glColor3f(0.48+col,0.27+col,0.44+col);
        glBegin(GL_POLYGON);
        glVertex2f(79,152);
        glVertex2f(79,194);
        glVertex2f(94,194);
        glColor3f(0.0+col,0.0+col,0.0+col);
        glVertex2f(94,155);
        glEnd();
        cloud(59,194);

        //pipe2
        glColor3f(0.44+col,0.48+col,0.27+col);
        glBegin(GL_POLYGON);
        glVertex2f(112,156);
        glVertex2f(112,198);
        glVertex2f(127,198);
        glColor3f(0.0+col,0.0+col,0.0+col);
        glVertex2f(127,158);
        glEnd();
        cloud(92,198);

        //pipe3
        glColor3f(0.27+col,0.48+col,0.44+col);
        glBegin(GL_POLYGON);
        glVertex2f(159,161);
        glVertex2f(159,203);
        glVertex2f(179,203);
        glColor3f(0.0+col,0.0+col,0.0+col);
        glVertex2f(179,160);
        glEnd();
        cloud(144,203);

        glRasterPos2i(100,90);
```

```
    for(int s=0;tag[s]!='\0';s++)
     {
          glColor4f(1.0,1.0,1.0,1.0);

     glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,tag[s]);
     }
   /* glBegin(GL_POLYGON);
    glColor3f(0.0, 0.0, 0.0);
    glVertex2f(200,-50 );
    glVertex2i(100, 0);
    glVertex2i(400, 0);
    glVertex2i(300, -50);
    glEnd();

    glBegin(GL_POLYGON);
    glColor3ub(165.0, 42.0, 42.0);
    glVertex2i(180, 0);
    glVertex2i(180, 55);
    glVertex2i(320, 55);
    glVertex2i(320, 0);
    glEnd();

    glBegin(GL_POLYGON);
    glColor3f(0.0, 0.0, 0.0);
    glVertex2i(360, 4);
    glVertex2i(350, -20);
    glVertex2i(360, -50);
    glEnd();*/

    glPopMatrix();
}

//ship movement left to right

void update(int value)
{

    if (position > 1500.0)
        position = -300.0f;

    position += _move;

    glutPostRedisplay();

    glutTimerFunc(22, update, 0);
}
/*void update3(int value)
{

    if (position > 1500.0)
        position = -300.0f;

    position += _move;

    glutPostRedisplay();

    glutTimerFunc(22, update, 0);
}
*/
///* Tree_Model *///
void Tree_Model_One()
{

    glPushMatrix();
    glTranslatef(110, 110, 0);
    circle(15);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(110, 100, 0);
```

```
    circle(15);
    glPopMatrix();

    glBegin(GL_POLYGON);
    glColor3ub(27.0, 38, 49);
    glVertex2f(109, 70);
    glVertex2f(109, 90);
    glVertex2f(111, 90);
    glVertex2f(111, 70);

    glEnd();
}
void Tree_Model_Two()
{

    glPushMatrix();
    glTranslatef(130, 130, 0);
    circle(5);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(125, 126, 0);
    circle(5);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(135, 126, 0);
    circle(5);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(130, 125, 0);
    circle(5);
    glPopMatrix();

    glBegin(GL_POLYGON);
    glColor3ub(27, 38, 49);
    glVertex2f(129, 110);
    glVertex2f(129, 124);
    glVertex2f(131, 124);
    glVertex2f(131, 110);

    glEnd();
}

void Tree_Model_Three()
{

    glBegin(GL_POLYGON);

    glVertex2f(125, 123);
    glVertex2f(133, 145);
    glVertex2f(141, 123);

    glEnd();

    glBegin(GL_POLYGON);
    glColor3ub(27, 38, 49);
    glVertex2f(132, 110);
    glVertex2f(132, 124);
    glVertex2f(134, 124);
    glVertex2f(134, 110);

    glEnd();
}

/// * Windmill_Stand_Model *///
void Windmill_Stand_Model()
{

    glColor3f(0.38, 0.41, 0.36);
```

```
    glBegin(GL_POLYGON);
    glVertex2i(375, 100);
    glVertex2i(380, 240);
    glVertex2i(384, 240);
    glVertex2i(390, 100);
    glEnd();
}

///* Windmill_Blades_Model *///

void Windmill_Blade()
{

    /// Blade_One
    glPushMatrix();
    glRotatef(spin, 0, 0, 90);
    glBegin(GL_POLYGON);
    glVertex2i(-5, 0);
    glVertex2i(-85, -36);
    glVertex2i(-83, -37);
    glVertex2i(-3, -8);
    glEnd();
    glPopMatrix();

    /// Blade_Two
    glPushMatrix();
    glRotatef(spin, 0, 0, 90);
    glBegin(GL_POLYGON);
    glVertex2i(0, 5);
    glVertex2i(45, 70);
    glVertex2i(50, 73);
    glVertex2i(5, 0);
    glEnd();
    glPopMatrix();

    /// Blade_Three
    glPushMatrix();
    glRotatef(spin, 0, 0, 90);
    glBegin(GL_POLYGON);
    glVertex2i(68, -78);
    glVertex2i(0, 0);
    glVertex2i(5, 5);
    glVertex2i(70, -77);
    glEnd();
    glPopMatrix();
}
///* Windmill_Final_Model *///
void Windmill()
{

    /// Windmill_Stand
    glColor3f(0.38, 0.41, 0.36);
    glPushMatrix();
    Windmill_Stand_Model();
    glPopMatrix();

    /// Windmill_Motor
    glColor3ub(208, 211, 212);
    glPushMatrix();
    glTranslatef(380, 250, 0);
    circle(10);
    glPopMatrix();

