PostScript

Salvador Ramirez Flandes (sram)

28 de noviembre de 2000

Índice General

1 Introduccion

PostScript es un lenguaje interprete con poderosas capacidades graficas. Su principal aplicacion es describir la apariencia de texto, formas graficas e imagenes. Esta descripcion es de alto nivel e independiente del dispositivo grafico a usar (pantalla grafica o impresora).

La interaccion con el interprete PostScript se puede visualizar en las siguientes figuras:

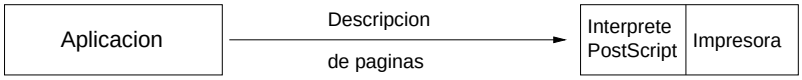

Fig 1. Modelo tradicional de impresion. Ejemplo: cualquier impresora PostScript

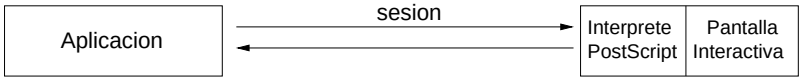

Fig 2. Modelo de despliegue de imagen por pantalla. Ejemplo: ghostview

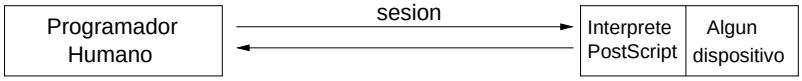

Fig 3. Modelo de lenguaje de programacion interactivo. Ejemplo: ghostscript

2 Ideas Basicas

PostScript puede ser visto desde los siguientes puntos de vista:

Un lenguaje de programacion de proposito general con primitivas graficas incorporadas

Un lenguaje de descripcion de pagina con cualidades de programacion

Un sistema interactivo para controlar dispositivos de salida raster (pantallas e impresoras)

Un formato para intercambio

2.1 Dispositivos de Salida Raster

El lenguaje *PostScript* (PS) puede interactuar con *Dispositivos de Salida Raster* (DSR) tales como impresoras laser, matriz de punto, de inyeccion de tinta y todo tipo de pantallas. Una imagen impresa o desplegada a un DSR es un arreglo rectangular de puntos llamados pixeles que pueden ser direccionados individualmente. Por ejemplo en DSR blanco y negro cada pixel puede ser blanco o negro. En otros DSR mas avanzados estos pixeles pueden ser tonalidades de grises o colores. Con esta tecnica se pueden reproducir complejas imagenes o texto dependiendo de la resolucion del DSR.

La resolucion del DSR es una medida del numero de pixeles por unidad de distancia a lo largo de dos dimensiones espaciales. La resolucion es tipicamente, pero no necesariamente, la misma horizontal y vertical. Por ejemplo muchas impresoras laser actuales tienen resoluciones entre 600 y 1200 pixel por pulgadas (dpi - dots per inch). A mayor resolucion mejor la calidad de la salida en el DSR, pero a mayor costo.

2.2 Scan Conversion

Un entidad grafica abstracta (como por ejemplo una linea, un circulo, un caracter de texto o una imagen) es renderizada en el DSR por un proceso llamado Scan Conversion. Este proceso consiste en que dada una descripcion PS de la entidad grafica, se determinan que pixeles se deben ajustar y que valor se les debe asignar para encontrar la mejor representacion posible de la entidad a la resolucion del DSR.

El proceso de Scan Conversion se realiza para cada entidad grafica que aparecera en la pagina. Cuando este proceso esta listo entonces los pixeles en la memoria del DSR representa la apariencia de la pagina y se puede entonces imprimir o desplegar.

La renderizacion de imagenes en tonalidades de grises en DSR cuyos pixeles solo pueden ser blanco y negros se lleva a cabo a traves de un proceso llamado *halftoning*. El arreglo de pixeles es dividido en un malla muy fina de acuerdo a un patron llamado *halftone screen.* Dentro de cada celda de esta malla algunos pixeles son pintados en blanco y otros en negro en proporcion al nivel de gris de la imagen original. Si la malla es suficientemente fina entonces las celdas de estas no son vistas y la ilusion de tonalidades de grises es lograda.

2.3 Lenguajes de Descripcion de Pagina

Teoricamente se podria pensar que cualquier aplicacion (como un procesador de texto por ejemplo) podria hacer una descripcion de pagina como un arreglo de paginas, cada una de estas siendo un arreglo de pixeles. Sin embargo una descripcion de este tipo seria dependiente del dispositivo y enorme para documentos grandes. Un lenguaje descripcion de pagina debe ser suficientemente compacto para almacenamiento y transmision, ademas de ser independiente del dispositivo.

Estas caracteristicas de compacticidad e independencia del dispositivo se pueden lograr se el lenguaje logra describir las paginas en terminos de entidades graficas abstractas de alto nivel en lugar de un arreglo de pixeles. Es en el DSR donde se genera el arreglo de pixeles a traves del proceso Scan Conversion.

2.4 Estructura del Lenguaje

Una descripcion de pagina PS bien estructurada consiste en un prologo y un script. El prologo es un conjunto de procedimientos a usar y se acostumbra a ponerlo en el comienzo del archivo. El script es la descripcion de los elementos de las paginas con ayuda de los procedimientos del prologo. Esto ayuda a que la descripcion de la pagina sea mas eficiente y compacta. Por ejemplo en el prologo se puede definir:

```
/ms {moveto show} bind ref
```

y en el script se puede llamar a este procemiento de la siguiente forma:

```
(texto cualquiera aqui) 100 200 ms
```

El script de un documento a imprimir por ejemplo, consiste tipicamente de una secuencia de paginas separadas. La descripcion de cada pagina puede usar los procedimientos del prologo pero debe ser independiente de las demas paginas del script.

3 Lenguaje de Programacion PostScript

En esta seccion se describira PostScript como un lenguaje de programacion, es decir se mostrara su sintaxis, sus tipos de datos, etc.

PostScript usa notacion postfija. En una operacion los operandos van despues de los operadores. Ej: “2 3 add”.

El interprete PS manipula entidades llamados Objetos PostScript (OP). Algunos OP son datos (numeros, booleanos, strings) mientras que otros son elementos de programa a ser ejecutados (nombres, operadores y procedimientos). El interprete opera ejecutando secuencia de OP. El efecto de la ejecucion de un OP depende del tipo de este, sus atributos y su valor. Por ejemplo la ejecucion de un OP numero provoca que el interprete ponga una copia de ese objeto en la pila de operandos. La ejecucion de un nombre provoca que el interprete busque su valor en el diccionario, lo obtenga y ejecute ese valor asociado. La ejecucion de un objeto operador provoca que el interprete realice una accion que posee incorporada, tal como la suma o el pintado de caracteres en memoria.

El inteprete PS consume un programa mediante la ejecucion de cada OP que encuentra. No es necesario por tanto compilar o interpretar el programa entero de una vez para hacer algo, sino que

3.1 Sintaxis

Existen tres codificaciones para programas PS: ASCII, binary token y secuencia de objetos binarios. ASCII es la codificacion mas usada y recomendada.

