DMF—um metafile gráfico para diálogo de captura de dados

Luiz Cristovão Gomes Coelho

July 10, 1996

Abstract

Este documento descreve a sintaxe de arquivos *metafile* usados nos programas |ed| e |draw|.

1 Introdução

A linguagem de entrada de dados descrita no programa |ed| utiliza um metafile gráfico que é gerado pelo programa |draw|. O arquivo descrito aqui se assemelha aos demais metafiles existentes para descrição de imagens bidimensionais. A diferença, e portanto justificativa de existência deste metafile, está na possibilidade de se agrupar primitivas gráficas. Neste artigo, descrevemos a sintaxe e semântica do metafile gráfico |DMF| (|draw metafile|).

Não é necessário o conhecimento do metafile gráfico para utilização dos programas |draw| e |ed| pois o |draw| gera estes arquivos com um diálogo gráfico interativo.

Os conceitos teóricos sobre captura de dados sobre desenhos, e uma descrição do programa |ed|, podem ser encontrados em [1].

2 Conceitos léxicos

Um *metafile* é um arquivo texto e é formado por *registros*. Cada registro começa com um comando na primeira coluna da linha. Linhas em branco são ignoradas. Comentários devem conter o caracter |#| na primeira coluna e vão até o fim da linha. Assim, uma linha começando com |#| é um comentário.

Cada registro é divido em *campos*, que são separados por brancos ou *tabs*.

Os registros são agrupados em *blocos*. Um bloco começa com um registro especial, cujo primeiro campo é da forma *diretiva* seguida de *branco* ou *tab*.

Um bloco termina quando começa um outro bloco, ou quando o arquivo termina. Os blocos reconhecidos por |DMF| são mostradois na Tabela 1.

Cada bloco possui uma sequência obrigatória de informações. A sequência comum para as primitivas gráficas, com excessão do texto é:

```
diretiva atributos
numero de pontos
sequência de pontos
```

Cada um destes blocos será descrito em detalhes nas seções seguintes.

diretiva	nome	significado
w	*window*	determinação da "window"
g	*group*	início de um grupo
p	*polygon*	linha poligonal
t	*text*	texto
m	*mark*	marcas
a	*arc*	arco de círculo
c	*circle*	círculo completo
b	*bezier*	bezier
q	*quit*	fim do metafile

Table 1: Blocos do DMF.

3 Blocos de contrôle

4 Bloco *window*

Este bloco define a janela a ser usado na visualização das primitivas seguintes. Um bloco window é normalmente o primeiro dos comnados de um metafile. A sintaxe deste bloco é:

`w` *xmin xmax ymin ymax*

Os campos $xmin$, $xmax$, $ymin$ e $ymax$ contém os limites da janela a ser usada.

5 Bloco *group*

Este bloco define o início de um grupo de primitivas. Todas as primitivas gráficas seguintes pertencerão à este grupo. A sintaxe deste bloco é:

`g` *"name"*

O campo *name* que aparece no bloco é o nome grupo de primitivas relacionado com o arquivo *template*, descrito no artigo [1], que é usado pelo programa |ed|.

Considere o seguinte exemplo esquemático:

```
g "global"
p ...
c ...
t ...
t ...
g "new"
```

Neste exemplo, o nome do grupo é |global| seguido por uma poligonal um círculo e dois textos. Sabe-se que o grupo terminou pelo comando |g new| que indica o início do grupo |new|.

6 Bloco *polygon*

Este bloco contém as linhas poligonais que o draw pode fazer. A sintaxe deste bloco é:

```
p
número de pontos
x y
x y
...
```

`p` *b color style width i color style index*

O primeiro campo de atributos seguinte ao comando |p|, o |b|, indica que serão fornecidas as características da borda, |color| indicando a cor, |style| indicando o padrão e |width| que indica a espessura.

As possibilidades de cores são limitadas às seguintes:

```
background
white
red
green
blue
yellow
magenta
grey
brown
cyan
orange
violet
coral
pink
turqouise
beige
```

É importante ressaltar que a possibilidade de cores é limitada pelo terminal que se está usando. De forma que, se a cor não estiver disponível o interpretador de metafile deve usar a mais próxima disponível.

A espessura da borda indica o fator de escala a ser aplicado. Valores maiores que 1.0 indicam bordas mais grossas e menores que 1.0 bordas mais finas.

O padrão que as linhas podem assumir são:

```
_ cheia;
-- tracejada;
..  pontilhada;
-.  ou .- tracejada pontilhada.
```

O campo seguinte, o |i|, indica que serão fornecidos os atributos do interior da poligonal, |color| indica a cor do interior e |style| indica o padrão de *fill area* a ser adotado. Este padrão pode ser: |hollow|, |solid|, |pattern| ou |hatch|.

Se o padrão de *fill area* for |pattern| deve ser indicado um |index| que descreve o índice em uma tabela de *dithering*. Este índice pode variar entre 1 a 65 e o seu crescimento indica a diminuição de pontos (*pixels*) no interior da área.

Se o padrão de *fill area* for |hatch| o campo |index| contém um dos tipos de padrão seguintes:

```
- horizontal;
| vertical;
/ +45 graus;
\ -45 graus;
x +45-45 graus.
```

Qualquer dos dois conjuntos de campos pode ser suprimido indicando que a poligonal não possui borda ou interior preenchido.

O campo seguinte indica o número de vértices que descrevem a poligonal e os seguintes são as coordenadas destes vértices.

