Program 11

Aim: Write a program to implement Z-buffer algorithm. Perform the algorithm on a set of at least 5 objects in a viewing coordinate system.

Theory:

Z-buffer, which is also known as the Depth-buffer method is one of the commonly used method for hidden surface detection. It is an Image space method. Image space methods are based on the pixel to be drawn on 2D. For these methods, the running time complexity is the number of pixels times number of objects. And the space complexity is two times the number of pixels because two arrays of pixels are required, one for frame buffer and the other for the depth buffer.

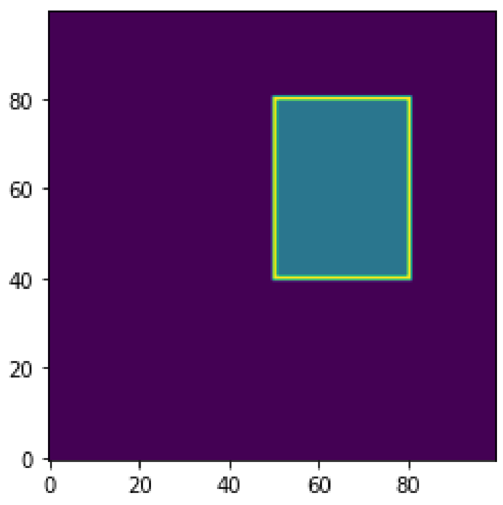
Algorithm:

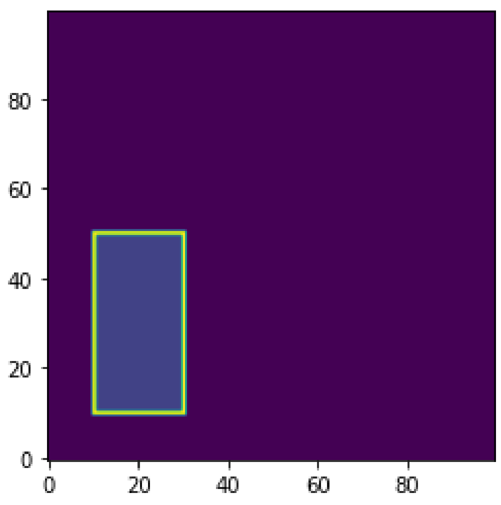
```
First of all, initialize the depth of each pixel.
i.e,  d(i, j) = infinite (max length)
Initialize the color value for each pixel
as c(i, j) = background color
for each polygon, do the following steps :

for (each pixel in polygon's projection)
{
    find depth i.e, z of polygon
    at (x, y) corresponding to pixel (i, j)

    if (z < d(i, j))
    {
        d(i, j) = z;
        c(i, j) = color;
    }
}
```

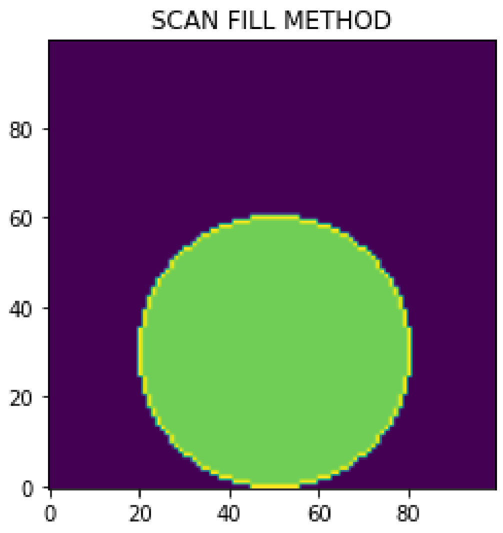
Code:

