Lab report on
Computer Graphics

Course Code : CSE-4105

Submitted to:

Md. Mohibullah
Assistant professor
Department of CSE
Comilla University

Submitted By:

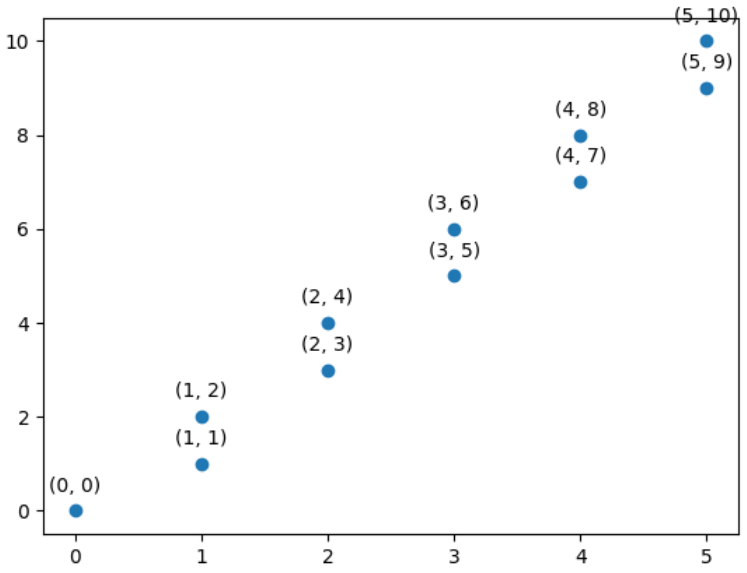
Syed Khaled Hossain
Id: 11908047
Session: 2018-19
Dept. of CSE

Date of Submission: 16 June, 2023.

1. Draw a line using DDA Algorithm

Source code:

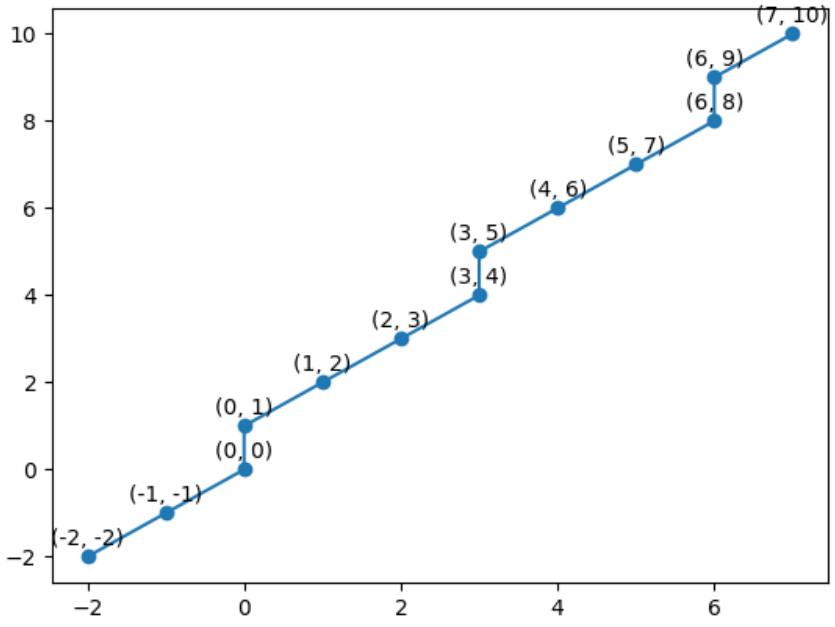
```
import matplotlib.pyplot as plt
def round(a):
    a1 = int(a)
    a2 = a1+1
    if a-a1>=a2-a:
        return a2    return a1

def DDA(x1,y1,x2,y2):
    if x1>x2 or y1>y2:
        x1,x2=x2,x1
        y1,y2=y2,y1

    dx = x2 - x1
    dy = y2 - y1
    m = dy/dx
    if m<0:
        print("Slope is negative")
        return
    step = max(abs(dx),abs(dy))
    xinc = dx / step
    yinc = dy / step
    x = x1
    y = y1
    X = [x]
    Y = [y]
    # Draw line
    for i in range(step):
        X.append(round(x+xinc))
        Y.append(round(y+yinc))
        x += xinc
        y += yinc
    # Plot the points
    plt.scatter(X,Y)
    for i in range(len(X)):
        plt.annotate(f'({X[i]}, {Y[i]})', (X[i], Y[i]),
textcoords="offset points", xytext=(0,10), ha='center')

    plt.show()

DDA(0,0,5,10)
```



