Complex Problem Assignment

Student Name: Mayank Singh **UID:22BCS10205**
Branch: CSE **Section/Group: 22BCS_IOT-612/B**
Semester: 6 **Date of Performance:10/04/2025**
Subject Name: Computer Graphics Lab **Subject Code: 22CSH-352**

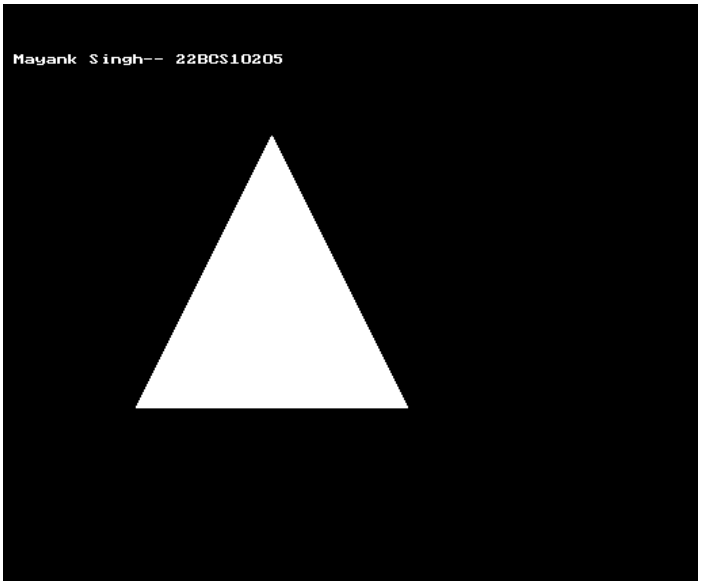
Question 1

1. **Problem Statements:** Write a program to draw and fill a triangle using the scan-line algorithm.
2. **Aim:** To understand the concept of area filling using scan- line methods.
3. **Objective:** Learn to implement scan-line polygon filling and manage edge detection.
4. **Code:**

```
#include <graphics.h>
#include <conio.h>
#include <dos.h>
void scanFill(int x[], int y[], int n) {
   int i, j, temp;
   int dx, dy;
   float slope[10];
   for (i = 0; i < n; i++) {
      dx = x[(i + 1) % n] - x[i];
      dy = y[(i + 1) % n] - y[i];
      if (dy == 0)
         slope[i] = 1.0;
      else if (dx == 0)
         slope[i] = 0.0;
      else
         slope[i] = (float)dx / dy;
   }
   for (int scanline = 0; scanline < 480; scanline++) {
      int interX[10], count = 0;
    for (i = 0; i < n; i++) {
         if (((y[i] <= scanline) && (y[(i + 1) % n] > scanline)) ||
            ((y[i] > scanline) && (y[(i + 1) % n] <= scanline))) {
            interX[count] = (int)(x[i] + slope[i] * (scanline - y[i]));
            count++;
         }
      }
          for (i = 0; i < count - 1; i++) {
         for (j = 0; j < count - 1 - i; j++) {
            if (interX[j] > interX[j + 1]) {
               temp = interX[j];
               interX[j] = interX[j + 1];
               interX[j + 1] = temp;
            }
         }
      }
```

```
        for (i = 0; i < count; i += 2) {
            line(interX[i], scanline, interX[i + 1], scanline);
        }
    }
}
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\Turboc3\\BGI");
    int x[3], y[3];
    x[0] = 200; y[0] = 100;
    x[1] = 300; y[1] = 300;
    x[2] = 100; y[2] = 300;
    line(x[0], y[0], x[1], y[1]);
    line(x[1], y[1], x[2], y[2]);
    line(x[2], y[2], x[0], y[0]);
    scanFill(x, y, 3);
    outtextxy(10, 40, "Mayank Singh-- 22BCS10205");
    getch();
    closegraph();
    return 0;
     }
```

5. Output:



Question 2

1. **Problem Statements:** Implement a Bezier curve generator to draw smooth curves by specifying four control points.
2. **Aim:** To explore curve generation techniques used in graphical applications.
3. **Objective:** Learn the mathematics behind Bezier curves and practice stepwise generation using parametric equations.

