Computer Graphics (UCS505)

Project on

Hilly Landscape

Submitted By

Anchal	102103227
Praval	102103400
Chetna Sharma	102103396

Group No. 10

B.E. Third Year – COE15

Submitted To:

Ms. Jyoti

Computer Science and Engineering Department

Thapar Institute of Engineering and

Technology Patiala – 147001

Table of Contents

Introduction to Project

It is a Computer Graphics Project aimed at creating a captivating visualization of a hilly landscape. It is developed using OpenGL, the project incorporates various elements such as a sailing ship, water bodies, and a flying plane to offer an immersive experience to the viewer.

1. **Project Overview:**

   The primary objective of the project is to demonstrate the potential of computer graphics in rendering dynamic environments. Through the implementation of interactive elements and user-controlled features, the project aims to engage the audience and showcase the creative possibilities of OpenGL programming.

2. **Key Components:**

   Ship Rendering: The central focus of the project is the rendering of a sailing ship, comprising different components such as the base, sides, and pipes.

   Environmental Elements: Various environmental elements, including mountains, water bodies, and a sun/moon, are rendered to create a realistic hilly landscape.

   Interactive Features: The project incorporates interactive features such as user-controlled ship movement and the ability to trigger firecrackers, enhancing user engagement and interactivity.

   User Interface: An introduction page provides essential project details, including the project title, college name, department, and project creators names.

3. **Implementation:**

   OpenGL Usage: OpenGL primitives such as polygons, lines, and arcs are extensively used to render the graphical elements of the scene.

   User Input Handling: Keyboard input is utilized to enable user interaction, allowing users to control the movement of the ship and trigger interactive elements.

4. Conclusion:

"TIET SHIP" successfully demonstrates the capabilities of computer graphics in creating visually appealing and interactive environments. The project serves as a testament to the creative potential of OpenGL programming and showcases the possibilities for future projects in the field of computer graphics.

Computer Graphics Concepts Used

Graphics Primitives: The code uses various graphics primitives such as lines, polygons, and arcs to draw different objects like the sea, mountains, and the ship. These primitives are used to create basic shapes, which are then combined to form more complex objects. For example, the ship is made up of multiple polygons and arcs (lines 152-244).

2-D Geometrical Transformations: The glTranslatef function is used to translate objects to their desired positions. This is used to create the illusion of motion for the ship and the plane. For instance, the ship function translates the ship to the position x (line 142), and the plane function translates the plane to the position pl (line 317).

Color Tables: The glColor3f function is used to set the color of different objects. This is used to create visually distinct objects, such as the sea, mountains, and the ship. For example, the sea color is set using glColor3f (line 261), and the ship s color is set using glColor3f with varying RGB values (lines 154-156, 160-162, 166-168, 172-174, 178-180, 184-186).

Viewing & Clipping in 2-D: The code uses the gluOrtho2D function to set up the viewing frustum, which defines the visible area in the 2D space. This is used to establish the coordinate system and the aspect ratio of the rendering window (line 102).

Fundamentals of Computer Graphics: The code demonstrates the application of computer graphics in creating a visually engaging animation. It covers various aspects of computer graphics, such as rendering, transformation, and color representation.

Input-Output Devices: Although not explicitly shown in the code, the program uses the keyboard and mouse as input devices to control the animation and change the background color.

Lines: The most common algorithm for drawing lines in computer graphics is the Bresenham's line algorithm. This algorithm is used to draw a line between two points (x1, y1) and (x2, y2) in a rasterized image. The algorithm works by iterating through the pixels along the line and determining whether to fill each pixel based on the error term. The error term is calculated as the difference between the actual and expected values of the y-coordinate. The algorithm is efficient and accurate, making it a popular choice for drawing lines in computer graphics.

Polygons: Polygons are drawn in computer graphics by filling the area enclosed by a set of line segments. The most common algorithm for filling polygons is the scanline algorithm. This algorithm works by dividing the polygon into horizontal slices and filling each slice with a solid color. The algorithm is efficient and accurate, making it a popular choice for drawing polygons in computer graphics.

Arcs: Arcs are drawn in computer graphics by calculating the coordinates of the points along the arc and connecting them with line segments. The most common algorithm for drawing arcs is the parametric form algorithm. This algorithm works by calculating the coordinates of the points along the arc using the parametric form of the circle equation. The algorithm is efficient and accurate, making it a popular choice for drawing arcs in computer graphics.

User Defined Functions

1. **frontscreen():** This function is responsible for displaying the introduction page of the project. It uses OpenGL commands to draw text and lines on the screen, presenting information such as the project title, college name, department, etc.

2. **Drawarc():** This function is used to draw arcs, representing the sun or moon in the sky. It calculates the vertices of the arc based on the start angle, end angle, center coordinates, and radius provided.

3. **cloud(int m, int n):** This function is used to draw clouds on the screen. It calls the Drawarc() function multiple times to create circular shapes at different positions and sizes, resembling clouds.

4. **ship(float x):** This function draws the ship on the screen. It consists of multiple polygons representing different parts of the ship, such as the base, sides, pipes, etc. The ship's position can be adjusted horizontally using the x parameter.

5. **water():** This function draws the water body at the bottom of the screen. It's a simple rectangle filled with a blue color, representing the sea or ocean.

6. **mountain2() , mountain() , mountain3() :** These functions draw mountains of different shapes and colors on the screen. They consist of multiple polygons representing mountain peaks at different positions and heights.

