Computer Graphics Laboratory Manual

Contents

Lab 6: Cohen–Sutherland Line Clipping

Aim

To implement Cohen–Sutherland line clipping algorithm in Python and display the clipped line.

Theory

Cohen–Sutherland algorithm clips a line to a rectangular window by assigning region codes to the line endpoints. Based on these codes, the line is accepted, rejected, or clipped until it lies inside the window.

Part A: Basic Program (Terminal Output)

Program

```
# Cohen-Sutherland Line Clipping (Simple)

INSIDE = 0
LEFT = 1
RIGHT = 2
BOTTOM = 4
TOP = 8

def find_code(x, y, xmin, ymin, xmax, ymax):
    code = INSIDE

    if x < xmin:
        code = code + LEFT
    elif x > xmax:
        code = code + RIGHT

    if y < ymin:
        code = code + BOTTOM
    elif y > ymax:
        code = code + TOP

    return code


def cohen_sutherland(x1, y1, x2, y2, xmin, ymin, xmax, ymax):

    code1 = find_code(x1, y1, xmin, ymin, xmax, ymax)
    code2 = find_code(x2, y2, xmin, ymin, xmax, ymax)

    while True:
```

```
        if code1 == 0 and code2 == 0:
            print("Clipped Line:", x1, y1, x2, y2)
            break

        elif (code1 & code2) != 0:
            print("Line is outside the window")
            break

        else:
            if code1 != 0:
                code_out = code1
            else:
                code_out = code2

            if code_out & TOP:
                x = x1 + (x2-x1)*(ymax-y1)/(y2-y1)
                y = ymax

            elif code_out & BOTTOM:
                x = x1 + (x2-x1)*(ymin-y1)/(y2-y1)
                y = ymin

            elif code_out & RIGHT:
                y = y1 + (y2-y1)*(xmax-x1)/(x2-x1)
                x = xmax

            elif code_out & LEFT:
                y = y1 + (y2-y1)*(xmin-x1)/(x2-x1)
                x = xmin

            if code_out == code1:
                x1 = x
                y1 = y
                code1 = find_code(x1, y1, xmin, ymin, xmax, ymax)
            else:
                x2 = x
                y2 = y
                code2 = find_code(x2, y2, xmin, ymin, xmax, ymax)


xmin = 10
ymin = 10
xmax = 100
ymax = 100

cohen_sutherland(0, 0, 120, 120, xmin, ymin, xmax, ymax)
```

Result

The clipped line coordinates are printed in the terminal.

Part B: Program with Visualization

Program

```
# Cohen-Sutherland with Visualization (Simple)

import matplotlib.pyplot as plt

INSIDE = 0
LEFT = 1
RIGHT = 2
BOTTOM = 4
TOP = 8

def find_code(x, y, xmin, ymin, xmax, ymax):
    code = INSIDE

    if x < xmin:
        code = code + LEFT
    elif x > xmax:
        code = code + RIGHT

    if y < ymin:
        code = code + BOTTOM
    elif y > ymax:
        code = code + TOP

    return code


def cohen_sutherland(x1, y1, x2, y2, xmin, ymin, xmax, ymax):

    code1 = find_code(x1, y1, xmin, ymin, xmax, ymax)
    code2 = find_code(x2, y2, xmin, ymin, xmax, ymax)

    while True:

        if code1 == 0 and code2 == 0:
            return x1, y1, x2, y2

        elif (code1 & code2) != 0:
            return None

        else:
```

```
            if code1 != 0:
                code_out = code1
            else:
                code_out = code2

            if code_out & TOP:
                x = x1 + (x2-x1)*(ymax-y1)/(y2-y1)
                y = ymax

            elif code_out & BOTTOM:
                x = x1 + (x2-x1)*(ymin-y1)/(y2-y1)
                y = ymin

            elif code_out & RIGHT:
                y = y1 + (y2-y1)*(xmax-x1)/(x2-x1)
                x = xmax

            elif code_out & LEFT:
                y = y1 + (y2-y1)*(xmin-x1)/(x2-x1)
                x = xmin

            if code_out == code1:
                x1 = x
                y1 = y
                code1 = find_code(x1, y1, xmin, ymin, xmax, ymax)
            else:
                x2 = x
                y2 = y
                code2 = find_code(x2, y2, xmin, ymin, xmax, ymax)


def draw(original, clipped, xmin, ymin, xmax, ymax):

    plt.plot([xmin, xmax, xmax, xmin, xmin],
             [ymin, ymin, ymax, ymax, ymin])

    x1,y1,x2,y2 = original
    plt.plot([x1,x2],[y1,y2],'--')

    if clipped != None:
        cx1,cy1,cx2,cy2 = clipped
        plt.plot([cx1,cx2],[cy1,cy2])

    plt.title("Cohen-Sutherland Line Clipping")
    plt.axis("equal")
    plt.grid(True)
    plt.show()
```

```
xmin = 10
ymin = 10
xmax = 100
ymax = 100

original = (0, 0, 120, 120)
clipped = cohen_sutherland(0, 0, 120, 120, xmin, ymin, xmax, ymax)
draw(original, clipped, xmin, ymin, xmax, ymax)
```