2. Draw a line using Direct Line Algorithm

Source code:

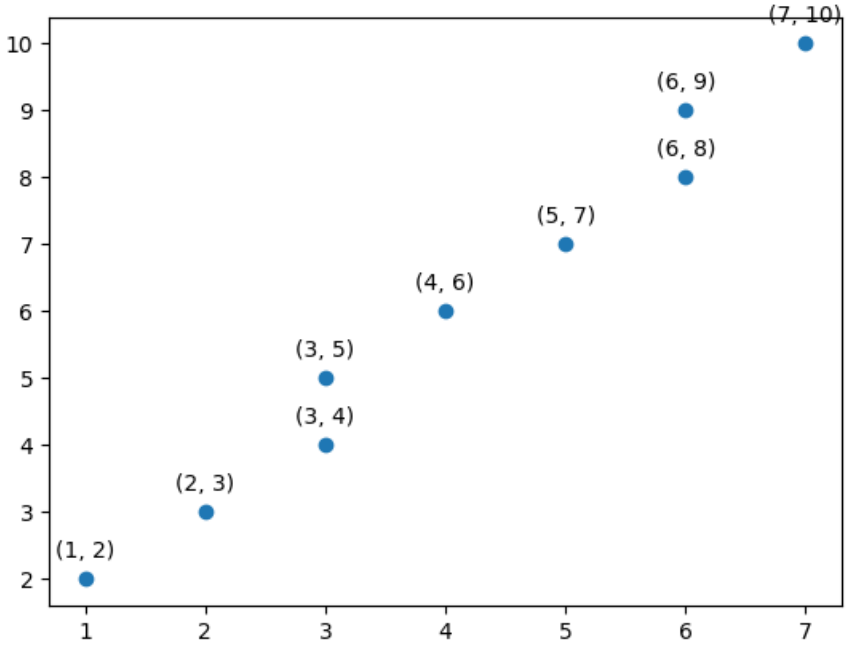
```
import matplotlib.pyplot as plt

def round(a):
    a1 = int(a)
    a2 = a1+1
    if a-a1>=a2-a:
        return a2
    return a1

def direct_line(x1,y1,x2,y2):
    if x1>x2:
        x1,x2=x2,x1
        y1,y2=y2,y1

    m = abs(y1-y2)/abs(x1-x2)
    if m<0:
        print("Invalid input")
        return
    b = y1-m*x1
    X = []
    Y = []
    while x1<=x2 or y1<=y2:
        X.append(round(x1))
        Y.append(round(y1))
        if m<=1:
            x1 +=1
            y1 = m*x1+b
        else:
            y1+=1
            x1 = (y1-b)/m
    # Plot the points
    plt.plot(X,Y,marker='o')
    for i in range(len(X)):
        plt.annotate(f'({X[i]}, {Y[i]})', (X[i], Y[i]),
textcoords="offset points", xytext=(0,5), ha='center')

    plt.show()

direct_line(-2,-2,7,10)
```



