Lab – 5

10.1. Line clipping algorithm (Cohen-Sutherland).

#include <graphics.h>

#include <conio.h>

void and\_operation(int c1[4], int c2[4], int c[4])

{

    for (int i = 0; i < 4; i++)

    {

        c[i] = (c1[i] == 0 || c2[i] == 0) ? 0 : 1;

    }

}

void or\_operation(int c1[4], int c2[4], int c[4])

{

    for (int i = 0; i < 4; i++)

    {

        c[i] = (c1[i] == 0 && c2[i] == 0) ? 0 : 1;

    }

}

void code(int x, int y, int Xmin, int Ymin, int Xmax, int Ymax, int c[4])

{

    c[0] = (y > Ymax) ? 1 : 0;

    c[1] = (y < Ymin) ? 1 : 0;

    c[2] = (x > Xmax) ? 1 : 0;

    c[3] = (x < Xmin) ? 1 : 0;

}

int main()

{

    int gd = DETECT, gm;

    initgraph(&gd, &gm, "");

    int x1 = 56, y1 = 117;

    int x2 = 425, y2 = 385;

    int Xmin = 150, Ymin = 100, Xmax = 500, Ymax = 350;

    line(x1, y1, x2, y2);

    rectangle(Xmin, Ymin, Xmax, Ymax);

    setcolor(RED);

    int c1[4];

    int c2[4];

    int c[4];

    code(x1, y1, Xmin, Ymin, Xmax, Ymax, c1);

    code(x2, y2, Xmin, Ymin, Xmax, Ymax, c2);

    or\_operation(c1, c2, c);

    int sum = 0;

    for (int i = 0; i < 4; i++)

    {

        sum += c[i];

    }

    if (sum == 0)

    {

        line(x1, y1, x2, y2);

    }

    else

    {

        int cc[4];

        and\_operation(c1, c2, cc);

        sum = 0;

        for (int i = 0; i < 4; i++)

        {

            sum += cc[i];

        }

        if (sum == 0)

        {

            float m = (float)(y2 - y1) / (x2 - x1);

            int x = x1, y = y1, xx = x2, yy = y2;

            if (c[0] == 1)

            {

                xx = x1 + (Ymax - y1) / m;

                yy = Ymax;

            }

            if (c[1] == 1)

            {

                x = x1 + (Ymin - y1) / m;

                y = Ymin;

            }

            if (c[2] == 1)

            {

                yy = y1 + (Xmax - x1) \* m;

                xx = Xmax;

            }

            if (c[3] == 1)

            {

                y = y1 + (Xmin - x1) \* m;

                x = Xmin;

            }

            line(x, y, xx, yy);

            line(x + 1, y + 1, xx + 1, yy + 1);

            line(x - 1, y - 1, xx - 1, yy - 1);

        }

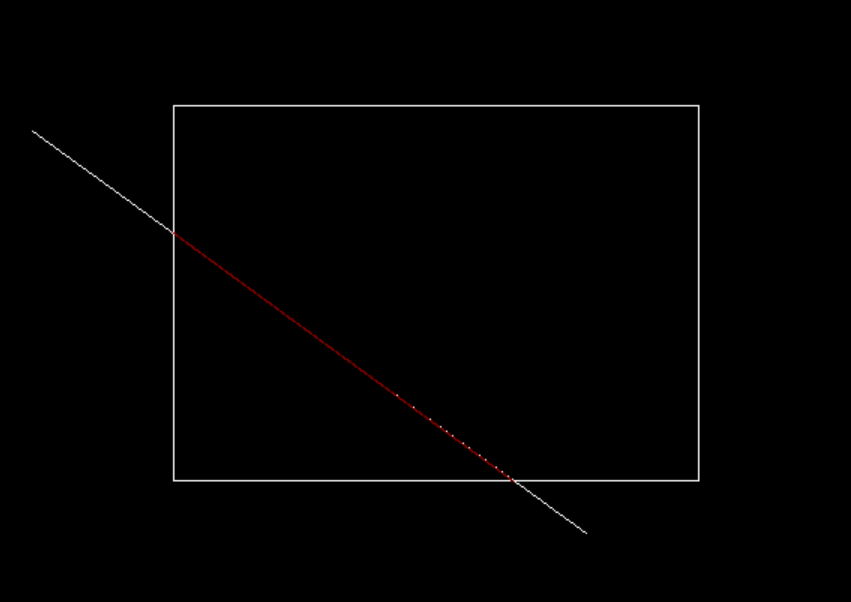
    }

    getch();

    closegraph();

    return 0;

}



10.2. Line clipping algorithm (Liang-Barsky).

#include <graphics.h>

#include <conio.h>

float max(float a, float b)

{

    return (a > b) ? a : b;

}

float min(float a, float b)

{

    return (a < b) ? a : b;

}

int main()

