Matriz de transformación en PostScript

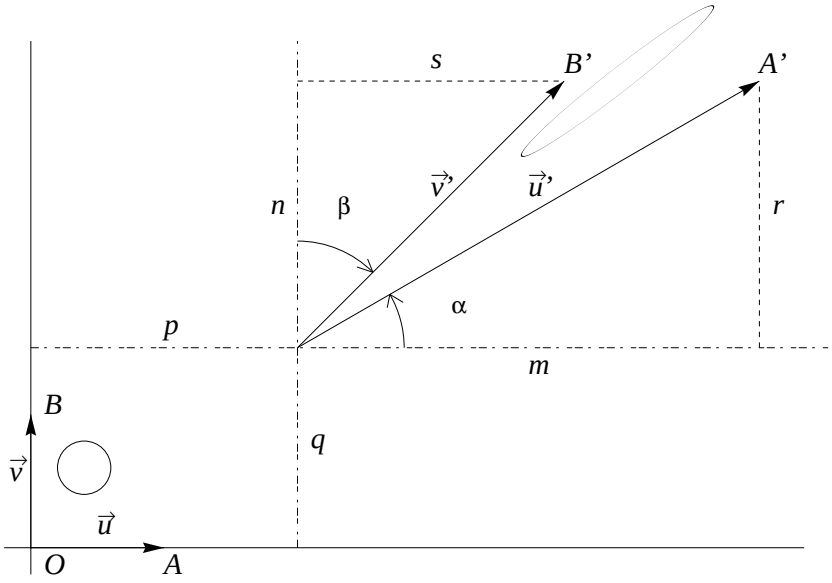
En PostScript, las operaciones de traslación (*translate*), rotación (*rotate*) y cambio de escala (*scale*) quedan abarcadas en una transformación afín de la forma:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} m & s \\ r & n \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} p \\ q \end{pmatrix} \qquad \text{o bien} \qquad \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} m & s & p \\ r & n & q \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Ponemos a continuación la forma de dicha matriz en cuatro casos particulares: 1) Al comienzo de un programa, la matriz de transformación es por defecto la matriz identidad. 2) Para una traslación ($p\ q$ *tanslate*). 3) Para un cambio de escala ($m\ n$ *scale*) y 4) Para una rotación de ángulo θ (θ *rotate*).

1) $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ 2) $\begin{pmatrix} 1 & 0 & p \\ 0 & 1 & q \\ 0 & 0 & 1 \end{pmatrix}$ 3) $\begin{pmatrix} m & 0 & 0 \\ 0 & n & 0 \\ 0 & 0 & 1 \end{pmatrix}$ 4) $\begin{pmatrix} \cos\theta & -\operatorname{sen}\theta & 0 \\ \operatorname{sen}\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Tenemos la siguiente interpretación geométrica de los coeficientes r y s, determinando la imagen de un par de vectores ortonormales en el origen de coordenadas.



$$\begin{pmatrix} m & s \\ r & n \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} p+m \\ q+r \end{pmatrix} \equiv A'$$

$$\begin{pmatrix} m & s \\ r & n \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} + \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} p+s \\ q+n \end{pmatrix} \equiv B'$$

de donde se tiene entonces que:

$$r = m \operatorname{tag}\alpha \qquad s = n \operatorname{tag}\beta;$$

así r está relacionado con la desviación respecto a la horizontal y s con la desviación respecto a la vertical. En una rotación, $\alpha = \theta$ y $\beta = -\theta$.

La matriz de transformación será singular si $mn - rs = mn(1 - \operatorname{tag}\alpha \operatorname{tag}\beta) = 0$, o sea si $m = 0$ o $n = 0$ o $\alpha + \beta = \pi/2$. Cuando se pongan algunos de estos valores se producirá un error.

