

**UNIVERSIDADE FEDERAL DE VIÇOSA**  
**DEPARTAMENTO DE INFORMÁTICA**  
**Inf 390 – Computação gráfica**  
**Prof. Salles Magalhães**

**Trabalho 1 (parte 1)**

O trabalho poderá ser feito em dupla

Regras gerais de trabalhos de INF390:

<https://docs.google.com/document/d/1yvvr6TrK3B9wBg7rHQNwHIJZ34I01rBtq0P7wFSrRA/edit> (atenção: **leia TUDO**, mesmo se já tiver cursado alguma disciplina comigo)

Você deverá criar um programa em C++ capaz de processar um arquivo descrevendo uma imagem (ou seja, uma imagem em formato vetorial) e, a partir disso, gerar uma imagem em formato raster (no formato PPM). Apenas funções e classes da biblioteca padrão do C++/STL poderão ser utilizadas.

O arquivo de entrada começará sempre com uma linha contendo dois inteiros W H separados por um espaço em branco. Tais valores representam, respectivamente, a largura e altura da imagem (seus pixels terão coordenadas y,x entre 0,0 e H-1,W-1). Inicialmente você deverá criar uma imagem em branco com essas dimensões.

A seguir, há uma série de comandos (um por linha) representando os componentes da imagem. Você deverá interpretar tais comandos na ordem em que eles aparecem e aplicá-los à imagem atual (é importante seguir a ordem, já que alguma forma pode se sobrepôr a outra). Quando chegar no final de arquivo (ou seja, quando a imagem terminar), a imagem rasterizada deverá ser salva na saída padrão (cout/stdout) utilizando o formato descrito abaixo.

Obs:

- Algum comando poderá levar ao preenchimento de pixels fora da área da imagem. Nesse caso, tais pixels não deverão ser preenchidos (exemplo: se um retângulo estiver parcialmente fora da imagem → apenas a parte do lado de dentro deverá ser rasterizada).
- Cálculos de ponto flutuante devem ser feitos utilizando variáveis double.
- Teste bastante seu algoritmo para garantir que ele está funcionando corretamente! Códigos vistos em sala podem estar simplificados demais por motivos didáticos e, assim, tais algoritmos podem precisar ser melhorados.
- Você deverá ENTENDER e, depois, implementar todos algoritmos (não copie e cole código, mesmo os vistos em sala!).
- Formatos de imagens vetoriais funcionam basicamente dessa forma. Um exemplo é o formato SVG, que é basicamente um XML descrevendo a imagem (veja o exemplo de imagem e XML disponível aqui:

[https://en.wikipedia.org/wiki/Scalable\\_Vector\\_Graphics#:~:text=Scalable%20Vector%20Graphics%20\(SVG\)%20is, support%20for%20interactivity%20and%20animation.&text=SVG%20images](https://en.wikipedia.org/wiki/Scalable_Vector_Graphics#:~:text=Scalable%20Vector%20Graphics%20(SVG)%20is, support%20for%20interactivity%20and%20animation.&text=SVG%20images)

[%20and%20their%20behaviors.indexed%2C%20scripted%2C%20and%20compressed. \).](#)

Poderíamos ter criado um rasterizador de SVG, mas isso poderia complicar um pouco o trabalho e tirar o foco da rasterização.

## Comandos

No caso de figuras geométricas, elas deverão ser desenhadas sem preenchimento (ou seja, apenas a borda com 1 pixel deverá ser desenhada). Os valores (r,g,b) indicam a cor da borda dessas figuras. Todos valores são inteiros.

- pixel y x r g b

Marca o pixel das coordenadas (y,x) com a cor (r,g,b).

- retangulo y1 x1 y2 x2 r g b

Desenha um retângulo cujos pontos extremos são (y1,x1) e (y2,x2).

- segmento y1 x1 y2 x2 r g b

Desenha um segmento ligando o ponto (x1,y1) ao ponto (x2,y2). Tal segmento deverá ser rasterizado utilizando o algoritmo de Bresenham.

- segmentosimples y1 x1 y2 x2 r g b

Mesmo que o anterior, mas utilizando o algoritmo simples (melhorado) de rasterização visto em sala.

- polyline N y1 x1 y2 x2 y3 x3 .... yN xN r g b

Desenha uma *polyline* composta pelos N-1 segmentos (y1,x1 - y2,x2) ; (y2,x2 - y3,x3) ; ( y3,x3 - y4,x4) ; .... (y(N-1),x(N-1) - yN xN). Cada segmento deverá ser rasterizado utilizando o algoritmo de Bresenham. N será pelo menos 2 (ou seja, a polyline será no mínimo um segmento).

- circulo y x raio r g b

Desenha um círculo com centro (y,x) e raio “raio”. O círculo deverá ser rasterizado utilizando o algoritmo de Bresenham.

- preencher4 y x r g b

Preenche a região conexa a (y,x) com a cor (r,g,b) utilizando vizinhança “4-conexa”. Pixels vizinhos são conexos se possuem a mesma cor.

- preencher8 y x r g b

Mesmo que o anterior, mas utiliza vizinhança “8-conexa”.


## Saída

Após ler todos comandos (ou seja, chegar ao final do arquivo de entrada), você deverá gravar a imagem em um arquivo utilizando o formato descrito abaixo. O caminho do arquivo será o primeiro argumento do seu programa (ou seja, ele será executado com um comando similar a “./a.out saída.pbm < entrada.txt”)

A primeira linha de sua saída deverá conter apenas “P3”. A segunda linha deverá conter “W H” (ou seja, a largura e altura da imagem). A terceira linha deverá conter 255.

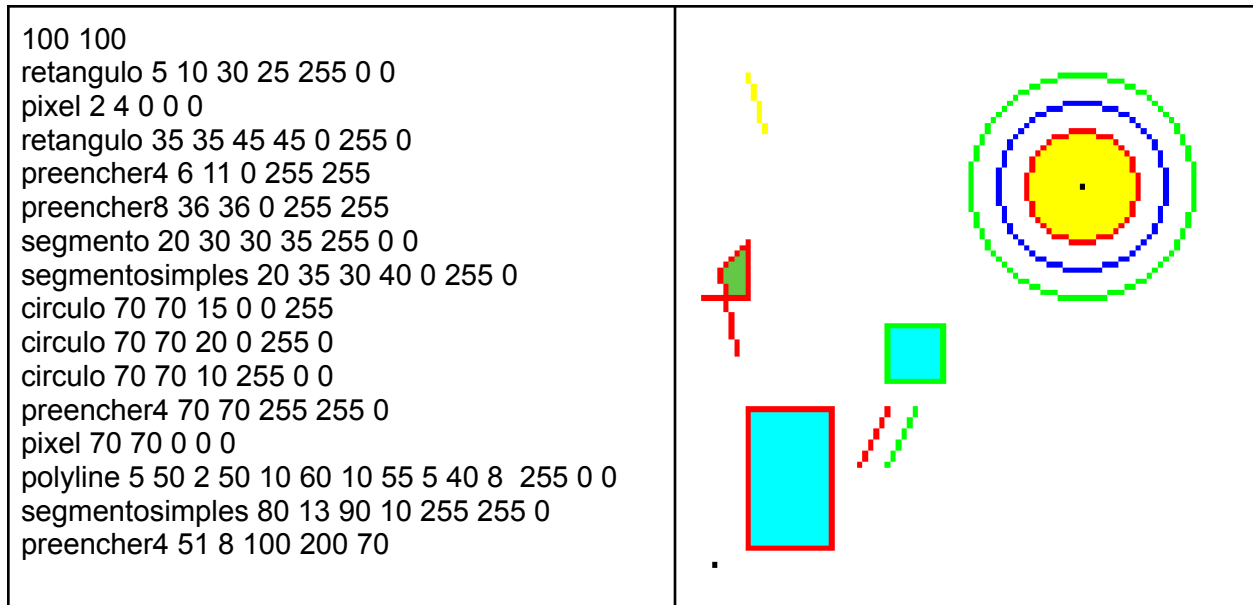
Finalmente, imprima todos os pixels da imagem. O primeiro pixel a ser impresso deverá ser o do canto superior esquerdo da imagem. O segundo será o pixel à direita dele, e assim por diante. Ao imprimir um pixel, imprima as três coordenadas (r,g,b). Imprima um espaço em branco após cada número.

Exemplo de saída: imagem com 2 pixels de largura e 3 de altura. O pixel do canto superior esquerdo é vermelho, o à sua direita é verde. Os dois do meio são brancos e os dois de baixo são pretos.

<pre>P3 2 3 255 255 0 0 0 255 0 255 255 255 255 255 255 0 0 0 0 0 0</pre>	
---	--

Se essa saída for salva com a extensão “.ppm” ela poderá ser visualizada utilizando algum visualizador de imagem! (isso pode ser feito facilmente no Linux -- no Windows o suporte não é tão bom). A string “P3” é um “código mágico” utilizado para determinar o formato do arquivo (há outras versões do ppm -- por exemplo, versão onde os pixels estão codificados em um formato binário, sendo mais compacto mas menos legível por humanos).

## Exemplo de entrada e saída



## Arquivo README

Seu trabalho deverá incluir um arquivo README.

Tal arquivo conterá:

- Nome/matricula
- Informações sobre fontes de consulta utilizadas no trabalho
- Escreva um parágrafo descrevendo as principais diferenças que você achou entre o formato vetorial e o raster. Quais são as vantagens e desvantagens de cada um?

## Entrega

O trabalho deverá ser entregue pelo Submittity. O código deverá estar em um arquivo main.cpp (ele será compilado utilizando o comando “g++ main.cpp”)

## Avaliacao

Seu programa deverá compilar/funcionar no Linux (sugere-se que você o desenvolva no Linux -- você deverá pelo menos testá-lo nesse Sistema Operacional antes de entregá-lo).

## Duvidas

Dúvidas sobre este trabalho deverão ser postadas no sistema Piazza. Se esforce para implementá-lo e não hesite em postar suas dúvidas!

# Avaliacao do codigo

O código do seu trabalho também será avaliado. Tente mantê-lo bem organizado, com comentários, etc.

**Adicione comentários explicando os passos de cada algoritmo implementado.**