{

    int gd = DETECT, gm;

    initgraph(&gd, &gm, "");

    int x1 = 56, y1 = 117;

    int x2 = 425, y2 = 385;

    int Xmin = 150, Ymin = 100, Xmax = 500, Ymax = 350;

    line(x1, y1, x2, y2);

    rectangle(Xmin, Ymin, Xmax, Ymax);

    setcolor(RED);

    float p[4];

    p[0] = x1 - x2;

    p[1] = x2 - x1;

    p[2] = y1 - y2;

    p[3] = y2 - y1;

    float q[4];

    q[0] = x1 - Xmin;

    q[1] = Xmax - x1;

    q[2] = y1 - Ymin;

    q[3] = Ymax - y1;

    float t1 = 0, t2 = 1;

    int flag = 1;

    for (int i = 0; i < 4; i++)

    {

        if (p[i] == 0)

        {

            printf("\nLine is parallel");

            for (int i = 0; i < 4; i++)

            {

                if (q[i] < 0)

                {

                    flag = 0;

                    printf("\nLine is Completely Outside");

                    break;

                }

            }

            break;

        }

        else

        {

            float t = q[i] / p[i];

            if (p[i] < 0)

            {

                t1 = max(t1, t);

            }

            if (p[i] > 0)

            {

                t2 = min(t2, t);

            }

        }

    }

    if (flag)

    {

        if (t1 < t2)

        {

            int x = x1, y = y1, xx = x2, yy = y2;

            if (t1 != 0)

            {

                x = x1 + t1 \* (x2 - x1);

                y = y1 + t1 \* (y2 - y1);

            }

            if (t2 != 1)

            {

                xx = x1 + t2 \* (x2 - x1);

                yy = y1 + t2 \* (y2 - y1);

            }

            line(x, y, xx, yy);

            line(x + 1, y + 1, xx + 1, yy + 1);

            line(x - 1, y - 1, xx - 1, yy - 1);

        }

        else if (t1 > t2)

        {

            printf("\nLine is Outside the window");

        }

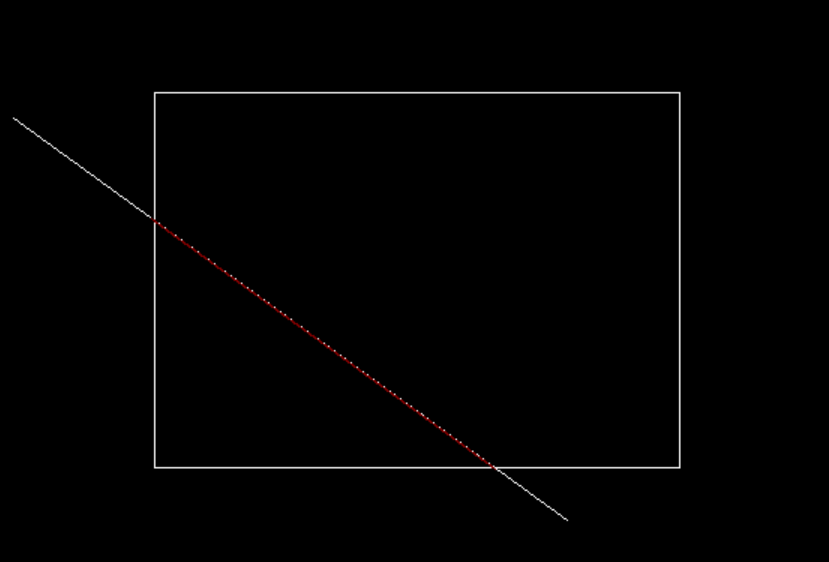
    }

    getch();

    closegraph();

    return 0;

}



11. ⁠create any one cartoon character and apply any one transformation technique.

#include <stdio.h>

#include <graphics.h>

#include <math.h>

void drawFace(int x, int y, int radius)

{

    setcolor(YELLOW);

    setfillstyle(SOLID\_FILL, YELLOW);

    fillellipse(x, y, radius, radius);

    setcolor(WHITE);

    setfillstyle(SOLID\_FILL, WHITE);

    fillellipse(x - radius / 3, y - radius / 3, radius / 6, radius / 6);

    fillellipse(x + radius / 3, y - radius / 3, radius / 6, radius / 6);

    setcolor(BLACK);

    setfillstyle(SOLID\_FILL, BLACK);

    fillellipse(x - radius / 3, y - radius / 3, radius / 12, radius / 12);

    fillellipse(x + radius / 3, y - radius / 3, radius / 12, radius / 12);

    setcolor(RED);

    arc(x, y + radius / 4, 200, 340, radius / 3);

}

void scaleFace(int x, int y, int radius, float scaleX, float scaleY)

{

    int newRadiusX = radius \* scaleX;

    int newRadiusY = radius \* scaleY;

    drawFace(x, y, newRadiusX < newRadiusY ? newRadiusX : newRadiusY);

}

int main()

{

*// Translation & Scaling*

    int gd = DETECT, gm;

    initgraph(&gd, &gm, "");

    int x = 300, y = 200, radius = 100;

    float sx = 0.5, sy = 0.6;

    int tx = 250, ty = 80;

    drawFace(x, y, radius);

    scaleFace(x + tx, y + ty, radius, sx, sy);

    getch();

    closegraph();

    return 0;

}

