Computer Graphics Assignment-3

Clipping

Name-Ashish Goyal

Id-2016ucp1100

Batch-A (1, 2)

Cohen Sutherland Line clip Algorithm

Code:

```
from graphics import *
##############################
#   less_one   slope -1 to 1


boundary=[]
color="black"
color1="red"  #correct
color2="blue" #clipped
def zero_to_one(x0,y0,x1,y1,color):  #slope 0 to 1
    a=y1-y0
    b=-1*(x1-x0)
    di=2*a+b
    dne=2*(a+b)
    de=2*a
```

```
    y=y0
    for x in range(x0,x1,1):
        #pixel=Point(x,y)
        #pixel.draw(win_obj,"red")
        pt = Point(x,y)
        pt.setOutline(color)
        pt.draw(win_obj)
        #time.sleep(0.01)
        if(di>0):
            y=y+1
            di=di+dne
        else:
            di=di+de
        boundary.append([x,y])
        print("point"+"["+str(x)+","+str(y)+"]")
def zero_to_n_one(x0,y0,x1,y1,color): #slope -1 to 0
    a=y1-y0
    b=-1*(x1-x0)
    di=2*a-b
    dne=2*(a-b)
    de=2*a
    y=y0
    for x in range(x0,x1,1):
        #pixel=Point(x,y)
        #pixel.draw(win_obj,"red")
```

```
        pt = Point(x,y)
        pt.setOutline(color)
        pt.draw(win_obj)
        #time.sleep(0.01)
        if(di>0):
            di=di+de
        else:
            y=y-1
            di=di+dne
        boundary.append([x,y])
        print("point"+"["+str(x)+","+str(y)+"]")
#################################
#  greater_one   slope <-1 and >1
def pure_greater_one(x0,y0,x1,y1,color):  #slope >1
    b=-1*(y1-y0)
    a=x1-x0
    dne=2*(a+b)
    de=2*a
    di=2*a+b
    x=x0
    for y in range(y0,y1,1):
        #pixel=Point(x,y)
        #pixel.draw(win_obj,"red")
        pt = Point(x,y)
        pt.setOutline(color)
```

```
        pt.draw(win_obj)
        #time.sleep(0.01)
        if(di>0):
            x=x+1
            di=di+dne
        else:
            di=di+de
        boundary.append([x,y])
        print("point"+"["+str(x)+","+str(y)+"]")
def less_negative_one(x0,y0,x1,y1,color):  #slope <-1
    a=x1-x0
    b=-1*(y1-y0)
    di=2*a-b
    dne=2*(a-b)
    de=2*a
    x=x0
    for y in range(y0,y1,1):
        #pixel=Point(x,y)
        #pixel.draw(win_obj,"red")
        pt = Point(x,y)
        pt.setOutline(color)
        pt.draw(win_obj)
        #time.sleep(0.01)
        if(di>0):
            di=di+de
```

```
        else:
            x=x-1
            di=di+dne
        boundary.append([x,y])
        print("point"+"["+str(x)+","+str(y)+"]")
################################

def less_one(x0,y0,x1,y1,color):  #slope -1 to 1
    a=y1-y0
    b=-1*(x1-x0)
    if(a<0):
        zero_to_n_one(x0,y0,x1,y1,color)
    else:
        zero_to_one(x0,y0,x1,y1,color)

#the greater_one cases are mirror image of less_one cases so simply replace x and y

def greater_one(x0,y0,x1,y1,color):  #slope > 1 and <-1
    a=x1-x0
    b=-1*(y1-y0)
    if(a<0):
        less_negative_one(x0,y0,x1,y1,color)
    else:
        pure_greater_one(x0,y0,x1,y1,color)
```

```
def helper(x0,y0,x1,y1,color):
    boundary.append([x0,y0])
    boundary.append([x1,y1])

    if(abs(x0-x1)<abs(y0-y1)):              #slope > 1 and <-1
        if(y1>y0):
