

**UNIVERSIDADE FEDERAL DA PARAÍBA**

**DEPARTAMENTO DE INFORMÁTICA**

**INTRODUÇÃO A COMPUTAÇÃO GRÁFICA**

**Aluno**: Everaldo da Costa Mendes

**Matrícula:** 20160112738

**TRABALHO 1 – RASTERIZAÇÃO EM C/C++**

João Pessoa - PB

Fevereiro - 2019

**Introdução:**

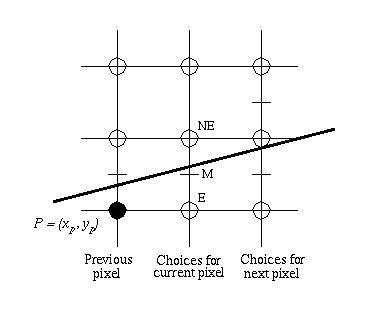
Nesse relatório iremos demonstrar como foi implementado um algoritmo para a rasterização de pontos e linhas.

A rasterização destas primitivas será feita através da escrita direta na memória. Como os sistemas operacionais atuais protegem a memória quanto ao acesso direto, é utilizado um framework, fornecido pelo professor, que simula o acesso à memória de vídeo.

**Algoritmo de Bresenham:**

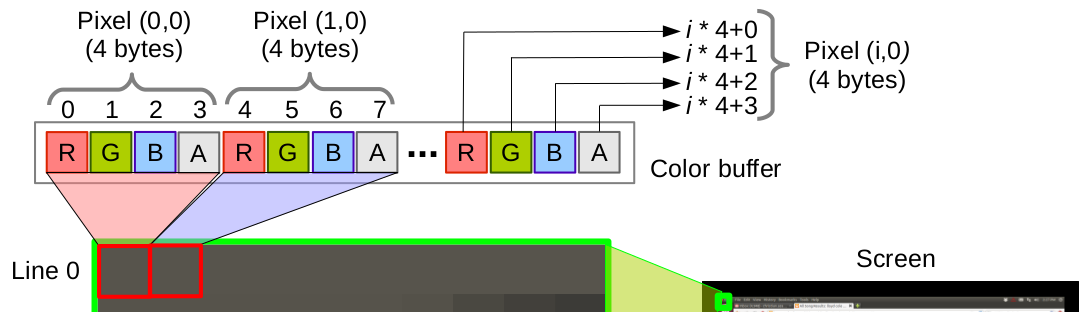
O algoritmo de Jack Elton Bresenham, é baseado na ideia do ponto médio. E tem como objetivo selecionar o número mínimo de pixel para a linha ficar fina, buscando seguir o modelo matemático.

Cada círculo da imagem abaixo representa o ponto central de um pixel, entre os círculos existe um ponto médio (Midpoint M) exatamente no meio entre cada círculo, baseado nesse ponto o algoritmo decide qual pixel será escolhido para ser aceso. Caso a reta passe por baixo do ponto médio, a próxima posição com relação ao pixel atual (x, y) será (x+1, y), por outro lado, se a reta passar por cima, o próximo pixel será o (x+1, y+1), garantindo que o próximo pixel a ser aceso será o mais próximo da reta.



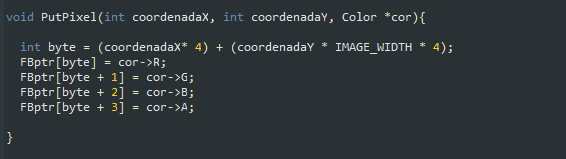
**Ponto**

Um pixel na memória dos nossos computadores são, normalmente, representados por 4 bytes de informação, cada byte representa respectivamente Red (Vermelho), Blue (Azul), Green (verde) e Alpha(transparência ou brilho). Estes 4 componentes misturados produzem uma gama de cores grande e comumente o suficiente para os trabalhos realizados nos computadores.

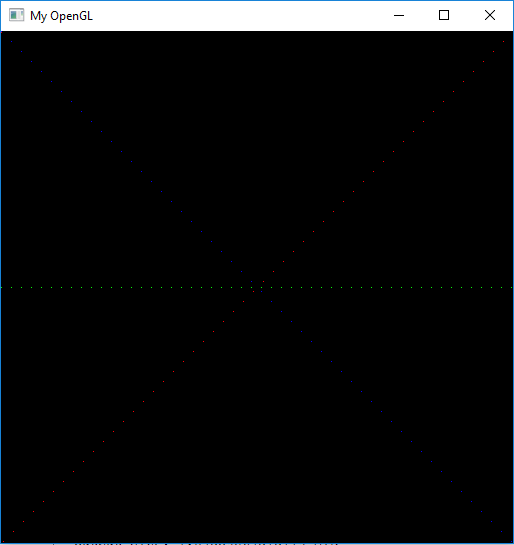


Na memória do computador a tela é representada de forma contínua, ela é uma linha que tem 4\*largura\*altura bytes de tamanho, como queremos pintar um pixel de coordenadas (x, y) é necessário fazer 4\*x + 4\*y\*largura.