Exemplo:

```
p b white -- 1 i green solid
5
 -1.25 8.25
 1.25 8.25
 1.25 10.75
 -1.25 10.75
 -1.25 8.25
```

7 Bloco *text*

Este bloco descreve uma primitiva do tipo texto. A sintaxe deste bloco é:

t *color font path halign valign vupx vupy height expf*

```
"text"
1
x y
```

Esta primitiva possui os atributos relacionados com o GKS-PUC. Não se pretende reescrever cada um deles, apenas citá-los sucintamente.

O atributo |color| é o mesmo para todas as primitivas gráficas.

O atributo |font| possui as seguintes possibilidades:

```
string
char
iso_646
complex_single_space
greek_single_space
complex
greek
simplex_script_i
complex_script_i
```

```
simplex_greek_i
complex_greek_i
complex_cyrillic_i
gothic_english_i
gothic_german_i
gothic_italian_i
gothic_ii
little_complex_ii
little_ii
sanserif_ii
script_ii
simplex_ii
triplex_ii
triplex_script_ii
european_ii
complex_iii
italic_iii
simplex_iii
txt_iii
hebrew_i
```

Que são as fontes atualmente disponíveis na versão 3.4 do GKS-PUC.

O atributo |path| possui as seguintes variações: |right|, |left|, |up| e |down|.

O atributo |halign| indica o alinhamento horizontal e pode ser: |normal|, |left|, |centre| e |right|.

Da mesma forma o atributo |valign| indica o alinhamento vertical com: |normal|, |top|, |cap|, |half|, |base| e |bottom|.

Os atributos |vupx| e |vupy| indicam as coordenadas do *view up vector*.

O atributo |height| fornece a altura do texto.

Todos os atributos devem ser fornecidos na ordem indicada.

Exemplo:

```
t white complex_single_space right centre half 0 1 0.15 1
"var."
1
 0.0248322 9.90911
```

8 Bloco *mark*

Bloco de definição das *polymarkers*. A sintaxe deste bloco é:

```
m                   color type size
número de pontos
x y
x y
```

...

Os tipos de marcas disponíveis são:

`.` *point*

`+` *plus*

`*` *asterisk*

`o` *circle*

`x` *xis*

Exemplo:

```
m     white * 1
4
 -5.33808 2.23009
 2.22907 3.4805
 2.41662 0.111143
 -4.08521 -3.32061
```

9 Bloco *arc*

Bloco de definição de arcos de círculo. A sintaxe é:

`a` *b color style width i color style index*
`3`
`x y`
`x y`
`x y`

Os atributos são idênticos aos do bloco *polygon*. O número de vértices de arcos de círculo é sempre 3 e o usuário tem que especificá-lo na atual versão do |DMF|. O primeiro vértice indica o centro do arco. A distância do primeiro até o segundo descreve o raio. O terceiro vértice fornece o ângulo no sentido anti-horário.

Exemplo:

```
a     b white __ 1    i white solid
3
 0.638081 -0.805309
 6.13642 6.13231
 -7.33613 4.07326
```

10 Bloco *circle*

Bloco de definição de círculos completos. A sintaxe é:

`c` *b color style width i color style index*
`2`

```
x y
x y
```

Os atributos são também os mesmos do bloco *polygon*. Os primeiro vértice indica o centro e o segundo indica um ponto sobre o círculo.

Exemplo:

```
c     b white __ 1    i white solid
2
 1.5096 -5.63717
 4.01082 0.859889
```

11 Bloco *bezier*

Bloco de definição de beziers. A sintaxe do bloco |bezier| é:

b *b color style width i color style index*

```
4
x y
x y
x y
x y
```

A bezier descrita aqui é uma cúbica e os quatro pares de coordenadas descrevem seus pontos de contrôle.

Exemplo:

```
b     b white __ 1    i white solid
4
 -10.754 -3.37611
 -12.5563 -0.949581
 -8.99284 4.16685
 -6.05452 0.235934
```

12 Bloco *quit*

Este bloco define o fim do *metafile*. A sintaxe deste bloco é:

```
q
```

A interpretação do *metafile* termina quando este bloco é encontrado. Embora dispensável, este bloco serve para manter alterações temporárias no arquivo, sem precisar "comentar" trechos inativos.

13 Um exemplo

Como ilustração, apresenta-se um exemplo de *metafile* gráfico que possui algumas das primitivas citadas.

```
# eg output generated by Draw at Fri Jan  8 14:19:39 1993

W -9.43214 50.7179 -20.8437 26.8437
g "symbol"
p b white __ 1 i white solid
6
 0 11
 8 7
 8 3
 0 7
 -8 3
 -8 7
p b white __ 1 i white solid
6
 -2.38419e-07 -5
 8 -1
 8 3
 -2.38419e-07 -1
 -8 3
 -8 -1
g "major_text"
t white complex_single_space right centre half 0 1 3 1
"PETROBRAS"
1
 30 4
p b white __ 1 i background solid
3
 43.3 5.9
 43.7 6.4
 43.8 6.3
g "minor_text"
t white complex_single_space right centre half 0 1 1 1
"PETROLEO BRASILEIRO S.A."
1
 29 -4.76837e-07
q
```

Um interpretador para este *metafile* deveria produzir a seguinte imagem:

References

[1] L. H. de Figueiredo, C. S. de Souza, M. Gattass e L. C. G. Coelho, *Geração de interfaces para captura de dados sobre desenhos*, Anais do SIBGRAPI V, 1992, 169–175.

Figure 1: Interpretação de um *metafile*.