En la codificacion ASCII los espacios en blanco separan una construccion sintactica de otra. Uno o mas espacio son tratados como si fueran uno solo. Los caracteres tratados como espacios son los siguientes:

Octal	Hex	Decimal	Nombre
000	00	0	Nulo (nul)
011	09	9	Tab (tab)
012	0A	10	Line-feed (LF)
014	0C	12	Form-feed (FF)
015	0D	13	Carriage-return (CR)
040	20	32	Space (SP)

Los caracteres CR y LF son tambien llamados caracteres newline. Un CR seguido inmediatamente por un LF son tratados juntos como un solo newline.

Los caracteres (,),<,>,[,],{,},/, y % son especiales. Ellos delimitan entidades sintacticas tales como strings, procedimientos, nombres literales y comentarios.

Un comentario en PS se forma desde un % hasta el siguiente newline no importando la existencia de cualquier caracter de por medio.

3.1.1 Numeros

Los numeros en PS pueden ser:

Enteros con signo: 123, -98, 43445, 0, +17

Reales: -0.002, -.01, 123.6e10. 1E-5, 0.0 y

Numeros con base (radix numbers): 8#1777, 16#FFFE, 2#1000

Si un entero exede el limite de implementacion para ese tipo de dato entonces es convertido a real. Si un real exede su limite entonces un error de "limitcheck" ocurre. Los mismo anterior sucede para los radix numbers.

3.1.2 Strings

Hay 3 convenciones para definir strings:

Como texto literal encerrado entre (y)

Como dato codificado hexadecimal encerrado entre < y >

Como dato codificado ASCII base-85 encerrado entre <~ y ~> (PS2)

Dentro de un string el backslash \ se usa como caracter de escape para incluir caracteres especiales tales como : \n, \r, \t, \b, \f, \\, \(, \), \ddd. Este ultimo es codigo octal del caracter y sirve para poner caracteres especiales como ñ o á, ó, etc. pues estos ultimos no son recomendados de usar directamente.

3.1.3 Nombres

Toda construccion sintactica compuesta de caracteres regulares y que no puede ser interpretado como numero es tratado como un objeto nombre o un nombre ejecutable. Cualquier caracter excepto delimitadores o espacios pueden formar parte de un nombre. Cuando inmediatamente delante del nombre hay un / entonces este es un nombre literal, en caso contrario es un nombre ejecutable. Mas adelante se entendera la diferencia entre estos dos tipos de nombre.

3.1.4 Arreglos

Los caracteres [y] delimitan un arreglo. Por ejemplo: "[123 /abc (xyz)]" es un objeto arreglo que contiene el entero 123, el nombre literal abc y el string xyz.

3.1.5 Procedimientos

Los caracteres { y } delimitan un arreglo ejecutable o procedimiento. Por ejemplo: "{add 2 div}". Un procedimiento no es ejecutado de inmediato, sino que se pone en la pila de operandos y se ejecuta si es explicitamente invocado.

3.1.6 Diccionarios

Las entidades << y >> delimitan un diccionario. Un diccionario es un arreglo de duplas clave-valor. Por ejemplo: "<<clave1 valor1 clave2 valor2 ... claven valorn>>". En PS1 y PS2 el comportamiento de los diccionarios cuando se llenan es distinto. En PS1 un error dictfull ocurre mientras que en PS2 el diccionario se agranda automaticamente.
Diccionarios locales standard:

userdict: Diccionario que puede ser escrito y que es el usado normalmente por los programas PS, por ejemplo cuando se crean procedimientos con el operador def. Es el diccionario local standard y es inicialmente el diccionario top en la pila de diccionarios.

errordict: Diccionario de errores.

$error: Diccionario accesado por las funciones incorporadas para el manejo de errores.

statusdict:

FontDirectory: Diccionario para las definiciones de fonts. Normalmente es read-only pero es actualizado por el operador definefont y consultado por findfont.

Diccionarios globales standard:

systemdict: diccionario read-only que asocia los nombres de todos los operadores PS (add por ejemplo) con sus respectivos valores (las acciones incorporadas en el interprete que las implementa). Es el ultimo diccionario en la pila de diccionarios.

globaldict: (PS2) es un diccionario que puede ser escrito en la memoria global. Este diccionario se encuentra en entre el systemdict y el userdict.

GlobalFontDirectory: (PS2) diccionario para la definicion de fonts en la memoria virtual global. Es actualizado por definefont y consultado por findfont.

3.2 Tipos de datos y Objetos

Cada objeto tiene un tipo, algunos atributos y un valor. Existen objetos simples y compuestos. En objetos simples el copiado de objetos es con tipo, atributos y valores. En objetos compuestos no se copia el valor sino una referencia solamente. A continuacion una lista objetos tanto simples como compuestos:

Objetos simples	Objetos compuestos
boolean	array
fontID	condition (Display PS)
integer	dictionary
mark	file
name	gstate (nivel 2)
null	lock (Display PS)
operator	packedarray (nivel 2)
real	string
save	

El objeto file es el objeto archivo y puede ser por ejemplo stdin o stdout por ejemplo. El objeto mark es un objeto especial para denotar una posicion en la pila de operandos. Todo objeto recien creado tiene valor null. El objeto boolean puede ser “true” o “false”.

3.2.1 Literales y ejecutables

Cada objeto en PS es literal o ejecutable. La diferencia entre estos es como el interprete los trata. A un objeto literal el interprete PS lo trata como dato y lo pone en la pila de operandos para su posterior uso en alguna operacion. A un objeto ejecutable en cambio, el interprete PS lo ejecuta. En este ultimo caso el interprete PS se comporta asi:

> Si el objeto es un nombre ejecutable entonces se busca este nombre en el diccionario del contexto actual y se ejecuta su valor asociado
>
> Si el objeto es un operador ejecutable entonces se usa realiza la accion incorporada (built-in) en el lenguaje, tal como una suma por ejemplo
>
> Si el objeto es un arreglo ejecutable (o procedimiento) se ejecutan los elementos de tal arreglo en la secuencia correspondiente

Algunas construcciones sintacticas entonces producen objetos literales y otros ejecutables:

Enteros, reales y strings constantes son siempre objetos literales

Los nombres son literales si son precedidos por un / y ejecutables sino

Los operadores [y] cuando se ejecutan producen un arreglo literal de objetos. De la misma forma << y >> producen un objeto diccionario literal. Los operadores { y } finalmente crean un arreglo ejecutable o procedimiento.

3.2.2 Accesos o permisos

El acceso es un atributo de los objetos. Los accesos son: unlimited, read-only, execute-only, none.

3.3 Pilas (stacks)

El IPS maneja 4 pilas que representan el estado de ejecucion de un programa PS.

Pila de operandos. Mantiene los objetos PS que son operandos para una operacion posterior o son el resultado de alguna operacion llevada a cabo. Esta pila puede es manipulada por el programa PS.

Pila diccionario. Mantiene objetos diccionario. Esta pila tambien es manipulada por el programa PS.

Pila de ejecucion. Mantiene la secuencia de objetos ejecutables a ejecutar. Esta pila es manipulada por el IPS, puede ser leida por el programa PS pero no modificada.

Pila de estado grafico.

3.4 Ejecucion

La semantica de la ejecucion difiere de un tipo de objeto a otro. Esta ejecucion puede ser inmediata o diferida.