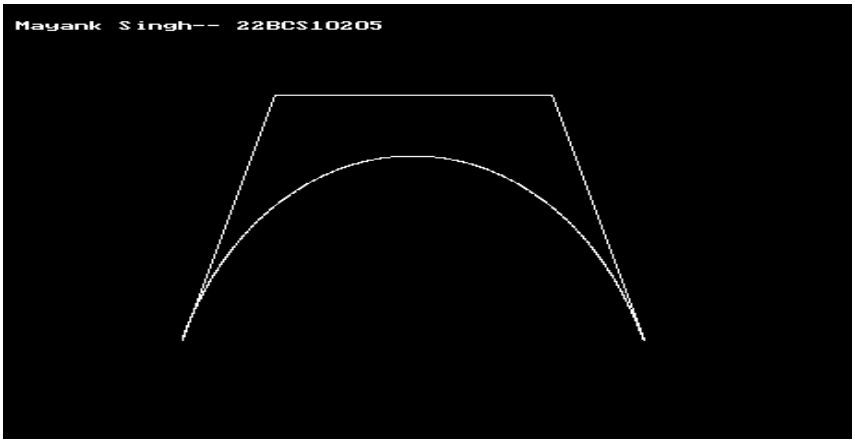
4. Code:

```
#include <graphics.h>
#include <conio.h>
#include <math.h>
#include <dos.h>

void bezier(int x[4], int y[4]) {
   float t;
   int px, py;
   for (t = 0.0; t <= 1.0; t += 0.001) {
      px = (int)((1 - t)*(1 - t)*(1 - t) * x[0] +
               3 * t*(1 - t)*(1 - t) * x[1] +
               3 * t*t*(1 - t) * x[2] +
               t*t*t * x[3]);
      py = (int)((1 - t)*(1 - t)*(1 - t) * y[0] +
               3 * t*(1 - t)*(1 - t) * y[1] +
               3 * t*t*(1 - t) * y[2] +
               t*t*t * y[3]);

      putpixel(px, py, WHITE);
      delay(1);     }
}
int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "C:\\Turboc3\\BGI");
   int x[4] = {100, 150, 300, 350};
   int y[4] = {300, 100, 100, 300};
   for (int i = 0; i < 3; i++) {
      line(x[i], y[i], x[i + 1], y[i + 1]);
   }
   bezier(x, y);
   outtextxy(10, 40, "Mayank Singh-- 22BCS10205");
   getch();
   closegraph();
   return 0;
}
```