            greater_one(x0,y0,x1,y1,color)
        else:
            greater_one(x1,y1,x0,y0,color)
    else:
        if(x1>x0):                          #slope -1 to 1
            less_one(x0,y0,x1,y1,color)
        else:                               #we always increase x by 1 therefore start point should always less, so swap both points
            less_one(x1,y1,x0,y0,color)



##############################################################################################################

INSIDE = 0  #0000
LEFT = 1    #0001
RIGHT = 2   #0010
BOTTOM = 4  #0100
TOP = 8     #1000
```

```
def computeOutcode(x, y):
    outcode = 0
    if x < xmin:
        outcode =outcode | LEFT
    elif x > xmax:
        outcode =outcode | RIGHT
    if y < ymin:
        outcode =outcode | BOTTOM
    elif y > ymax:
        outcode =outcode | TOP

    return outcode

def clipping(x0,y0,x1,y1):
    outcode0=computeOutcode(x0,y0)
    outcode1=computeOutcode(x1,y1)

    done=False
    accept=False
    while True:
        if (outcode0==0 and outcode1==0):#trivial accept
            accept=True
            break
        elif (outcode0 & outcode1):#trivial reject
            done=True
```

```
    break
else:
    if outcode0>outcode1:
        outcodeOut=outcode0
    else:
        outcodeOut=outcode1
    if outcodeOut & TOP:           # point  above the clip ( diagram)
        x = x0 + (x1 - x0) *(ymax - y0) / (y1 - y0)
        y = ymax

    elif outcodeOut & BOTTOM:    # point  below the clip
        x = x0 + (x1 - x0) *(ymin - y0) / (y1 - y0)
        y = ymin

    elif outcodeOut & RIGHT:  # point  right of the clip
        y = y0 + (y1 - y0) * (xmax - x0) / (x1 - x0)
        x = xmax

    elif outcodeOut & LEFT:     # point left of the clip
        y = y0 + (y1 - y0) *(xmin - x0) / (x1 - x0)
        x = xmin

    #intersection point x,y
    #  replace o/s point with intersection
    if outcodeOut == outcode1:
```

```
            x1 = x
            y1 = y
            outcode1 = computeOutcode(x1, y1)
        else:
            x0 = x
            y0 = y
            outcode0 = computeOutcode(x0,y0)
    if accept:
        helper(int(x0),int(y0),int(x1),int(y1),color1)
    if done:
        #print("Boo")
        helper(int(x0),int(y0),int(x1),int(y1),color2)
```

```
##########################################################################################################
```

```
#A GraphWin object represents a window on the screen
```

```
t=4#sides of rectangle
lista=[]

xmin=int(input("enter xmin coordinate of window:"))
ymin=int(input("enter ymin coordinate of window:"))
xmax=int(input("enter xmax coordinate of window:"))
ymax=int(input("enter ymax coordinate of window:"))



x0=int(input("enter x0 coordinate of line to be clipped:"))
y0=int(input("enter y0 coordinate of line to be clipped:"))
x1=int(input("enter x1 coordinate of line to be clipped:"))
y1=int(input("enter y1 coordinate of line to be clipped:"))



lista.append([xmin,ymin])
lista.append([xmax,ymin])
lista.append([xmax,ymax])
lista.append([xmin,ymax])
list_ini_point=lista[0]
#x0=list_ini_point[0]
```