Por ejemplo:

```
40 60 add 2 div
```

1. El IPS en este caso carga en la pila de operandos el valor entero 40

2. Se encuentra el entero 60 y lo carga tambien en la pila de operandos

3. Se encuentra add, se busca en la pila de diccionarios por este nombre. Se encuentra en systemdict y se ejecuta su accion incorporada, la suma de los dos ultimos operandos de la pila de operandos

4. Luego de llevarse a cabo tal operacion se carga 100 en la pila de operandos y se eliminan los operandos anteriores

5. Luego el IPS carga 2 en la pila de operandos y ejecuta div, despues de lo cual cargaria 50 en la pila de operandos. Esta ejecucion es inmediata.

Ahora considerese el siguiente ejemplo:

```
/promedio {add 2 div} def
40 60 /promedio
```

1. El IPS encuentra el nombre literal “promedio” que lo pone en la pila de operandos.

2. Se encuentra un procedimiento que tambien lo pone en la pila de operandos.

3. Se encuentra def, que lo busca en la pila de diccionarios y encuentra este en el diccionario systemdict y por tanto se ejecuta la accion incorporada asociada que es sacar los dos ultimos operandos de la pila de operandos y ponerlos en el diccionario actual (userdict tipicamente) con “promedio” como clave y {add 2 div} como valor.

4. El IPS finalmente encuentra los operandos 40 y 60 que los pone en la pila de operandos y luego se encuentra “promedio” que se busca en la pila de diccionarios y se encuentra con valor {add 2 div}, se ejecuta este procedimiento entonces y se carga en la pila de operandos el valor 50.

3.4.1 Estructuras de control

En PS, las estructuras de control tales como condicionales o iteraciones son llevadas a cabo por operadores que toman procedimientos como operandos. Por ejemplo:

```
a b gt {a} {b} ifelse
```

1. El interprete encuentra los nombre ejecutables a y b y los busca en la pila de diccionarios. En este caso se asumen numeros y se ejecutan cargandolos en la pila de operandos

2. El operador gt saca los dos operandos, los compara y pone true en la pila de operandos si el primero es mayor que el segundo y false en cualquier otro caso.

3. Se encuentran los procedimientos {a} y {b} que se ponen en la pila de operandos.

4. El operador ifelse toma tres operandos, un boolean y dos procedimientos. Se sacan estos tres operandos de la pila de operandos. Si el primer operando es true entonces se ejecuta {a}, en otro caso se ejecuta {b].

El resultado de este estamento es poner entonces en la pila de operandos el valor mayor entre a y b, dado que ambos procedimientos tienen aquellos numeros tan solo como cuerpo.

Los operadores de control son:

if, ifelse

exec: ejecuta un objeto incondicionalmente

for, repeat, loop y forall

exit: escapa de un loop

stop: termino prematuro, quiebra toda secuencia de ejecucion

countexecstack, execstack: se usan para leer la pila de ejecucion

Ejemplos:

```
% provocara poner en la pila 4 veces el string “abc” sacando previamente
% los operandos 4 y el procedimiento de la pila
4 {(abc)}repeat

% aqui el tope de la pila termina con el valor 4
8 4 {1 sub} repeat

% equivale al codigo-C: for(i=0; i<=8; i++) proc;
```

% si el incremento es negativo entonces el <= se cambia por un >=
0 2 8 {proc} for

% lo siguiente pondra los valores 3.0, 2.5, 2.0, 1.5 y 1.0 en la pila
3 -0.5 1 {}for

% en cada iteracion del for el actual valor del contador es puesto en
% la pila, asi
0 1 4 {add} for
% resultara en un 10 al final.

Los operadores binarios de comparacion son:

eq, ne

gt, ge, le, lt

and, or, xor, true, false, bitshift

3.4.2 Operadores de pila

dup: duplica un objeto
Ejemplo:
% Lo siguiente es un procedimiento para elevar numeros al cubo
/cubo {dup dup mul mul}def
3 cubo

exch: intercambia los elementos del tope de la pila

pop: elimina el elemento del tope de la pila

copy: duplica porciones de la pila de operandos

roll: trata una porcion de la pila como una cola circular

index: accesa la pila como si fuera un arreglo indexable

mark: marca una posicion en la pila

clear: limpia la pila

count: cuenta el numero de elementos de la pila

counttomark: cuenta los elementos sobre la marca mas alta (mas adelante se describe mejor este operador)

3.4.3 Operadores aritmeticos y matematicos

Estas funciones toman argumentos tanto enteros como reales y producen resultados enteros o reales segun corresponda. Si no se puede definir un resultado entonces ocurre un error PS undefinedresult.

Operadores de dos argumentos: **add, sub, mul, div, idiv, mod**

Operadores de un argumento: **abs, neg, ceiling, floor, round, truncate**

Funciones matematicas y trigonometricas: **sqrt, exp, ln, log, sin, cos, atan**

Funciones para generacion de numeros pseude-random: **rand, srand, rrand**

3.4.4 Operadores de arreglos, diccionarios y strings

get: toma un objeto compuesto y un indice (clave en el caso de diccionarios) y retorna elemento componente del objeto compuesto asociado a tal indice

put: guarda un elemento en un objeto compuesto (no se aplica a objetos packed array pues ellos son read-only)

copy: copia el valor de un objeto compuesto a otro objeto compuesto del mismo tipo

length: retorna el numero de elementos del objeto compuesto

forall: accesa todos los elementos del objeto compuesto en secuencia, llamando un procedimiento para cada uno

getinterval: crea un nuevo objeto que comparte un subintervalo del arreglo, packed array o string (no se aplica a diccionarios)

putinterval: sobreescribe un subintervalo de un arreglo o string con el contenido de otro (no se aplica a diccionarios ni packed array)

aload-astore: (solo para arreglos y packed arrays) transfiere todos los elementos de un arreglo a o desde la pila de operandos

setpacking-currentpacking:

begin-end: (solo para diccionarios) pone o saca diccionarios de la pila de diccionarios

def-store: (solo para diccionarios) asocia claves con valores en diccionarios en la pila de diccionarios. **load** y **where** busca por claves en ellos

countdictstack-cleardictstack-dictstack: manipulan la pila de diccionarios

known: consulta si una clave esta presente en un diccionario especifico

maxlength: retorna la capacidad maxima de un diccionario

undef: (PS2) elimina una clave de un diccionario

<<->>: (PS2) construye un diccionario

search-anchorsearch: (solo strings) realiza busqueda y matching de string

token: scanea los caracteres de un string de acuerdo a las reglas de sintaxis del PS, sin ejecutar los objetos resultantes.

3.4.5 Operadores de tipo, atributo y conversion

type: retorna el tipo de un operando como nombre de objeto (intergertype, realtype, etc)

xcheck-rcheck-wcheck: consulta al objeto literal/ejecutable y accesan los atributos del objeto

cvlit-cvx: cambian el atributo literal/ejecutable de un objeto

readonly-executeonly-noaccess: reducen los atributos de acceso de un objeto

cvi-cvr: intercambia el tipo de dato de un objeto entre real y entero y lo interpreta como tal

cvn: convierte un string en un nombre de objeto definido por los caracteres del string

cvs-cvrs: convierte un objeto de algun tipo a una representacion string

3.5 Manejo de memoria

Un programa PS se ejecuta en un ambiente con 5 componentes principales: pilas, memoria virtual, archivos entrada y salida y el estado grafico.