    /// Windmill_Rotary_Blades
    glColor3ub(208, 211, 212);
    glPushMatrix();
    glTranslatef(380, 251, 0);
    Windmill_Blade();
    glPopMatrix();
}
/// plane//
```

```
void plane()
{

    glPushMatrix();
    // glTranslatef(position,0.0f,0.0f);
    glBegin(GL_POLYGON); // 1t

    // glColor3ub(247, 249, 249);

    glVertex2f(20.0f, 20.0f);
    glVertex2f(25.0f, 20.0f);
    glVertex2f(25.0f, 21.0f);

    glVertex2f(22.0f, 21.0f);

    glEnd();

    //lines

    //flag
     glColor3f(0.0f, 0.0f, 0.0f);
    glBegin(GL_LINES);
    glVertex2f(22.5, 20);
    glVertex2f(22.5, 10);
    glEnd();

    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
     glColor4f(1.0f, 1.0f, 1.0f,0.5f);
    glBegin(GL_POLYGON);
    glVertex2f(22.5, 17.0);
    glVertex2f(22.5, 10.0);
    glVertex2f(19.0, 13.5);
    glVertex2f(22.5, 17.0);
    glEnd();

    //name
   /* glRasterPos2i(19,12);
    glColor3f(0.0,0.0,0.0);
    for(int s=0;name[s]!='\0';s++)
     {

     glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,name[s]);
     }*/
glDisable(GL_BLEND);

    glBegin(GL_POLYGON); // 1t

    glColor3f(255.0f, 0.0f, 0.0f);

    glVertex2f(25.0f, 22.0f);
    glVertex2f(24.0f, 21.0f);

    glVertex2f(25.0f, 21.0f);

    glEnd();

    glBegin(GL_POLYGON); // window1

    glColor3f(255.0f, 0.0f, 0.0f);
    glVertex2f(23.7f, 20.2f);
    glVertex2f(24.2f, 20.2f);
    glVertex2f(24.2f, 20.7f);

    glVertex2f(23.7f, 20.7f);

    glEnd();

    glBegin(GL_POLYGON); // door
```

```
    glColor3f(255.0f, 0.0f, 0.0f);
    glVertex2f(21.8f, 20.0f);
    glVertex2f(22.2f, 20.0f);
    glVertex2f(22.2f, 20.6f);

    glVertex2f(21.8f, 20.6f);

    glEnd();

    glBegin(GL_POLYGON); // window2

    glColor3f(255.0f, 0.0f, 0.0f);
    glVertex2f(22.8f, 20.2f);
    glVertex2f(23.3f, 20.2f);
    glVertex2f(23.3f, 20.7f);

    glVertex2f(22.8f, 20.7f);

    glEnd();

    glPopMatrix();
}
//both planes
void update1(int value)
{

    if (position1 < -400.0)
        position1 = 1200.0f;

    position1 -= _move1;

    glutPostRedisplay();

    glutTimerFunc(22, update1, 0);
}

/// Model_End
///=============================================================================================================///

///=================///
///*   Object  *///
///=================///

///* plane *///
void plane1()
{
    glColor3ub(241, 196, 15);
    glPushMatrix();

    glTranslatef(position1, 0, 0);
    glTranslatef(0, 360, 0);
    glScaled(10.0f, 10.0f, 0.0f);
      plane();
  /* glRasterPos2i(19,12);
    for(int s=0;name[s]!='\0';s++)
     {
     glColor3f(1.0,1.0,1.0);
     glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,name[s]);
     }*/
 //name
    glRasterPos2i(19,12.5);
    glColor3f(0.0,0.0,0.0);
    for(int s=0;name[s]!='\0';s++)
     {

     glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,name[s]);
     }

    glPopMatrix();
}
```

```
void plane2()
{
    glColor3ub(247, 249, 249);
    glPushMatrix();
    glTranslatef(position1, 0, 0);
    glTranslatef(100, 320, 0);
    glScaled(10.0f, 10.0f, 0.0f);
    plane();
    glRasterPos2i(17,12);
    glColor3f(0.0,0.0,0.0);
    for(int s=0;name2[s]!='\0';s++)
     {

     glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,name2[s]);
     }
    glPopMatrix();
}

///* Sun *///
void Sun()
{
    glColor3f(s3, s2, s1);
    glPushMatrix();
    Sun_Model();
    glPopMatrix();
}

///* Cloud_One_Model_One *///
void cloud_one()
{
    glPushMatrix();
    glTranslatef(cx, 225, 0); //-40
    cloud_model_one();
    glPopMatrix();
}

void star()
{
    if (condition == true)

    {
        glBegin(GL_POINTS);         // Each set of 4 vertices form a quad
        glColor3ub(247, 249, 249);  // Red
        glVertex2f(500.1f, 500.3f); // x, y
        glVertex2f(550.1f, 504.0f);

        glVertex2f(150.1f, 504.0f);

        glVertex2f(310.1f, 504.0f);

        glVertex2f(261.0f, 505.0f);

        glVertex2f(453.1f, 506.0f);

        glVertex2f(616.1f, 507.0f);

        glVertex2f(763.1f, 508.0f);

        glVertex2f(587.1f, 524.0f);

        glVertex2f(954.1f, 574.0f);

        glVertex2f(231.1f, 585.0f);

        glVertex2f(275.1f, 566.0f);

        glVertex2f(852.1f, 557.0f);

        glVertex2f(476.1f, 548.0f);

        glVertex2f(140.1f, 509.0f);
```

```
        glVertex2f(350.1f, 510.0f);