3. Draw a line using Bresenham's Algorithm

Source code:

```
import matplotlib.pyplot as plt
def bresenham_line(x1,y1,x2,y2):
    if x1>x2 or y1>y2:
        x1,x2=x2,x1
        y1,y2=y2,y1
    dx = x2-x1
    dy = y2-y1
    m = dy/dx
    if m>=1:
        c1 = 2*dx
        c2 = 2*(dx-dy)
        p = c1-dy
    else:
        c1 = 2*dy
        c2 = 2*(dy-dx)
        p = c1-dx
    X=[x1]
    Y=[y1]
    got=0
    while got<2:
        if p<0:
            p += c1
            if m >= 1:
                y1 += 1
            else:
                x1 += 1
        else:
            p += c2
            y1+=1;x1+=1

        if got>0:
            got+=1
        if(x1==x2 and y1==y2):
            got=1
        if x1-2>x2 and y1-2>y2:
            break
        if(x1<=x2 and y1<=y2):
            X.append(x1)
            Y.append(y1)

plt.scatter(X,Y)
    for i in range(len(X)):
        plt.annotate(f'({X[i]}, {Y[i]})', (X[i], Y[i]),
```

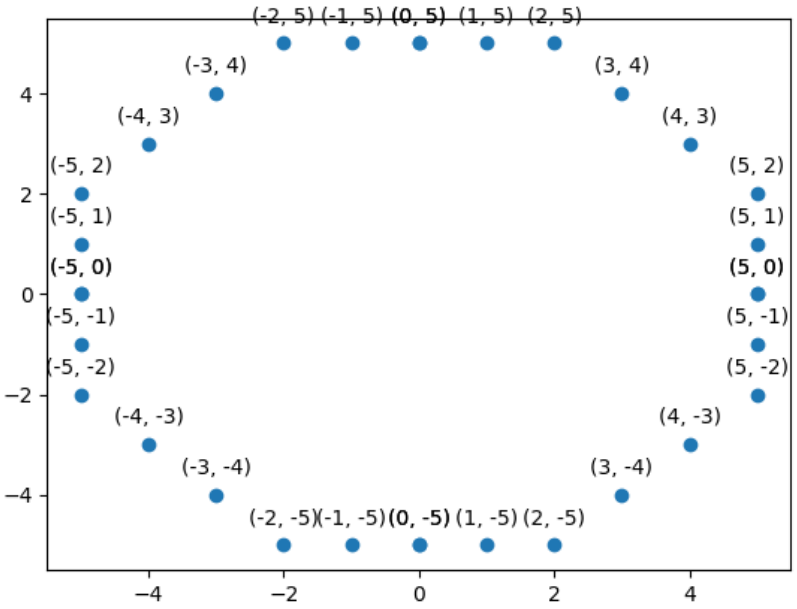


```
textcoords="offset points", xytext=(0,10), ha='center')
    plt.show()
bresenham_line(1,2,7,10)
```

4. Draw a circle using Bresenhum's Algorithm

Source code:

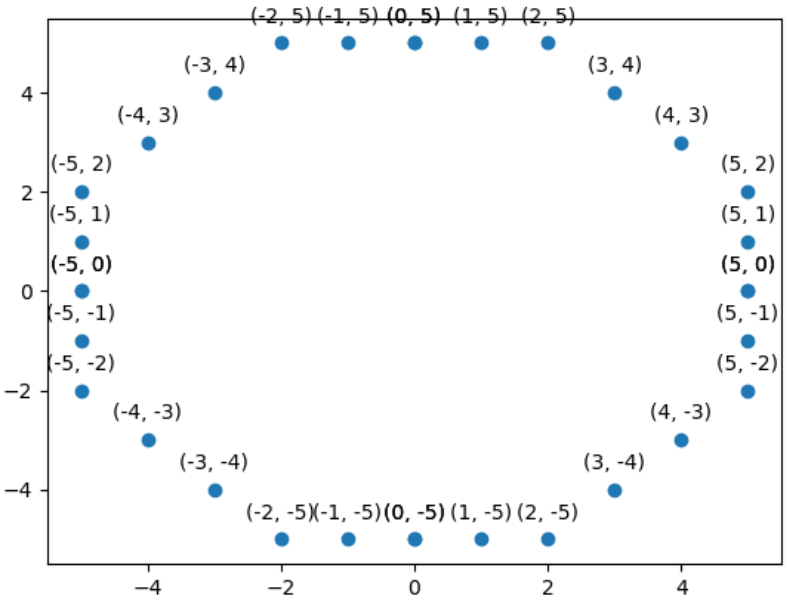
```
import matplotlib.pyplot as plt
def bresenham_circle(h,k,r):
    x, y = 0,r
    p = 3-2*r
    X = []
    Y = []
    while(x<=y):
        X.append(x+h)
        Y.append(y+k)
        X.append(y+k)
        Y.append(x+h)
        X.append(-x+h)
        Y.append(y+k)
        X.append(y+k)
        Y.append(-x+h)
        X.append(-x+h)
        Y.append(-y+k)
        X.append(-y+k)
        Y.append(-x+h)
        X.append(x+h)
        Y.append(-y+k)
        X.append(-y+k)
        Y.append(x+h)
        if p<0:
            p = p+4*x+6
        else:
            p = p+4*(x-y)+10
            y-=1
        x+=1
    plt.scatter(X,Y)
    for i in range(len(X)):
        plt.annotate(f'({X[i]}, {Y[i]})', (X[i], Y[i]),
textcoords="offset points", xytext=(0,10), ha='center')
    plt.show()
bresenham_circle(0,0,5)
```