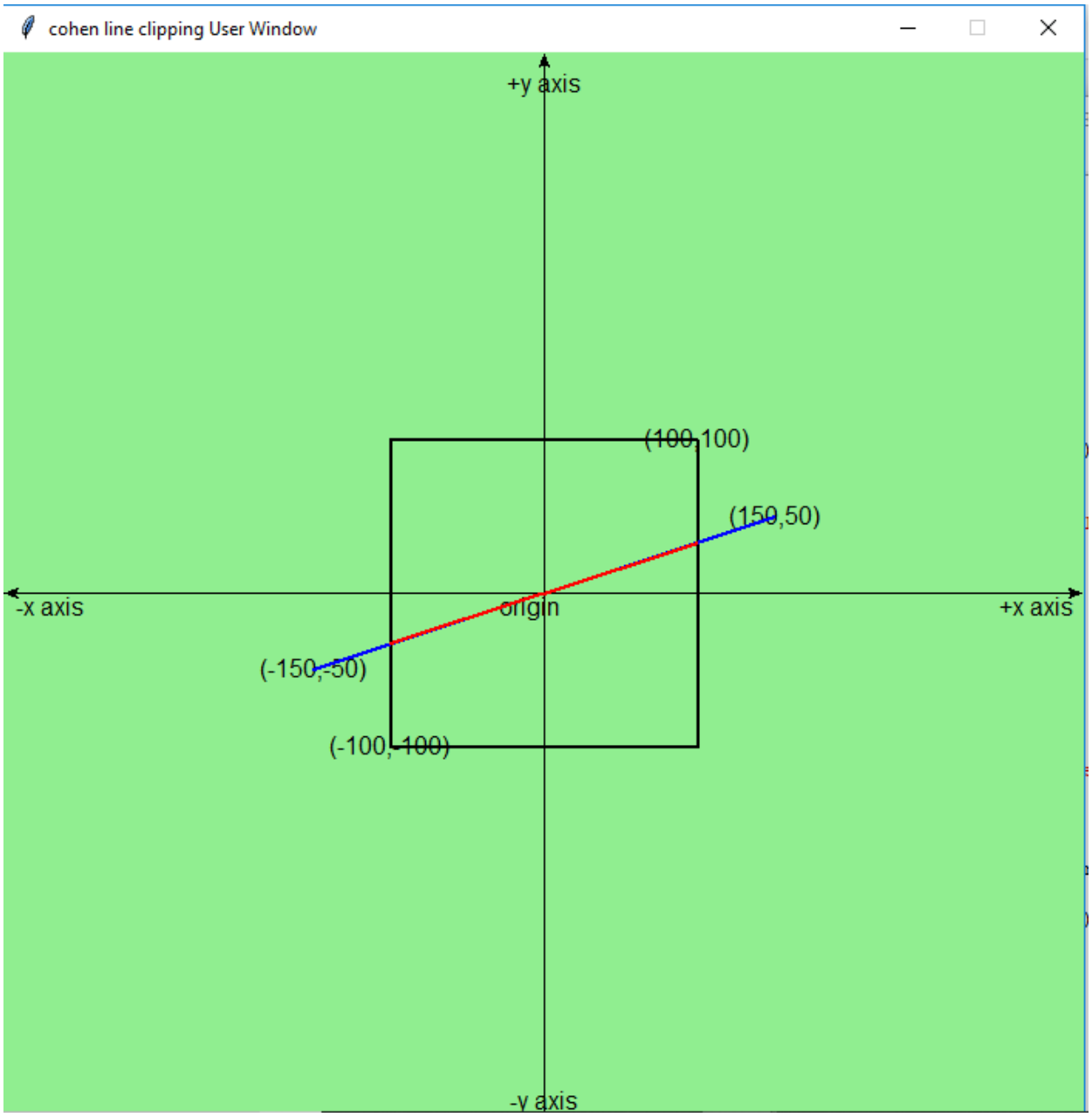
```
#y0=list_ini_point[1]

lista.append(list_ini_point)

#x01=x0

#y01=y0



win_obj=GraphWin("cohen line clipping User Window",700,700) #set viewport size  700,700 are device coordinates

win_obj.setBackground("Light Green")

win_obj.setCoords(-350,-350,350,350) #set window  use coordinates are set

x_axis=Line(Point(-350,0),Point(350,0))   #obj for x axis

y_axis=Line(Point(0,-350),Point(0,350))    #obj for y axis


x_axis.setOutline("Black")

y_axis.setOutline("Black")

x_axis.setArrow('both')

y_axis.setArrow('both')

x_axis.draw(win_obj)

y_axis.draw(win_obj)


info_x=Text(Point(320,-10),"+x axis")

info_x.draw(win_obj)

info_nx=Text(Point(-320,-10),"-x axis")

info_nx.draw(win_obj)

info_y=Text(Point(0,330),"+y axis")
```

```
info_y.draw(win_obj)
info_ny=Text(Point(0,-330),"-y axis")
info_ny.draw(win_obj)

origin=Text(Point(-10,-10),"origin")
origin.draw(win_obj)

#for rectangle
initial_point=Text(Point(xmin,ymin),"("+str(xmin)+","+str(ymin)+")")
initial_point.draw(win_obj)
final_point=Text(Point(xmax,ymax),"("+str(xmax)+","+str(ymax)+")")
final_point.draw(win_obj)



#for line
initial_point=Text(Point(x0,y0),"("+str(x0)+","+str(y0)+")")
initial_point.draw(win_obj)
final_point=Text(Point(x1,y1),"("+str(x1)+","+str(y1)+")")
final_point.draw(win_obj)

#edge_table=[]
j=0
while (j!=t):
```

```
    list_t1=lista[j]
    list_t2=lista[j+1]
    x01=list_t1[0]
    y01=list_t1[1]
    x02=list_t2[0]
    y02=list_t2[1]
    helper(x01,y01,x02,y02,color)
    j=j+1
#print(edge_table)
#scanline()
helper(x0,y0,x1,y1,color2)
clipping(x0,y0,x1,y1)


win_obj.getMouse()
win_obj.close()
```