        glVertex2f(061.1f, 511.0f);

        glVertex2f(463.1f, 512.0f);

        glVertex2f(822.1f, 513.0f);

        glVertex2f(110.1f, 514.0f);

        glVertex2f(966.1f, 515.0f);

        glVertex2f(211.1f, 516.0f);

        glVertex2f(313.1f, 517.0f);

        glVertex2f(869.1f, 518.0f);

        glVertex2f(639.1f, 519.0f);

        glVertex2f(106.1f, 520.0f);

        glVertex2f(140.1f, 709.0f);
        glVertex2f(350.1f, 810.0f);

        glVertex2f(061.1f, 911.0f);

        glVertex2f(463.1f, 412.0f);
        glVertex2f(822.1f, 713.0f);

        glVertex2f(110.1f, 614.0f);

        glVertex2f(966.1f, 815.0f);

        glVertex2f(211.1f, 916.0f);

        glVertex2f(313.1f, 1017.0f);

        glVertex2f(869.1f, 718.0f);

        glVertex2f(639.1f, 919.0f);

        glVertex2f(106.1f, 902.0f);
        glVertex2f(106.1f, 908.0f);
        glVertex2f(106.1f, 620.0f);
        glVertex2f(250.1f, 630.0f);
        glVertex2f(106.1f, 906.0f);
        glVertex2f(116.1f, 530.0f);
        glVertex2f(980.1f, 980.0f);
        glVertex2f(900.1f, 930.0f);
        glVertex2f(858.1f, 666.0f);
        glVertex2f(845.1f, 642.0f);
        glVertex2f(900.1f, 900.0f);
        glVertex2f(910.1f, 903.0f);
        glVertex2f(915.1f, 908.0f);
        glVertex2f(919.1f, 920.0f);
        glVertex2f(930.1f, 903.0f);
        glVertex2f(935.1f, 905.0f);
        glVertex2f(940.1f, 905.0f);
        glVertex2f(945.1f, 910.0f);

        glVertex2f(600.1f, 525.0f);

        glVertex2f(650.1f, 535.0f);

        glVertex2f(690.1f, 490.0f);
        glVertex2f(550.1f, 600.0f);
        glVertex2f(500.1f, 600.0f);

        glEnd();
    }
```

```
}

bool con()
{
    return condition = true;
}

float updatex()
{

    return n1 = -700, s1 = 241, s2 = 240, s3 = 236;
}
float updatex1()
{

    return n1 = 1000, s1 = 0, s2 = 1, s3 = 1;
}

void night()
{

    glColor3f(.0, 0.0, 0.0);

    glPushMatrix();

    glBegin(GL_POLYGON);

    glVertex3i(0, n1, 0);
    glVertex3i(0, 1000, 0);
    glVertex3i(1200, 1000, 0);
    glVertex3i(1200, n1, 0);
    glPopMatrix();

    glEnd();
}

///* Cloud_Two_Model_one *///

void cloud_two()
{
    glPushMatrix();
    glTranslatef(bx + 100, 290, 0);
    cloud_model_one();
    glPopMatrix();
}

///* Cloud_Three_Model_Two *///

void cloud_three()
{
    glPushMatrix();
    glTranslatef(ax - 80, 230, 0);
    cloud_model_Two();
    glPopMatrix();
}
///* Cloud_Four_Model_Two *///

void cloud_four()
{
    glPushMatrix();
    glTranslatef(dx + 300, 275, 0);
    cloud_model_Two();
    glPopMatrix();
}
///* Cloud_Five_Model_Three *///
void cloud_five()
{

    glPushMatrix();
    glTranslatef(ax + -300, 310, 0);
    cloud_model_Three();
```

```
    glPopMatrix();
}
///* Cloud_Six_Model_Three *///
void cloud_six()
{

    glPushMatrix();
    glTranslatef(cx + -500, 390, 0);
    cloud_model_Three();
    glPopMatrix();
}

///* House_One *///
void house_one()
{
    glPushMatrix();
    glTranslatef(0, 200, 0);
    house1();
    glPopMatrix();
}

///* House_Two *///
/*
void house_two(){
    glPushMatrix();
    glTranslatef(450,200,0);
    house();
    glPopMatrix();
}
*/
///* House_Two *///
void house_three()
{
    glPushMatrix();
    glTranslatef(320, 237, 0);
    house();
    glPopMatrix();
}
///* Hill_volkano***///
void hill_volkano()
{
    glPushMatrix();
    glTranslatef(0, 200, 0);
    hill_big1();
    glPopMatrix();
}
///* Hill_big_Two *///
void Hill_Big_Two()
{
    glPushMatrix();
    glTranslatef(550, 180, 0); //-20
    hill_big2();
    glPopMatrix();
}
///* Hill_Small_One *///
void Hill_Small_One()
{
    glPushMatrix();
    glTranslatef(0, 200, 0);
    hill_small();
    glPopMatrix();
}
/// * Tilla_One_Model_One *///

void Tilla_One()
{

    glPushMatrix();
    glTranslatef(0, 200, 0);
    Tilla_One_Model();
    glPopMatrix();
```

```
}
/// * Tilla_Two_Model_Two *///
void Tilla_Two()
{

    glPushMatrix();
    glTranslatef(0, 200, 0);
    Tilla_Two_Model();
    glPopMatrix();
}
/// * Tilla_Three_Model_Two *///
void Tilla_Three()
{

    glPushMatrix();
    glTranslatef(400, 200, 0);
    Tilla_Two_Model();
    glPopMatrix();
}
/// * Tilla_Four_Model_One *///
void Tilla_Four()
{

    glColor3f(0.833, 1., 0.0);
    glPushMatrix();
    glTranslatef(380, 200, 0);
    Tilla_One_Model();
    glPopMatrix();
}
///* Tree_1 *///
void Tree_One()
{
    glColor3ub(46, 204, 13.0);
    glPushMatrix();
    glTranslatef(0, 200, 0);
    Tree_Model_One();
    glPopMatrix();
}