Los objetos simples en PS contienen su valor en ellos mismos, mientras que los objetos compuestos contienen su valor en algun lugar de la memoria virtual. Las variables que referencian a objetos compuestos solo son referencias a direcciones dentro de la memoria virtual de PS. El operador dup por ejemplo pone una segunda copia del objeto arreglo en la pila de operandos, sin embargo este objeto arreglo es una referencia, por lo que lo que se duplica realmente es la referencia pero no el valor mismo.

Existen dos tipos de memoria virtual, local y global. En la memoria virtual lcoal es donde los objetos compuestos que un programa PS maneja se crean. Los operadores **save** y **restore** sirven para grabar y recuperar el estado de la memoria virtual local en la ejecucion de un programa. Estos operadores son entendidos para ser usados en programas estructurados tales como descripciones de paginas. En tales casos save y restore sirven las siguientes funciones:

> Un documeno consiste de un prologo y un script. El prologo contiene definiciones que son usadas a lo largo de todo el documento. El script contiene una secuencia de paginas independientes. Cada pagina tiene un **save** al comienzo y un **restore** al final, inmediatamente antes del operador **showpage**. Asi cada pagina comienza su ejecucion con las condiciones iniciales establecidas en el prologo.
>
> Una pagina muchas veces contiene substructura de programa tales como imagenes, cuya ejecucion necesita ser encapsulada. El programa encapsulado puede hacer cambios en la memoria virtual local para sus propios fines. Para no interferirse entre el programa y el encapsulado se puede encerrar entre **save** y **restore** el programa encapsulado.
>
> Durante la ejecucion de un programa PS, se pueden crear muchos objetos compuestos tanto por el mismo programa como por el scanner PS. El operador **restore** entonces puede ser usado para liberar la memoria usada desde el ultimo **save**. Asi la ejecucion periodica de **save** y **restore** ayuda a no llenar la memoria virtual local.

En PS2 se implementa un sistema automatico para la liberacion de variables que ya no son mas accesibles al programa. Este sistema se llama **recolector de basura** (garbage collector). Por ejemplo, en el siguiente codigo:

/a (string 1) def
/a (string 2) def
(algun texto) show

El objeto string “string 1” ya no es mas accesible. De la misma forma el objeto “algun texto” tampoco pues show consume ese operando pero no lo almacena.

3.6 Recursos

El lenguaje PS tiene muchos recursos que pueden ser usados para definir la apariencia de una pagina o de los objetos componentes de esas paginas. Todos estos recursos no pueden estar en memoria virtual a la vez y por tanto PS provee de diccionarios con claves y referencias hacia los lugares (de la memoria o medio adicional de almacenamiento) donde se encuentran fisicamente esos recursos.

Existen 5 operadores que operan recursos: **findresource**, **resourcestatus**, **resourceforall**, **defineresource** y **undefineresource**.

La categoria de recursos mas usados en PS son los fonts, tal como Times-Roman, a los cuales se les puede aplicar los operadores **scalefont** o **makefont**. Se proveen unos operadores rapidos para fonts:

findfont : equivalente a /Font findresouce

definefont: equivalente a /Font defineresource

undefinefont: equivalente a /Font undefineresource

3.7 Binding anticipado de nombres

Normalmente cuando el scanner del lenguaje PS encuentra un nombre ejecutable en el programa, este crea un objeto nombre ejecutable pero no busca el valor del nombre. El valor del nombre se busca cuando el IPS ejecuta tal objeto nombre. Asi la busqueda en diccionarios de la pila de diccionarios ocurre en tiempo de ejecucion.

Por ejemplo en :

/promedio {add 2 div} def

en la definicion no se buscan los operadores add ni div, sino que cada vez que el procedimiento /promedio es invocado. Esto puede cambiarse usando el operador **bind**.

3.7.1 El operador Bind

El operador bind provoca la busqueda de los nombres ejecutables en todo el procedimiento. Por cada nombre cuyo valor sea un operador, se reemplaza el nombre por el objeto operador. Esto ocurre en el momento en que bind es ejecutado. Esto tiene dos ventajas:

> Un procedimiento en el que se ha usado bind en su definicion, siempre ejecutara los operadores que fue instruido a ejecutar cuando fue definido, aun cuando los nombre de los operadores hayan sido redefinidos.
>
> Un procedimiento definidido asi se ejecuta mas rapido pues solo una vez se buscan los objetos operadores en las pilas, las demas veces se ejecutan los operadores directamente. Tipicamente los procedimientos definidos en el prologo se hacen con bind para estos propositos.

3.7.2 Nombres evaluados inmediatamente

Cuando el scanner PS encuentra un nombre precedido por dos slash (//), inmediatamente se busca su nombre en los diccionarios y se reemplaza el nombre por su valor actual. Este reemplazo se hace inmediatamente incluso para procedimientos cuya ejecucion es diferida. Si el scanner no pudo encontrar un valor para tal nombre entonces un error **undefined** ocurre.

4 Grafica

Los operadores graficos del lenguaje PS describen la apariencia de las paginas que seran reproducidas en un DSR. Estos operadores se dividen en seis grupos:

> Operadores de estado grafico. Estos operadores manipulan la estructura de dato llamada “estado grafico” (EG), que es el marco global donde los otros operadores graficos se ejecutan.
>
> Operadores de sistemas de coordenadas y matriz. El EG incluye la matriz de transformacion actual (current transformation matrix - CTM) que mapea las coordenadas usadas en el programa PS a las coordenadas del DSR. Lo operadores de este grupo manipulan la CTM para llevar a cabo toda combinacion de traslaciones, rotaciones, cambios de escala, etc.

Operadores de construccion de rutas (path). El EG incluye la ruta actual que define formas y trayectorias de lineas. Los operadores en este grupo empiezan una nueva ruta, agregan segmentos de linea y curvas a la ruta actual y cierran la ruta actual.

Operadores de pintado. Estos operadores pintan elementos graficos tales como lineas, areas rellenas e imagenes en la memoria del DSR. Los operadores de pintado son controlados por la ruta actual, color actual y otros parametros en el GS.

Operadores de caracteres y fonts. Estos operadores seleccionan y pintan caracteres de fonts. Dado que PS trata a los caracteres de los fonts como formas graficas generales, muchos de los operadores de fonts podrian ser agrupados con los operadores de construccion de path o de pintado. Sin embargo las estructuras de dato y mecanismos para operar con fonts son bastante especializadas.

Operadores de configuracion de dispositivo y de salida. Estos operadores transmiten o configuran la salida del programa PS al dispositivo del DSR.

4.1 Modelo de Imagen

Un programa PS construye una imagen pintando en una pagina en areas seleccionadas. El IPS acumula las marcas hechas por los operadores de pintado en la pagina actual.

Cuando un programa PS comienza, la pagina actual esta completamente en blanco. Cada operador que se ejecuta deja sus marcas en la pagina actual. Si una marca se superpone a otra ya existente entonces la ultima marca opaca complementamente a la anterior sin importar los colores en cuestion. Una vez que una pagina ha sido completamente compuesta se invoca el operador **showpage** que renderiza las marcas acumuladas al DSR y luego reinicia la pagina actual a blanco de nuevo[1].

Los principales operadores de pintado (OP), entre muchos otros, son: **fill** (pinta un area), **stroke** (pinta lineas), **image** (pinta imagenes), **show** (pinta formas de caracteres). Los OP toman varios parametros tanto implicitos como explicitos. Entre los implicitos esta el path actual usado por fill, stroke y show.

[1]En algunos casos las marcas puestas por los operadores de pintado son renderizadas inmediatamente al DSR. Esto es usual en programas interactivos de despliegue.

Un path consiste de una secuencia de puntos, lineas y curvas tanto conectadas como desconectadas y que juntas describen formas y sus posiciones. Un path se construye por la aplicaciones sucesiva de los operadores de construccion de path (OCP), cada uno de los cuales modifica el path actual, por ejemplo agregandole un nuevo elemento.

Entre los OCP estan **newpath**, **lineto**, **curveto**, **arc** y **closepath**. Cabe destacar que los OCP no ponen marcas en la pagina actual sino los OP. Los OCP crean las formas que los OP pintaran poniendo las marcas en la pagina actual. Algunos operadores tales como **ufill** y **ustroke** combinan construccion de path con pintado en una simple instruccion por razones de eficiencia.

Otros parametros implicitos de los OP son el color actual, el ancho actual de la linea, el font actual y muchos otros.

Un programa PS tipicamente contiene muchas instancias de la siguiente secuencia de pasos:

1. Construir un path usando los OCP
2. Configurar algunos parametros implicitos si es necesario
3. Realizar una OP

Existe aun otro parametro implicito a los OP, el *current clipping path* (CCP). Este parametro delimita el area de la pagina actual a un area en la cual se puede pintar. Inicialmente esta area es el total del area de imagen de la pagina actual. Usando el operador **clip**, un programa PS puede recortar esta area a cualquier forma deseada. Un OP puede intentar pintar fuera de esta area pero su marca sera ignorada.

A continuacion un par de ejemplos:

```
% el siguiente ejemplo dibuja un cuadrado
% para convertir a cm los 1/72 pulgadas hay que multiplicar por 72/2.54
= 28.346
/cm {28.346 mul} def
5 setlinewidth %ancho de las lineas
.5 cm .5 cm moveto
0 cm 1 cm rlineto
1 cm 0 cm rlineto
0 cm -1 cm rlineto
closepath
```

stroke

Todos los parametros implicitos de los OP se guardan en una estructura llamada “estado grafico” (EG), que se vera a continuacion.

4.2 Estado Grafico

El estado grafico (EG) es la estructura que mantiene los parametros implicitos de los OP. El EG no es un objeto sino que mantiene el estado de muchos objetos.

Existen dos mecanismos para grabar y restaurar el EG. Un mecanismo es la pila de estado grafico (PEG). Los operadores usados en este mecanismo son **gsave** (hace push de una copia del EG a la pila) y **grestore** (hace pop del EG de la pila). El otro mecanismo esta disponible en PS2 y usa el objeto gstate en memoria virtual. El operador **gstate** crea un nuevo objeto gstate. El operador **currentgstate** copia el EG a un objeto gstate dado como operando. El operador **setgstate** reemplaza el EG actual por el EG guardado en el objeto gstate dado como operando.

El EG se compone de parametros tanto dependientes como independientes del dispositivo. Los parametros dependientes del dispositivo son los parametros usados en el proceso de scan-conversion descrito anteriormente. Una descripcion de pagina independiente del dispositivo no deberia alterar estos ultimos parametros.

4.2.1 Parametros independientes del dispositivo

CTM. Tipo: array. Matriz actual de transformacion que mapea posiciones desde coordenadas de usuario a coordenadas de dispositivo. Esta matriz es modificada por los operadores de sistemas de coordenadas.

color. Tipo: varios. Valor inicial: negro. El color usado durante las operaciones de pintado. Este valor es interpretado de acuerdo al siguiente parametro: el espacio del color. (ver setgray, setrgbcolor, sethsbcolor, setcmykcolor, setcolor y setpattern, donde estas tres ultimas son PS2).

color space. Tipo: array. Valor inicial: DeviceGray. (PS2). Determina como los colores son interpretados.

position. Tipo: 2 numeros. Valor inicial: indefinido. Localizacion del “punto actual” (current point) en el espacio del usuario, es decir, las coordenadas del ultimo elemento del path actual.

path. Tipo: interno. Valor inicial: vacio. El path actual construido por los OCP.

clipping path. Tipo: interno. Valor inicial: toda la imagen visible de pagina. Un path que define las frontera de un area sobre la cual los OP trabajaran.

font. Tipo: dictionary. Valor inicial: un diccionario invalido de font. El conjutno de formas graficas (caracteres) que definen el actual typeface.

line width. Tipo: number. Valor inicial: 1. El ancho de las lineas a ser dibujadas por el operador stroke.

line cap. Tipo: integer. Valor inicial: 0. Un codigo que especifica la forma de los puntos finales de un path abierto (ver setlinecap).

line join. Tipo: integer. Valor inicial: 0. Un codigo que especifica la forma de las uniones entre segmentos conectados de una linea definida por pedazos (ver setlinejoin).

miter limit. Tipo: number. Valor inicial: 10.

dash pattern. Tipo: varios. Valor inicial: linea solida normal. Descripcion del patron a usar para dibujar lineas (ver setdash).

stroke adjust. Tipo: boolean. Valor inicial: tipicamente false para impresoras, true para pantallas. (PS2). Especifica si ajustar el ancho de lineas cuando estas puedan ser demasiado pequen~as para el ojo humano.

4.2.2 Parametros dependientes del dispositivo

color rendering. Tipo: dictionary. Valor inicial: dependiente de la instalacion. (PS2). Describe como transformar especificaciones de colores basados en CIE a valor de colores del dispositivo.

overprint. Tipo: boolean. Valor inicial: false. (PS2). Cuando se generan separaciones de color, especifica si el pintado en una separacion causa la modificacion del area correspondiente en la otra separacion o no.

black generation. Tipo: proc. Valor inicial: dependiente de la instalacion. (PS2). Calcula la cantidad de negro a usar cuando se convierten colores de RGB a CMYK.

undercolor removal. Tipo: proc. Valor inicial: dependiente de la instalacion. (PS2). Calcula la reduccion en la cantidad de cyan, magenta y amarillo a compensar por la cantidad de negro agregado por el parametro anterior.

transfer. Tipo: proc. Valor inicial: dependiente de la instalacion. Funcion que ajusta el valor de gris o componentes de color para corregir ???...

halftone

halftone phase

flatness

device

4.3 Sistemas de Coordenadas y Transformaciones

Los paths y formas son especificados en coordenadas cartesianas en relacion a la pagina actual.