Example1:

```
================ RESTART: C:\Users\Ashish\Desktop\py\cohen.py
enter xmin coordinate of window:-100
enter ymin coordinate of window:-100
enter xmax coordinate of window:100
enter ymax coordinate of window:100
enter x0 coordinate of line to be clipped:-150
enter y0 coordinate of line to be clipped:-50
enter x1 coordinate of line to be clipped:150
enter y1 coordinate of line to be clipped:50
```


cohen line clipping User Window
+y axis
(100,100)
(150,50)
-x axis
origin
+x axis
(-150,-50)
(-100,-100)
-y axis

Parametric line clip Algorithm

Code:

```
from graphics import *
##############################
#   less_one   slope -1 to 1


boundary=[]
color="black"
color1="red"  #correct
color2="blue" #clipped
def zero_to_one(x0,y0,x1,y1,color):  #slope 0 to 1
    a=y1-y0
    b=-1*(x1-x0)
    di=2*a+b
    dne=2*(a+b)
    de=2*a
    y=y0
    for x in range(x0,x1,1):
        #pixel=Point(x,y)
        #pixel.draw(win_obj,"red")
        pt = Point(x,y)
        pt.setOutline(color)
        pt.draw(win_obj)
        #time.sleep(0.01)
        if(di>0):
```

```
            y=y+1
            di=di+dne
        else:
            di=di+de
        boundary.append([x,y])
        print("point"+"["+str(x)+","+str(y)+"]")
def zero_to_n_one(x0,y0,x1,y1,color): #slope -1 to 0
    a=y1-y0
    b=-1*(x1-x0)
    di=2*a-b
    dne=2*(a-b)
    de=2*a
    y=y0
    for x in range(x0,x1,1):
        #pixel=Point(x,y)
        #pixel.draw(win_obj,"red")
        pt = Point(x,y)
        pt.setOutline(color)
        pt.draw(win_obj)
        #time.sleep(0.01)
        if(di>0):
            di=di+de
        else:
            y=y-1
            di=di+dne
```

```
        boundary.append([x,y])
        print("point"+"["+str(x)+","+str(y)+"]")
################################
#  greater_one   slope <-1 and >1
def pure_greater_one(x0,y0,x1,y1,color):  #slope >1
    b=-1*(y1-y0)
    a=x1-x0
    dne=2*(a+b)
    de=2*a
    di=2*a+b
    x=x0
    for y in range(y0,y1,1):
        #pixel=Point(x,y)
        #pixel.draw(win_obj,"red")
        pt = Point(x,y)
        pt.setOutline(color)
        pt.draw(win_obj)
        #time.sleep(0.01)
        if(di>0):
            x=x+1
            di=di+dne
        else:
            di=di+de
        boundary.append([x,y])
        print("point"+"["+str(x)+","+str(y)+"]")
```

```
def less_negative_one(x0,y0,x1,y1,color):  #slope <-1
    a=x1-x0
    b=-1*(y1-y0)
    di=2*a-b
    dne=2*(a-b)
    de=2*a
    x=x0
    for y in range(y0,y1,1):
        #pixel=Point(x,y)
        #pixel.draw(win_obj,"red")
        pt = Point(x,y)
        pt.setOutline(color)
        pt.draw(win_obj)
        #time.sleep(0.01)
        if(di>0):
            di=di+de
        else:
            x=x-1
            di=di+dne
        boundary.append([x,y])
        print("point"+"["+str(x)+","+str(y)+"]")
################################

def less_one(x0,y0,x1,y1,color):  #slope -1 to 1
    a=y1-y0
```

```
    b=-1*(x1-x0)
    if(a<0):
        zero_to_n_one(x0,y0,x1,y1,color)
    else:
        zero_to_one(x0,y0,x1,y1,color)

#the greater_one cases are mirror image of less_one cases so simply replace x and y

def greater_one(x0,y0,x1,y1,color):  #slope > 1 and <-1
    a=x1-x0
    b=-1*(y1-y0)
    if(a<0):
        less_negative_one(x0,y0,x1,y1,color)
    else:
        pure_greater_one(x0,y0,x1,y1,color)



def helper(x0,y0,x1,y1,color):
    boundary.append([x0,y0])
    boundary.append([x1,y1])

    if(abs(x0-x1)<abs(y0-y1)):              #slope > 1 and <-1
        if(y1>y0):
            greater_one(x0,y0,x1,y1,color)
        else:
```

```
        greater_one(x1,y1,x0,y0,color)
    else:
        if(x1>x0):                  #slope -1 to 1
            less_one(x0,y0,x1,y1,color)
        else:                       #we always increase x by 1 therefore start point should always less, so swap both points
            less_one(x1,y1,x0,y0,color)



##############################################################################################################

def parameter():
    te=0.0
    tl=1.0
    diff_p_x=x1-x0
    diff_p_y=y1-y0



    #case1
    nd=-1*diff_p_x

    if nd!=0:
        temp=(1)*(x0-xmin)/nd
        if(nd>0):
            tl=min(temp,tl)
        else:
```

```
        te=max(te,temp)
#case2
nd=1*diff_p_x
if nd!=0:
    temp=(-1)*(x0-xmax)/nd
    if(nd>0):
        tl=min(temp,tl)
    else:
        te=max(te,temp)
#case3
nd=-1*diff_p_y
if nd!=0:
    temp=(1)*(y0-ymin)/nd
    if(nd>0):
        tl=min(temp,tl)
    else:
        te=max(te,temp)
#case4
nd=1*diff_p_y
if nd!=0:
    temp=(-1)*(y0-ymax)/nd
    if(nd>0):
        tl=min(temp,tl)
    else:
        te=max(te,temp)
```

```
    if te>tl:
        return


    a=x0+(x1-x0)*te
    b=y0+(y1-y0)*te

    c=x0+(x1-x0)*tl
    d=y0+(y1-y0)*tl

    helper(int(a),int(b),int(c),int(d),color1)



################################################################################
######################

#A GraphWin object represents a window on the screen



t=4#sides of rectangle
lista=[]
```

```
xmin=int(input("enter xmin coordinate of window:"))
ymin=int(input("enter ymin coordinate of window:"))
xmax=int(input("enter xmax coordinate of window:"))
ymax=int(input("enter ymax coordinate of window:"))



x0=int(input("enter x0 coordinate of line to be clipped:"))
y0=int(input("enter y0 coordinate of line to be clipped:"))
x1=int(input("enter x1 coordinate of line to be clipped:"))
y1=int(input("enter y1 coordinate of line to be clipped:"))



lista.append([xmin,ymin])
lista.append([xmax,ymin])
lista.append([xmax,ymax])
lista.append([xmin,ymax])
list_ini_point=lista[0]
#x0=list_ini_point[0]
#y0=list_ini_point[1]
lista.append(list_ini_point)
#x01=x0
#y01=y0
```