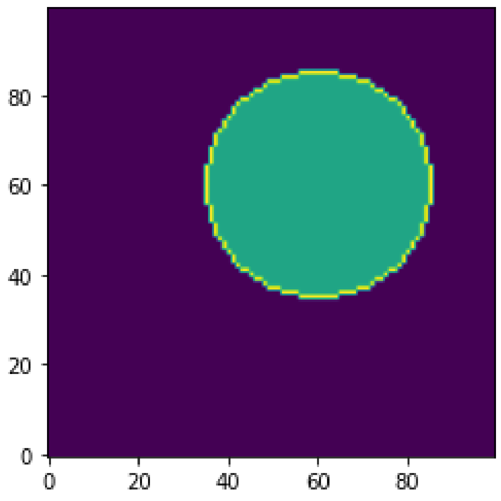
```python
import matplotlib.pyplot as plt
import numpy as np

def circle_imp(x0, y0, r):
    x = 0
    y = r
    pts_x = [x0+x, x0+x, x0+y, x0-y]
    pts_y = [y0+y, y0-y, y0+x, y0+x]
    d = 1 - r
    while y >= x:
        dE = 2*x + 1
        dSE = 2*x - 2*y + 1
        if d <= 0:
            d += dE
        else:
            d += dSE
            y = y - 1
        pts_x.append(x0+x)
        pts_x.append(x0-x)
        pts_x.append(x0+x)
        pts_x.append(x0-x)
        pts_x.append(x0+y)
        pts_x.append(x0-y)
        pts_x.append(x0+y)
        pts_x.append(x0-y)
        pts_y.append(y0+y)
        pts_y.append(y0+y)
        pts_y.append(y0-y)
        pts_y.append(y0-y)
        pts_y.append(y0+x)
        pts_y.append(y0+x)
        pts_y.append(y0-x)
        pts_y.append(y0-x)
        x += 1
    return pts_x, pts_y

def midpt_line(x0, y0, xn, yn):

    dy = yn-y0
    dx = xn-x0
    pts_x = []
    pts_y = []
    pts_x.append(x0)
    pts_y.append(y0)
    x = x0
    y = y0
    if dy >= 0 and dx >= 0:
        if dy > dx:
            d = 2*dx - dy
            dE = 2*dx
            dNE = 2*(dx-dy)
            for i in range(1, yn-y0 + 1):
                if d <= 0:
                    d = d + dE
                else:
                    d = d + dNE
                    x = x + 1
                pts_x.append(x)
                pts_y.append(y + i)
        else:
            d = 2*dy - dx
            dE = 2*dy
            dNE = 2*(dy-dx)
            for i in range(1, xn-x0+1):
                if d <= 0:
                    d = d + dE
                else:
                    d = d + dNE
                    y = y + 1
                pts_x.append(x + i)
                pts_y.append(y)
    else:
        if abs(dy) > abs(dx):
            dy = abs(dy)
            dx = abs(dx)
            d = 2*dx - dy
            dE = 2*dx
            dNE = 2*(dx-dy)
            for i in range(1, abs(yn-y0)+1):
                if d <= 0:
                    d = d + dE
                else:
                    d = d + dNE
                    x = x + 1
                pts_x.append(x)
                pts_y.append(y - i)
        else:
            dy = abs(dy)
            dx = abs(dx)
            d = 2*dy - dx
            dE = 2*dy
            dNE = 2*(dy-dx)
            for i in range(1, abs(xn-x0)+1):
                if d <= 0:
                    d = d + dE
                else:
                    d = d + dNE
                    y = y - 1
                pts_x.append(x + i)
                pts_y.append(y)
    return pts_x, pts_y

def FloodFill(screen, x, y, newC, oldC):

    currPix = screen[y, x]
    if currPix == oldC:
        screen[y, x] = newC
        FloodFill(screen, x + 1, y, newC, oldC)
        FloodFill(screen, x - 1, y, newC, oldC)
        FloodFill(screen, x, y + 1, newC, oldC)
        FloodFill(screen, x, y - 1, newC, oldC)

    x1 = np.zeros((100, 100))
    coor_y, coor_x = midpt_line(50, 40, 80, 40)
    for i, j in zip(coor_x, coor_y):
        x1[i, j] = 255
    coor_y, coor_x = midpt_line(50, 40, 50, 80)
    for i, j in zip(coor_x, coor_y):
        x1[i, j] = 255
    coor_y, coor_x = midpt_line(50, 80, 80, 80)
    for i, j in zip(coor_x, coor_y):
        x1[i, j] = 255
    coor_y, coor_x = midpt_line(80, 80, 80, 40)
    for i, j in zip(coor_x, coor_y):
        x1[i, j] = 255
    FloodFill(x1, 60, 50, 100, 0)
    plt.figure()
    plt.imshow(x1)
    plt.gca().invert_yaxis()
    obj1 = np.where(x1 > 0)

    plt.figure()
    x2 = np.zeros((100, 100))
    coor_y, coor_x = midpt_line(10, 10, 10, 50)
    for i, j in zip(coor_x, coor_y):
        x2[i, j] = 255
    coor_y, coor_x = midpt_line(10, 50, 30, 50)
    for i, j in zip(coor_x, coor_y):
        x2[i, j] = 255
    coor_y, coor_x = midpt_line(30, 50, 30, 10)
    for i, j in zip(coor_x, coor_y):
        x2[i, j] = 255
    coor_y, coor_x = midpt_line(10, 10, 30, 10)
    for i, j in zip(coor_x, coor_y):
        x2[i, j] = 255
    FloodFill(x2, 20, 20, 50, 0)
    plt.figure()
    plt.imshow(x2)
    plt.gca().invert_yaxis()
    obj2 = np.where(x2 > 0)

    plt.figure()
    x3 = np.zeros((100, 100))
    coor = []
    coor_y, coor_x = circle_imp(50, 30, 30)
    for i, j in zip(coor_x, coor_y):
        x3[i, j] = 255
        coor.append((j, i))
    FloodFill(x3, 50, 20, 200, 0)
    plt.figure()
    plt.imshow(x3)
    plt.title('SCAN FILL METHOD')
    plt.gca().invert_yaxis()
    obj3 = np.where(x3 > 0)

    plt.figure()
    x4 = np.zeros((100, 100))
    coor = []
    coor_y, coor_x = circle_imp(60, 60, 25)
    for i, j in zip(coor_x, coor_y):
        x4[i, j] = 255
        coor.append((j, i))
    FloodFill(x4, 60, 60, 150, 0)
    plt.figure()
    plt.imshow(x4)
    plt.gca().invert_yaxis()
    obj4 = np.where(x4 > 0)

    x5 = np.zeros((100, 100))
    coor_y, coor_x = midpt_line(70, 10, 90, 10)
    for i, j in zip(coor_x, coor_y):
        x5[i, j] = 255
    coor_y, coor_x = midpt_line(90, 10, 90, 30)
    for i, j in zip(coor_x, coor_y):
        x5[i, j] = 255
    coor_y, coor_x = midpt_line(70, 30, 90, 30)
    for i, j in zip(coor_x, coor_y):
        x5[i, j] = 255
    coor_y, coor_x = midpt_line(70, 10, 70, 30)
    for i, j in zip(coor_x, coor_y):
        x5[i, j] = 255
    FloodFill(x5, 80, 20, 250, 0)
    plt.figure()
    plt.imshow(x5)
    plt.gca().invert_yaxis()
    obj5 = np.where(x5 > 0)


    depth = np.full((100, 100), 10000)
    new = np.zeros((100, 100))

    for poly in [(obj1, 100, 5), (obj2, 50, 2), (obj3, 200, 10), (ob
j4, 150, 20), (obj5, 250, 1)]:
        d = poly[2]
        c = poly[1]
        poly = poly[0]
        for i, j in zip(poly[0], poly[1]):
            if depth[i, j] > d:
                depth[i, j] = d
                new[i, j] = c
    plt.figure()
    plt.imshow(new)
    plt.gca().invert_yaxis()
```

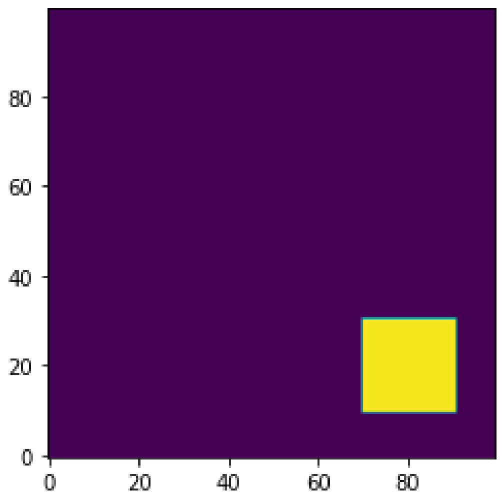
Output:

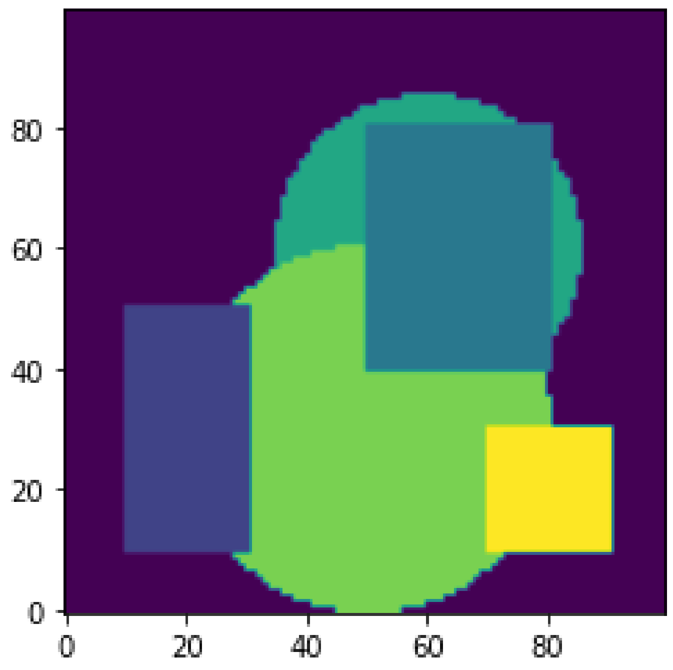

0
20
40
60
80


0
20
40
60
80


SCAN FILL METHOD
0
20
40
60
80


0
20
40
60
80


0
20
40
60
80



0
20
40
60
80
0
20
40
60
80