



UFRR

UNIVERSIDADE FEDERAL DE RORAIMA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

CLEUSON SOUSA SANTOS

**COMPUTAÇÃO GRÁFICA: ALGORITMOS DE RASTERIZAÇÃO DE
CIRCUNFERÊNCIA**

Boa Vista – Roraima,
Novembro de 2022

CLEUSON SOUSA SANTOS

**COMPUTAÇÃO GRÁFICA: ALGORITMOS DE RASTERIZAÇÃO DE
CIRCUNFERÊNCIA**

Relatório apresentada ao Departamento de
Ciência da Computação da Universidade
Federal de Roraima, como requisito parcial
para a obtenção de nota na Disciplina
Computação Gráfica.

Orientador

ProfºDr. Luciano Ferreira Silva

Boa Vista – Roraima,

Novembro de 2022

RELATÓRIO

Discente: Cleuson Sousa Santos

Tema: Rasterização de circunferências por meios de algoritmos: Equação Paramétrica, Incremental com Simetria e Bresenham.

O programa desenvolvido permite desenhar retas por meios dos algoritmos apresentados em sala de aula: Analítico, DDA e Bresenham. Implementado de maneira que se pode ver o trabalho de cada algoritmo e comparar os resultados.

1. Grid de Pixels

Segundo o padrão utilizado pela indústria, e o que consta na literatura, o sistema de coordenadas de pixel, define a origem de um destino de renderização no canto superior esquerdo, e avança por coordenadas positivas no eixo x para a direita e no eixo y para baixo.

Com a finalidade de tornar a rasterização mais didática e intuitiva, levou-se em consideração o sistema de coordenadas de pixels, no entanto, foi dada preferência e por transformar o sistema de coordenadas para o sistema euclidiano mais próximo do utilizado nas aulas, de tal maneira que torne melhor compreensível o resultado pretendido, tanto na renderização do resultado quanto na comparação de resultados.

A grid foi implementado como uma malha com 80 pixels de altura, com 80 pixels de largura, mais uma coluna e linha para representar o valor 0. A malha foi dividida em valores variando de -40 a 40, tanto no eixo x, quanto no eixo y.

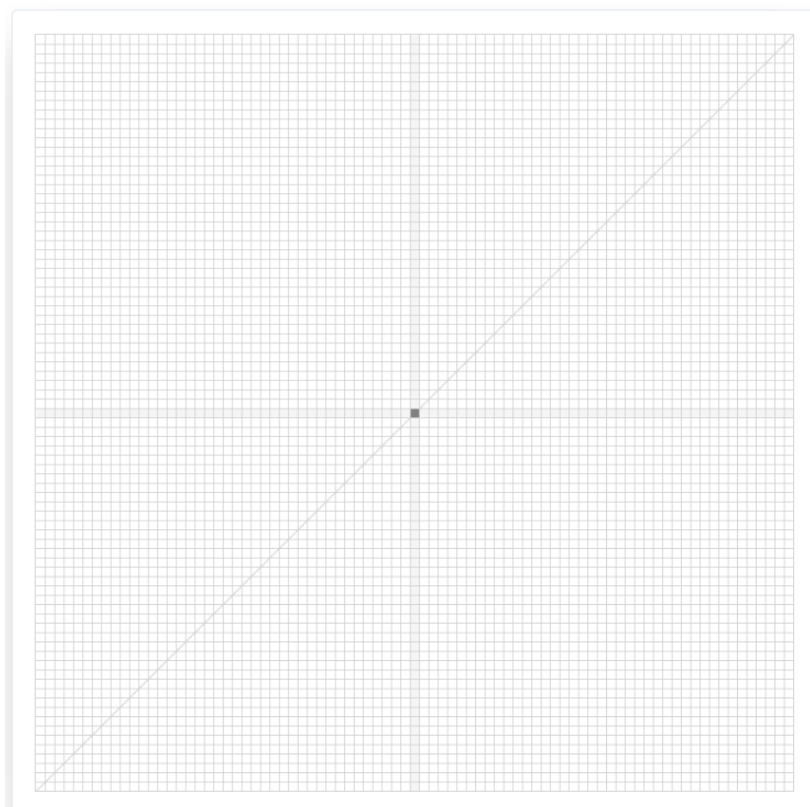


Fig 1. Malha de pixels

2. Controles

Foi implementado um conjunto de controles que possibilite a escolha do algoritmo que será utilizado para renderizar a linha, além de determina a coordenada inicial e a final da linha.

Método	
<input checked="" type="radio"/>	Bresenham
<input type="radio"/>	Incremental
<input type="radio"/>	Paramétrico

Centro	
x	0
y	0

Raio	
r	0

Fig 2. Controle para escolha de parâmetros da linha e comparação de algoritmo

3. Rasterização

Ao se escolher o algoritmo e entrar com as coordenadas, a circunferência será mostrada na malha de pixels, atualizando sempre que se altera campo do formulário de controle, automaticamente.