///* Tree_2 *///
void Tree_Two()
{
    glColor3ub(46, 204, 13.0);
    glPushMatrix();
    glTranslatef(540, 200, 0);
    Tree_Model_One();
    glPopMatrix();
}

///* Tree_3 *///
void Tree_Three()
{
    glColor3ub(46, 204, 13.0);
    glPushMatrix();
    glTranslatef(750, 200, 0);
    Tree_Model_One();
    glPopMatrix();
}
///* Tree_4 *///
void Tree_Four()
{
    glColor3ub(46, 204, 13.0);
    glPushMatrix();
    glTranslatef(292, 240, 0);
    Tree_Model_One();
    glPopMatrix();
}

///* Tree_5 *///
void Tree_Five()
{
```

```
    glColor3ub(46, 204, 13.0);
    glPushMatrix();
    glTranslatef(30, 180, 0); //-20
    Tree_Model_Two();
    glPopMatrix();
}

///* Tree_6 *///
void Tree_Six()
{
    glColor3ub(46, 204, 13.0);
    glPushMatrix();
    glTranslatef(50, 190, 0); //-10
    Tree_Model_Two();
    glPopMatrix();
}
///* Tree_7 *///
void Tree_Seven()
{
    glColor3ub(46, 204, 13.0);
    glPushMatrix();
    glTranslatef(322, 200, 0);
    Tree_Model_Two();
    glPopMatrix();
}

///* Tree_8 *///
void Tree_Eight()
{
    glColor3ub(46, 204, 13.0);
    glPushMatrix();
    glTranslatef(350, 185, 0); //-15
    Tree_Model_Two();
    glPopMatrix();
}

///* Tree_9 *///
void Tree_Nine()
{
    glColor3ub(46, 204, 13.0);
    glPushMatrix();
    glTranslatef(760, 125, 0); //-75
    Tree_Model_Two();
    glPopMatrix();
}

///* Tree_10 *///
void Tree_Ten()
{
    glColor3ub(46, 204, 13.0);
    glPushMatrix();
    glTranslatef(90, 198, 0); //-2
    Tree_Model_Three();
    glPopMatrix();
}

///* Tree_11 *///
void Tree_Eleven()
{
    glColor3ub(46, 204, 13.0);
    glPushMatrix();
    glTranslatef(125, 200, 0);
    Tree_Model_Three();
    glPopMatrix();
}

///* Tree_12 *///
void Tree_Twelve()
{
    glColor3ub(46, 204, 13.0);
    glPushMatrix();
```

```
    glTranslatef(408, 178, 0); //-22
    Tree_Model_Three();
    glPopMatrix();
}

/// * Windmill *///
void Windmill_One()
{
    glColor3f(0.11, 0.23, 0.36);
    glPushMatrix();
    glTranslatef(0, 190, 0); //-10
    Windmill();
    glPopMatrix();
}

void Windmill_Two()
{
    glColor3f(0.11, 0.23, 0.36);
    glPushMatrix();
    glTranslatef(508, 130, 0); //-70
    Windmill();
    glPopMatrix();
}
void tree()
{

    glColor3f(0.11, 0.23, 0.36);
    glPushMatrix();
    glTranslatef(-120, -110, 0); //-90
    Windmill();
    glPopMatrix();
}
void fire2()
{

    glColor3f(1, 1, 0);
    glPushMatrix();
    glTranslatef(130, 390, 0);
    glScaled(x, x, 0);
    fire();

    glPopMatrix();
}

void fire1()
{

    glColor3f(0.7, 0.0, 0.0);
    glPushMatrix();
    glTranslatef(63, 390, 0);
    glScaled(y, y, 0);
    fire();

    glPopMatrix();
}

/// Object_End
///=============================================================================================================///

///=========================///
///* Display Function *///
///=========================///

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0.0, 0.0, 1.0);

    night();
    star();
    brown_hill();
```

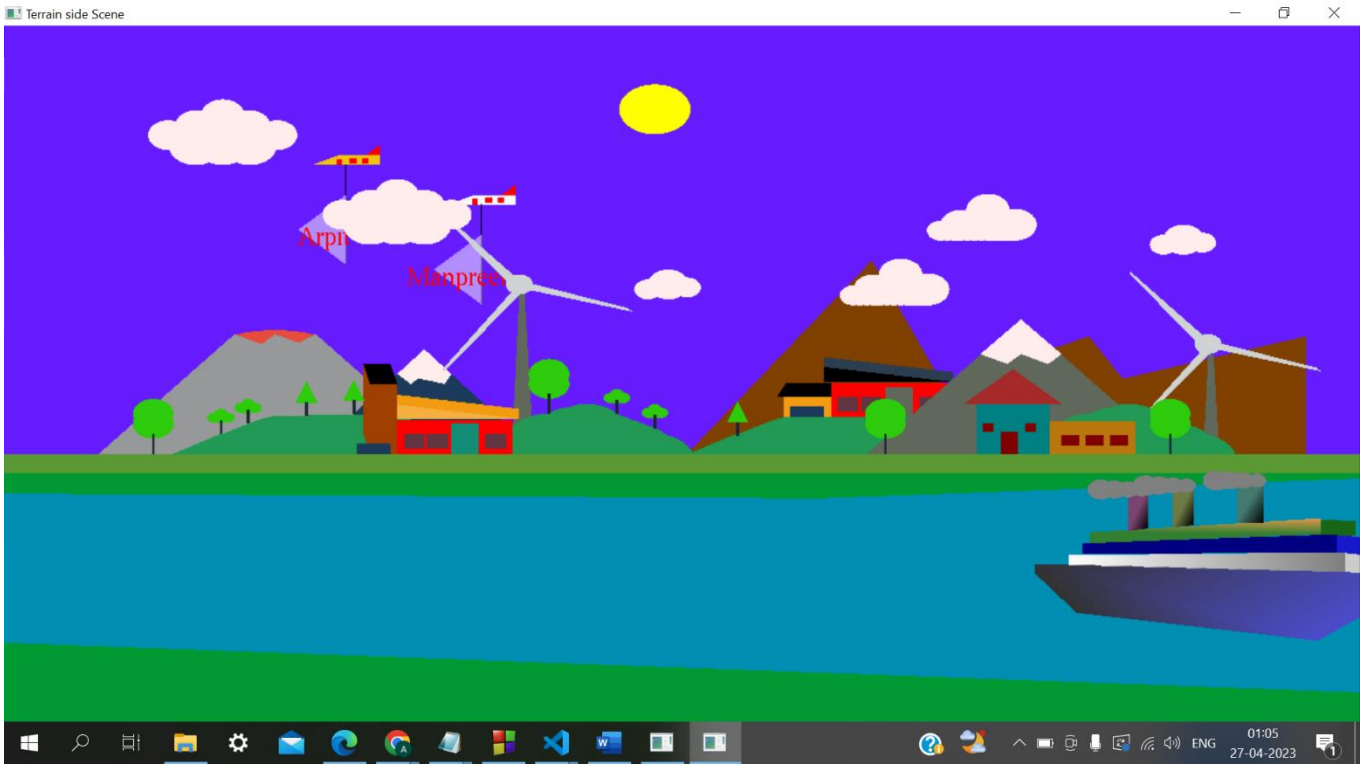
```
    Sun();
    plane1();
    plane2();

    fire1();
    fire2();
    tree();

    hill_volkano();
    Tilla_Four();

    house_three();

    Hill_Big_Two();
    Hill_Small_One();

    cloud_three();
    cloud_four();

    Windmill_One();
    Windmill_Two();

    Tilla_One();
    Tilla_Two();
    Tilla_Three();

    house_one();
    cloud_one();
    house2();

    Tree_One();
    Tree_Two();
    Tree_Three();
    Tree_Four();
    Tree_Five();
    Tree_Six();
    Tree_Seven();
    Tree_Eight();
    Tree_Nine();
    Tree_Ten();
    Tree_Eleven();
    Tree_Twelve();

    cloud_two();
    cloud_five();
    cloud_six();
    field();
    Boat();

    glFlush();
}
///=======================///
///* Speed & Movement *///
///=======================///
///* Sun_Move *///
///cloud movement right left
//in sync with windmill rotation
void move_right()
{

    //sun_move();

    spin = spin + .8;
    ax = ax + .70;
    bx = bx + .80;
    cx = cx + .90;
    dx = dx + .70;

    if (cx > 1000)
    {
```

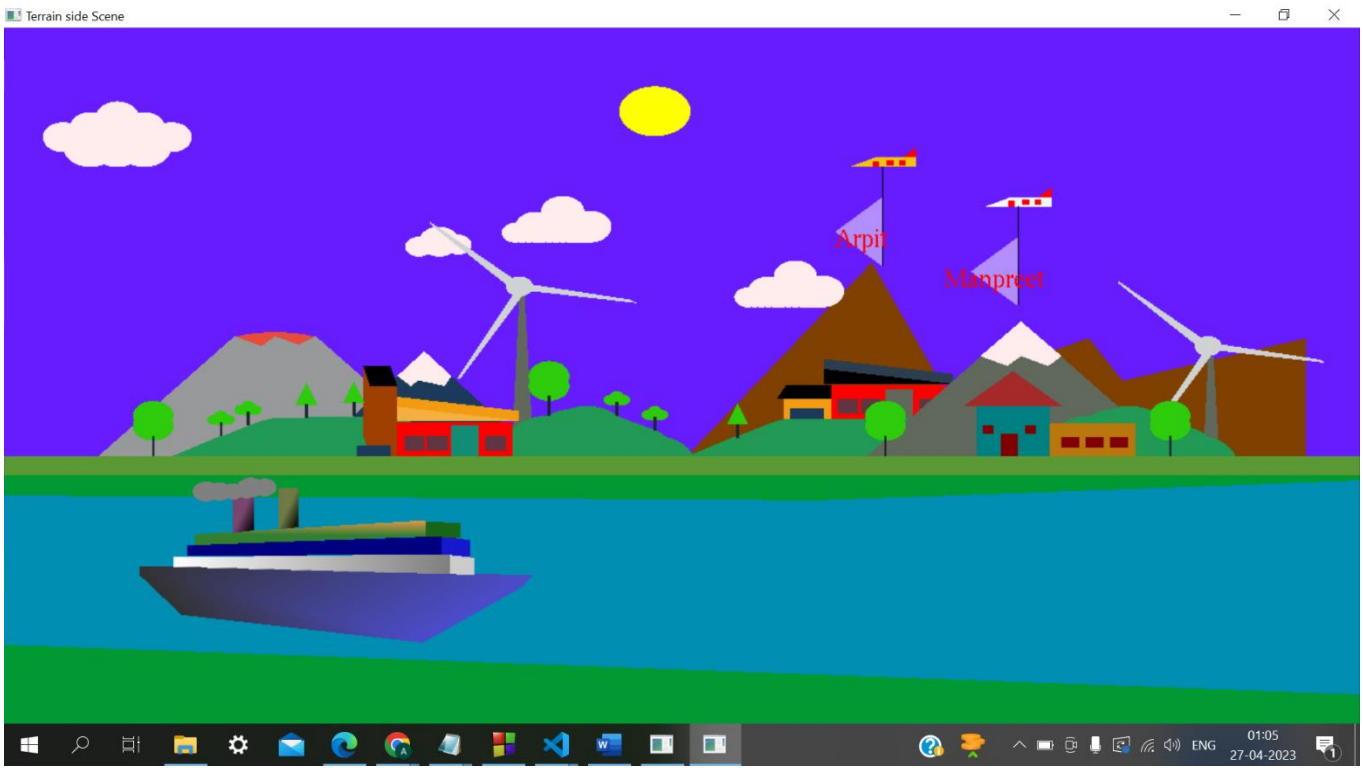
```
        cx = -300;
    }
    if (bx > 1000)
    {
        bx = -400;
    }
    if (cx > 1000)
    {
        cx = -400;
    }
    if (dx > 1000)
    {
        dx = -500;
    }

