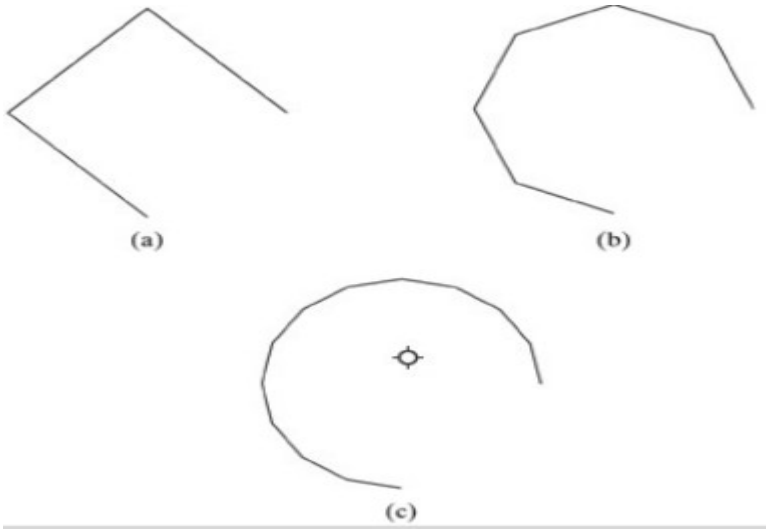
Rasterização de Curvas

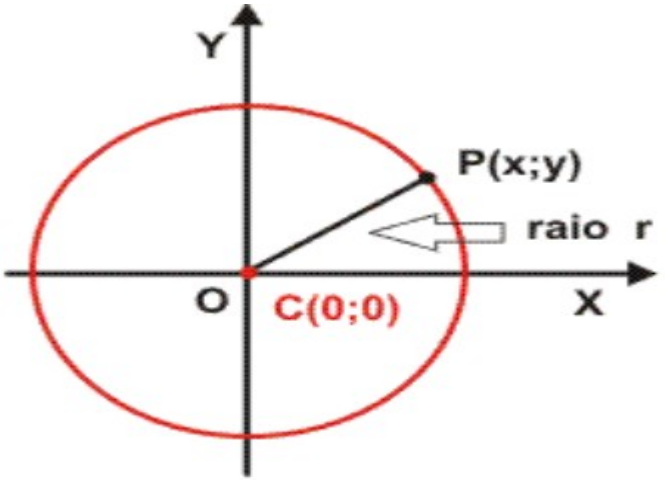
Curvas

- Podem ser representadas por polilinhas
  - a) 3 segmentos
  - b) 6 segmentos
  - c) 20 segmentos
- Pode-se usar os algoritmos de rasterização de retas



Circunferência

- Pontos equidistam de um ponto central
- Caso o ponto central seja a origem:

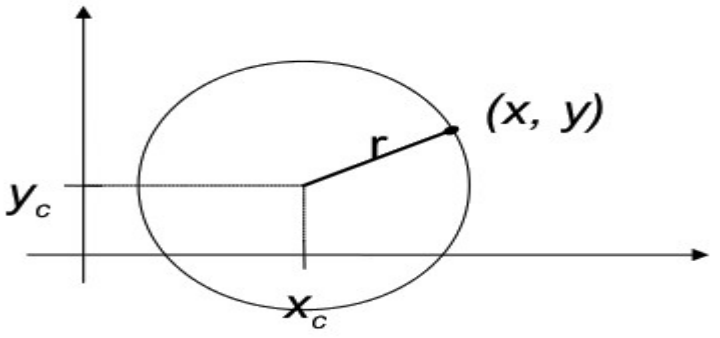


$$r^2 = x^2 + y^2 \Rightarrow \boxed{x^2 + y^2 - r^2 = 0}$$

Circunferência

- Considerando o centro em (x_c, y_c)

  $(x - x_c)^2 + (y - y_c)^2 = r^2$



- Algoritmo baseado na equação da circunferência
  - Para cada x entre: $(x_c - r) \leq x \leq (x_c + r)$

    $y = y_c \pm \sqrt{r^2 - (x - x_c)^2}$

Circunferência

- Algoritmo baseado na equação da circunferência

trinket Python3 Run Share

main.py

```
1  from PIL import Image, ImageDraw
2  import math
3
4  im = Image.new('L',(200, 200),255)
5  draw = ImageDraw.Draw(im)
6
7  r=20
8  xc=80
9  yc=80
10
11 for x in range(-r,r):
12   y=int(math.sqrt(r*r-x*x))
13   draw.point((xc+x,yc+y),0)
14   draw.point((xc+x,yc-y),0)
15
16 im = im.resize((800,800))
17 im.show()
18 im.save("a.png")
```

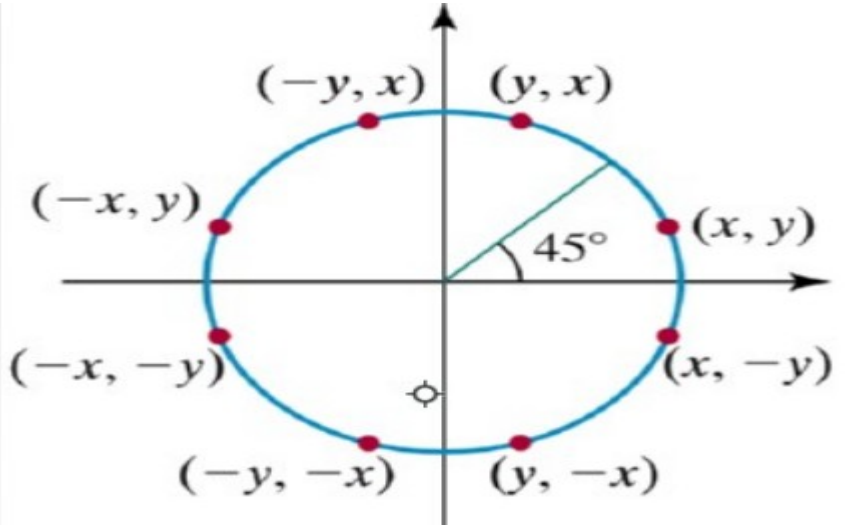
Powered by trinket

Problemas

- Ineficiente
  - Multiplicações e cálculo de raiz quadrada
  - Desenho descontínuo