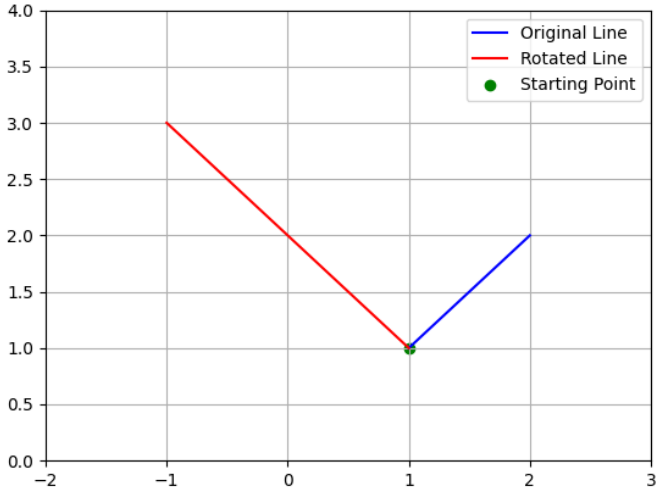
5. Draw a circle using midpoint Algorithm

Source code:

```
import matplotlib.pyplot as plt
def midpoint_circle(h,k,r):
    x, y = 0,r
    p = 1-r
    X = []
    Y = []
    while(x<=y):
        X.append(x+h)
        Y.append(y+k)
        X.append(y+k)
        Y.append(x+h)
        X.append(-x+h)
        Y.append(y+k)
        X.append(y+k)
        Y.append(-x+h)
        X.append(-x+h)
        Y.append(-y+k)
        X.append(-y+k)
        Y.append(-x+h)
        X.append(x+h)
        Y.append(-y+k)
        X.append(-y+k)
        Y.append(x+h)
        if p<0:
            p = p+2*x+3
        else:
            p = p+2*(x-y)+5
            y-=1
        x+=1
    plt.scatter(X,Y)
    for i in range(len(X)):
        plt.annotate(f'({X[i]}, {Y[i]})', (X[i], Y[i]),
textcoords="offset points", xytext=(0,10), ha='center')
    plt.show()
midpoint_circle(0,0,5)
```