    glutPostRedisplay();
}

void mouse(int key, int state, int x, int y)
{
    switch (key)
    {
    case GLUT_LEFT_BUTTON:
        if (state == GLUT_DOWN)
        {
            glutIdleFunc(move_right);
        }
        break;
    case GLUT_MIDDLE_BUTTON:
    case GLUT_RIGHT_BUTTON:
        if (state == GLUT_DOWN)
        {
            glutIdleFunc(NULL);
        }
        break;
    default:
        break;
    }
}
void handleKeypress(unsigned char key, int x, int y)
{
    switch (key)
    {
    case 'v':
        r(1);
        sound();
        break;
    case 'g':
        r(1, 1);
        break;
    case 'i':
        _move = 0.0f;

        break;
    case 'o':
        _move = 5.0f;
        break;

    case 'k':
        _move1 = 0.0f;
        break;
    case 'l':
        _move1 = 3.0f;

        break;

    case 'n':

        updatex();
        condition = true;
        break;
```

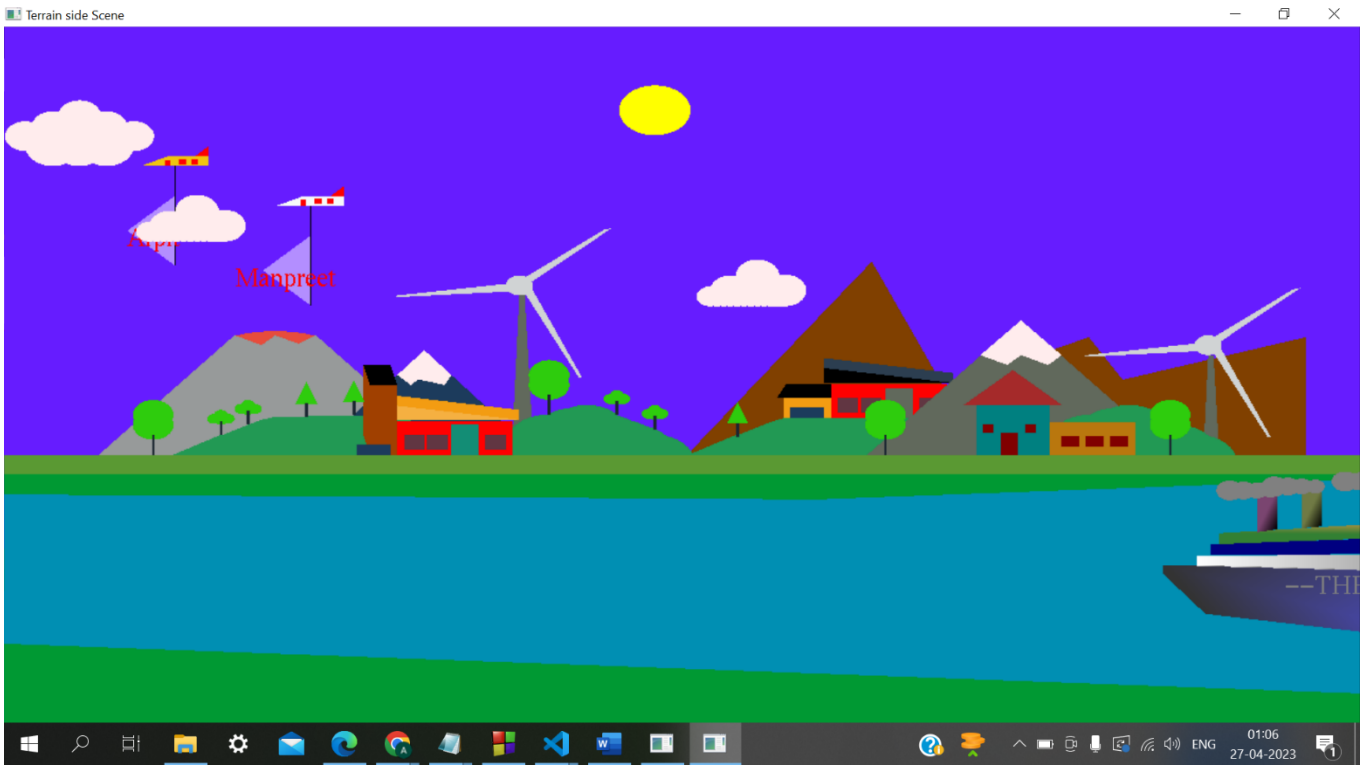
```
    case 'd':
        updatex1();
        condition = false;
        break;