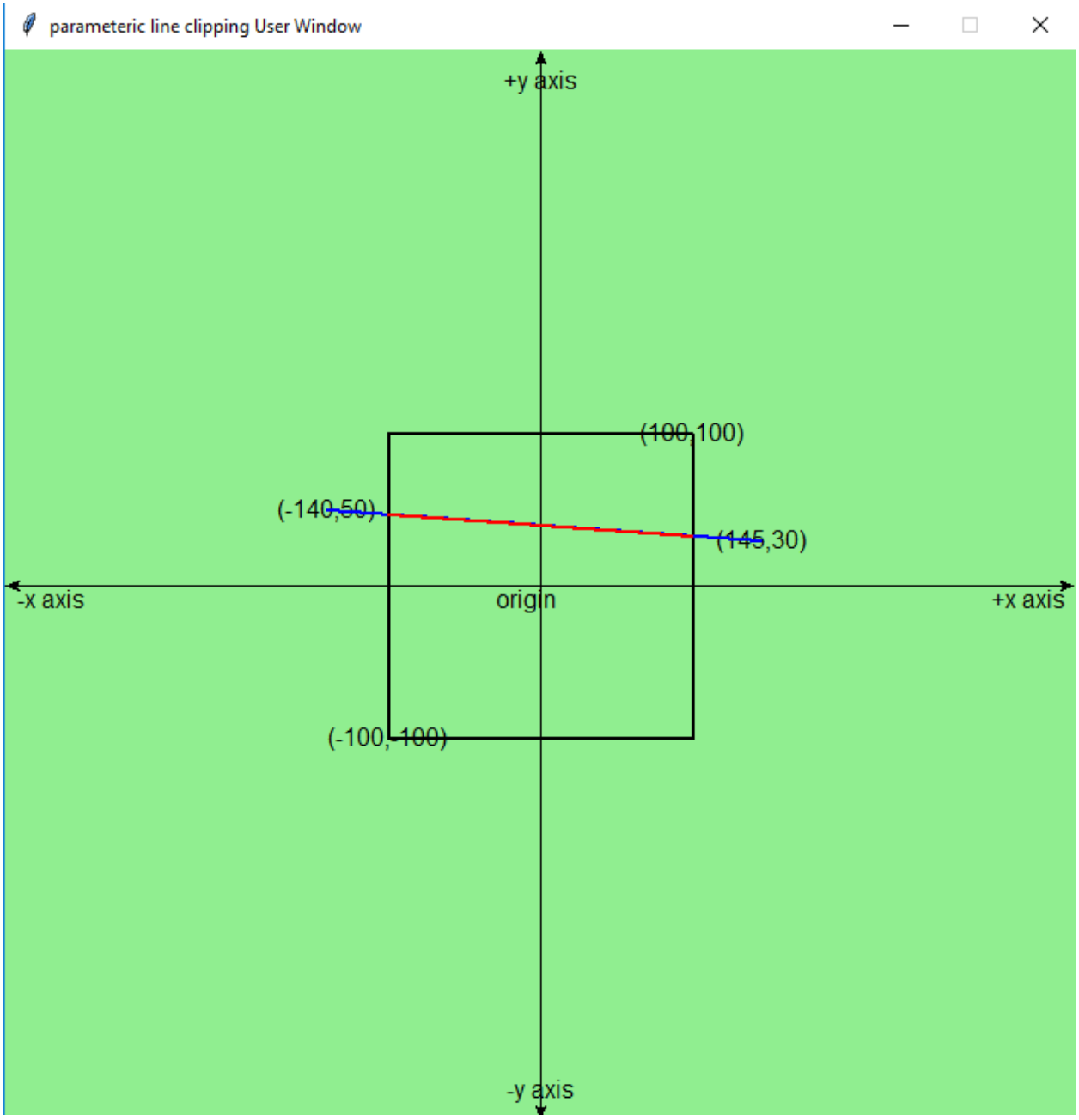
```
win_obj=GraphWin("parameteric line clipping User Window",700,700) #set viewport size 700,700 are device coordinates

win_obj.setBackground("Light Green")

win_obj.setCoords(-350,-350,350,350) #set window  use coordinates are set

x_axis=Line(Point(-350,0),Point(350,0))   #obj for x axis

y_axis=Line(Point(0,-350),Point(0,350))    #obj for y axis


x_axis.setOutline("Black")

y_axis.setOutline("Black")

x_axis.setArrow('both')

y_axis.setArrow('both')

x_axis.draw(win_obj)

y_axis.draw(win_obj)


info_x=Text(Point(320,-10),"+x axis")

info_x.draw(win_obj)

info_nx=Text(Point(-320,-10),"-x axis")

info_nx.draw(win_obj)

info_y=Text(Point(0,330),"+y axis")

info_y.draw(win_obj)

info_ny=Text(Point(0,-330),"-y axis")

info_ny.draw(win_obj)


origin=Text(Point(-10,-10),"origin")

origin.draw(win_obj)
```

```
#for rectangle
initial_point=Text(Point(xmin,ymin),"("+str(xmin)+","+str(ymin)+")")
initial_point.draw(win_obj)
final_point=Text(Point(xmax,ymax),"("+str(xmax)+","+str(ymax)+")")
final_point.draw(win_obj)




#for line
initial_point=Text(Point(x0,y0),"("+str(x0)+","+str(y0)+")")
initial_point.draw(win_obj)
final_point=Text(Point(x1,y1),"("+str(x1)+","+str(y1)+")")
final_point.draw(win_obj)

#edge_table=[]
j=0
while (j!=t):
    list_t1=lista[j]
    list_t2=lista[j+1]
    x01=list_t1[0]
    y01=list_t1[1]
    x02=list_t2[0]
    y02=list_t2[1]
```

```
    helper(x01,y01,x02,y02,color)
    j=j+1
#print(edge_table)
#scanline()
helper(x0,y0,x1,y1,color2)
parameter()


win_obj.getMouse()
win_obj.close()
```