5. Output:



Question 3

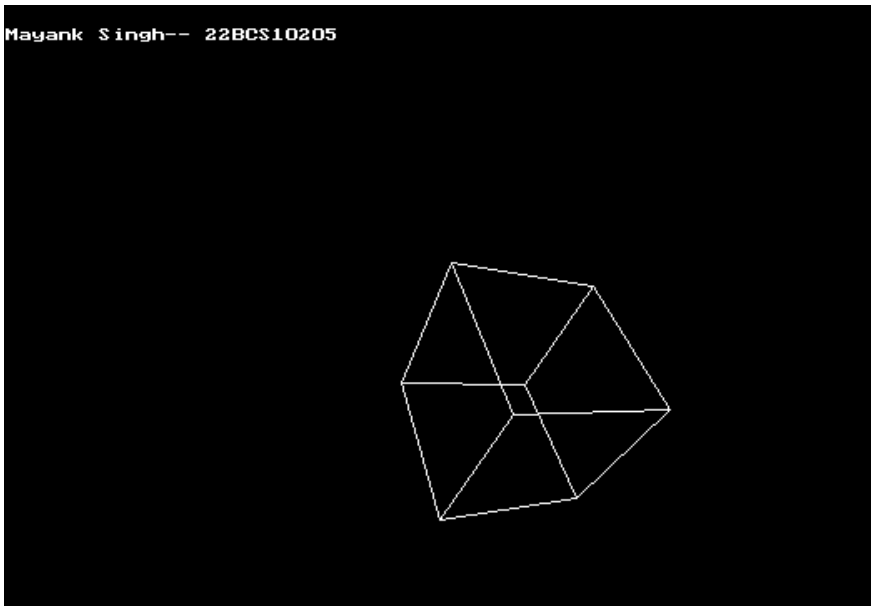
1. **Problem Statements:** Develop a program to animate a rotating cube in 3D space with perspective projection.
2. **Aim:** To simulate 3D object rotation and display perspective depth.
3. **Objective:** Understand 3D transformations, including rotation, and apply projection techniques for visualization.
4. **Code:**

```
#include <graphics.h>
#include <conio.h>
#include <math.h>
#include <dos.h>

#define PI 3.14159265
struct Point3D {
   float x, y, z;
};
const float distance = 300;
Point3D cube[8] = {
   {-50, -50, -50}, {50, -50, -50},
   {50, 50, -50}, {-50, 50, -50},
   {-50, -50, 50}, {50, -50, 50},
   {50, 50, 50}, {-50, 50, 50}
};
void rotateX(Point3D &p, float angle) {
   float rad = angle * PI / 180;
   float y = p.y;
   float z = p.z;
   p.y = y * cos(rad) - z * sin(rad);
   p.z = y * sin(rad) + z * cos(rad);
}
void rotateY(Point3D &p, float angle) {
   float rad = angle * PI / 180;
   float x = p.x;
   float z = p.z;
   p.x = x * cos(rad) + z * sin(rad);
   p.z = -x * sin(rad) + z * cos(rad);
}
void project(Point3D p, int &x2d, int &y2d) {
   float factor = distance / (distance + p.z);
   x2d = (int)(p.x * factor) + getmaxx() / 2;
   y2d = (int)(p.y * factor) + getmaxy() / 2;
}
void drawCube(Point3D p[]) {
   int edges[12][2] = {
      {0,1}, {1,2}, {2,3}, {3,0},
      {4,5}, {5,6}, {6,7}, {7,4},
      {0,4}, {1,5}, {2,6}, {3,7}
```

```
    };
    for (int i = 0; i < 12; i++) {
        int x1, y1, x2, y2;
        project(p[edges[i][0]], x1, y1);
        project(p[edges[i][1]], x2, y2);
        line(x1, y1, x2, y2);
    }
}
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\Turboc3\\BGI");
    Point3D rotated[8];
    float angleX = 0, angleY = 0;
    while (!kbhit()) {
        cleardevice();
        for (int i = 0; i < 8; i++) {
            rotated[i] = cube[i];
            rotateX(rotated[i], angleX);
            rotateY(rotated[i], angleY);
        }
        drawCube(rotated);
        outtextxy(10, 20, "Mayank Singh-- 22BCS10205");
        delay(50);
        angleX += 2;
        angleY += 3;
    }
    getch();
    closegraph();
    return 0;
  }
```