Lab 7: Liang–Barsky Line Clipping

Aim

To implement Liang–Barsky line clipping algorithm in Python and display the clipped line.

Theory

Liang–Barsky algorithm uses the parametric equation of a line and finds the visible part of the line inside a rectangular window using simple calculations.

Part A: Basic Program (Terminal Output)

Program

```
# Liang-Barsky Line Clipping (Simple)

def liang_barsky(x1, y1, x2, y2, xmin, ymin, xmax, ymax):

    dx = x2 - x1
    dy = y2 - y1

    u1 = 0
    u2 = 1

    p = [-dx, dx, -dy, dy]
    q = [x1-xmin, xmax-x1, y1-ymin, ymax-y1]

    i = 0
    while i < 4:

        if p[i] == 0:
            if q[i] < 0:
                print("Line is outside the window")
                return
        else:
            t = q[i] / p[i]

            if p[i] < 0:
                if t > u1:
                    u1 = t
            else:
                if t < u2:
                    u2 = t
        i = i + 1

    if u1 > u2:
```

```
        print("Line is outside the window")
        return

    nx1 = x1 + u1 * dx
    ny1 = y1 + u1 * dy
    nx2 = x1 + u2 * dx
    ny2 = y1 + u2 * dy

    print("Clipped Line:", nx1, ny1, nx2, ny2)


xmin = 10
ymin = 10
xmax = 100
ymax = 100

liang_barsky(0, 0, 120, 120, xmin, ymin, xmax, ymax)
```

Part B: Program with Visualization

Program

```
# Liang-Barsky with Visualization (Simple)

import matplotlib.pyplot as plt

def liang_barsky(x1, y1, x2, y2, xmin, ymin, xmax, ymax):

    dx = x2 - x1
    dy = y2 - y1

    u1 = 0
    u2 = 1

    p = [-dx, dx, -dy, dy]
    q = [x1-xmin, xmax-x1, y1-ymin, ymax-y1]

    i = 0
    while i < 4:

        if p[i] == 0:
            if q[i] < 0:
                return None
        else:
            t = q[i] / p[i]

            if p[i] < 0:
```

```
                if t > u1:
                    u1 = t
            else:
                if t < u2:
                    u2 = t
        i = i + 1

    if u1 > u2:
        return None

    nx1 = x1 + u1 * dx
    ny1 = y1 + u1 * dy
    nx2 = x1 + u2 * dx
    ny2 = y1 + u2 * dy

    return nx1, ny1, nx2, ny2


def draw(original, clipped, xmin, ymin, xmax, ymax):

    plt.plot([xmin, xmax, xmax, xmin, xmin],
             [ymin, ymin, ymax, ymax, ymin])

    x1,y1,x2,y2 = original
    plt.plot([x1,x2],[y1,y2],'--')

    if clipped != None:
        cx1,cy1,cx2,cy2 = clipped
        plt.plot([cx1,cx2],[cy1,cy2])

    plt.title("Liang-Barsky Line Clipping")
    plt.axis("equal")
    plt.grid(True)
    plt.show()


xmin = 10
ymin = 10
xmax = 100
ymax = 100

original = (0, 0, 120, 120)
clipped = liang_barsky(0, 0, 120, 120, xmin, ymin, xmax, ymax)
draw(original, clipped, xmin, ymin, xmax, ymax)
```