Son seis los coeficientes fundamentales que determinan la transformación, con los que se confecciona la matriz del lenguaje PostScript siguiente:

$$[m\ r\ s\ n\ p\ q]$$

Cuando se usan los operadores *translate, rotate, scale*, etc., el intérprete PostScript calcula automáticamente, a partir de los argumentos de los operadores nombrados, la matriz de transformación. No obstante, tal matriz puede utilizarse directamente para establecer transformaciones geométricas de objetos, utilizando el operador *concat*. Otros usos de la matriz de transformación son para transformar caracteres de un juego de caracteres por medio del operador *makefont*, o para modificar la representación de una imagen a partir de su descripción digitalizada, utilizando el operador *image*. Existen muchos más operadores que tienen como uno de sus argumentos una matriz de transformación (citemos, por ejemplo, aquellos que sustituyen una matriz por otra, el que multiplica matrices, el que obtiene la inversa, etc.). Discutiremos sólo los tres citados antes con algunas de sus aplicaciones.

• Operador *concat*

En el dibujo de arriba hemos transformado un circunferencia de radio 10 puntos y de centro en (20,30) mediante la matriz de transformación [3 2 2 2 100 75] en una elipse, lo cual se logra con el siguiente programa en PostScript utilizando el operador *concat*:

```
/circunferencia {20 30 10 0 360 arc} def
/matriz [3 2 2 2 100 75] def

circunferencia stroke
matriz concat
circunferencia stroke
```

Otro ejemplo de utilización del operador *concat* es para poner un texto inclinado o con sobran (cosa que también se puede hacer mediante el operador *makefont*, como veremos).

```
/Times-Roman findfont 50 scalefont setfont
/texto (Sombra)  def

gsave
  0 0 moveto
  .5 setgray
  [1 0 2 -2 0 0] concat
  texto show
grestore
0 0 moveto
[1 0 1 1 0 0] concat
texto show
```

• Operador *makefont*

En PostScript cada juego de caracteres está asociado a un matriz de transformación que determina la escala, la orientación y la posición de los caracteres. Esta matriz, como sabemos, consta de seis elementos, con índices que varían del 0 al 5:

$$[m\ r\ s\ n\ p\ q]$$

m es el parámetro (índice 0 de la matriz) que representa la longitud en puntos del carácter.
r (índice 1) está relacionado con el ángulo θ que forma el carácter con la horizontal, $r = m \tan\theta$.
s (índice 2) está relacionado con el ángulo β del carácter con la vertical, $s = n \tan\beta$.
n representa la altura en puntos del carácter.
p, q representan el desplazamiento en puntos, con respecto al origen de coordenadas; una posible utilización es para colocar subíndices y superíndices.

Esta matriz es el parámetro del operador *makefont* que transforma el juego de caracteres, por lo que [m 0 0 m 0 0] *makefont* es equivalente a m *scalefont*.

Podemos poner en una misma página el texto sombreado anterior y otros más, mediante el siguiente programa:

```
/Times-Roman findfont [50 0 100 -100 0 0] makefont setfont
/texto (Sombra)  def
0 0 moveto
gsave
 .5 setgray
  texto show
grestore
/Times-Roman findfont [50 0 50 50 0 0] makefont setfont
texto show
/Helvetica findfont  [45 15 -4 20 -300 -10] makefont setfont
-2 0 (Texto en escalera) ashow
```

En el texto que se ha escrito en escalera, además de deformar la cadena de caracteres se ha reducido el espacio entre ellos con el operador *ashow*

Otra posible utilización de *makefont* aplicada a una matriz de transformación es separar cada carácter de una cadena y aplicarle a cada uno de ellos una transformación. Valga como ejemplo el siguiente programa:

```
/fuente /Times-Roman findfont def
/texto (LaTeX2e y PostScript) def
/matriztr [40 0 0 60 0 0] def
/carac 1 string def  /numcarac 0 def
/longmedia texto length 1 add 2 div def
0 -165 translate 100 0 moveto
0 1 texto length 1 sub     { dup
  /numcarac numcarac 1 add def
  /matriztr
     matriztr 2 numcarac longmedia sub 5 mul dup sin exch cos div 75 mul put
     matriztr 3 35 2.5 2 numcarac longmedia sub exch exp mul add put
  def
  fuente matriztr makefont setfont
  texto exch get
  carac exch 0 exch put
  -2 0 carac ashow         } for
```

• Operador *image*

El operador *image* permite tratar información digitalizada para lo que necesita, entre sus cinco argumentos, una matriz de transformación que le ha de indicar en qué ha de convertir un cuadrado de unidad (1/72 pulgadas) de lado según sea los parámetros de dicha matriz $[m\ r\ s\ n\ p\ q]$.