7. **flag(float x):** This function draws a flag on the ship. It includes a pole and a triangular flag with a red color. The x parameter adjusts the flag's position horizontally.

8. **crackers():** This function seems to be intended for drawing fireworks or crackers. However, in the provided code, it is commented out (// if(cf==1))

and not being used in the display function.

9. **plane():** This function draws an airplane flying across the screen. It consists of several polygons representing different parts of the airplane, such as the body, wings, etc. The plane's position can be adjusted horizontally.

These functions collectively create the scene displayed by the program, incorporating various elements like the ship, water, mountains, flag, and airplane. Each function focuses on drawing a specific component of the scene using OpenGL primitives and transformations.

Code

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


float pos = 0, x = 0, col = 0, i, p, c, cf
= 0, pr, pl = 0;
int screen = 0;


//Introduction Page
char name[35] = "\n - \n ";
char college[100] = "\n(TIET)
Thapar Institute of Engineering
Technology       ";
char dept[100] = "Computer Science
& Engineering Department ";
char heading[100] = "Made by :::
Anchal , Praval Kaushal and Chetna
Sharma  ";
char row1[100] = "Submitted to :::
Jyoti mam ";
char row2[100] = "
---Computer Graphics-UCS505";
char emsg[100] = "Press ENTER to
start";
char title[50] = "SCENERY
VIEW...!";
char dash[100] = "----------------------
-----------------------------------------
-----------------";


void frontscreen(void)
{
   glPushMatrix();
   glTranslatef(50, 100, 0);
   glClearColor(0.8, 0.8, 1.0, 1.0);
   int s;

glClear(GL_COLOR_BUFFER_BIT)
;


   glRasterPos2i(184, 480);
//displays college name
   for (s = 0; college[s] != '\0'; s++)
   {
      glColor3f(0.0, 0.0, 1.0);

glutBitmapCharacter(GLUT_BITMA
P_TIMES_ROMAN_24, college[s]);
   }


   glRasterPos2i(180, 450);
//displays dept.
   for (s = 0; dept[s] != '\0'; s++)
   {
      glColor3f(1.0, 0.0, 0.0);

glutBitmapCharacter(GLUT_BITMA
P_TIMES_ROMAN_24, dept[s]);
   }


   glRasterPos2i(210, 380);
//displays project name
   for (s = 0; title[s] != '\0'; s++)
   {
      glColor3f(0.0, 0.0, 0.0);

glutBitmapCharacter(GLUT_BITMA
P_TIMES_ROMAN_24, title[s]);
```

```
    }


    glRasterPos2i(155, 370);
//displays dashes
    for (s = 0; dash[s] != '\0'; s++)
    {
        glColor3f(0.0, 0.0, 0.0);

glutBitmapCharacter(GLUT_BITMA
P_9_BY_15, dash[s]);
    }


    glRasterPos2i(170, 340);
//displays heading of table
    for (s = 0; heading[s] != '\0'; s++)
    {
        glColor3f(0.0, 0.0, 0.0);

glutBitmapCharacter(GLUT_BITMA
P_TIMES_ROMAN_24, heading[s]);
    }


    glRasterPos2i(155, 330);
//displays dashes
    for (s = 0; dash[s] != '\0'; s++)
    {
        glColor3f(0.0, 0.0, 0.0);

glutBitmapCharacter(GLUT_BITMA
P_9_BY_15, dash[s]);
    }


    glRasterPos2i(170, 310);
//displays 1st row in table
    for (s = 0; row1[s] != '\0'; s++)
    {
        glColor3f(0.0, 0.0, 0.0);

glutBitmapCharacter(GLUT_BITMA
P_TIMES_ROMAN_24, row1[s]);
    }


    glRasterPos2i(155, 370);
//displays dashes
    for (s = 0; dash[s] != '\0'; s++)
    {
        glColor3f(0.0, 0.0, 0.0);

glutBitmapCharacter(GLUT_BITMA
P_9_BY_15, dash[s]);
    }


    glRasterPos2i(170, 290);
//displays 2nd row in table
    for (s = 0; row2[s] != '\0'; s++)
    {
        glColor3f(0.0, 0.0, 0.0);

glutBitmapCharacter(GLUT_BITMA
P_TIMES_ROMAN_24, row2[s]);
    }


    glRasterPos2i(155, 270);
//displays dashes
    for (s = 0; dash[s] != '\0'; s++)
    {
        glColor3f(0.0, 0.0, 0.0);

glutBitmapCharacter(GLUT_BITMA
P_9_BY_15, dash[s]);
    }
```

```
   glRasterPos2i(200, 100);
//displays "enter msg"
   for (s = 0; emsg[s] != '\0'; s++)
   {
      glColor3f(1.0, 0.0, 0.5);

glutBitmapCharacter(GLUT_BITMA
P_TIMES_ROMAN_24, emsg[s]);
   }

   glPopMatrix();
   glFlush();
}

//sun/moon
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();
}

//smoke
void cloud(int m, int n)
{
   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);
   }
}

void ship(float x)
{
   glPushMatrix();
   glTranslatef(x, 0, 0);
   //base
   glColor3f(0.2 + col, 0.2 + col, 0.2 +
col);
   glBegin(GL_POLYGON);
   glVertex2f(10, 119);
   glVertex2f(10, 110);
   glVertex2f(41, 70);
   glColor3f(0.3 + col, 0.3 + col, 0.8 +
col);
   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);
   glPopMatrix();
}


void water()
{
   glBegin(GL_POLYGON);
   glColor3f(0.2 + col, 0.2 + col, 0.6 +
col);
   glVertex2f(00, 00);
   glVertex2f(00, 300);
   glVertex2f(1024, 300);
   glVertex2f(1024, 00);
   glEnd();
}


void mountain2()
{
   float a, b;
   glColor3f(0.6 + col, 0.4 + col, 0.2 +
col);
   for (a = 0, b = 300; a < 1025; a = a
+ 80)
   {
      glBegin(GL_POLYGON);
      glVertex2f(-40 + a, b);
      glVertex2f(10 + a, b + 140);
      glVertex2f(60 + a, b);
      glEnd();
   }
}
```