4.3.1 Coordenadas de usuario y Coordenadas de dispositivo

Las coordenadas especificadas en un programa PS se refieren a un sistema de coordenadas que siempre es el mismo sin importar donde se vaya a desplegar o imprimir la pagina. Estas son las coordenadas del espacio de usuario (EU). Las coordenadas que finalmente maneja el dispositivo cuando el programa PS se interpreta son las coordenadas del espacio de dispositivo (ED) y varian de un dispositivo a otro dado que unos tienen mas resolucion que otros, las salidas en algunos son mas grandes que en otros, etc.

Los operandos de los OP estan en el EU. El IPS automaticamente transforma EU a ED en un proceso no visible al programa PS. Un programa PS raramente debe considerar las ED, esta es la clave de PS para lograr una descripcion de pagina independiente del dispositivo.

Se puede define un sistema de coordenadas en PS por:

La locacalizacion del origen

La orientacion del eje x e y

El largo de las unidades a lo largo de cada eje

Inicialmente, el EU predeterminado se define como:

El origen en la esquina inferior izquierda de la pagina o ventana

Orientacion positiva hacia arriba y a la derecha respectivamente para cada eje

El largo de una unidad tanto en el eje y como en el eje x es 1/72 pulgadas[2]

4.3.2 Transformaciones

Una matriz de transformacion especifica como transformar las coordenadas de un espacio de coordenadas a otro. Por ejemplo, la estructura ya vista, EG, incluye la matriz actual de transformacion (CTM) que describe la transformacion del EU al ED.

Los elementos de una matriz especifican los coeficientes de un par de ecuaciones lineales en x e y que generan los x e y transformdos.

Los operadores de matrices de transformaciones mas usados son aquellos que modificanel CTM en el EG. Estos operadores no crean una nueva matriz de transformaciones de la nada sino que modifican la actual. Entre los operadores que modifican el EU se encuentran:

translate. Mueve el origen del EU a una nueva posicion en la pagina actual sin modificar las orientaciones ni los largos de las unidades de los ejes.

rotate. rota los ejes de EU alrededor del origen en algun angulo y sin modificar la posicion del origen ni los largos de las unidades de los ejes.

scale. modifica los largos de las unidades de los ejes independientemente en el eje x e y. No modifica ni el origen ni la orientacion de los ejes.

concat. aplica una transformacion lineal dada al sistema de coordenadas del EU.

Con el uso de estos operadores se puede por ejemplo definir un sistema de coordenadas para cada elemento dentro de una misma pagina, lo que permite despues escalar o mover los elementos de una pagina muy facilmente.

[2]El taman~o predeterminado de la unidad (1/72 pulgadas) es aproximadamente lo que se conoce como un punto (“point”), una unidad de medida usada ampliamente en la industria de la impresion. Sin embargo no hay una definicion universal de un punto.

Ejemplo:

```
% A continuacion se define un procedimiento para construir un path
% cuadrado unitario en el sistema de coordenadas del EU con el origen
% predeterminado
/box {newpath
0 0 moveto
0 1 lineto
1 1 lineto
1 0 lineto
closepath
} def

% Se graba el EG actual y se crea uno nuevo a modificar
gsave

% Se modifica la matriz de transformacion asi cada cosa dibujada
% sera 72 veces mas grande, es decir, cada unidad sera una pulgada
% y no 1/72 pulgadas
72 72 scale

% Se dibuja un cuadrado en el origen.  Los lados de este miden 1 pulgada
box fill

% Se cambia la matriz de transformacion asi el origen ahora sera en (2,2)
2 2 translate

% Se dibuja otro cuadrado. Este se dibuja en (2,2)
box fill

% Se vuelve al EG predeterminado
grestore
```

4.3.3 Representacion y Manipulacion de las Matrices

Una matriz bi-dimensional es descrita matematicamente por una matriz 3x3:

$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

En PS esta matriz es representada por un arreglo de seis elementos: $\begin{bmatrix} a & b & c & d & t_x & t_y \end{bmatrix}$ omitiendo los elementos de la tercera columna que siempre son valores constantes. Esta matriz transforma un par de coordenadas (x, y) en (x', y') de acuerdo a las siguientes ecuaciones lineales:

$x' = ax + cy + t_x$

$y' = bx + dy + t_y$

Las transformaciones comunes son facilemente expresadas en esta notacion matricial. Una traslacion especificada por un desplazamiento $(t_x; t_y)$ es descrita por la matriz:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

El escalamiento por un factor s_x en la dimension x y s_y en la dimension y es descrito por:

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

La rotacion en sentido horario alrededor del origen por un angulo se describe:

$$\begin{bmatrix} \cos & \text{sen} & 0 \\ \text{sen} & \cos & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

El lenguaje PS puede expresar cualquier transformacion como una secuencia de estas operaciones basicas realizadas en un orden determinado. Una propiedad importante de la notacion matricial para las transformaciones es que un programa puede unir una secuencia de operaciones (hartas matrices) en una sola operacion (una sola matriz). Esta union de operaciones se realiza a traves de la multiplicaciones de las matrices correspondientes.

Los operadores **translate**, **scale** y **rotate** concatenan o unen la CTM con la matriz que describe la transformacion deseada, produciendo asi una nueva matriz que describe la combinacion de la original y las transformaciones adicionales.

$nuevaCTM = Transformacion \; originalCTM$

4.4 Construccion de Path

En PS un path define formas, trayectorias y regiones de todo tipo. Un path se compone de segmentos rectos y curvos que se pueden conectar con otros o no. La topologia de un path no esta restringida.

Un path en PS es representado por estructuras internas al IPS y aunque estas estrucuturas no son visibles al programador, un programa PS controla directamente su construccion y uso. El current path es parte del EG. Los operadores **gsave** y **grestore** guardan y restauran el current path como a los demas miembros del EG.

Un subpath es una secuencia de segmentos conectados. Un path se compone de uno o mas subpath desconectados. El operador **closepath** que conecta el final de un subpath a su comienzo.

El operador **newpath** sirve para inicializar un path vacio, para luego aplicar los demas operadores que agregaran nuevos segmentos al path actual. Sin embargo siempre debe ir un **moveto** antes de la aplicacion de la creacion del primer segmento. El punto final del ultimo segmento agregado a un path se llama punto actual. Si el path actual esta vacio entonces este punto final esta indefinido. La mayoria de los operadores de path (OCP) agregan segmentos al path actual en el punto actual. Si el punto actual esta indefinido entonces se ejecuta el error **nocurrentpoint**.

Los OCP mas comunes son:

moveto. Crea un nuevo punto actual sin agregar segmentos al path actual, asi se comienza un nuevo subpath del path actual.

lineto. agrega un segmento de linea recta al path, conectando el punto actual anterior al nuevo.

arc, arcn, arct y arcto. Agregan un arco de circulo al path actual.

curveto. Agrega un seccion de un curve cubica Bezier al path actual.

rmoveto, rlineto y rcurveto. Lo mismo que las anteriores pero especifican nuevos puntos como desplazamientos respecto del punto actual.

closepath. Agrega un segmento de linea recta conectando el punto actual al punto de comienzo del subpath actual (usualmente el punto especificado por moveto mas recientemente), cerrando el actual subpath.