        glutPostRedisplay();
    }
}
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition(50, 50);
    glutInitWindowSize(1800, 900);
    glutCreateWindow("Terrain side Scene");
    init();
    glutDisplayFunc(display);
    glutTimerFunc(20, update, 0);
    glutTimerFunc(20, update1, 0);
    glutTimerFunc(20, update2, 0);
    glutMouseFunc(mouse);
    glutKeyboardFunc(handleKeypress);
    glutMainLoop();
}
```

5. Output Screenshots

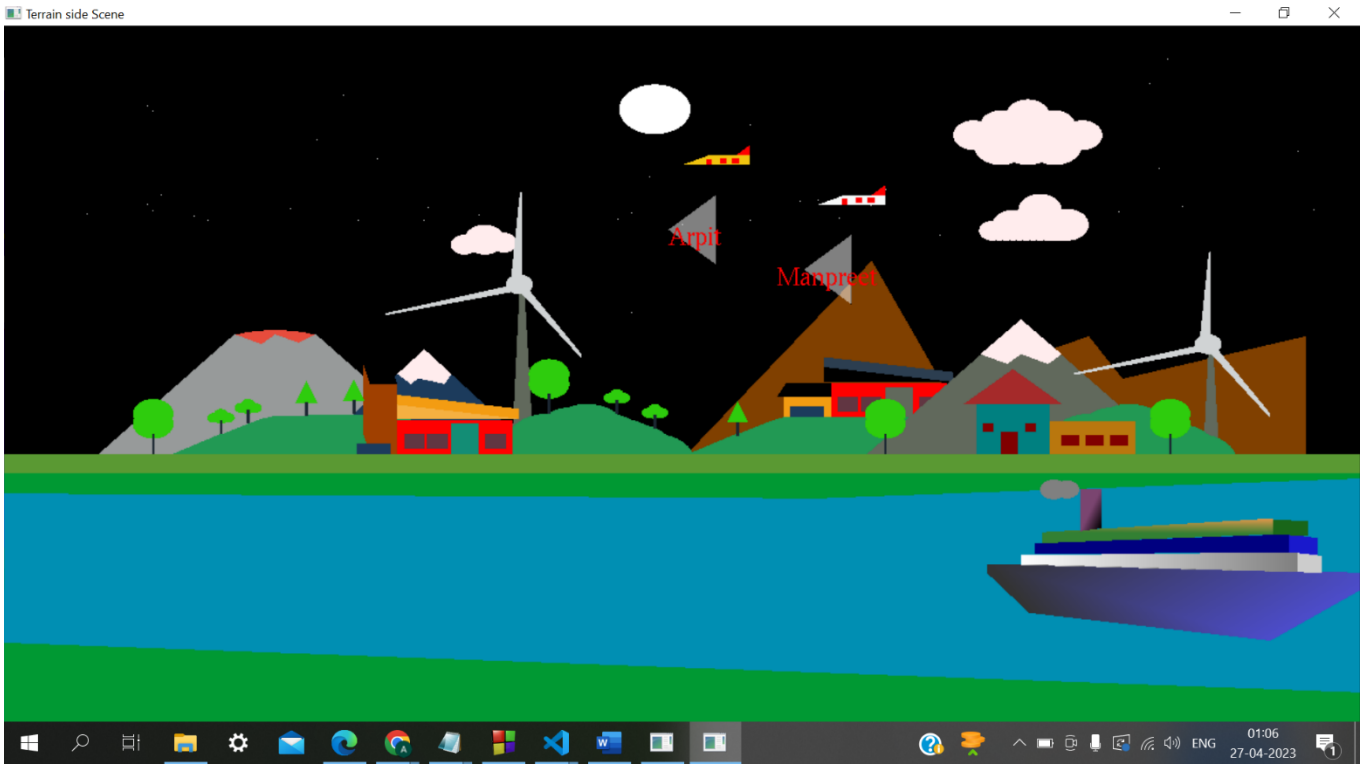
- Movement of ship, clouds, name tagged planes and rotation of windmill



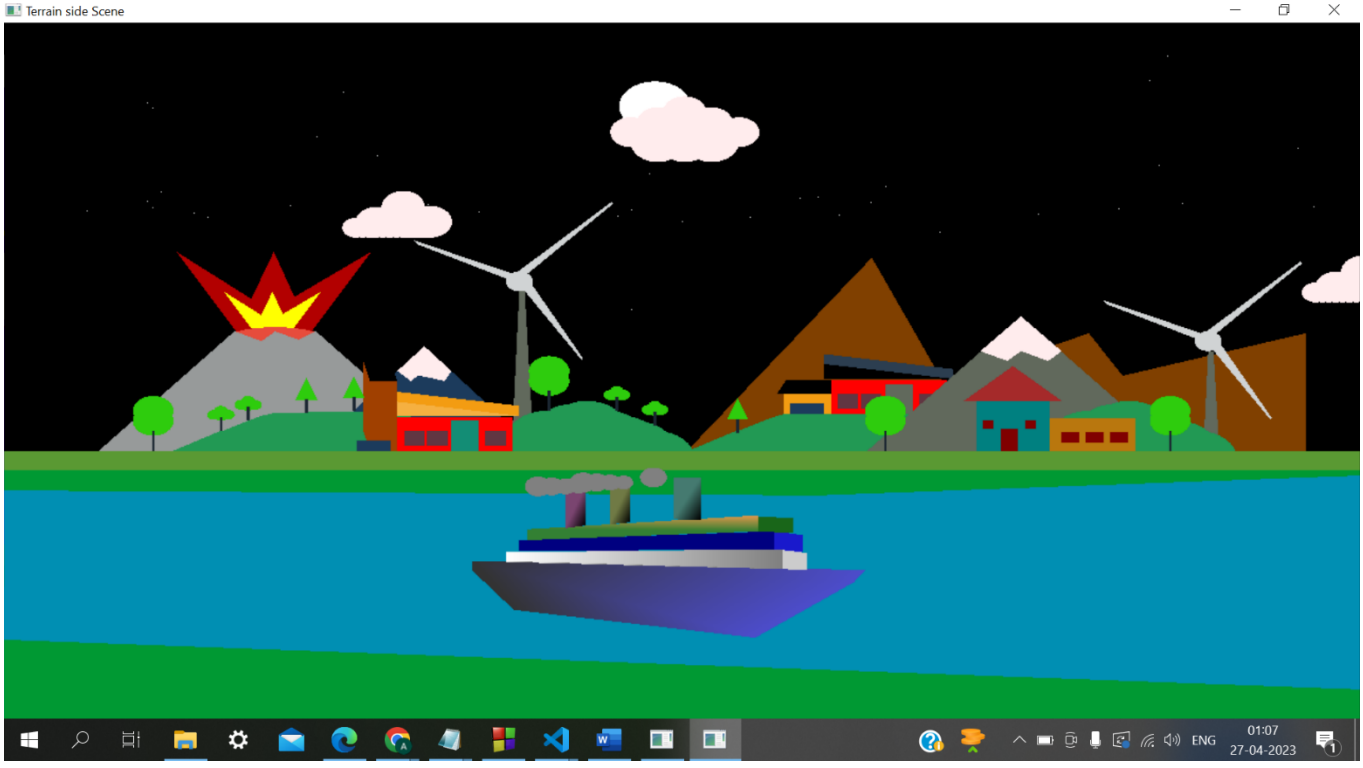

Terrain side Scene
Arpit
Manpreet
ENG
01:05
27-04-2023

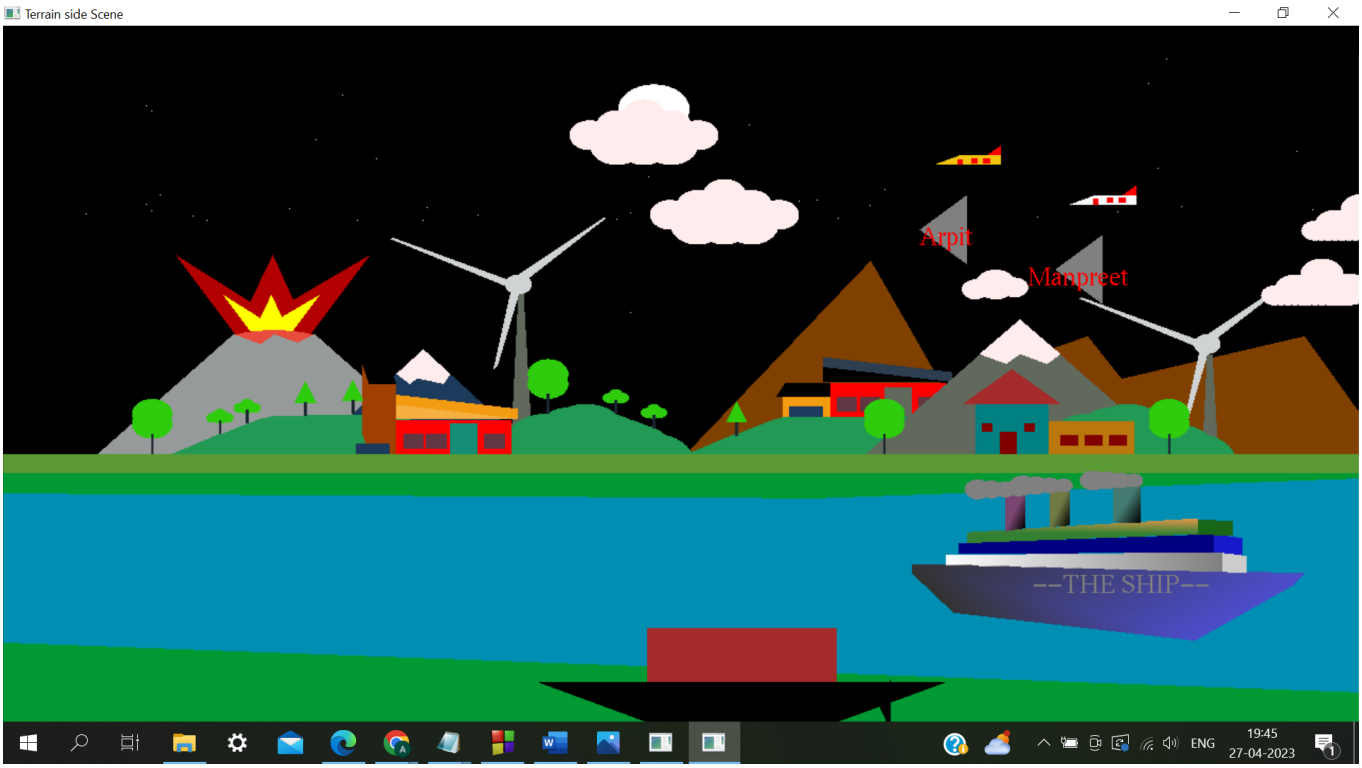

Terrain side Scene
Manpreet
ENG
01:06
27-04-2023

- Night scene with stars



- Volcano eruption with burst sound





- Two ship movement task