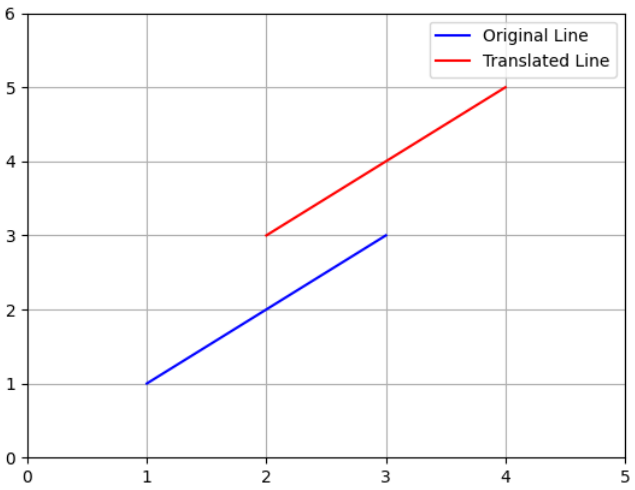
```python
import matplotlib.pyplot as plt
import numpy as np

def rotate_line(x1, y1, x2, y2, angle):
    x2 -= x1
    y2 -= y1
    angle_rad = np.radians(angle)
    cos_theta = np.cos(angle_rad)
    sin_theta = np.sin(angle_rad)
    new_x2 = x2 * cos_theta - y2 * sin_theta
    new_y2 = x2 * sin_theta + y2 * cos_theta
    new_x2 += x1
    new_y2 += y1
    plt.plot([x1, x2], [y1, y2], 'b', label='Original Line')
    plt.plot([x1, new_x2], [y1, new_y2], 'r', label='Rotated Line')
    plt.scatter(x1, y1, color='g', label='Starting Point')

    plt.xlim(min(x1, x2, new_x2) - 1, max(x1, x2, new_x2) + 1)
    plt.ylim(min(y1, y2, new_y2) - 1, max(y1, y2, new_y2) + 1)

    plt.legend()
    plt.axis('on')
    plt.grid(True)
    plt.show()

x1 = 1
y1 = 1
x2 = 3
y2 = 3
angle = 90
rotate_line(x1, y1, x2, y2, angle)
```



7. Program to show the translation of a line

Source Code:

```
import matplotlib.pyplot as plt
def translate_line(x1, y1, x2, y2, tx,ty):

    new_x1 = x1 + tx
    new_y1 = y1 + ty
    new_x2 = x2 + tx
    new_y2 = y2 + ty

    plt.plot([x1, x2], [y1, y2], 'b', label='Original Line')
    plt.plot([new_x1, new_x2], [new_y1, new_y2], 'r', label='Translated
Line')

    plt.xlim(min(x1, x2, new_x1, new_x2) - 1, max(x1, x2, new_x1,
new_x2) + 1)
    plt.ylim(min(y1, y2, new_y1, new_y2) - 1, max(y1, y2, new_y1,
new_y2) + 1)

    plt.legend()
    plt.axis('on')
    plt.grid(True)
    plt.show()

x1 = 1
y1 = 1
x2 = 3
y2 = 3
tx = 1
ty = 2

translate_line(x1, y1, x2, y2, tx, ty)
```



8. Program to scale a triangle

Code :

```
import matplotlib.pyplot as plt

def scale_triangle(x1, y1, x2, y2, x3, y3, scale_factor):
    new_x1 = x1 * scale_factor
    new_y1 = y1 * scale_factor
    new_x2 = x2 * scale_factor
    new_y2 = y2 * scale_factor
    new_x3 = x3 * scale_factor
    new_y3 = y3 * scale_factor

    plt.plot([x1, x2, x3, x1], [y1, y2, y3, y1], 'b', label='Original
Triangle')
    plt.plot([new_x1, new_x2, new_x3, new_x1], [new_y1, new_y2, new_y3,
new_y1], 'r', label='Scaled Triangle')

    plt.xlim(min(x1, x2, x3, new_x1, new_x2, new_x3) - 1, max(x1, x2,
x3, new_x1, new_x2, new_x3) + 1)
    plt.ylim(min(y1, y2, y3, new_y1, new_y2, new_y3) - 1, max(y1, y2,
y3, new_y1, new_y2, new_y3) + 1)

    plt.legend()
    plt.axis('on')
    plt.grid(True)
    plt.show()

x1 = 1
y1 = 2
x2 = 5
y2 = 5
x3 = 6
y3 = 3
scale_factor = 2

scale_triangle(x1, y1, x2, y2, x3, y3, scale_factor)
```