Example2:

```
============== RESTART: C:\Users\Ashish\Desktop\py\parameter.py
enter xmin coordinate of window:-100
enter ymin coordinate of window:-100
enter xmax coordinate of window:100
enter ymax coordinate of window:100
enter x0 coordinate of line to be clipped:-140
enter y0 coordinate of line to be clipped:50
enter x1 coordinate of line to be clipped:145
enter y1 coordinate of line to be clipped:30
```


parameteric line clipping User Window
+y axis
(100,100)
(-140,50)
(145,30)
-x axis
origin
+x axis
(-100,-100)
-y axis

Cohen Sutherland Polygon clip Algorithm

Code:

```
from graphics import *

boundary=[]



def zero_to_one(x0,y0,x1,y1,color):  #slope 0 to 1
    a=y1-y0
    b=-1*(x1-x0)
    di=2*a+b
    dne=2*(a+b)
    de=2*a
    y=y0
    for x in range(x0,x1,1):
        #pixel=Point(x,y)
        #pixel.draw(win_obj,"red")
        pt = Point(x,y)
        pt.setOutline(color)
        pt.draw(win_obj)
        #time.sleep(0.01)
        if(di>0):
            y=y+1
            di=di+dne
        else:
```

```
            di=di+de
        boundary.append([x,y])
        print("point"+"["+str(x)+","+str(y)+"]")
def zero_to_n_one(x0,y0,x1,y1,color): #slope -1 to 0
    a=y1-y0
    b=-1*(x1-x0)
    di=2*a-b
    dne=2*(a-b)
    de=2*a
    y=y0
    for x in range(x0,x1,1):
        #pixel=Point(x,y)
        #pixel.draw(win_obj,"red")
        pt = Point(x,y)
        pt.setOutline(color)
        pt.draw(win_obj)
        #time.sleep(0.01)
        if(di>0):
            di=di+de
        else:
            y=y-1
            di=di+dne
        boundary.append([x,y])
        print("point"+"["+str(x)+","+str(y)+"]")
#################################
```

```
#  greater_one   slope <-1 and >1
def pure_greater_one(x0,y0,x1,y1,color):  #slope >1
    b=-1*(y1-y0)
    a=x1-x0
    dne=2*(a+b)
    de=2*a
    di=2*a+b
    x=x0
    for y in range(y0,y1,1):
        #pixel=Point(x,y)
        #pixel.draw(win_obj,"red")
        pt = Point(x,y)
        pt.setOutline(color)
        pt.draw(win_obj)
        #time.sleep(0.01)
        if(di>0):
            x=x+1
            di=di+dne
        else:
            di=di+de
        boundary.append([x,y])
        print("point"+"["+str(x)+","+str(y)+"]")
def less_negative_one(x0,y0,x1,y1,color):  #slope <-1
    a=x1-x0
    b=-1*(y1-y0)
```

```
    di=2*a-b
    dne=2*(a-b)
    de=2*a
    x=x0
    for y in range(y0,y1,1):
        #pixel=Point(x,y)
        #pixel.draw(win_obj,"red")
        pt = Point(x,y)
        pt.setOutline(color)
        pt.draw(win_obj)
        #time.sleep(0.01)
        if(di>0):
            di=di+de
        else:
            x=x-1
            di=di+dne
        boundary.append([x,y])
        print("point"+"["+str(x)+","+str(y)+"]")
################################

def less_one(x0,y0,x1,y1,color):  #slope -1 to 1
    a=y1-y0
    b=-1*(x1-x0)
    if(a<0):
        zero_to_n_one(x0,y0,x1,y1,color)
```

```
    else:
        zero_to_one(x0,y0,x1,y1,color)

#the greater_one cases are mirror image of less_one cases so simply replace x and y

def greater_one(x0,y0,x1,y1,color):  #slope > 1 and <-1
    a=x1-x0
    b=-1*(y1-y0)
    if(a<0):
        less_negative_one(x0,y0,x1,y1,color)
    else:
        pure_greater_one(x0,y0,x1,y1,color)



def helper(x0,y0,x1,y1,color):
    boundary.append([x0,y0])
    boundary.append([x1,y1])

    if(abs(x0-x1)<abs(y0-y1)):              #slope > 1 and <-1
        if(y1>y0):
            greater_one(x0,y0,x1,y1,color)
        else:
            greater_one(x1,y1,x0,y0,color)
    else:
        if(x1>x0):                          #slope -1 to 1
```

```
            less_one(x0,y0,x1,y1,color)
        else:                       #we always increase x by 1 therefore start point should always less, so swap both points
            less_one(x1,y1,x0,y0,color)



##############################################################################################################

def intersect(s,p,clipedge):
    #horizontal
    if clipedge[0][1] == clipedge[1][1]:
        y = clipedge[1][1]
        x = s[0] + (y - s[1])*(p[0]-s[0])/(p[1]-s[1])
    else:#vertical
        x = clipedge[1][0]
        y = s[1] + (x - s[0])*(p[1]-s[1])/(p[0]-s[0])

    return [int(x),int(y)]

def inside(point,clipedge):
    if clipedge[1][0] > clipedge[0][0]: # bottom
        if point[1] >= clipedge[0][1]:
            return 1
    if clipedge[1][1] > clipedge[0][1]: # right
        if point[0] <= clipedge[1][0]:
            return 1
```

```
    if clipedge[1][0] < clipedge[0][0]: # top
        if point[1] <= clipedge[0][1]:
            return 1
    if clipedge[1][1] < clipedge[0][1]: # left
        if point[0] >= clipedge[1][0]:
            return 1
    return 0

def polygon_clip(in_vertex_array,inlen,clipedge):
    out_vertex_array=[]
    s=in_vertex_array[inlen-1]
    for i in range(inlen):
        p=in_vertex_array[i]
        if (inside(p,clipedge)):
            if(inside(s,clipedge)):
                out_vertex_array.append(p)
            else:
                intersect_point=intersect(s,p,clipedge)
                out_vertex_array.append(intersect_point)
                out_vertex_array.append(p)
        elif (inside(s,clipedge)):
            intersect_point=intersect(s,p,clipedge)
            out_vertex_array.append(intersect_point)
        s=p
    return out_vertex_array
```

```
#-----------------------------------------------------------------------------------------------------------

x_min = int(input("Enter min x coordinate for window: "))

y_min = int(input("Enter min y coordinate for window: "))

x_max = int(input("Enter max x coordinate for window: "))

y_max = int(input("Enter max y coordinate for window: "))


print("Enter the number of vertices of polygon")

c=int(input())

coordinate = []

for i in range(c):

    x = int(input("Enter x coordinate of vertex:"))

    y = int(input("Enter y coordinate of vertex:"))

    point=[]

    point.append(x)

    point.append(y)

    coordinate.append(point)

temp=[]

temp=coordinate[0]

coordinate.append(temp)

print(coordinate)
```

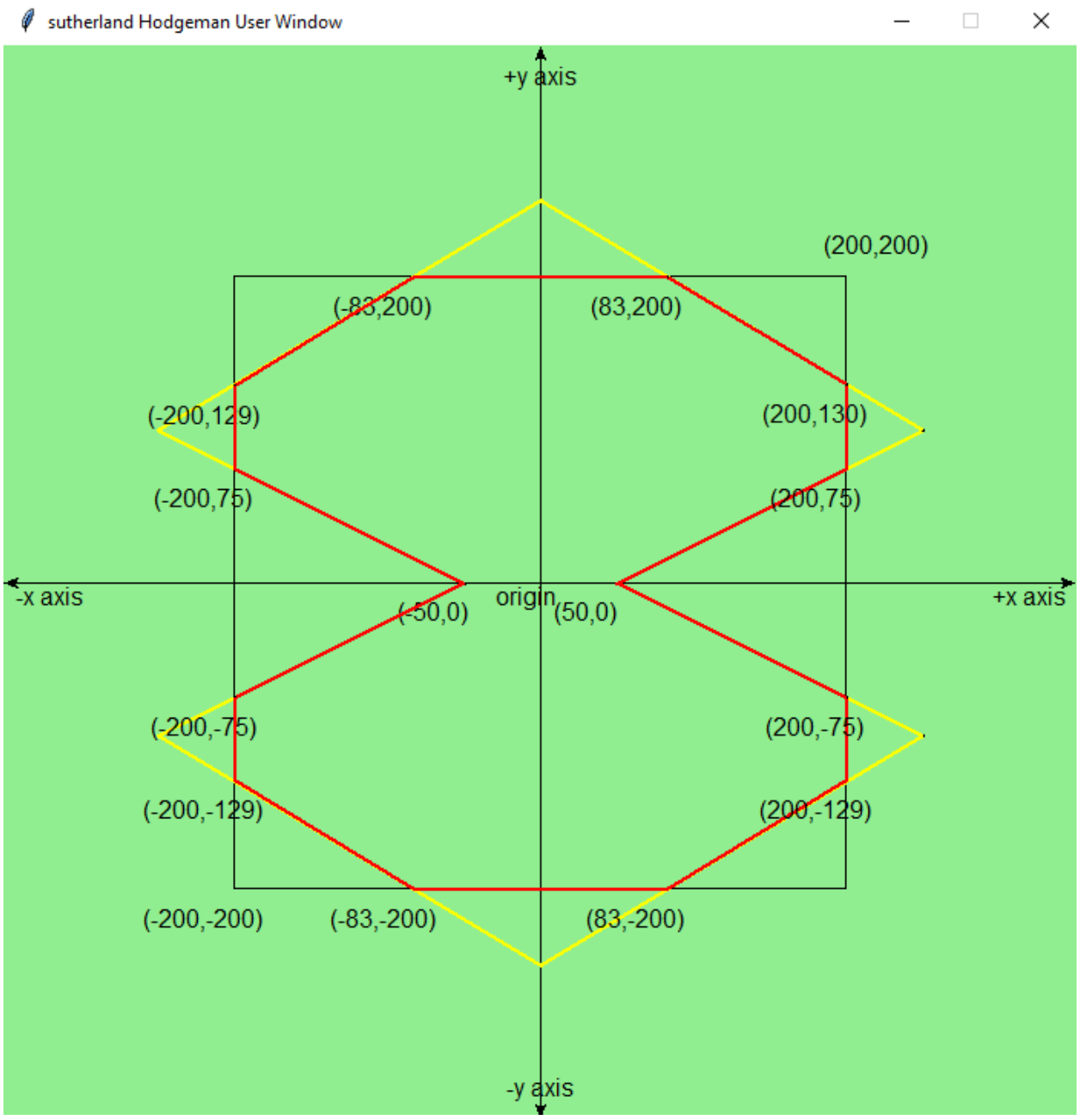
```
win_obj=GraphWin("sutherland Hodgeman User Window",700,700) #set viewport size  700,700 are device coordinates

win_obj.setBackground("Light Green")

win_obj.setCoords(-350,-350,350,350) #set window  use coordinates are set

x_axis=Line(Point(-350,0),Point(350,0))   #obj for x axis

y_axis=Line(Point(0,-350),Point(0,350))    #obj for y axis


x_axis.setOutline("Black")

y_axis.setOutline("Black")

x_axis.setArrow('both')

y_axis.setArrow('both')

x_axis.draw(win_obj)

y_axis.draw(win_obj)


info_x=Text(Point(320,-10),"+x axis")

info_x.draw(win_obj)

info_nx=Text(Point(-320,-10),"-x axis")

info_nx.draw(win_obj)

info_y=Text(Point(0,330),"+y axis")

info_y.draw(win_obj)

info_ny=Text(Point(0,-330),"-y axis")

info_ny.draw(win_obj)


origin=Text(Point(-10,-10),"origin")

origin.draw(win_obj)
```

```
rectangle = Rectangle(Point(x_min,y_min),Point(x_max,y_max))
rectangle.draw(win_obj)
display = Text(Point(x_max+20,y_max+20),"("+str(x_max)+","+str(y_max)+")")
display.draw(win_obj)
display = Text(Point(x_min-20,y_min-20),"("+str(x_min)+","+str(y_min)+")")
display.draw(win_obj)

#previous
for i in range(c):
    x0 = coordinate[i][0]
    y0 = coordinate[i][1]
    x1 = coordinate[i+1][0]
    y1 = coordinate[i+1][1]
    Point(x0,y0).draw(win_obj)
    helper(x0,y0,x1,y1,"yellow")

coordinate = polygon_clip(coordinate,len(coordinate),[[x_min,y_min],[x_max,y_min]])
coordinate = polygon_clip(coordinate,len(coordinate),[[x_max,y_min],[x_max,y_max]])
coordinate = polygon_clip(coordinate,len(coordinate),[[x_max,y_max],[x_min,y_max]])
coordinate = polygon_clip(coordinate,len(coordinate),[[x_min,y_max],[x_min,y_min]])

c=len(coordinate)
temp=[]
temp=coordinate[0]
```

```
coordinate.append(temp)
print(coordinate)
for i in range(c):
    x0 = coordinate[i][0]
    y0 = coordinate[i][1]
    x1 = coordinate[i+1][0]
    y1 = coordinate[i+1][1]
    Point(x0,y0).draw(win_obj)
    display = Text(Point(x0-20,y0-20),"("+str(x0)+","+str(y0)+")")
    display.draw(win_obj)
    helper(x0,y0,x1,y1,"red")



win_obj.getMouse()
win_obj.close()
```

Example3:

```
========== RESTART: C:\Users\Ashish\Desktop\py\Polygon_clipping.py
Enter min x coordinate for window: -200
Enter min y coordinate for window: -200
Enter max x coordinate for window: 200
Enter max y coordinate for window: 200
Enter the number of vertices of polygon
8
Enter x coordinate of vertex:0
Enter y coordinate of vertex:250
Enter x coordinate of vertex:-250
Enter y coordinate of vertex:100
Enter x coordinate of vertex:-50
Enter y coordinate of vertex:0
Enter x coordinate of vertex:-250
Enter y coordinate of vertex:-100
Enter x coordinate of vertex:0
Enter y coordinate of vertex:-250
Enter x coordinate of vertex:250
Enter y coordinate of vertex:-100
Enter x coordinate of vertex:50
Enter y coordinate of vertex:0
Enter x coordinate of vertex:250
Enter y coordinate of vertex:100
```


sutherland Hodgeman User Window
+y axis
(200,200)
(-83,200)
(83,200)
(-200,129)
(200,130)
(-200,75)
(200,75)
-x axis
origin
+x axis
(-50,0)
(50,0)
(-200,-75)
(200,-75)
(-200,-129)
(200,-129)
(-200,-200)
(-83,-200)
(83,-200)
-y axis