Octantes

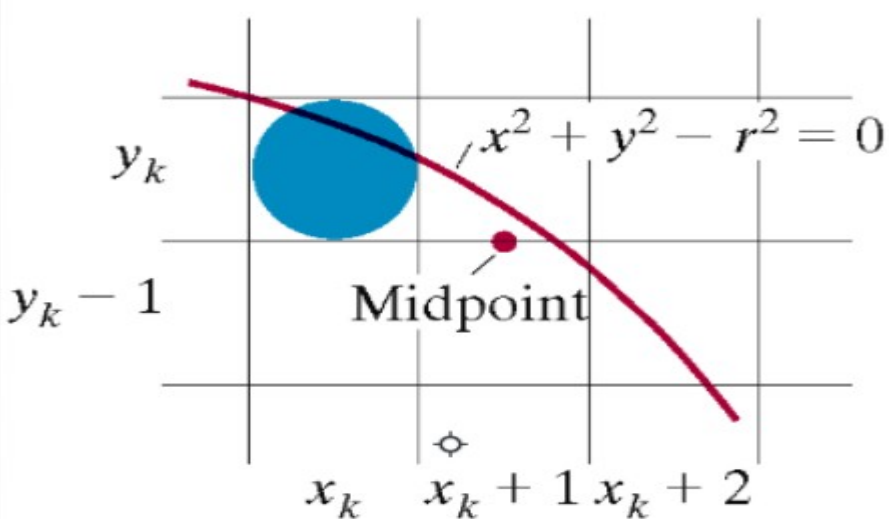
- Melhorar descontinuidades
  - Utilizando simetrias



Algoritmo de Ponto Médio

- Variante do algoritmo de Bresenham

f(x,y) { <0 (dentro do cric.)
=0 (no circ.)
>0 (fora do circ.)



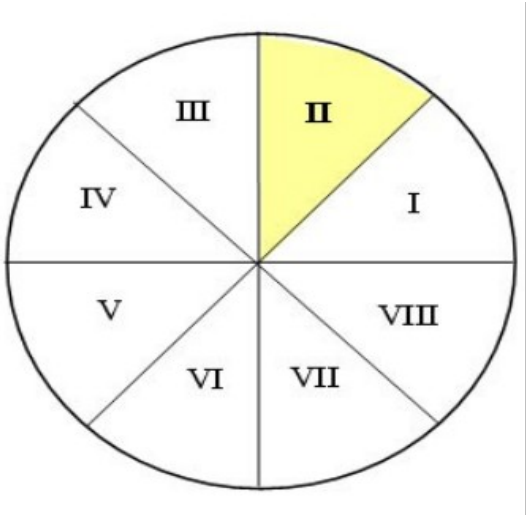
Algoritmo de Ponto Médio

- A cada passo, determina o melhor pixel
  - Similar ao algoritmo de reta, usa parâmetro de decisão: p
  - $p_k = (x_k + 1)^2 + (y_k – 1/2)^2 – r^2$
    - Se $p_k < 0$, escolher y_k
    - Se $p_k >= 0$, escolher yk+1
  - $p_0 = 1 – r$
  - $p_{k+1} = p_k + 2x_{k+1} + 3$ (se $p_k < 0$)
  - $p_{k+1} = p_k + 2x_{k+1} + 5 – 2y_{k+1}$ (se $p_k >= 0$)

Algoritmo de Ponto Médio

- Escolhendo o segundo octante
  - Demais octantes por simetria

OCTANTE	X_n	Y_n
I	Y	X
II	$\boldsymbol{X}$	$\boldsymbol{Y}$
III	$-X$	Y
IV	$-Y$	X
V	$-Y$	$-X$
VI	$-X$	$-Y$
VII	X	$-Y$
VIII	Y	$-X$

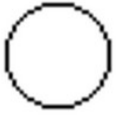


Algoritmo de Ponto Médio

main.py

```python
from PIL import Image, ImageDraw
import math

def drawSimetricPoints(draw, xc, yc, x, y):
  draw.point((xc+y,yc-x),0) # I
  draw.point((xc+x,yc-y),0) # II
  draw.point((xc-x,yc-y),0) # III
  draw.point((xc-y,yc-x),0) # IV
  draw.point((xc-y,yc+x),0) # V
  draw.point((xc-x,yc+y),0) # VI
  draw.point((xc+x,yc+y),0) # VII
  draw.point((xc+y,yc+x),0) # VIII

def drawCircle(draw,xc,yc,r):
  p=1-r
  y=int(r/math.sqrt(2))
  x=0

  drawSimetricPoints(draw, xc, yc, x, y)
  while y>x:
    if p<0:
      p+=2*x+3
    else:
      y-=1
      p+=2*(x-y)+5

    x+=1
    drawSimetricPoints(draw, xc, yc, x, y)

im = Image.new('L',(200, 200),255)
draw = ImageDraw.Draw(im)

r=20
xc=80
yc=80

drawCircle(draw,xc,yc,r)

im = im.resize((800,800))
im.show()
im.save("a.png")
```

Powered by trinket



a.png

Exercício

- Usando os algoritmos de ponto médio para circunferências desenhe um cacho de uvas como na imagem a seguir: