

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA
E TECNOLOGIA DE MATO GROSSO DO SUL
CAMPUS JARDIM**



**M E T A
P O S T**

UMA INTRODUÇÃO AO ESTUDANTE

**NICOLE RODRIGUES CUELLAR
ESTEVÃO VINÍCIUS CANDIA**



SOBRE A AUTORA

Oi, meu nome é Nicole, tenho 16 anos e sou estudante do curso técnico integrado em Informática do ITMS, Campus Jardim.

Minha primeira impressão sobre a linguagem MetaPost foi um misto de curiosidade com um certo medo. Mas ao começar a estudar um pouco a linguagem, entender um pouco mais de como funcionava, fui percebendo que, com certeza, não seria algo impossível de se aprender. Quando comecei a entender melhor a sintaxe dos códigos, tudo começou a fluir mais facilmente e isso gerou um sentimento agradável. Acredito que seja muito possível estudantes de ensino médio (como eu) aprenderem a utilizar o MetaPost.

Espero que no decorrer do estudo dessa apostila, você também se surpreenda e aprenda Matemática utilizando esta linguagem.

Bons estudos!

SUMÁRIO

Introdução	4
Mas o que, de fato, é o MetaPost?	4
Seção 1 – Figuras geométricas no MetaPost	7
Tópico 1: Triângulo	7
Tópico 2: Retângulo	9
Tópico 3: Circunferência	10
Tópico 4: Quadrado	11
Seção 2 – Decorando as figuras no MetaPost	15
Variáveis de substituição	15
Tipos de caneta	17
Tipos de traçado	18
Rótulos	20
Cores	23
Seção 3 – Exercícios para praticar	26
Exercício 1	26
Exercício 2	29
Exercício 3	31
Considerações finais	32
Referências bibliográficas	34
Apêndice – Resposta dos exercícios	35

Introdução

Esse trabalho tem como objetivo apresentar um conhecimento básico sobre o que é o MetaPost e como é o seu funcionamento.

Na primeira seção serão abordadas algumas figuras básicas e seus códigos. Nessa parte é explicado o conceito matemático de cada figura, é apresentado também um código e sua figura e alguns exercícios para treinamento.

Na segunda seção haverá mais algumas explicações e exemplos de códigos e figuras além das básicas já citadas e mais alguns exercícios. Serão explicadas como decorar as figuras com cores, preenchimento, além de como funcionam os rótulos (como colocá-los, quais os comandos, entre outros).

Na terceira seção, por fim, são apresentados alguns exercícios para que os conhecimentos abordados na apostila sejam colocados em prática.

Mas o que, de fato, é o MetaPost?

MetaPost é uma linguagem gráfica derivada do METAFONT. O METAFONT foi criado por Donald Knuth, porém o MetaPost foi desenvolvido, cerca de dez anos depois, por John Douglas Hobby, que seguiu com as pesquisas de Knuth sobre o METAFONT, chegando a uma versão melhorada, o MetaPost. Essa linguagem foi desenvolvida para gerar imagens e é baseada em cálculos matemáticos.

O MetaPost gera imagens vetoriais, que são imagens formadas a partir de vetores matemáticos, ou seja, ao invés de termos o mapeamento de cada um dos pixels da imagem, teremos uma fórmula que indica uma forma. Isso significa que imagens vetoriais não possuem pixels.

Essas imagens são diferentes das imagens rasterizadas (bitmap), que é o formato mais comum de imagem, o qual é como um mapa que indica a posição de cada pixel que compõe a imagem. Sendo assim, as imagens vetoriais acabam sendo melhores, pois podem ser redimensionadas o quanto for necessário e mesmo assim não perdem sua qualidade.

Existe um site com um MetaPost Previewer, isto é, um pré-visualizador de saídas de códigos MetaPost, que permite aos usuários testar seus códigos. Esse site foi desenvolvido pelo professor Troy Henderson, da Universidade de Mobile, Alabama, EUA, e auxilia muito o trabalho com o MetaPost. No site, há uma caixa de texto, onde deve-se colocar os comandos, e ao lado há o espaço onde são mostradas as figuras geradas pela implementação do código escrito. Caso haja algum erro no código, não se obtém o resultado esperado, mas sim uma mensagem informando o equívoco.

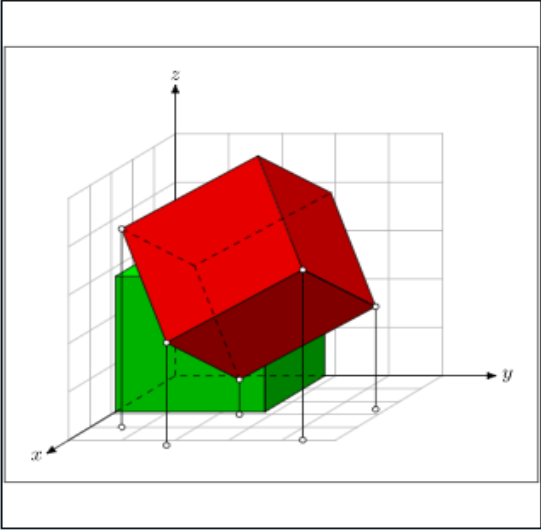
MetaPost Preview

by **Troy Henderson**
MetaPost version 2.000 (TeX Live 2017) (kpathsea version 6.2.3)

```

prologues:=3;
verbatimx
%&latex
\documentclass{minimal}
\begin{document}
etex
beginfig(0);
u:=25; % 25 = 25bp = 25 PostScript points = 30/72 in
wi:=10; % width in units u
he:=9; % height in units u
hoehe:=he*u; % height
breite:=wi*u; % width
transform t, Ixy;
pair P[];
path p[];
color wuerfel[], versch; % 3D vectors: MetaPost type
"color"
rotX:=45; % angle of rotation around the x axis
rotY:=45; % angle of rotation around the y axis
rotZ:=45; % angle of rotation around the z axis
versch:=(1.6, 1, 2.7); % translation (in
mathematical units)
P0=(3.2, 2.2)*u; % origin in MetaPost coordinates
(bp)
P1=(-.6, -.4)/1.5; % x axis in mathematical
endfig;
end

```



SVG
 PNG

Figura 1 – Confira: <http://www.tlhiv.org/mppreview/>

SEÇÃO 1 – Códigos para figuras geométricas no METAPOST

Nesta seção serão abordadas quatro figuras geométricas básicas que provavelmente você já conhece. Primeiramente são apresentados os conceitos matemáticos de cada uma delas, e em seguida o código que é utilizado para gerar essa figura no MetaPost. Nesta mesma seção também serão colocados alguns exercícios ao final de cada tópico de figura para que você, estudante, tente desenvolver o que foi visto.

Tópico 1: Triângulo

Triângulos são polígonos formados por três lados. Os polígonos, por sua vez, são figuras geométricas formadas por segmentos de reta que, dois a dois, tocam-se em seus pontos extremos, mas que não se cruzam em qualquer outro ponto.

Os triângulos são classificados quanto às medidas dos lados de três maneiras:

- Escaleno: triângulo que possui todos os lados com medidas diferentes;
- Isósceles: triângulo que possui dois lados com medidas iguais e o terceiro com medida diferente;
- Equilátero: triângulo que possui três lados com medidas iguais.

No MetaPost, quando é necessário fazer um segmento de reta, normalmente é usada a sintaxe --. Esses dois tracinhos servem para criar um

caminho retilíneo entre dois pontos. Isso se dá porque o MetaPost trabalha com coordenadas cartesianas para identificar a localização dos pontos. Dessa forma, o código a seguir gera um segmento de reta de 2 cm:

```
draw (0,0)--(2cm,0);
```



Figura 2 – Segmento de reta

Note que o código é formado por duas componentes. A primeira é o comando `draw` que é utilizado para desenhar caminhos. A segunda parte é `(0,0)--(2cm,0)` que é um caminho reto entre a origem, ponto $(0,0)$, e o ponto $(2\text{cm},0)$. É fundamental utilizar unidades de medida para os desenhos. O MetaPost aceita unidades como mm, cm, etc. Também, vale frisar que a sintaxe `--` precisa estar junta, sem espaços, para que o MetaPost a interprete corretamente. Para finalizar uma linha de comando, é preciso colocar o ponto-e-vírgula `;` ao final para que o MetaPost interprete que o comando está completo e faça o desenho solicitado.

Com isso em mente, um exemplo de comando para desenhar um triângulo é:

```
draw (0,0)--(3cm,0)--(0,3cm)--(0,0);
```

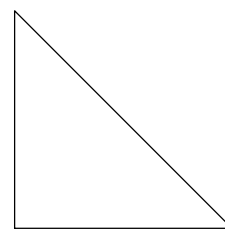


Figura 3 – Triângulo

O desenho é feito a partir dos vértices do triângulo. Perceba que o triângulo possui três vértices, sendo estes $(0,0)$, $(3\text{cm},0)$ e $(0,3\text{cm})$. Os três segmentos podem ser desenhados de uma só vez em uma mesma linha de código. Assim, $(0,0)--(3\text{cm},0)$ gera o cateto horizontal, $(3\text{cm},0)--(0,3\text{cm})$ a hipotenusa e $(0,3\text{cm})--(0,0)$ o cateto vertical. Para finalizar o comando, basta colocar o `;` e, voilá: o triângulo está desenhado.

Exercício 1:

Utilizando o MetaPost, desenhe um triângulo com dois lados medindo 6 cm cada.

Tópico 2: Retângulo

Um retângulo é uma figura geométrica plana formada por quatro lados (quadrilátero) e apresenta os quatro ângulos internos congruentes (mesma medida) e retos (90°).

```
draw (0,0)--(0,2cm)--(4cm,2cm)--(4cm,0)--cycle;
```

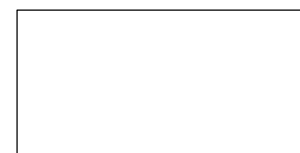


Figura 4 – Retângulo

Nesse exemplo, o comando desenha os lados do retângulo um por um, de acordo com as medidas. Iniciando no ponto $(0,0)$, depois partindo para o segundo ponto $(0,2\text{cm})$, em seguida para o terceiro $(4\text{cm},2\text{cm})$, logo depois

$(4\text{cm}, 0)$ e no final, é usado o comando `cycle`, que tem como objetivo fechar a figura, voltando ao ponto inicial. Esse comando é importante pois será usado para preencher figuras com cores. Isto será explicado na próxima seção.

Exercício 2:

Desenvolva um retângulo com 4 cm de comprimento e 6 cm de largura.

Tópico 3: Circunferência

A circunferência é uma figura geométrica plana formada pela união de pontos equidistantes, ou seja, possuem a mesma distância de um ponto fixo chamado de centro.

Circunferência e círculo não são a mesma coisa, há uma diferença muito importante entre eles: a circunferência é a borda do círculo. Isso significa que a circunferência é uma figura unidimensional, ou seja, é uma linha. Já o círculo é uma figura plana, bidimensional (possui duas dimensões).

Existe um comando específico para desenharmos uma circunferência no MetaPost, o `fullcircle`, que se refere a um caminho circular fechado de diâmetro unitário centrado na origem.

Exemplo:

```
draw fullcircle scaled 3cm;
```

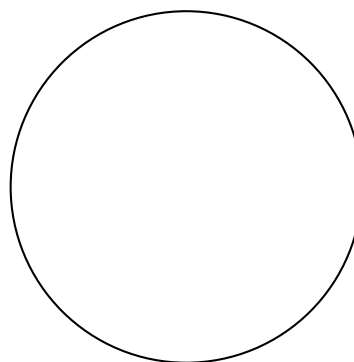


Figura 5 – Circunferência

Note que neste código foi utilizado o comando `scaled`. Esse comando é uma transformação que o MetaPost possui que pode ser aplicada a várias variáveis diferentes. No caso da circunferência, a variável é um caminho, que forma uma circunferência de diâmetro 1. O comando `scaled` multiplica essa medida pelo valor numérico que o acompanha, no caso, 3 cm. Assim, o resultado é uma circunferência que possui diâmetro de 3 cm.

Exercício 3:

Faça uma circunferência com 5 cm de diâmetro.

Tópico 4: Quadrado

Quadrados são figuras geométricas planas que devem possuir as seguintes características:

- deve ser um polígono com quatro lados;
- todos os quatro lados devem ser iguais;

- todos os ângulos internos devem ser retos.

Como já dito antes, quando é necessário fazer um segmento de reta, normalmente é usada a sintaxe --. Nada impede o usuário de utilizar essa sintaxe para construir um quadrado da mesma forma que foram feitos os retângulos. No entanto, assim como para o círculo, existe um comando específico para gerar caminhos quadrados. Este comando é o `unitsquare`, que nada mais é do que um caminho definido por:

```
(0,0)-(1,0)-(1,1)-(0,1)-cycle
```

Este caminho é um quadrado com lado unitário.

```
Draw unitsquare scaled 3cm;
```

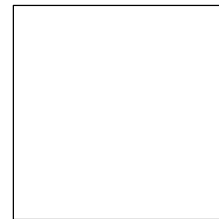


Figura 6 – Quadrado

O desenho, na verdade, é feito aumentando o tamanho do quadrado unitário (`unitsquare`) por meio do comando `scaled` em 3 cm.

Exercício 4:

Utilizando o MetaPost, desenhe um quadrado com os lados medindo 5 cm.

Quando o comando `scaled` é utilizado, ele aumenta proporcionalmente tanto as coordenadas x (horizontal) como as coordenadas y (vertical) de um caminho. No MetaPost, existem comandos predefinidos que servem quando há o interesse de escalar apenas o x ou o y , ou escalá-los com valores diferentes. O exemplo a seguir mostra como fazer isso utilizando `xscaled` e `yscaled`. Esse exemplo mostra como alterar as dimensões de um `unitsquare` de forma que o comprimento e a altura do retângulo a ser formado fiquem diferentes.

```
draw unitsquare xscaled 4cm yscaled 6cm;
```



Figura 7 – Retângulo

Neste exemplo, o `unitsquare` é escalado primeiramente nas coordenadas x por meio do `xscaled`, por 4 cm. Então, o lado unitário acaba se transformando em um lado com 4 cm. Depois, com o `yscaled` transforma as coordenadas y , ou seja, a altura, do quadrado unitário multiplicando por 6 cm. O resultado é um caminho a ser desenhado pelo comando `draw` que contém um retângulo com base medindo 4 cm e a altura medindo 6 cm, conforme ilustrado na Figura 7.

Exercício 5:

Faça um retângulo, usando `xscaled` e `yscaled`, com medida de 2cm por 4cm.

SEÇÃO 2 – Decorando as figuras no MetaPost

Nesta segunda seção, serão citadas e explicadas algumas variáveis utilizadas no MetaPost. Também serão abordados alguns códigos específicos para estilizar as figuras desenvolvidas, como comandos para gerar rótulos, para inserir cores, entre outros.

Variáveis de substituição

Primeiramente, o que são variáveis?

Na programação, definimos variável como um espaço na memória do computador destinado a um dado que é alterado durante a execução do algoritmo, por exemplo, um código do MetaPost. Utiliza-se variáveis para facilitar a escrita de códigos muito extensos. Para declarar uma variável no início do código, você deve escrever o tipo da variável juntamente com uma sequência de letras que será a sua string. Por exemplo, definir no início do código `numeric r;` indica que `r` é uma variável do tipo `numeric`.

A variável mais simples utilizada no MetaPost é a que é chamada de `numeric`. Ela armazena valores numéricos para serem utilizados durante a execução de um código. As letras atribuídas a essa variável não precisam ser declaradas no início, pois quaisquer sequências não declaradas o MetaPost já atribui como sendo numéricas. No entanto, todos os outros tipos de variáveis precisam ser declarados no início para não haver erros.

Para facilitar a escrita de um código com bastante quantidade numéricas repetidas, pode-se utilizar uma variável do tipo `numeric` para substituí-la durante a escrita do código. Por exemplo, definindo no MetaPost a variável numérica “u”, como sendo uma unidade usada para servir de padrão para 1 cm, podemos escrever apenas “u” no código quando for necessário definir o tamanho de segmentos. Veja só:

```
u=1 cm;  
draw (0,0)--(3u, 0)--(4u, 2u)--(u,u)--cycle;  
label.llft("O", (0,0));  
label.bot("A", (3u, 0));  
label.rt("B", (4u, 2u));  
label.ulft("C", (u, u));
```

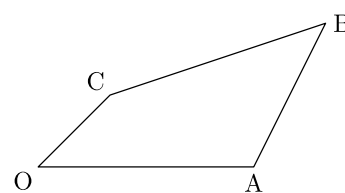


Figura 8 – Quadrilátero

No exemplo acima é possível visualizar como a variável `u` é utilizada para facilitar na hora de escrever o código de uma figura. Note que não foi necessário escrever `numeric u` no começo do código, pois o MetaPost já entende que se refere a uma variável do tipo `numeric`. Você pode utilizar qualquer sequência de letras para definir uma variável como essa. As variáveis que não são aceitas são as que definem comando pré-existentes no MetaPost.

Esse ainda é um código pequeno, mas em códigos maiores, poder usar essa variável de substituição com toda certeza é de grande ajuda.

Tipos de caneta

Ao fazer um desenho com o MetaPost, existe um tipo de variável (tipo `pen`) que define como será feito o traçado das linhas. É como se você escolhesse com que caneta (em inglês, *pen*) você fará o desenho. Existem dois tipos de canetas já pré-configuradas no MetaPost. São elas `pencircle` e `pensquare`. A diferença entre elas é que na `pencircle`, a ponta da caneta é redonda, como se fosse uma caneta esferográfica que é utilizada para escrever em papel. Já na `pensquare` a ponta da caneta tem um formato quadrado. A caneta padrão dos códigos é a `pencircle`. Se você não escolher outro tipo, ela será a pré-selecionada.

O comando `scaled` é usado para definir a largura do traçado da caneta.

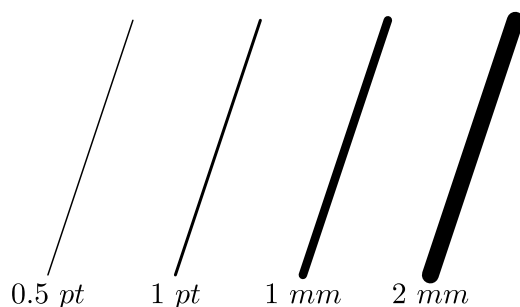


Figura 9 - Grossuras de traçados

A imagem acima mostra de forma bem prática algumas medidas que podem ser utilizadas nas canetas. Agora, você entenderá como utilizá-las num desenho.

Há duas formas simples de escolher a caneta a ser utilizada num desenho do MetaPost. A primeira forma é utilizar o comando `withpen` como um recurso de um comando `draw`. Por exemplo,

```
draw (0,0)--(3cm, 0) withpen pencircle scaled 1 mm;
```

desenha um segmento de 3 cm utilizando a caneta `pencircle` com espessura de 1 mm. Esse comando não altera a caneta utilizada em todo o desenho, apenas nesse comando `draw`. No entanto, se o intuito é utilizar várias vezes a mesma caneta para outras partes da figura, você pode digitar uma linha de código como

```
pickup pencircle scaled 1 mm;
```

que a sua figura será totalmente desenhada com esta caneta para os códigos `draw` desta linha de comando para baixo.

Tipos de traçado

No MetaPost também é possível tracejar as linhas desenhadas por meio do comando `dashed`, que define o tipo de tracejado. Dessa forma,

```
draw dashed
```

Onde `draw` representa o caminho e `dashed` o tipo de tracejado que será escolhido, tendo em vista que os dois estilos mais usados são os traços comuns e os pontilhados. Para desenhar os traços usa-se o comando `evenly`.

```
u=3cm;  
draw (u,0)--(0,u) dashed evenly scaled 2;  
draw (0,u)--(0,0) dashed evenly;  
draw (0,0)--(u,0) dashed evenly scaled 0.5;
```

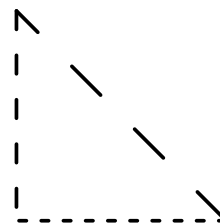


Figura 10 - Triângulo tracejado

O exemplo acima demonstra um código pronto de uma figura tracejada.

É importante lembrar que nesse tipo de código também usamos o comando `scaled` para escalar o tamanho do tracejado que for escolhido. Veja um exemplo com alguns espaçamentos diferentes do padrão:

```
u=3cm;  
draw (0,u)--(0,0) dashed evenly scaled 0.1;  
draw (0.5u,u)--(0.5u,0) dashed evenly scaled 0.3;  
draw (u,u)--(u,0) dashed evenly scaled 0.6;  
draw (1.5u,u)--(1.5u,0) dashed evenly scaled 0.8;  
draw (2u,u)--(2u,0) dashed evenly scaled 1;  
draw (2.5u,u)--(2.5u,0) dashed evenly scaled 2;
```

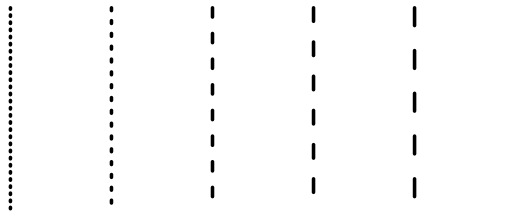


Figura 11 - Espaçamento dos tracejados

Já para desenvolver uma figura pontilhada, é usado o comando `withdots`, dessa maneira:

```
u=3cm;  
draw (0,0)--(u,0) dashed withdots;  
draw (u,0)--(0,u) dashed withdots;  
draw (0,u)--(0,0) dashed withdots;
```

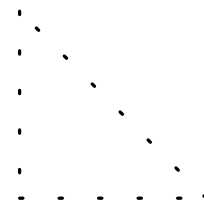


Figura 12 - Triângulo Pontilhado

Da mesma forma, pode ser utilizado o comando `scaled` para que a figura fique mais visível.

```
u=3cm;  
draw(0,0)--(u,0)--(u,u)--(0,u)--cycle  
dashed withdots scaled 0.5;
```



Figura 13 - Quadrado Pontilhado

Note que ao escalar o valor 0.5 o espaçamento entre os pontos diminui e fica mais fácil visualizar o pontilhado, pois aumenta a quantidade de pontinhos.

Rótulos

Para integrar textos as figuras no MetaPost utilizam-se alguns comandos específicos. Como por exemplo o `label`, que é um comando usado para rotular objetos em geral. Tem também o `dotlabel`, que diferentemente do anterior, é específico para marcar pontos dados por suas coordenadas.

A utilização desse comando é feita da seguinte maneira:

```
label.<sufixo> (<nome do rótulo>, <posição>);
```

O comando `dotlabel` tem a sintaxe similar:

```
dotlabel.<sufixo> (<nome do rótulo>, <posição>);
```

A componente `<sufixo>` indica qual será a posição do rótulo em relação ao ponto da imagem em que ele será incluído. Tais sufixos são `lft`, `rt`, `top`, `bot`, `ulft`, `urt`, `llft` e `lrt`. No caso de não haver uso de sufixo algum, o rótulo

é centralizado na <posição> especificada no código. A imagem abaixo ajuda a entender o posicionamento de cada rótulo.

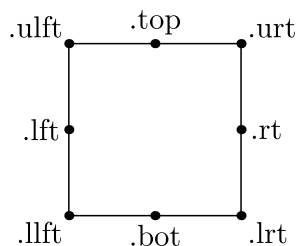


Figura 14 - Posições de Rótulo

A segunda componente do código é o <nome do rótulo>. Para escrever esse nome há duas possibilidades:

1. Escrever o rótulo entre `"`. Ao fazer isso, o MetaPost escreve com letra padrão os caracteres alfanuméricos escritos entre as aspas.
2. Escrever o rótulo iniciando por `btex` e finalizando por `etex`. Esse código ativa o ambiente LaTeX dentro do MetaPost. Assim, é possível escrever símbolos e, também, escrita matemática dentro do desenho. Para isso, é necessário iniciar e finalizar o código com `$`.

E a componente <posição> pode ser escrita de várias formas como utilizando as coordenadas específicas dos pontos, ou utilizando-se de operações vetoriais entre eles. Por exemplo, se forem definidos dois pontos A e B , a coordenada do ponto médio pode ser definida por $(A+B)$.

Aqui vai um exemplo de um segmento de reta rotulado:

```
u=1cm;  
draw (1u,2u)--(3u,5u);  
dotlabel.lft("M", (1u,2u));  
dotlabel.rt("N", (3u,5u));  
label.ulft(btex $\sqrt{13}$~cm$ etex,  
(2u,3.5u));
```

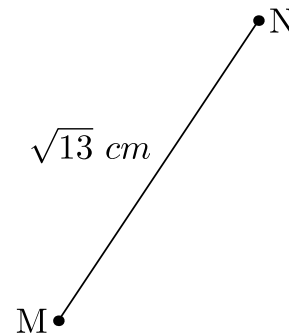


Figura 15 - Segmento de Reta

Este código é iniciado com o desenho de um segmento de reta. Em seguida começa-se a colocar os rótulos na figura, por meio do comando `label` e `dotlabel` junto aos seus sufixos (`lft`, `rt`, `ulft`), que, como dito antes, são utilizados para definir em que posição o rótulo será estabelecido. Como havia interesse em marcar as extremidades do segmento MN , os pontos M e N foram marcados juntos com os seus rótulos. Por isso, foi utilizado o `dotlabel` para essa tarefa. Já o ponto médio não tinha necessidade de ser marcado, pois precisava-se apenas da posição do mesmo para colocar a medida em cm. Dessa maneira, o comando mais apropriado era apenas o `label`. A posição do rótulo foi escrita a partir dos pares ordenados do local em que era desejado colocá-los.

Essa mesma figura poderia ser feita com o código:

```
u=1cm; pair M,N;  
M:=(1u,2u); N:=(3u,5u);  
draw M--N;  
dotlabel.lft("M", M);  
dotlabel.rh("N", N);  
label.ulft(btex $\sqrt{13}~cm$ etex,  
(M+N)/2);
```

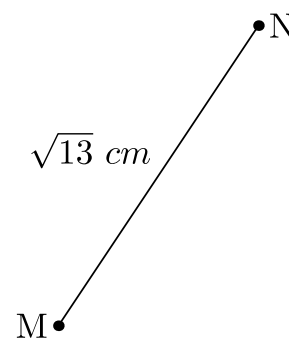


Figura 16 - Segmento de Reta

Note que no começo do código foram declaradas duas variáveis M e N do tipo `pair`. Esse tipo de variável refere-se a pares ordenados. Utilizar essa variável simplifica o código e ajuda a fazer os cálculos necessários, como o do ponto médio.

Cores

Na linguagem MetaPost, também é possível usar outras cores além de preto, tanto para fazer linhas coloridas, ou até mesmo preencher a área das figuras. Para isso, é utilizado o comando `fill` que preenche caminhos e curvas fechadas, como aqueles comandos que usam `cycle` no final. O uso do `cycle` ao final dos caminhos a serem pintados é obrigatório, pois senão o comando `fill` não funcionará.

O MetaPost já possui algumas cores pré-programadas, e assim como todos os outros comandos, essas cores são definidas em inglês. As básicas são `blue`, para azul; `green`, para verde; `red`, para vermelho.

Veja o exemplo:

```
u=2cm;  
fill fullcircle scaled 2u withcolor blue;  
draw fullcircle scaled 2u;
```

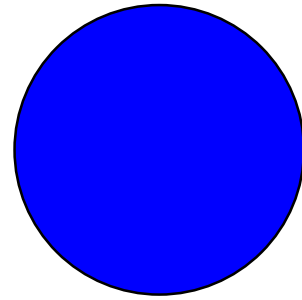


Figura 17 - Círculo azul

Também é possível utilizar o código RGB das cores, para determinar tonalidades diferentes das primárias. Por exemplo, a cor lilás, em RGB, é (200, 162, 200). O MetaPost possui um tipo para definir cores, a partir da quantidade de vermelho, verde e azul delas. Para isso, basta escrever uma tripla ordenada nessa ordem com a porcentagem de cada valor. Como os números de um código RGB vão de 0 a 255, basta dividir as três componentes por 255 no código. Veja o exemplo a seguir.

```
u=1cm;  
fill unitssquare scaled 3u  
withcolor (200/255, 162/255,  
200/255);  
draw unitssquare scaled 3u;
```

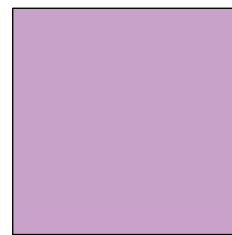


Figura 18 - Quadrado Lilás

Tons de cinza podem ser descritos a partir de um número decimal entre 0 e 1, sendo 0 a cor preta e 1 a cor branca.

Veja a imagem a seguir com diversas tonalidades.

```
u=1cm;  
fill unitsquare scaled 4u withcolor 0.2;  
fill unitsquare scaled 3u withcolor 0.4;  
fill unitsquare scaled 2u withcolor 0.6;  
fill unitsquare scaled 1u withcolor 0.8;  
draw unitsquare scaled 1u;  
draw unitsquare scaled 2u;  
draw unitsquare scaled 3u;  
draw unitsquare scaled 4u;
```

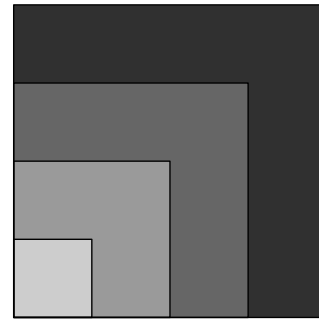


Figura 19 - Tonalidades de Cinza

Viu só quantas possibilidades existem no MetaPost? E tudo isso, na verdade, é só uma parte de tudo que conseguimos desenvolver nessa linguagem tão ampla. Para conhecer mais, acesse a página oficial do MetaPost, lá você encontrará o manual. Não deixe também de pesquisar trabalhos sobre o assunto, caso tenha curiosidade. Aproveite!

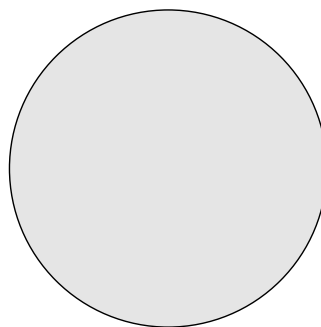
SEÇÃO 3 – Exercícios para praticar

Nesta terceira seção, serão disponibilizados alguns exercícios para praticar os conhecimentos que foram apresentados no decorrer da apostila. Para desenvolvê-los, você deve usar o MetaPost Previewer.

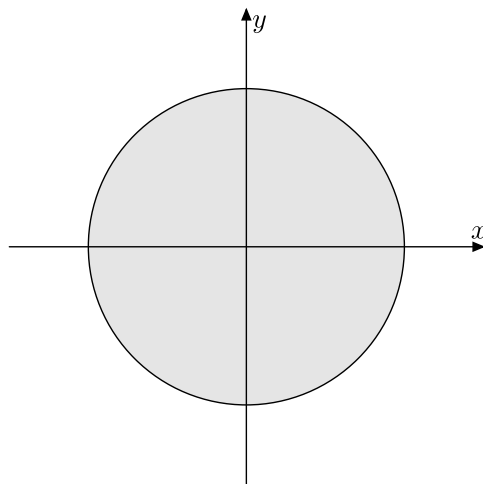
Exercício 1

Neste exercício você criará um ciclo trigonométrico com a marcação de um ponto.

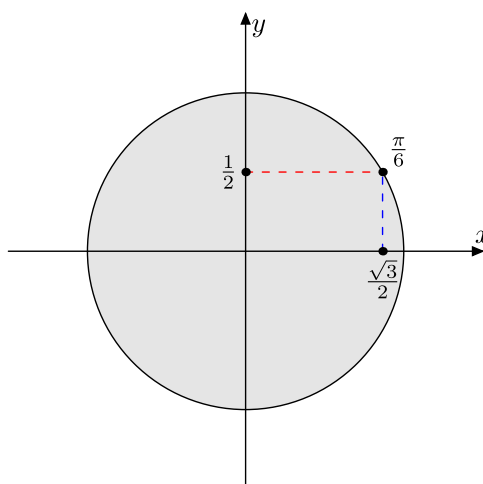
- (a) Para começar, crie um círculo com 2 cm de raio e o preencha com a cor 0.9.



(b) Trace os dois eixos do plano cartesiano e coloque os rótulos (x e y) nos seus devidos lugares. Dica: Você pode desenhar um segmento de reta com seta utilizando o comando `drawarrow` da mesma forma que usa o `draw`.

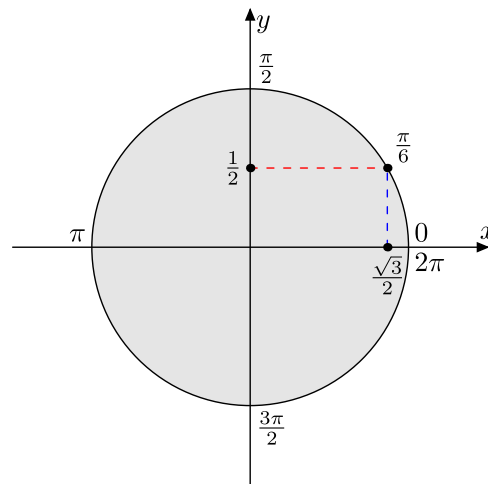


(c) Marque o ponto $\frac{\pi}{6}$ no ciclo trigonométrico e indique o valor de seu seno e cosseno.



(d) Marque os pontos que separam os quadrantes da circunferência:

$$0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}, 2\pi.$$



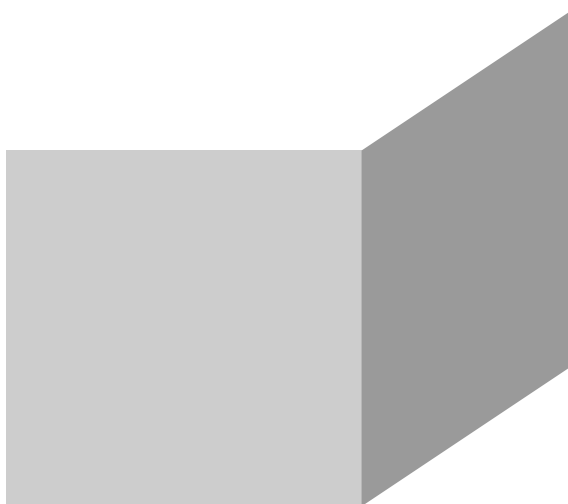
Exercício 2

Neste exercício, você fará a representação de um cubo sombreado.

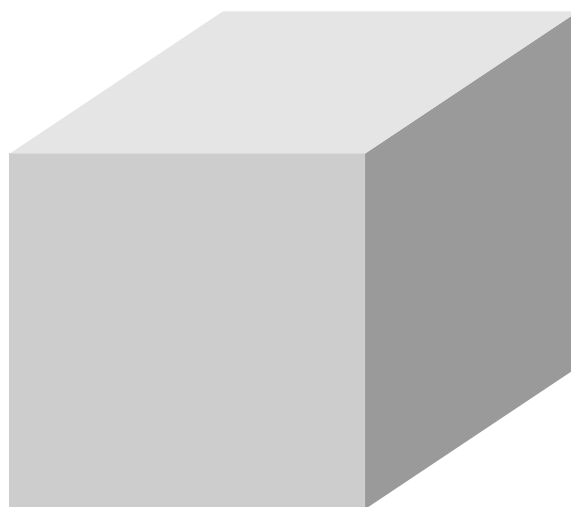
- (a) Desenhe um quadrado com medida de 5 cm e preencha-o com a cor 0.8.



- (b) Agora, faça uma lateral do cubo e a preencha com uma cor um pouco mais escura.



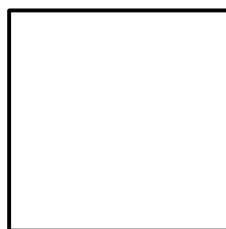
(c) Faça agora a face superior do cubo, preenchendo-a com uma cor um pouco mais clara que a das outras faces.



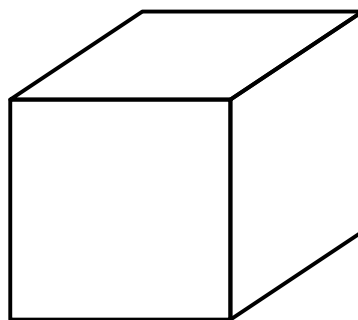
Exercício 3

Neste exercício, você criará a representação linear de um cubo.

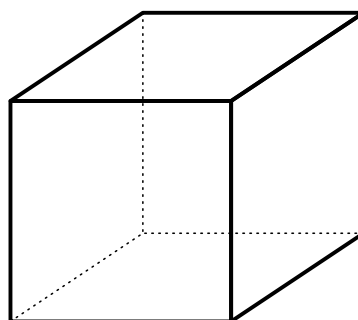
- (a) Crie um quadrado com medida de 3 cm.



- (b) Desenhe os outros lados do cubo, conforme a figura abaixo.