5. Output:



Question 4

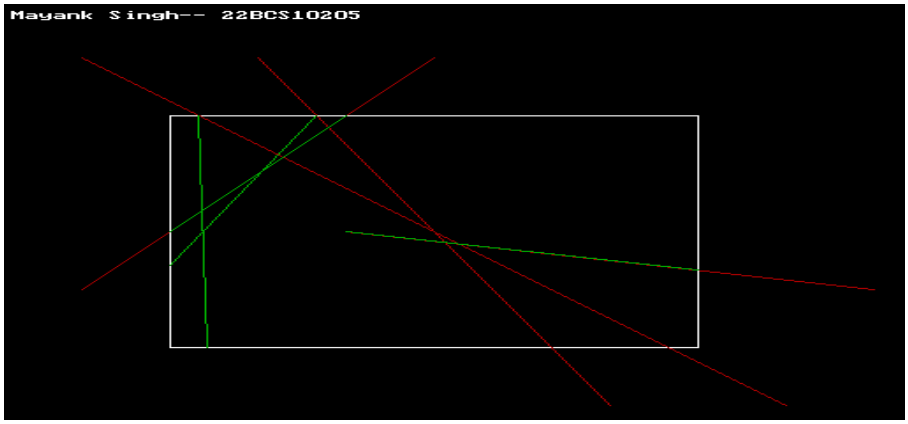
1. **Problem Statements:** Create a program to implement the Cohen-Sutherland line-clipping algorithm for a set of multiple intersecting lines.
2. **Aim:** To efficiently clip lines using a standard algorithm.
3. **Objective**: Understand and apply region codes and boundary conditions for clipping in a viewport.
4. **Code:**

```
#include <graphics.h>
#include <conio.h>
#define LEFT 1
#define RIGHT 2
#define BOTTOM 4
#define TOP 8
int xmin = 100, ymin = 100, xmax = 400, ymax = 300;
int getCode(int x, int y) {
    int code = 0;
    if (x < xmin) code |= LEFT;
    if (x > xmax) code |= RIGHT;
    if (y < ymin) code |= BOTTOM;
    if (y > ymax) code |= TOP;
    return code;
}
void cohenSutherland(int x1, int y1, int x2, int y2) {
    int code1 = getCode(x1, y1);
    int code2 = getCode(x2, y2);
    int accept = 0;
    while (1) {
        if ((code1 | code2) == 0) {
            accept = 1;
            break;
        } else if (code1 & code2) {
            break;
        } else {
            int codeOut;
            int x, y;
            codeOut = code1 ? code1 : code2;
            if (codeOut & TOP) {
                x = x1 + (x2 - x1) * (ymax - y1) / (y2 - y1);
                y = ymax;
            } else if (codeOut & BOTTOM) {
                x = x1 + (x2 - x1) * (ymin - y1) / (y2 - y1);
                y = ymin;
            } else if (codeOut & RIGHT) {
                y = y1 + (y2 - y1) * (xmax - x1) / (x2 - x1);
                x = xmax;
            } else {
                y = y1 + (y2 - y1) * (xmin - x1) / (x2 - x1);
                x = xmin;
```

```
            }

            if (codeOut == code1) {
               x1 = x;
               y1 = y;
               code1 = getCode(x1, y1);
            } else {
               x2 = x;
               y2 = y;
               code2 = getCode(x2, y2);
            }
         }
      }
      if (accept) {
         setcolor(GREEN);
         line(x1, y1, x2, y2);
      }
   }
   int main() {
      int gd = DETECT, gm;
      initgraph(&gd, &gm, "C:\\Turboc3\\BGI");
      rectangle(xmin, ymin, xmax, ymax);
      outtextxy(10, 10, "Mayank Singh-- 22BCS10205");
      setcolor(RED);
      line(50, 50, 450, 350);
      line(150, 50, 350, 350);
      line(200, 200, 500, 250);
      line(50, 250, 250, 50);
      delay(1000);
      cohenSutherland(50, 50, 450, 350);
      cohenSutherland(150, 50, 350, 350);
      cohenSutherland(200, 200, 500, 250);
      cohenSutherland(50, 250, 250, 50);
      getch();
      closegraph();
      return 0;
   }
```