Comenzamos con un ejemplo rústico hecho a mano que representa dos guaguas, una veloz y otra detenida:

```
gsave
  450 60 translate
  72 36 scale
  8 8 1 [8 0 0 8 0 0] {<bd420000c0d6d6c0>} image
grestore
gsave
  350 60 translate
  72 30 scale
  8 8 1 [8 0 3 8 0 0] {<bd420000c0d6d6c0>} image
grestore
```

El operador *image* posee cinco argumentos, que en este ejemplo son:

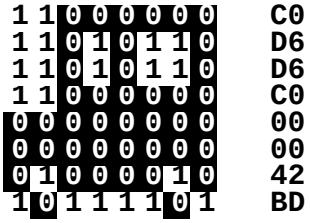
8 número pixel por línea.

8 número de líneas de la imagen.

1 número de bits por pixel. Valores posibles: 1 (blanco y negro), 2 (4 niveles de gris), 4 (16 niveles de gris) y 8 (256 niveles de gris).

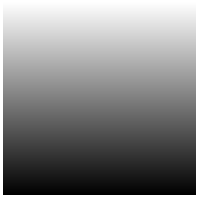
[8 0 3 8 0 0] matriz de transformación.

<BD420000C0D6D6C0> Proporciona los caracteres necesarias para la descripción de la imagen. Cada par de caracteres representa un número en base hexadecimal que origina una línea de la imagen; este número, en binario, tiene ocho caracteres de unos y ceros, que corresponden a blanco y negro, respectivamente. Así, la primera línea que corresponde a la ruedas, esta descrita por BD (hexadecimal) o 10111101 (binario). El diagrama de la derecha puede aclarar esto.



Damos un segundo ejemplo donde los códigos de los caracteres que describen la imagen están generados por una rutina o procedimiento.

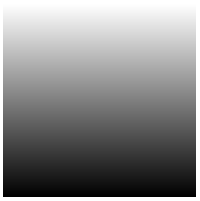
```
/cadena 256 string def
/imagen{ gsave translate scale image grestore} bind def
0 1 255 {cadena exch dup put} bind for
1 256 8 [1 0 0 256 0 0] {cadena}
72 72 424 16 imagen
```



Se produce un efecto similar con el siguiente programa, pero ahora podemos lograr que el degradado comience y termine en cualquier nivel de gris desde el 0 al 1:

```
/cadena 256 string def
/degradado{
     1 index sub
     exch 255 mul
     0 1 255 { cadena exch dup
               4 index mul
```

```
              3 index add cvi
              put
             } for
    pop pop
    1 256 8 [1 0 0 256 0 0] {cadena} image
         } bind def
gsave 400 50 translate  72 72 scale 0 1 degradado grestore
gsave 300 50 translate  72 72 scale .8 .3 degradado grestore
```



Un tercer ejemplo, que tal vez sea la mayor aplicación del operador *image*, es obtener una imagen digitalizada, por ejemplo con una escaner, almacenada bajo forma de un programa Postscript, y utilizar estos datos para modificar, si se desea, la imagen que ellos producen.

El siguiente programa está generado con un escaneado

```
%!PS-Adobe-2.0 EPSF-1.2
%%BoundingBox: 0 0 288 240
%%Creator: iPhotoPlus
%%EndComments
%iPhotoPlus Entity:   792   660 1 1  198    99 1
/width   792 def
/height   660 def
/glevel 1 def
/dpi 198 def
/istr   99 string def
/nheight height neg def
/dot {72.0 dpi div mul} def
/imageturkey
{ width height glevel [width 0 0 nheight 0 height]
  { currentfile istr readhexstring pop } image
} def
gsave
0 0 translate
width dot height dot scale
imageturkey
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFF9FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

%  mas codigos exadecimales

FFFFFFFFFFFE01FFC03E003FFFFC03E01C3E01FFFFC7F01F80FFFFFC0FE07FC0E000FF83F3FC1F
847E00FE0FF1FFFFE01F7EFFF3FFFE3FFFFFFFFFFFFFFFFFF83FFFFF8FFFFFFFFFFFFFFFFFFFF
FFFFFFF9FFFFC3FFFFFFFFFFFFFFFFFFFFFFFFFFFF
grestore
showpage
```