```
void mountain()
{
   float a, b;
   glColor3f(0.8 + col, 0.6 + col, 0.4 +
col);
   for (a = 0, b = 300; a < 1025; a = a
+ 80)
   {
      glBegin(GL_POLYGON);
      glVertex2f(0 + a, b);
      glVertex2f(50 + a, b + 100);
      glVertex2f(100 + a, b);
      glEnd();
   }
}

void mountain3()
{
   float a, b;
   glColor3f(0.4 + col, 0.2 + col, 0.0 +
col);
   for (a = 0, b = 350; a < 1025; a = a
+ 80)
   {
      glBegin(GL_POLYGON);
      glVertex2f(0 + a, b);
      glVertex2f(50 + a, b + 180);
      glVertex2f(100 + a, b);
      glEnd();
   }
}

void flag(float x)
{
   int s;
   glPushMatrix();
   glTranslatef(x, 0, 0);
   glColor3f(0.0, 0.0, 0.0);
   glBegin(GL_POLYGON);
   glVertex2f(245, 160);
   glVertex2f(245, 250);
   glVertex2f(242, 250);
   glVertex2f(242, 160);
   glEnd();
   glColor3f(0.8, 0.1, 0.1);
   glBegin(GL_POLYGON);
   glVertex2f(245, 250);
   glVertex2f(275, 215);
   glVertex2f(245, 180);
   glEnd();
   glRasterPos2i(50, 80);
//displays college name
   for (s = 0; college[s] != '\0'; s++)
   {
      glColor3f(1.0, 1.0, 1.0);

glutBitmapCharacter(GLUT_BITMA
P_TIMES_ROMAN_24, name[s]);
   }
   glPopMatrix();
}
void crackers()
{
   // if(cf==1)
   {
```

```
        glColor3f(1, 0, 0);
        glBegin(GL_POLYGON);
        glVertex2f(100 + pos, 100 + pr);
        glVertex2f(100 + pos, 110 + pr);
        glVertex2f(101 + pos, 110 + pr);
        glVertex2f(101 + pos, 100 + pr);
        glEnd();
        glFlush();
    }
    glutPostRedisplay();
}

void init()
{
    glClearColor(0.0, 0.0, 0.0, 1.0);
    glLoadIdentity();
    gluOrtho2D(0, 1024, 0, 768);
}



void plane()
{
    glPushMatrix();
    glTranslatef(0 + pl, 700, 0);
    glColor3f(0.0, 0.0, 0.0);

glBegin(GL_POLYGON);//rectangul
ar body
    glVertex2f(0.0, 30.0 / 3);
    glVertex2f(0.0, 55.0 / 3);
    glVertex2f(135.0 / 3, 55.0 / 3);
    glColor3f(1.0, 0.0, 0.0);
    glVertex2f(135.0 / 3, 30.0 / 3);
    glEnd();


    glColor3f(0.0, 0.0, 0.0);
    glBegin(GL_POLYGON);//upper
triangle construction plane
    glVertex2f(135.0 / 3, 55.0 / 3);
    glVertex2f(150.0 / 3, 50.0 / 3);
    glVertex2f(155.0 / 3, 45.0 / 3);
    glVertex2f(160.0 / 3, 40.0 / 3);
    glVertex2f(135.0 / 3, 40.0 / 3);
    glEnd();


    glColor3f(0.0, 0.0, 0.0);

glBegin(GL_LINE_LOOP);//outline
of upper triangle plane
    glVertex2f(135.0 / 3, 55.0 / 3);
    glVertex2f(150.0 / 3, 50.0 / 3);
    glVertex2f(155.0 / 3, 45.0 / 3);
    glVertex2f(160.0 / 3, 40.0 / 3);
    glVertex2f(135.0 / 3, 40.0 / 3);
    glEnd();