- (c) Trace linhas pontilhadas dentro desse cubo, formando os lados ocultos.



Considerações Finais

No decorrer de todo o desenvolvimento deste projeto foram abordados diversos conhecimentos básicos sobre a linguagem MetaPost. No início da apostila, foi apresentado do que, de fato, se trata esta linguagem, para que você se informasse melhor sobre o assunto. Durante o desenvolvimento, foram ensinados códigos para a criação de algumas figuras básicas, como quadrados, triângulos, circunferências e retângulos. Além disso, também foram apresentados alguns códigos para embelezar as figuras, como, por exemplo, adicionar rótulos e cores. Para que os conteúdos que foram apresentados na apostila sejam praticados, também foram disponibilizados alguns exercícios de fixação. Você poderá ter acesso aos mais diversos tipos de funções e códigos do MetaPost nas referências bibliográficas desta apostila.

Espera-se que por meio dos conhecimentos aqui apresentados, desenvolva-se uma visão mais ampla sobre essa linguagem tão rica chamada MetaPost. Espera-se também que a partir dessa apostila vocês, estudantes, busquem aprender e explorar muito mais do MetaPost e de outras ferramentas incríveis que ainda não são tão reconhecidas.

Referências Bibliográficas

CANDIA, E. **A Matemática e o MetaPost**. Dissertação de Mestrado. 2018. Disponível em <http://sca.proformat-sbm.org.br/sca_v2/get_tcc3.php?id=160350039> Acesso em 28 jun. 2021.

FELIPE, L. **MetaPost no Ensino Médio**. Dissertação de Mestrado. 2019. Disponível em <http://sca.proformat-sbm.org.br/sca_v2/get_tcc3.php?id=170350228> Acesso em 28 jun. 2021.

HENDERSON, T. **MetaPost Previewer**. Disponível em <<http://www.tlhiv.org/mppreview>> Acesso em 28 jun. 2021.

HOBBY, J. **METAPOST - A USER'S MANUAL**. Technical Report 162, AT&T Bell Laboratories, Murray Hill, New Jersey, 2019. Disponível em <<http://www.tug.org/docs/metapost/mpman.pdf>> Acesso em 28 jun. 2021.

OSWALD, U. **MetaPost: A Very Brief Tutorial**. Disponível em <<http://www.ursoswald.ch/metapost/tutorial.pdf>> Acesso em 28 jun. 2021.

VICENTE, L. **Escrevendo Imagens com MetaPost**. Disponível em <<http://www.ime.usp.br/~alkaid/metapost.pdf>> Acesso em 28 jun. 2021.

APÊNDICE - Respostas dos Exercícios

SEÇÃO 1

Exercício 1:

```
draw (0,0)--(6cm,0)--(0,6cm)--(0,0);
```

Exercício 2:

```
draw (0,0)--(0,4cm)--(6cm,4cm)--(6cm,0)--cycle;
```

Exercício 3:

```
draw fullcircle scaled 5cm;
```

Exercício 4:

```
draw unitsquare scaled 5cm;
```

Exercício 5:

```
draw unitsquare xscaled 2cm yscaled 4cm;
```

SEÇÃO 3

Exercício 1

```
u=1cm;
fill fullcircle scaled 4u withcolor 0.9;
draw fullcircle scaled 4u;
drawarrow (-3u,0)--(3u,0);
drawarrow (0,-3u)--(0,3u);
label.top (btex  $x$  etex, (3u,0));
label.lrt (btex  $y$  etex, (0,3u));
draw (2u*0, 2u*0.5)--(2u*sqrt(3)/2, 2u*0.5) dashed evenly
withcolor red;
draw (2u*sqrt(3)/2, 2u*0)--(2u*sqrt(3)/2, 2u*0.5) dashed
evenly withcolor blue;
dotlabel.urt (btex  $\frac{\pi}{6}$  etex, (2u*sqrt(3)/2,
2u*0.5));
dotlabel.lft (btex  $\frac{1}{2}$  etex, (2u*0, 2u*0.5));
dotlabel.bot (btex  $\frac{\sqrt{3}}{2}$  etex, (2u*sqrt(3)/2,
2u*0));
label.urt (btex  $0$  etex, (2u,0));
label.lrt (btex  $2\pi$  etex, (2u,0));
label.urt (btex  $\frac{\pi}{2}$  etex, (0,2u));
label.ulft (btex  $\pi$  etex, (-2u,0));
label.lrt (btex  $\frac{3\pi}{2}$  etex, (0,-2u));
```

Exercício 2

```
u=1cm;  
fill unitsquare scaled u withcolor 0.8;  
fill (u,0)--(1.6u,0.4u)--(1.6u,1.4u)--(u,u)--cycle withcolor  
0.6;  
fill (0,u)--(u,u)--(1.6u,1.4u)--(0.6u,1.4u)--cycle withcolor  
0.9;
```

Exercício 3

```
u=1cm;  
draw unitsquare scaled u;  
draw (u,0)--(1.6u,0.4u)--(1.6u,1.4u)--(u,u)--cycle;  
draw (0,u)--(u,u)--(1.6u,1.4u)--(0.6u,1.4u)--cycle;  
pickup pencircle scaled 0.2;  
draw origin--(0.6u,0.4u) dashed withdots scaled 0.1;  
draw (0.6u,1.4u)--(0.6u,0.4u) dashed withdots scaled 0.1;  
draw (1.6u,0.4u)--(0.6u,0.4u) dashed withdots scaled 0.1;
```