Un path a ser usado mas de una vez en un programa PS, se puede definir como un procedimiento en el prologo PS y ser usado despues. Asi cada instancia del path puede ser construido y pintado en la pagina en los siguientes tres pasos:

1. Modificar la CTM si es necesario, para trasladar, escalar o rotar el path como se desee en la pagina.
2. Invocar el procedimiento para construir el path.
3. Ejecutar un OP para marcar el path en la pagina.

Por supuesto los tres pasos anterior se pueden encapsular en **gsave** y **grestore** para retornar a las condiciones predeterminadas despues que se ha creado el path.

Tal como se ha dicho un path no tiene restricciones en su topologia. Sin embargo la cantidad de segmentos que un path pueda tener puede eventualmente llenar la memoria disponible de un DSR, en cuyo caso un error **limitcheck** ocurrira.

4.5 Pintado

Los operadores de pintado (OP) marcan formas graficas en la pagina actual. Los operadores de proposito general principales aqui son **stroke** y **fill**. Otros operadores mas especializados son **image** (a describir mas adelante) y los operadores de fonts (tambien a describir mas adelante).

4.5.1 Stroke

El operador **stroke** dibuja una linea de algun ancho a lo largo del path actual. La apariencia de la linea final dibujada esta gobernada por los parametros del EG vistos anteriormente. Este operador trata cada subpath de un path separadamente:

> Si dos segmentos consecutivos estan conectados entonces la union entre ambos es tratada con el line join actual, que puede ser mitered, rounded o beveled (ver operador **setlinejoin**).
>
> Si un subpath esta abierto, las puntas desconectadas son tratadas con el line cap actual, que puede ser butt, rounded o square (ver operador **setlinecap**).
>
> Puntos en los cuales segmentos desconectados se unen o intersectan no reciben los tratos especiales de las puntas de los path o subpath. Es recomendable cerrar los path con **closepath** en lugar de un **lineto** explicito al comienzo.

4.5.2 Fill

El operador fill usa el color actual o patron para pintar la region entera encerrada por el path actual. Si el path consiste de varios subpaths desconectados entonces se pintan todos ellos. Si algun subpath esta cerrado entonces implicitamente se cierra antes de rellenada.

Para un path simple es relativamente sencillo determinar las regiones dentro del path y las que no. En un path complejo con muchas intersecciones entre el mismo esto no es tan obvio. PS puede utilizar dos algoritmos para determinar si una region se encuentra dentro de un path o no. Uno de estos algoritmo se llama “la regla del numero no cero”. Este algoritmo conceptualmente dibuja un rayo partiendo del punto en cuestion al infinito in alguna direccion y examina las intersecciones donde el segmento cruza el rayo hipotetico. Se empieza un contador en cero, se suma 1 al contador cada vez que un segmento cruza el rayo de izquierda a derecha y se resta 1 cada vez que un segmento del path cruza al rayo de derecha a izquierda. Despues de contar todos los cruces, si el resultado es cero entonces el punto esta fuera del path. En cualquier otro caso el punto esta dentro[3].

El otro algoritmo se llama “la regla par-impar”. Este algoritmo tambien dibuja hipoteticamente un rayo al infinito y cuenta los segmentos de path que cruza. Si este numero es impar entonces el punto esta dentro, si el numero es par entonces el punto esta afuera.

Estos dos algoritmos entregan el mismo resultado para path simples, pero arroja resultados diferentes en path mas complejos. El primer algoritmo es el mayormente usado en PS y es el standard para el operador **fill**. El operador **eofill** invoca el otro algoritmo.

4.5.3 Probando interioridad

En PS2 existen operadores capaces de determinar si un punto esta dentro de un path o si un path intersecta a otro. Estos operadores retornan un simple resultado boolean.

infill

instroke

[3]En caso de que un segmento coincida con el rayo o sea tangente a el se puede elegir otro rayo que no tenga este problema.

inufill y inustroke

ineofill y inueofill

4.6 Paths de Usuario (User Paths)

Un path de usuario es un procedimiento que consiste integramente de OCP y OP. Son una completa descripcion de un path en el EU. Ejemplo:

```
{
ucache % esto es opcional
100 200 400 500 setbbox % esto es requerido
150 200 moveto
250 200 400 390 400 460 curveto
400 480 350 500 250 500 curveto
100 400 lineto
closepath
} ufill
```

El operador **ufill** toma el procedimiento como operando y lo pinta su path definido. Al tomar el operador como procedimiento para la generacion de un path **ufill** invoca a **newpath** y **gsave** y **grestore** al comienzo y al final respectivamente, asi se genera y pinta un path sin dejar modificaciones en el EG.

Claramente se puede apreciar entonces que los operadores de paths de usuario se pueden describir a traves de OCP y OP. Sin embargo los operadores de paths de usuario ofrecen ciertas ventajas en eficiencia y conveniencia. Un path de usuario es auto-contenido, no depende de otros recursos como los del EG.

Los operadores permitidos en la construccion de paths de usuario son:

Operandos	Operador
	ucache
ll_x ll_y ur_x ur_y	setbbox
x y	moveto
dx dy	rmoveto
x y	lineto
dx dy	rlineto
x1 y1 x2 y2 x3 y3	curveto
dx1 dy1 dx2 dy2 dx3 dy3	rcurveto
x y r ang1 ang2	arc
x y r ang1 ang2	arcn
x1 y1 x2 y2 r	arct
	closepath

arct hace lo mismo que arcto pero no pone el resultado en la pila de operandos. El operador charpath no es permitido pues el path de usuario dependeria del font actual. Los operadores especiales de la construccion de paths de usuario son ucache y setbbox. El operador ucache provoca que el path de usuario se guarde en una memoria cache especial para acelear su ejecucion en usos frecuentes. El operador setbbox establece una caja de frontera en EU encerrando el path de usuario.

4.7 Forms

(PS2). Un form es una descripcion auto-contenida de alguna grafica, texto o imagen a ser pintada multiples veces. Estos forms en PS son definidos como procedimientos que ejecutan operador graficos. Lo que distingue a un form de un procedimiento es que el form es auto-contenido y se comporta de acuerdo a ciertas reglas. Dependiendo del EG la ejecucion de un form tendra distintos resultados de salida en apariencia. La salida de un form solo depende del EG en un momento dado.

4.8 Espacios de Color

Las facilidades que ofrece PS en este aspecto estan divididas en dos partes:

> Especificacion de color. PS puede especificar colores abstractos de una manera independiente del dispositivo.

Renderizacion de color. El IPS reproduce los colores en el DSR a traves de una serie de procesos que incluyen conversiones de colores, correcciones gamma, halftoning y scan conversion. Esta parte del PS esta gobernada por el IPS y el programador no deberia meterse aqui para lograr una descripcion independiente del dispositivo.

Un programa PS configura primero el espacio de colores con el operador **setcolorspace**. Luego se selecciona el color con el operador **setcolor**. Luego de esto los operadores **fill** o **stroke** pueden poner marcas en la pagina sobre el path actual. Tambien existen operadores que configuran el espacio de color y el color mismo en un solo paso: **setgray**, **setrgbcolor**, **sethsbcolor**, **setcmykcolor** y **setpattern**.

Para una imagen el color y espacio de color configurado en el EG no es usado, la imagen misma contiene la definicion de color en cada punto a la resolucion de la imagen.

4.9 Patrones (patterns)

(PS2). Cuando los operadores fill, stroke y show pintan areas de un pagina con el color actual, ellas son ordinariamente pintadas de un simple color. Es posible en PS pintar con el patron de alguna figura repetida. Esta figura que se repite en el pintado se llama patron.

El pintado con patrones se compone de los siguientes cuatro pasos:

1. Describir el prototipo del patron. Esto es realizado mediante la creacion de un diccionario que contiene informacion acerca del patron. Un elemento importante del diccionario es el **PaintProc**, que es un procedimiento que puede ser ejecutado para pintar un celda del patron (celda del patron es la figura que se repite).

2. Instanciar el patron. El operador **makepattern** copia el prototipo del patron en el diccionario y produce una instancia del patron en el actual EU. El taman~o del patron y su espaciamiento son determinados por la CTM en el momento de la ejecucion de **makepattern**.

3. Seleccionar el patron como color actual. Hay un espacio de color especial, llamado Pattern cuyos valores de colores son diccionarios patrones en lugar de valores numericos. El operador setpattern instala el patron como el color actual en un simple paso.

4. Invocar los OP tales como fill, stroke, imagemask o show para pintar con el patron seleccionado las areas a pintar.

(AQUI FALTA AGREGAR EJEMPLOS CREACION DE PATRONES, ETC)

4.10 Imagenes

Una imagene es un arreglo rectangular de valores de muestra, cada uno representando algun color. Cada muestra en el arreglo consta de 1, 3 o 4 componentes (escala de grises, RGB o CMYK). Cada una de estas componentes consta de 1, 2, 4, 8 o 12 bits, permitiendo 2, 4, 16, 256 o 4096 diferentes valores para cada componente.

(AQUI FALTA MUCHO DE IMAGENES)

5 Fonts

5.1 Uso de los fonts

Los pasos para el uso basico de fonts es:

1. Seleccionar un font a usar
2. Escalar ese font al taman~o deseado e instalarlo como el font actual en el EG
3. Especificar una posicion en la pagina
4. Pintar un string de caracteres alli

A continuacion un ejemplo:

```
/Helvetica findfont
12 scalefont setfont
288 720 moveto
(ABC) show
```

5.1.1 Seleccionando Fonts

Para hacer mas eficiente la seleccion de fonts se pueden crear procedimientos en el prologo con los fonts a usar despues a lo largo del documento. A continuacion se muestra un ejemplo:

```
% Estos procedimientos en el prologo
/FSD {findfont exch scalefont def} bind def
/SMS {setfont moveto show} bind def
/MS {moveto show} bind def

% Los siguientes elementos del diccionario con sus nombre
% asociados se ponen al comienzo del script
/F1 10 /Helvetica FSD
/F2 10 /Helvetica-Oblique FSD
/F3 10 /Helvetica-Bold FSD

% Y ahora se usan en las paginas de la siguiente forma:
(Esto esta en Helvetica.) 10 78 F1 SMS
(Esto tambien.) 10 66 MS
(Esto esta en Helvetica-Oblique.) 10 54 F2 SMS
(Esto esta en Helvetica-Bold.) 10 42 F3 SMS
(Al igual que esto.) 10 30 MS
```

En el ejemplo anterior, FSD toma un nombre de variable, un factor de escalamiento y un nombre de font. Genera un diccionario de font descrito por el nombre del font y su factor de escalamiento, luego ejecuta **def** para asociar el diccionario de font con el nombre de variable dado. SMS escribe un string dado en las coordenadas dadas usando el font dado. MS escribe un string en las coordenadas dadas.

En PS2 existe un operador muy util y eficiente para hacer la labor de **findfont**, **scalefont** y **setfont** en un solo paso. El operador se llama **selectfont**.

5.1.2 Efectos graficos especiales con fonts

El uso normal de show es pintar los caracteres rellenos entero con negro. Sin embargo se puede combinar el uso de operadores graficos con los operadores de fonts para lograr otros efectos.

El color usado por los OP esta determinado por el color actual en el EG. El color predeterminado es negro pero este puede ser cambiado con operadores tales como **setgray** u otro de los OP. Por ejemplo para pintar en 50% de gris:

```
/Helvetica-Bold findfont 48 scalefont setfont
20 40 moveto
.5 setgray
(ABC) show
```

Otras manipulaciones pueden ser llevadas a cabo tratando los caracteres como paths. Esto es posible para caracteres definidos vectorialmente y no bitmap. El operador **charpath** devuelve un path vacio en este ultimo caso. Ejemplo:

```
/Helvetica findfont 48 scalefont setfont
20 38 moveto
(ABC) false charpath
2 setlinewidth stroke
```

5.1.3 Posicionamiento de caracteres

El ancho de un caracter es el espacio que un caracter ocupa en la linea de un texto, es decir, el la distancia que el punto actual se mueve cuando el caracteres es escrito. En algunos fonts el ancho de los caracteres es constante y tales fonts se dicen monoespaciados. La mayoria de los fonts en cambio asocian un ancho diferente a cada caracter. Estos ultimos fonts son llamados fonts proporcionales.

La informacion del ancho de cada caracter es guardada en el diccionario de font. Es posible lograr efectos de modificacion de ese ancho con los operadores **show, xshow, yshow, xyshow, widthshow, ashow, awidthshow**. El operador **stringwidth** sirve para obtener el ancho de un string.

5.2 Tipos de Fonts

Existen tres tipos de Fonts:

> Type 0. Es un font compuesto por otros fonts, llamados fonts base. Estos fonts son una caracteristica PS2.
>
> Type 1. Es un font base que define las formas de los caracteres usando procedimientos codificados de una manera especial. Esta codificacion es descrita en el libro llamado “Adobe Type 1 Font Format”.

Type 3. Es un font base definido por el usuario a traves de procedimientos comunes que definen la forma de los caracteres. En el diccionario de font estos procedimientos son los valores de las entradas llamadas **BuilGlyph** o **BuildChar**.

6 Convenciones de comentarios

Todo programa PS debe contener algunas lineas y comentarios. La primera linea a contener es la siguiente:

%!PS-Adobe-1.0

Luego deben estar los comentarios de encabezado siguientes:

%%DocumentoFonts: font1 font2 ...

%%Title: titulo

%%Creator: nombre

%%CreationDate: fecha

%%For: nombre

%%Pages: numero

%%BoundingBox: llx lly urx ury : Coordenadas en el EU del fondo derecha (llx, lly) y de la derecha arriba (urx, ury).

%%EndComments : fin de los comentarios de encabezado

Y luego, en el cuerpo del programa PS, debe aparecer los siguientes:

%%EndProlog

%%Page: etiqueta numero : marca el comienzo de una pagina del script. Cada pagina numerada puede ser conocida dentro del documento con una etiqueta como xv por ejemplo.

%%PageFonts

%%Trailer : marca el final de la ultima pagina del documento. El siguiente codigo es entendido a ser informacion que antes en los comentarios se expreso como “atend”.