4. Implementação

O programa foi implementado com JavaScript, utilizando o Framework NextJS.

a. Algoritmo Equação Paramétrica.

```
/* -- Metodo Analitico -- */
export const parametric = (center, r) => {

  let circle = [];

  if (r === 0) {
    return circle;
  }

  let x = center.x + r;
  let y = center.y;

  for (let i = 1; i <= 360; i++) {
    circle.push(Point(Math.round(x), Math.round(y)));
    x = center.x + (r * Math.cos((Math.PI * i) / 180));
    y = center.y + (r * Math.sin((Math.PI * i) / 180));
  }

  return circle;
}
```

b. Algoritmo Incremental com Simetria.

```
/* -- Metodo Incremental -- */
export const increment = (center, r) => {

  /* Armazena pontos da Circuferencia */
  let circle = [];

  if (r === 0) {
    return circle;
  }

  /* Define o Angulo do Incremento */
  let theta = (1 / r);

  /* Sen e Cos */
  let cos = Math.cos(theta);
  let sin = Math.sin(theta);

  /* X e Y Inicial */
  let x = r;
  let y = 0;

  /* Adiciona Pontos Iniciais */
  circle = circle.concat(circle, joinOctantes(Math.round(x), Math.round(y),
center.x, center.y));

  // Calcula os pontos do Primeiro Octante
  // Por Simetria replica nos demais
  while (x > y) {

    /* valor temporario de x */
    let xn = x;

    /* Calcula nova coordenada x,y */
    x = (xn * cos) - (y * sin);
    y = (y * cos) + (xn * sin);

    /* Armazena pontos da circunferencia */
    circle = circle.concat(circle, joinOctantes(Math.round(x),
Math.round(y), center.x, center.y));

  }

  return circle;
}
```

c. Algoritmo Bresenham.

```
/* -- Metodo Bresenham -- */
export const bresenham = (center, r) => {

  /* Armazena pontos da Circuferencia */
  let circle = [];

  if (r === 0) {
    return circle;
  }

  /* X e Y Inicial */
  let x = 0;
  let y = r;
  let p = 1 - r;

  // Calcula os pontos do Primeiro Octante
  // Por Simetria replica nos demais
  while (x <= y) {

    circle = circle.concat(circle, joinOctantes(Math.round(x),
Math.round(y), center.x, center.y));

    if (p >= 0) {
      y = y - 1;
      p = p + 2 * x - 2 * y + 5;

    } else {
      p = p + 2 * x + 3;
    }

    x++;
  }

  return circle;
}
```

d. Método Auxiliar para gerar coordenadas dos demais octantes por simetria.

```
/* Funcao auxiliar para Juntar outros octantes por simetria */
export const joinOctantes = (x, y, xc, yc) => {

  let circle = [];

  // Primeiro Octante
  circle.push(Point(x + xc, y + yc));

  // Segundo Octante
  circle.push(Point(y + xc, x + yc));

  // Terceiro Octante
  circle.push(Point(-y + xc, x + yc));

  // Quarto Octante
  circle.push(Point(-x + xc, y + yc));

  // Quinto Octante
  circle.push(Point(-x + xc, -y + yc));

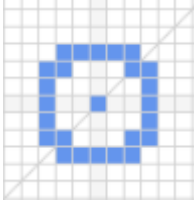
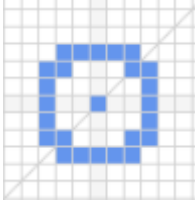
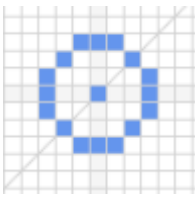
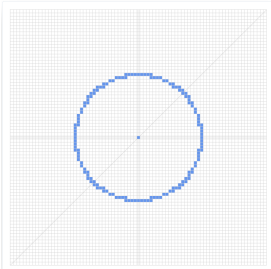
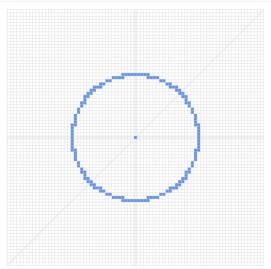
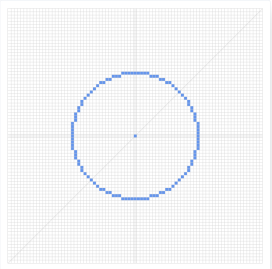
  // Sexto Octante
  circle.push(Point(-y + xc, -x + yc));

  // Setimo Octante
  circle.push(Point(y + xc, -x + yc));

  // Oitavo Octante
  circle.push(Point(x + xc, -y + yc));

  return circle;
}
```

5. Comparação

Coordernada	Paramétrico	Incremental	Bresenham
centro: (0,0) raio: 3			
centro: (0,0) raio: 20			

6. Considerações.

- Algoritmo da Equação Paramétrica, foi o que apresentou mais facilidade de implementação, no entanto, caso o parametro de paço seja definido com um valor alto, a circunferência apresenta descontinuidades, caso o valor seja muito baixo exige muito

processamento, nesse caso, optou-se por estabelecer o valor 1. Também aplicou-se arredondamento para evitar problemas ao plotar o ponto na mapa de pixel.

- b. Algoritmo Incremental com Simetria, corrige os problemas de continuidade, e utiliza a simetria da circunferência e só é necessário gerar as coordenadas para um octante, e utilizar simetria para as demais octantes. Pode apresentar problema de arredondamento, por isso é aplicado arredondamento.
- c. Algoritmo de Bresenham, também se utiliza da simetria da circunferência e busca as coordenadas apenas para um octante, e a partir desse se aplica aos demais octantes. Tem como principal vantagem a utilização somente de aritmética de inteiros.
- d. Não consideramos para circunferência raios negativos, e em caso de raio = 0, então retorna nenhuma coordenada.
- e. Para replicar as coordenadas por simetria, foi implementado um método a parte, que é utilizado tanto pelo algoritmo incremental quanto pelo algoritmo de Bresenham.
- f. O controle limita o raio ao valor máximo de 20 unidades, pois ao passar esse valor o custo computacional torna-se tão alto que o computador utilizado não consegue processar.