    glColor3f(1.0, 0.0, 0.0);
    glBegin(GL_POLYGON);//lower
triangle
    glVertex2f(135.0 / 3, 40.0 / 3);
    glVertex2f(160.0 / 3, 40.0 / 3);
    glVertex2f(160.0 / 3, 37.0 / 3);
    glVertex2f(145.0 / 3, 30.0 / 3);
    glVertex2f(135.0 / 3, 30.0 / 3);
    glEnd();
```

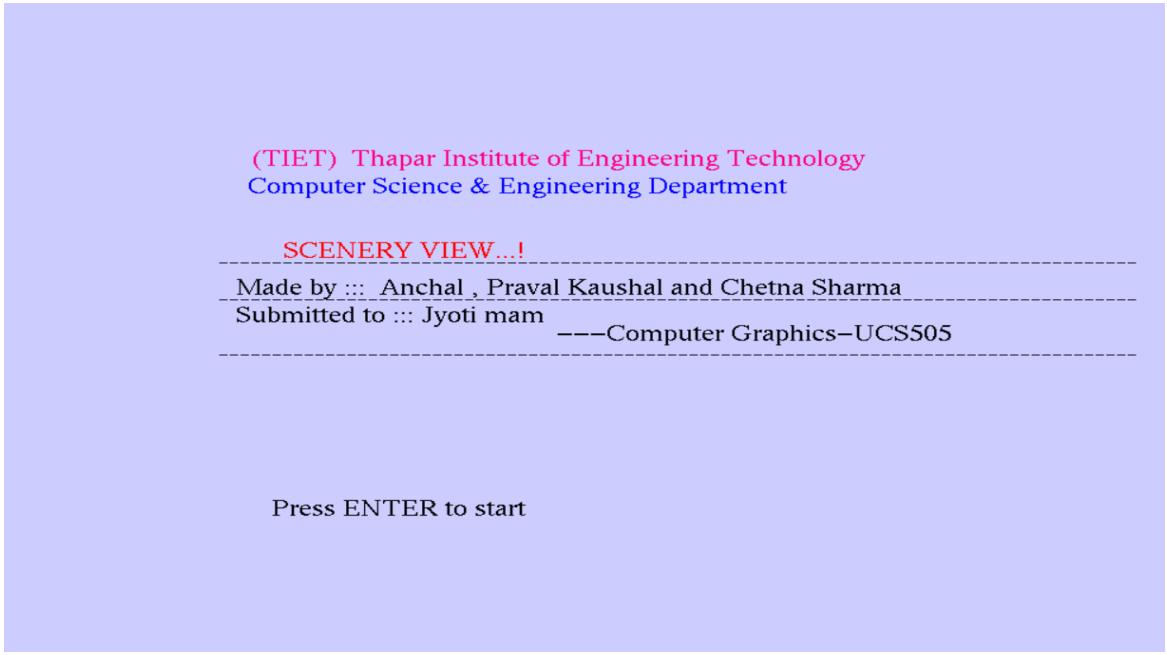
```
   glColor3f(1.0, 0.0, 0.0);
   glBegin(GL_POLYGON);//back
wing
   glVertex2f(0.0, 55.0 / 3);
   glVertex2f(0.0, 80.0 / 3);
   glVertex2f(10.0 / 3, 80.0 / 3);
   glVertex2f(40.0 / 3, 55.0 / 3);
   glEnd();


   glColor3f(1.0, 0.0, 0.0);
   glBegin(GL_POLYGON);//left
side wing
   glVertex2f(65.0 / 3, 55.0 / 3);
   glVertex2f(50.0 / 3, 70.0 / 3);
   glVertex2f(75.0 / 3, 70.0 / 3);
   glVertex2f(90.0 / 3, 55.0 / 3);
   glEnd();


   glColor3f(1.0, 0.0, 0.0);

glBegin(GL_POLYGON);//rightside
wing
   glVertex2f(70.0 / 3, 40.0 / 3);
   glVertex2f(100.0 / 3, 40.0 / 3);
   glVertex2f(80.0 / 3, 15.0 / 3);
   glVertex2f(50.0 / 3, 15.0 / 3);
   glEnd();
   glPopMatrix();
}


void display()
{

glClear(GL_COLOR_BUFFER_BIT)
;
   if (screen == 0)
   {
      frontscreen();
      glFlush();
   }
   else if (screen == 1)
   {
      mountain3();
      mountain2();
      mountain();
      water();
      if (cf == 1)
      {
         crackers();
         pr += 2;
      }
      ship(pos);
      flag(pos);

      plane();
      pl += 2;
      if (pl == 1200)
         pl = 0;

      if (i == 1)
      {
         glColor3f(0.9, 0.9, 0.5);
         Drawarc(0, 360, 200, 700,
10);
      }
      if (i == 2)
      {
```

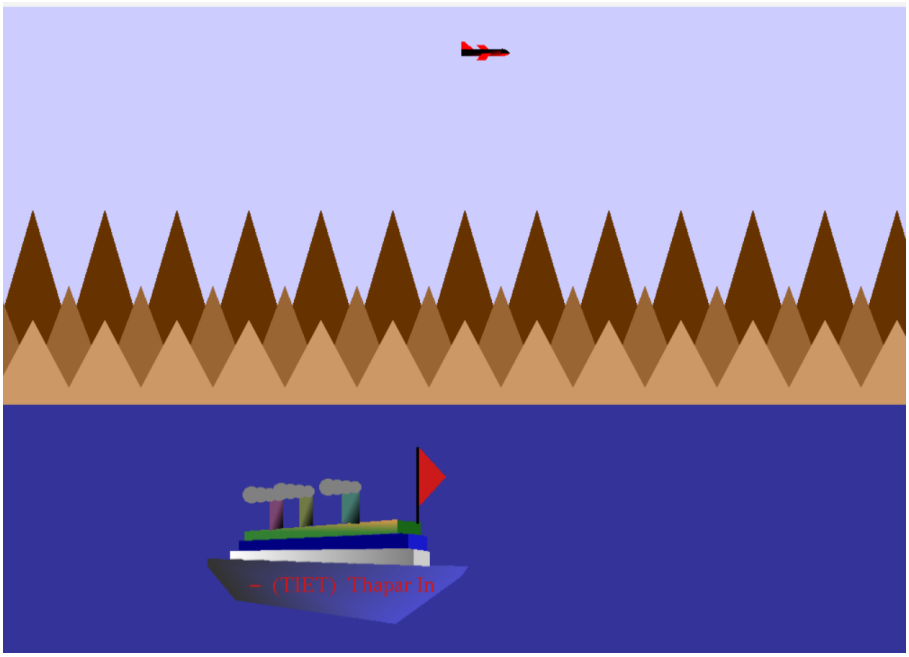
```
        glColor3f(0.8, 0.7, 0.4);
        Drawarc(0, 360, 400, 700,
10);
      }
      if (i == 3)
      {
        glColor3f(1.0, 1.0, 1.0);
        Drawarc(0, 360, 600, 700,
10);
      }
      if (i == 4)
      {
        glColor3f(1.0, 1.0, 1.0);
        Drawarc(0, 360, 800, 700,
10);
      }
      glFlush();
   }
   glutPostRedisplay();
   glutSwapBuffers();
   glFlush();
}

void keyboard(unsigned char key, int
x, int y)
{
   switch (key)
   {
   case 13:if (screen == 0)
   {
      screen = 1;
      glutPostRedisplay();
   }
        break;
   case 27:exit(0);
      break;
   case 'y':glClearColor(0.8, 0.5, 0.4,
1.0);
      col = -0.1;
      i = 0;
      glutPostRedisplay();
      break;
   case 'm':glClearColor(0.5, 0.5, 1.0,
1.0);
      i = 1;
      col = 0.2;
      glutPostRedisplay();
      break;
   case 'a':glClearColor(0.9, 0.9, 0.3,
1.0);
      i = 2;
      col = 0.3;
      glutPostRedisplay();
      break;
   case 'e':glClearColor(0.8, 0.5, 0.4,
1.0);
      col = -0.1;
      i = 0;
      glutPostRedisplay();
      break;
   case 'n':glClearColor(0.5, 0.5, 0.5,
1.0);
      i = 3;
      glutPostRedisplay();
      col = -0.2;
      break;
   case 'd':glClearColor(0.0, 0.0, 0.0,
```

```
0.0);
        i = 4;
        col = -0.3;
        glutPostRedisplay();
        break;
    case '6':pos += 2;
        break;
    case '4':pos -= 2;
        break;
    case 'z':cf = 1;
        pr = 0;
        glutPostRedisplay();
        break;
    }
}
void SpecialInput(int key, int x, int y)
{
    switch (key)
    {
    case GLUT_KEY_UP:
        break;
    case GLUT_KEY_DOWN:
        break;
    case GLUT_KEY_LEFT:
        pos -= 2;
        break;
    case GLUT_KEY_RIGHT:
        pos += 2;
        break;
    }
}

int main()
{

glutInitDisplayMode(GLUT_DOUBL
E | GLUT_RGB);
    glutInitWindowSize(1024, 768);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("\n-\n");
    glutSwapBuffers();
    glutKeyboardFunc(keyboard);
    glutDisplayFunc(display);
    glutSpecialFunc(SpecialInput);
    init();
    glutMainLoop();
}
```

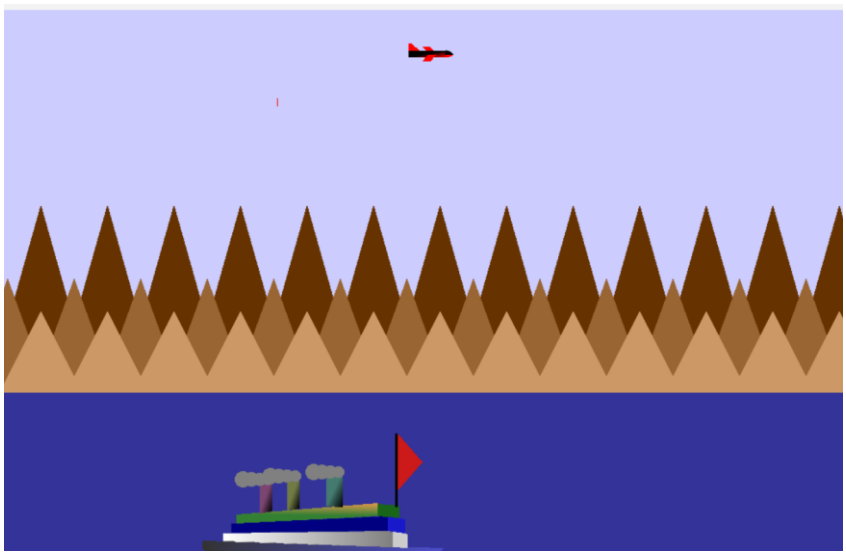
Output Screenshots

Press Enter to start

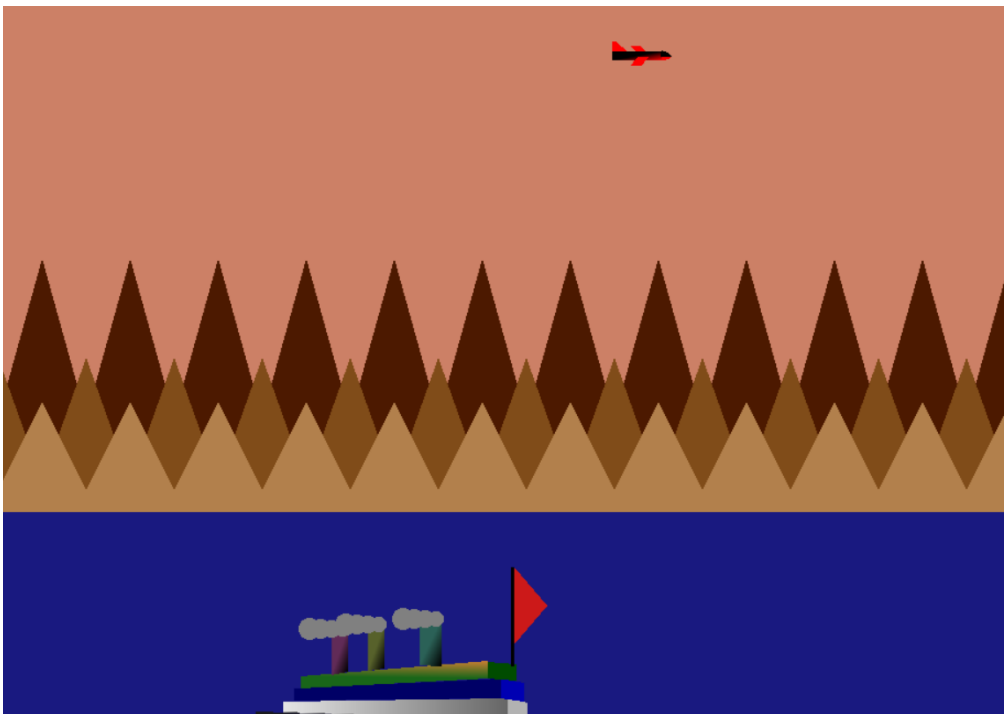




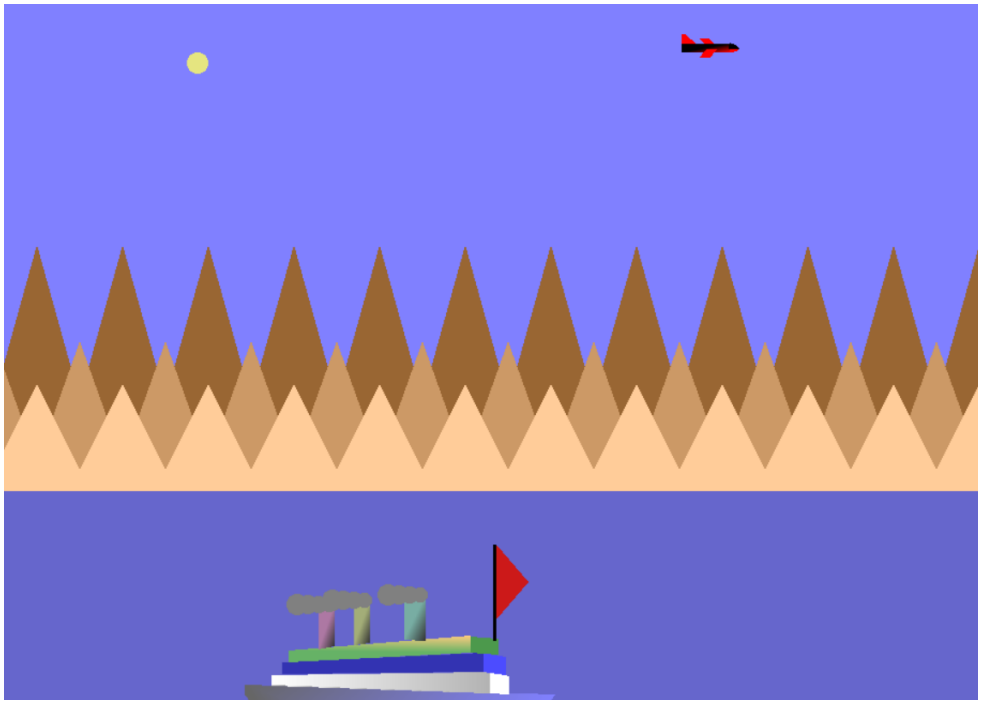
z -> Shoot rocket



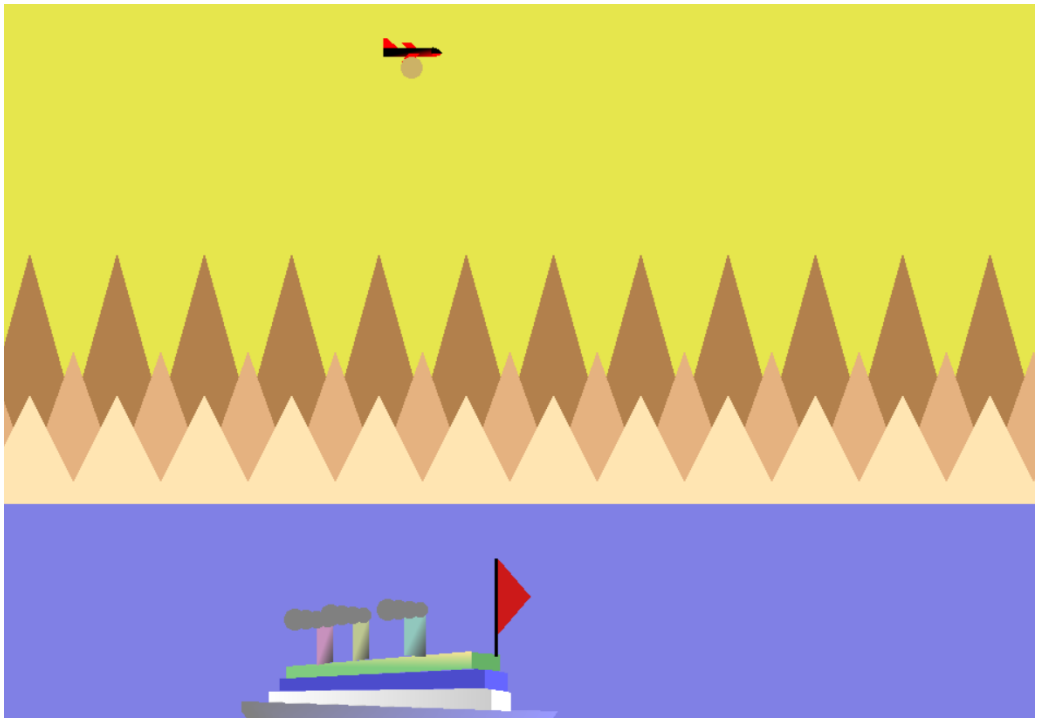
y -> Early Morning



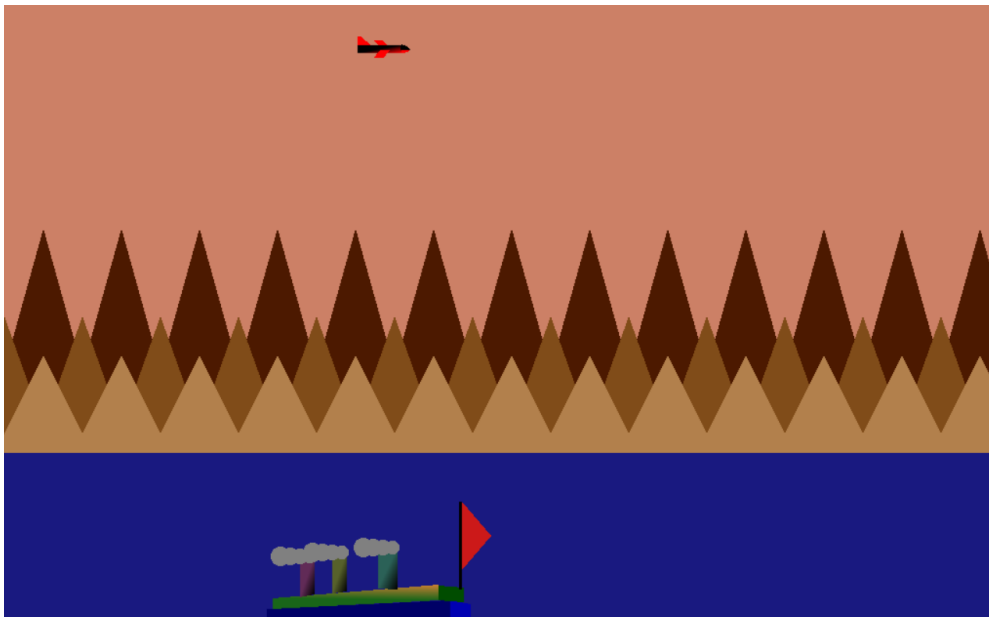
m -> Morning



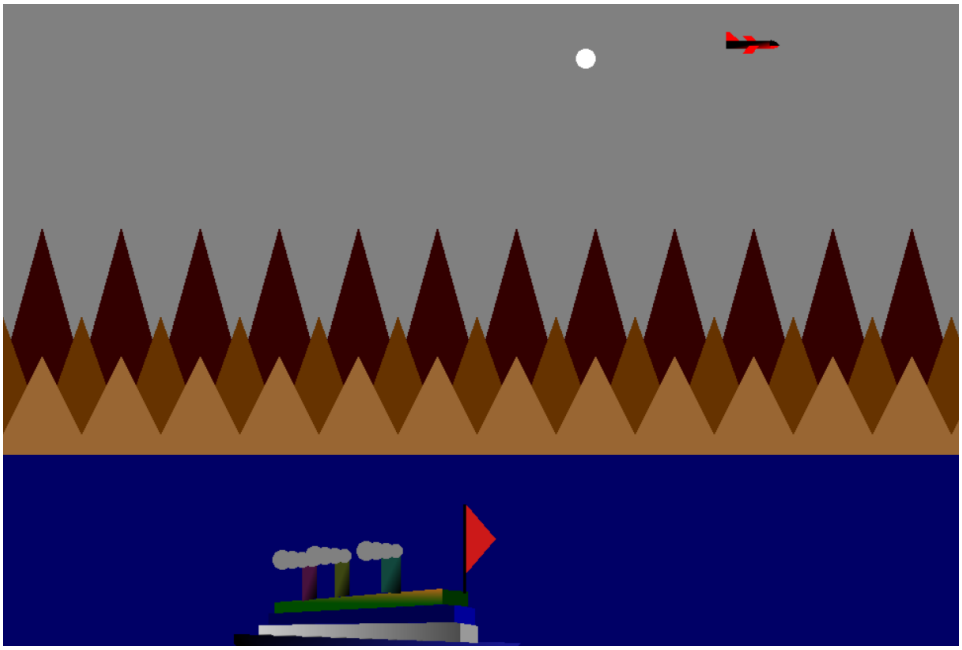
a -> Afternoon



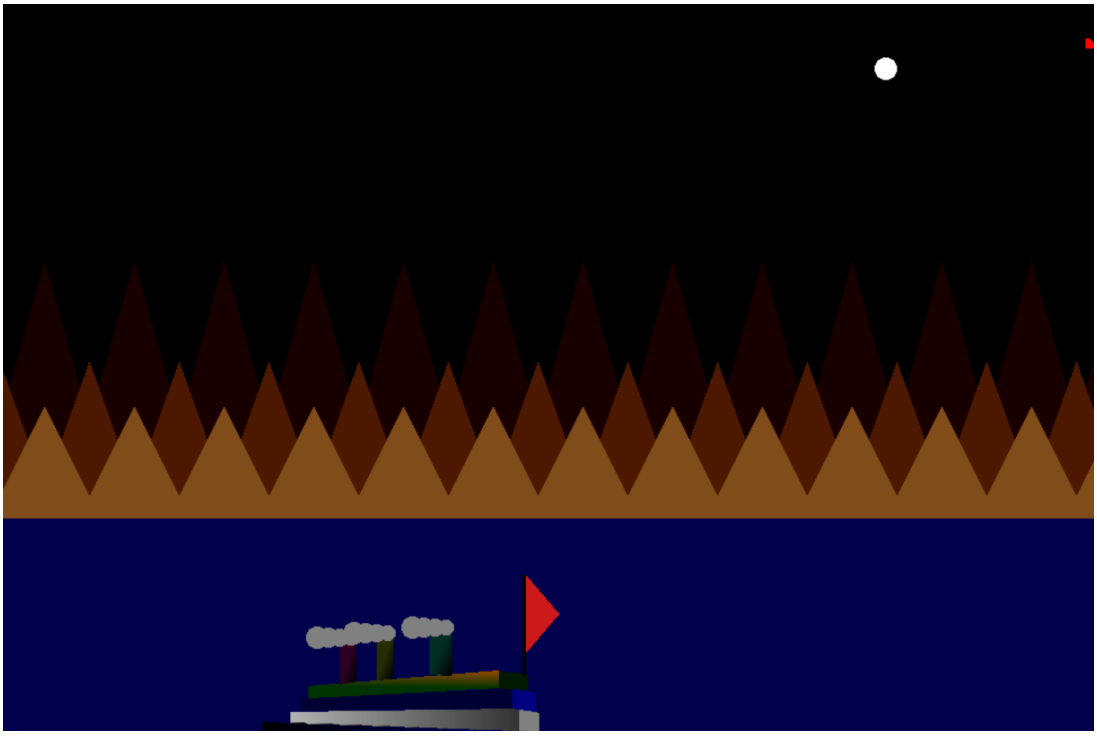
e -> Evening



n -> Night



d -> Mid Night



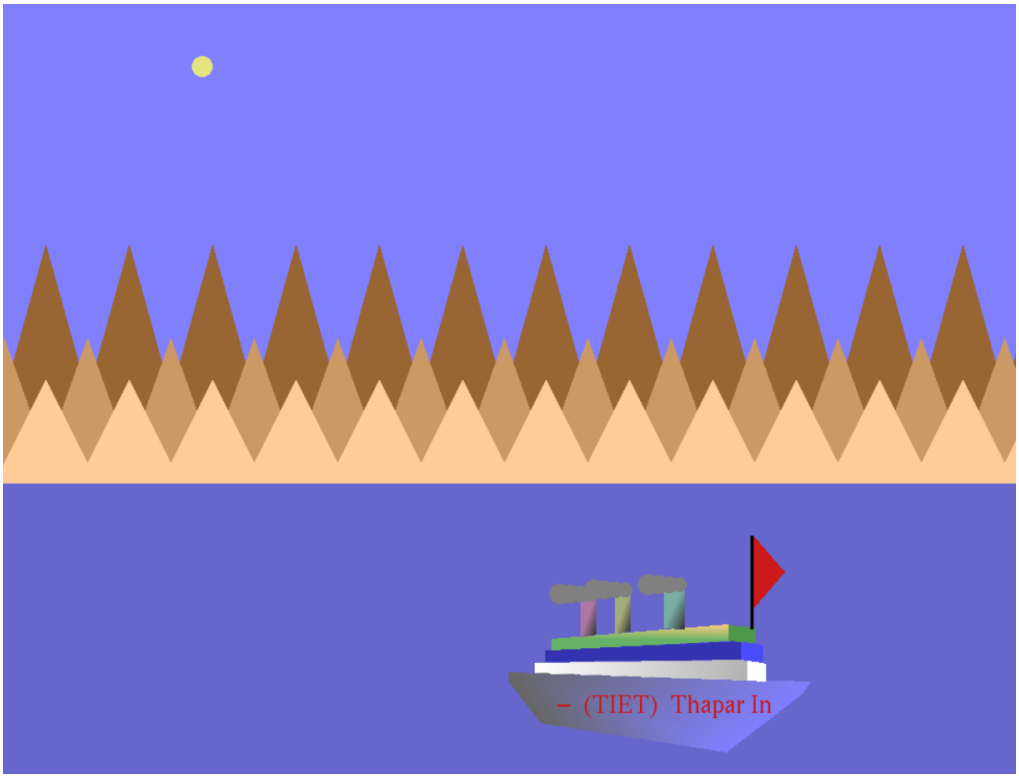
Num 6 -> Moving Ship Fwd

Num 4 -> Moving Ship backward

Esc -> Exit

Left Arrow -> Moving Ship backward

Right Arrow -> Moving Ship Forward


– (TIET) Thapar In