Colocando os conceitos em algoritmo, temos o exemplo anterior onde Fbptr é um ponteiro para o início da memória, e foi criado duas struct, Color que contém os componentes RGBA e Coordenadas que contém as coordenadas (x, y).



Resultado:



**Linha com Interpolação de Cores**

A implementação da linha foi baseada no algoritmo de Bresenham para achar o ponto médio.

void DrawLineBresenham(Coordenadas \* p0, Coordenadas \* p1, Color \* corInicial, Color \* corFinal)

{

int dX = p1->coordenadaX - p0->coordenadaX,

dY = p1->coordenadaY - p0->coordenadaY,

x = 1, y = 1,

d,coordX, coordY,i;

if(dX < 0) x = -x;

if(dY < 0) y = -y;

dX = abs(dX);

dY = abs(dY);

coordX = p0->coordenadaX;

coordY = p0->coordenadaY;

if (dX > dY) {

for(i = 0; i < dX; i++) {

PutPixel(coordX,coordY,corAuxiliar);

if(d < 0)

d += dY << 1;

else {

coordY += y;

d += (dY - dX) << 1;

}

coordX += x;

}

}

else {

d = (dX << 1) - dY;

for(i = 0; i < dY; i++) {

PutPixel(coordX,coordY,corAuxiliar);

if(d < 0)

d += dX << 1;

else {

coordX += x;

d += (dX - dY) << 1;

}

coordY += y;

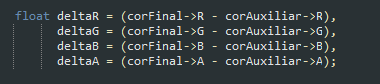
}

}

}

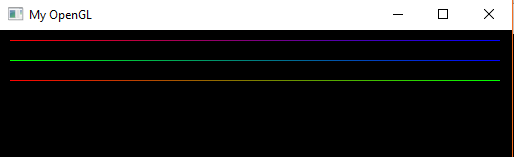
O método de Interpolação foi feito utilizando-se de duas cores que que foram pedidos como parâmetro, uma inicial e outra final, assim que recebemos nós usamos uma variável auxiliar para receber a cor inicial.

Em seguida, determinamos o deltaR, deltaG, deltaB e deltaA, que é a variação de cor do ponto inicial ao ponto final.



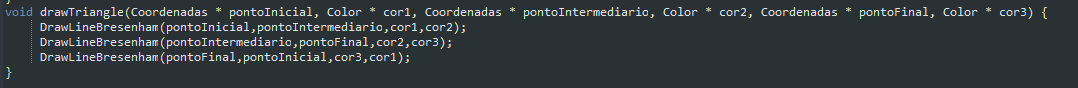
Após isso dividimos todos esses deltas obtidos pelo dX que é a variação da linha, e obtemos a quantidade de cor que varia de pixel para pixel de acordo com o desenho da linha.

Resultado:

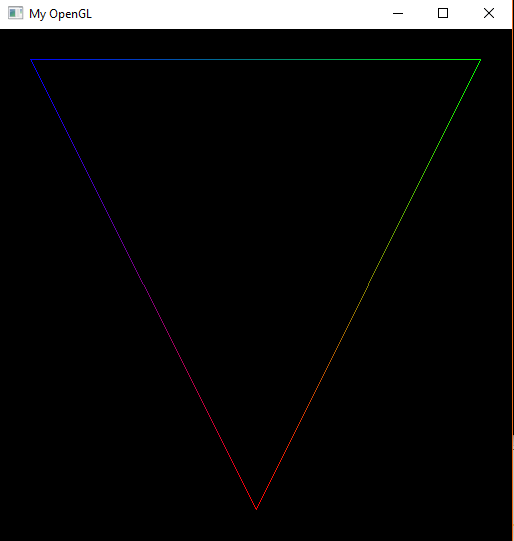


**Triângulo Usando Interpolação de Cores:**

O triangulo é bem mais fácil de fazer, pois ele é baseado no algoritmo da linha, vamos fazer 3 chamadas para desenhar 3 linhas, a partir dos vértices passados nos parâmetros.



Resultado:



**Problemas Encontrados**

Um dos principais problemas foi a implementação da interpolação de cores onde ocorreu diversas falhas para encontrar a variação de cores de pixel para pixel.

O principal problema foi o entendimento do algoritmo de Bresenham que requereu estudo sobre o assunto para conseguir implementar corretamente. Além desse houveram outros como a implementação da interpolação de cores onde ocorreu diversas falhas para encontrar a variação de cores de pixel para pixel.

**Referencias**

- <https://medium.com/@filhojoseildo/implementa%C3%A7%C3%A3o-de-algoritmos-de-rasteriza%C3%A7%C3%A3o-ef413aaccf3d>

- <https://medium.com/@filhojoseildo/implementa%C3%A7%C3%A3o-de-algoritmos-de-rasteriza%C3%A7%C3%A3o-ef413aaccf3d>

- <https://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm>

- <http://letslearnbits.blogspot.com/2014/10/icgt1-interpolacao-de-cores.html>

- Slides do professor Cristian