5. Output:



Question 5

1. **Problem Statements:** Write a program to visualize 3D transformations such as scaling, rotation, and translation on a cube.
2. **Aim:** To implement and visualize composite transformations in 3D space.
3. **Objective**: Gain insights into homogeneous coordinate systems and concatenation of transformation matrices.
4. **Code:**

```
#include <graphics.h>
#include <conio.h>
#include <math.h>
#include <dos.h>

#define PI 3.14159265
struct Point3D {
   float x, y, z;
};
Point3D cube[8] = {
   {-50, -50, -50}, {50, -50, -50},
   {50, 50, -50}, {-50, 50, -50},
   {-50, -50, 50}, {50, -50, 50},
   {50, 50, 50}, {-50, 50, 50}
};
float transform[4][4] = {
   {1,0,0,0},
   {0,1,0,0},
   {0,0,1,0},
   {0,0,0,1}
};
Point3D multiply(Point3D p) {
   Point3D res;
   res.x = p.x * transform[0][0] + p.y * transform[1][0] + p.z * transform[2][0] + transform[3][0];
   res.y = p.x * transform[0][1] + p.y * transform[1][1] + p.z * transform[2][1] + transform[3][1];
   res.z = p.x * transform[0][2] + p.y * transform[1][2] + p.z * transform[2][2] + transform[3][2];
   return res;
}
void resetMatrix() {
   int i, j;
   for (i = 0; i < 4; i++)
      for (j = 0; j < 4; j++)
         transform[i][j] = (i == j) ? 1 : 0;
}
void applyTranslation(float tx, float ty, float tz) {
   transform[3][0] += tx;
   transform[3][1] += ty;
   transform[3][2] += tz;
}
void applyScaling(float sx, float sy, float sz) {
```

```
    transform[0][0] *= sx;
    transform[1][1] *= sy;
    transform[2][2] *= sz;
}
void applyRotationX(float angle) {
    int i, j, k;
    float rad = angle * PI / 180;
    float rot[4][4] = {
        {1, 0, 0, 0},
        {0, cos(rad), sin(rad), 0},
        {0, -sin(rad), cos(rad), 0},
        {0, 0, 0, 1}
    };
    float temp[4][4];
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++) {
            temp[i][j] = 0;
            for (k = 0; k < 4; k++)
                temp[i][j] += transform[i][k] * rot[k][j];
        }
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
            transform[i][j] = temp[i][j];
}
void applyRotationY(float angle) {
    int i, j, k;
    float rad = angle * PI / 180;
    float rot[4][4] = {
        {cos(rad), 0, -sin(rad), 0},
        {0, 1, 0, 0},
        {sin(rad), 0, cos(rad), 0},
        {0, 0, 0, 1}
    };
    float temp[4][4];
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++) {
            temp[i][j] = 0;
            for (k = 0; k < 4; k++)
                temp[i][j] += transform[i][k] * rot[k][j];
        }
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
            transform[i][j] = temp[i][j];
}
void project(Point3D p, int &x, int &y) {
    float d = 300;
    float factor = d / (d + p.z);
    x = (int)(p.x * factor + getmaxx() / 2);
    y = (int)(p.y * factor + getmaxy() / 2);
```

```
}
void drawCube(Point3D p[]) {
    int i;
    int edges[12][2] = {
        {0,1}, {1,2}, {2,3}, {3,0},
        {4,5}, {5,6}, {6,7}, {7,4},
        {0,4}, {1,5}, {2,6}, {3,7}
    };
    for (i = 0; i < 12; i++) {
        int x1, y1, x2, y2;
        project(p[edges[i][0]], x1, y1);
        project(p[edges[i][1]], x2, y2);
        line(x1, y1, x2, y2);
    }
}
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\Turboc3\\BGI");
    Point3D transformed[8];
    int frame, i;
    for (frame = 0; frame < 300; frame++) {
        cleardevice();
        resetMatrix();
        applyTranslation(0, 0, 200);
        applyRotationX(frame);
        applyRotationY(frame);
        applyScaling(1.5, 1.5, 1.5);
        for (i = 0; i < 8; i++)
            transformed[i] = multiply(cube[i]);
        drawCube(transformed);
        outtextxy(10, 10, "Mayank Singh-- 22BCS10205");
        delay(50);
    }
    getch();
    closegraph();
    return 0;
     }
```

Output: