Experiment – 4

Theory:

There are several ellipse drawing algorithms used in computer graphics, each with its own advantages and disadvantages. Here are brief descriptions of a few popular algorithms:

1. Midpoint Ellipse Algorithm: This algorithm is one of the most popular ellipse drawing algorithms. It is based on the concept of finding the nearest integer pixel coordinate to the ideal ellipse path at each step. It uses integer arithmetic and is therefore faster and more efficient than algorithms that require floating-point calculations. This algorithm works by iterating over the four quadrants of the ellipse and drawing each pixel as a symmetric pair.

2. Bresenham's Ellipse Algorithm: This algorithm is an extension of Bresenham's line algorithm and is more efficient than the midpoint algorithm. It uses integer arithmetic to avoid the need for floating-point calculations, making it faster. The algorithm works by determining the most suitable pixel for drawing the ellipse at each stage of the process, which minimizes the error that accumulates over the ellipse.

3. Rational Bezier Curve Algorithm: This algorithm uses Bezier curves to draw the ellipse. It involves defining the ellipse as a Bezier curve and then dividing the curve into smaller segments to calculate the coordinates of each point. This algorithm produces very accurate ellipses but can be slower than other algorithms.

In summary, there are several ellipse drawing algorithms available, each with its own advantages and disadvantages. The choice of algorithm will depend on the specific needs of the application and the desired trade-off between speed and accuracy.

Algorithm:

```
function drawEllipse(xc, yc, rx, ry)
  x = 0
  y = ry
  p = ry^2 - rx^2*ry + 1/4*rx^2

  while 2*ry^2*x < 2*rx^2*y
    plotPoints(xc, yc, x, y)
    x = x + 1
    if p < 0 then
      p = p + 2*ry^2*x + ry^2
    else
```

```
            y = y - 1
            p = p + 2*ry^2*x - 2*rx^2*y + ry^2
        end if
    end while

    p = ry^2*(x+1/2)^2 + rx^2*(y-1)^2 - rx^2*ry^2

    while y >= 0
        plotPoints(xc, yc, x, y)
        y = y - 1
        if p > 0 then
            p = p - 2*rx^2*y + rx^2
        else
            x = x + 1
            p = p + 2*ry^2*x - 2*rx^2*y + rx^2
        end if
    end while
end function

function plotPoints(xc, yc, x, y)
    plotPixel(xc+x, yc+y)
    plotPixel(xc-x, yc+y)
    plotPixel(xc+x, yc-y)
    plotPixel(xc-x, yc-y)
end function
```