POSTSCRIPT - EN INTRODUKTION

Magnus Bondesson
Institutionen för Datavetenskap
1988-03-07, 1993-12-05, 2000-02-14, 2004-09-23

PostScript är ett språk som beskriver en dokumentsidas innehåll. En sida betraktas som grafisk, dvs den innehåller bildelement där dock merparten ofta utgörs av text. På engelska används beteckningen PDL (Page Description Language) för denna typ av språk.

Det ursprungliga syftet var att kunna skicka informationen om en sida mellan en dator och en skrivare. Det skedde tidigare endera genom att man lät datorn generera sidans bitmönster eller genom att man skickade mycket enkla kommandon (ren text i form av rader eller när det gäller grafik "rita ett streck" och ofta som en s k ESC-sekvens). I bägge fallen var sättet utrustningsberoende. En bitmönster för en sida med hög upplösning skulle belasta datorn mycket. PostScript är **utrustningsoberoende** och gör att kommunikationen kan ske på hög nivå och i mänskligt läsbar form (vilket reducerar överföringsproblemen). Det är dessutom **utvidgbart**, dvs en tillämpning kan arbeta på ännu högre nivå. En viktig förutsättning för PostScript är att datorkraften numera kan distribueras. För att t ex skrivaren skall kunna producera sidan, måste den innehålla en dator med stort primärminne som genererar bitmönstret utifrån den överförda beskrivningen, dvs postscriptprogrammet för sidan.

PostScript är idag en s k industristandard för kommunikation mellan en dator och högkvalitativa laserskrivare. Det har också använts för att beskriva innehållet i fönster på en datorskärm (NeWS för SUN och Display PostScript för bl a NeXT).

PostScript togs fram av företaget Adobe i USA, som startades i slutet av 1982. Dess framgång inleddes med introduktionen av laserskrivaren LaserWriter från Apple under januari 1985, ungefär ett år efter introduktionen av Macintosh. Denna laserskrivare var den första som innehöll en PostScript-tolk. Många andra företag har sedan nappat på kroken, och alla laserskrivare utom de billigaste fungerar så. Adobe har tydligen Apple att tacka för sina framgångar. Sannolikt är dock Apples tacksamhetsskuld ännu större. Utan denna typ av skrivare hade kanske användningsområdet för Mac varit mindre och därmed även Apples försäljningsframgångar (en del tyder dock på att Apple hade planerat ett eget överföringsformat, som överenstämmer med det som används internt i datorn). Det är för övrigt intressant att notera att Adobe startades av personer med ursprung hos Xerox PARC (Xerox Palo Alto Research Center) liksom att Apple hämtat de grundläggande idéerna till Macs användarsnitt därifrån (mot ersättning). Adobe, Apple och Xerox ligger bara några kilometer från varandra i förstäderna Mountain View, Cupertino respektive Palo Alto till San Francisco.

PostScript är inte ett språk som man normalt programmerar i, även om det är fullt möjligt. Utan tillämpningsprogrammet låter användaren arbeta intuitivt och genererar ett postscriptprogram som överförs till och sedan tolkas i skrivaren.

PostScript är ett rikt programspråk med betydande kraft när det gäller grafik. Diskussionen här går inte in på detaljer och bygger på att läsaren har en allmän kännedom om programspråk. Vi tar inte alls upp effektivitetsfrågor.

1.0 PostScript som programspråk

PostScript är ett **stackorienterat programspråk**, som använder sig av bakvänd polsk notation (det senare av prestandaskäl). Följande två rader adderar två tal respektive två variabler

```
13 7 add
a b add
```

PostScript använder sig som sagt av stackar. De båda mest intressanta är **operandstacken** (eng. operand stack) och **ordboksstacken** (eng. dictionary stack). På operandstacken placeras alla operander. T ex innebär de två raderna ovan att

1. Talet 13 läggs på operandstacken.
2. Talet 7 läggs på stacken.
3. De två översta elementen på stacken adderas, varpå de tas bort från stacken. och summan läggs på stacken.
4. Värdet på variabeln a hämtas från ordboksstacken och läggs på operandstacken.
5. Dito för variabeln b.
6. Som i punkt 3.

Ordboksstacken innehåller variabler och procedurer (eller operatorer). Mera precist finns i denna stack variablernas värden och beskrivningarna av procedurerna. Dessa kan vara systemdefinierade eller användardefinierade. I själva verket består stacken av två delar, dels systemordboken och dels användarordboken. Man kan dessutom lägga på ytterligare ordböcker, vilket förklarar benämningen stack, men det går vi inte in på närmare. Utan sådana extra ordböcker blir dock alla namn globala.

När ett namn, t ex en variabel eller en operator, påträffas som symbol genomletas i första hand användarordboken och därnäst systemordboken, vilket gör att man kan omdefiniera systemoperatorer.

Inga namn deklareras som i Pascal. En variabel med namnet summa ges värde på föjande sätt:

```
/summa uttryck som ger ett tal på stacken def
```

Enklast är en enkel initiering med konstant, t ex

```
/summa 7 def
```

som motsvarar `summa:=7` i Pascal. Men även pascals allmänna tilldelningssats har en motsvarighet. T ex blir `summa:=summa+1` i PostScript

```
/summa summa 1 add def
```

Beteckningen

```
/namn
```

är en symbol som används när man ger ett namn innebörd (första gången eller vid ändring). Symbolen placeras när den påträffas på operandstacken. Ovanpå denna placeras det framräknade uttrycksvärdet. Operatorn *def* verkar på de två översta operanderna. Den undersöker först om namnet redan finns i användar- eller systemordboken. Om namnet inte finns placeras det i användarordboken tillsammans med värdet och stacken minskas med två element. Om det finns i användarordboken ändras värdet i stället. Om det finns i systemordboken förs det in i användarordboken.

Andra operatorer med en uppenbar betydelse är *mul*, *div*, *sub*, *eq* (plus många andra).

En procedur definieras på motsvarande sätt med

```
/procedurnamn
{
        procedurkropp
} def
```

Parametrar förs över via stacken. En procedur som ökar översta elementet på stacken med 1 kan se ut så här:

```
/inc
{
       1 add
} def
```

Om det finns många parametrar är det sällan dessa kan hanteras i den ordning de finns på stacken. Då definieras variabler motsvarande dem med hjälp av exch-operatorn, se nedan.

När man beskriver de olika operatorerna i PostScript används ibland skrivsättet

```
arg1 arg2 arg3 operator arg4 arg5
```

Till vänster anges de översta elementen på stacken före utförandet av operatorn. Till höger anges vad dessa ersätts av när operationen utförts. Ett streck betecknar härvid ett tomt stackelement.

Det finns ett antal operatorer som bara manipulerar med stacken. En är *dup* som kopierar ett element och lägger det överst:

```
arg dup arg arg
```

Och naturligtvis finns en operator som bara tar bort översta elementet:

```
arg pop –
```

En annan är `exch` som byter plats på de båda översta elementen:

```
arg1 arg2 exch arg2 arg1
```

Denna operator är speciellt användbar när det gäller skrivande av procedurer med parametrar som ligger på stacken.

Exempel: En procedur som beräknar $x^2 + y^2$, där x och y placerats på stacken och lägger resultatet på stacken ges av

```
/length
{       /y exch def
        /x exch def
        x x mul y y mul add
} def
```

Observera att variablerna *x* och *y* nu blir globala. För att få lokalitet måste man införa en extra ordbok.

Naturligtvis finns olika styrkonstruktioner, t ex repetitionskonstruktionen

```
antal {kropp} repeat
```

som upprepar innehållet i kroppen det antal gånger som anges av antal. Antalsuppgiften placeras på stacken (eller finns där redan) och tas sedan bort.

En villkorskonstruktion är

```
villkor {sannkropp} {falskkropp} ifelse
```

Om villkoret är sant utförs sannkroppen annars falskkroppen.

Medan vi i andra sammanhang är vana vid att t ex stackar och vektorer är homogena, dvs bara innehåller objekt av en typ, är det en begränsning som saknas i PostScript.

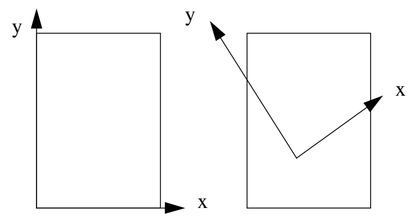
Uppdelningen i rader är godtycklig i ett Postscript-program, dvs byte till ny rad påverkar ej programmets funktion (undantag: i strängar är radslut signifikant).

Som i andra programspråk finns kommentarer. Sådana utgörs av

```
% text
```

2.0 Grafik: koordinatsystem och tillstånd

När man anger koordinater sker det i ett användarkoordinatsystem som är utrustningsoberoende. Detta koordinatsystem har som standard origo i papprets nedre vänstra hrn och en enhet motsvarar 1/72 tum, se vänstra figuren nedan. Vill man använda ett annat koordinatsystem går det utmärkt, genom att man gör en koordinattransformation sammansatt av en translation, rotation och skalning. Origo kan då hamna på annan plats på pappret, systemet kan vridas och enheten kan bli en en annan. Se högra figuren nedan.



Origo flyttas med

```
tx ty translate
```

Systemet vrids moturs en viss vinkel (i grader) med

```
vinkel rotate
```

och enheten görs till tum i stället för 1/72 tum med

```
72 72 scale
```

Detta känner vi igen från andra paket för grafik i 2D. Även här finns en bakomliggande transformationsmatris som man kan manipulera med på ett otal olika sätt. Detta är dessutom det enda sättet att placera ut ett viss objekt på olika sätt på pappret. Man skiljer således inte i PostScript på modellering och koordinattransformation.

I varje ögonblick finns det ett visst grafiskt tillstånd. Hit hör aktuellt koordinatsystem, linjetjocklek ritintensitet med mera. Man sparar ett visst grafiskt tillstånd (på en grafikstack) med operatorn

```
gsave
```

och återställer det med

```
grestore
```

Eftersom lagringen sker på en stack, kan sådana par nästlas.

Linetjockleken sätts med

```
tjocklek setlinewidth
```

Som enhet anävnds samma som i gällande koordinatsystem. Med talet 0 får man tunnast möjliga linje på aktuell utrustning.

Ritintensitet sätts med

```
intensitet setgray
```

varvid intensiteten är ett tal mellan 0 och 1 och 0 betyder svart och 1 vitt.

Koordinater kan anges som reella tal. Postscriptsystemet sköter om alla omräkningar till utrustningens egna koordinater och gör det på bästa möjliga sätt.

3.0 Grafik: Operatorer

De grundläggande operatorerna är *moveto* och *lineto* som används enligt

```
x y moveto
x y lineto
```

och flyttar pennan respektive definierar en linje. Den senare ritar dock inte i sig någon linje i sidminnet, utan används för att bygga upp geometriska objekt (på engelska kallade *path*). Ett sådant objekt kan innehålla (eventuellt från varandra separerade) delar som utgörs av linjestycken, bågar och bezierkurvor. Om objektet skall vara slutet är det bäst att ange operatorn *closepath* snarare än t ex ett avslutande *lineto* till startpunkten. Ett nytt tomt objekt skapas med operatorn *newpath*.

Det aktuella geometriska objektet (det som skapats med den senaste newpath-operatorn) ritas in i sidminnet med operatorn *stroke*. Vill man att objektet i stället skall fyllas med gällande intensitet invändigt används operatorn *fill* (i detta fall sluts eventuella separerade delar). I båda fallen görs ett automatiskt *newpath*, dvs objektet är inte längre åtkomligt.

Exempel: Följande procedur kan användas för att rita en rektangel. Som argument anges hörnen, först övre vänstra hörnpunkten och därnäst den undre högra.

```
/rectangle
{       /bottom exch def
        /right exch def
        /top exch def
        /left exch def
        newpath
        left top moveto right top lineto
        right bottom lineto left bottom lineto
        closepath       % eller ev left top lineto
} def
```

Tre rektanglar kan sedan ritas in i sidminnet med


```
100 100 200 50 rectangle stroke
300 100 400 50 rectangle fill
gsave
45 rotate
0 200 100 100 rectangle stroke
grestore
```

Den första rektangeln är ofylld, den andra fylld och den tredje något vriden.

En båge genereras med operatorn *arc*, som tar en mittpunkt, en radie samt startvinkel och slutvinkel (i grader) som argument:

```
x0 y0 radie startvinkel slutvinkel arc
```

Vinklarna räknas motsols.

Exempel: Proceduren *circle* ritar en cirkel vars mittpunkt och radie är argument

```
/circle
{       newpath
        % mittpunkt och radie finns redan på stacken
        0 360 arc stroke
} def
```

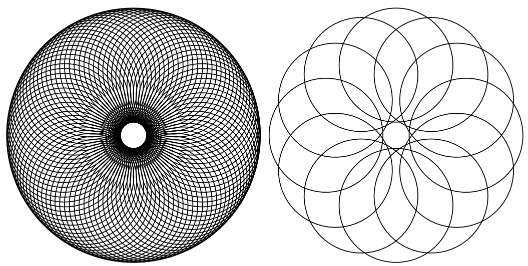
Proceduren *spiro* ritar sedan ett antal cirklar genom att rotera den senast ritade en aning. Antalet är argument till proceduren och vridningsvinkeln beräknas genom att dividera 360 grader med antalet.

```
/spiro
{       /nr exch def
        nr {
               80 0 65 circle
               360 nr div rotate
        } repeat
} def
```

Med

```
150 150 translate % Flyttar origo snett uppåt
90 spiro
300 0 translate % Flyttar origo ytterligare åt höger
12 spiro
```

ritas en vacker figur bestående av 90 respektive 12 roterade cirklar.



Operatorerna *moveto*, *lineto* och *curveto* finns också i relativa versioner.

Bezierkurvelement genereras med operatorn *curveto*, som tar tre punkter som argument

```
x1 y1 x2 y2 x3 y3 curveto
```

Den genererar en bezierkurva med gällande position som startpunkt (lämpligen gör man först ett *moveto*) och x3 y3 som slutpunkt och med x1 y1 och x2 y2 som styrpunkter.

När en sida är uppbyggd fullständigt i sidminnet och man vill ha den utskriven på skrivaren ges kommandot

```
showpage
```

Inget ritas förrän detta kommando exekverats!

Klippningsområde definieras genom att man bildar en "path" (geometriskt objekt) och sedan ger operatorn *clip*. Det finns systemvariabler som informerar om tillstånd. T ex placerar

```
currentpoint
```

koordinaterna för befintlig punkt på stacken.

PostScript kan även hantera rasterbilder (bitmönster).

4.0 Text

Det är naturligtvis huvudsakligen text som matas ut på en skrivare. I PostScript väljer man typsnitt, typsnittsvariation och storlek enligt följande förfarande:

```
/Times-Bold findfont 72 scalefont setfont
```

Först typsnittsnamnet (här Times) inklusive eventuell typsnittsvariation (här Bold, dvs fetstil; normalfallet skrivs för detta typsnitt Roman), därefter operatorn *findfont*, som tar fram en lagrad matematisk-geometrisk beskrivning av typsnittet i en normaliserad storlek, därefter storleken uttryckt i för tillfället gällande koordinatenhet, sedan operatorn *scalefont*, som skapar ett bitmönster i den aktuella typsnittsstorleken och till sist operatorn *setfont*, som gör typsnittet till det gällande. Lämpligen gör man en procedur för varje sådant val (eventuellt parametriserad med storleken).

En textsträng skrivs in i sidminnet med operatorn *show*. T ex hamnar texten 'Exempel 1:' på viss plats i sidminnet med

```
10 20 moveto
(Exempel 1:) show
```

Någon automatisk uppdelning på rader sker inte, utan det måste programmeraren själv sköta om. Om man tittar på en postscript-fil genererad av något dokumentproduktionssystem, t ex FrameMaker, så är texten uppbruten i en mängd delar. Detta beror dels på radbrytningsinstruktionerna, dels på att programmet måste sköta om alla typografiska finesser som t ex knipning (eng. kerning), som innebär att vissa bokstavskombinationer skall sättas tätare än andra. Dessutom inför programmet ofta egna kommandon i form av postscript-procedurer.

5.0 En förhandsbetittare (eng. previewer)

Ett okommersiellt program GhostScript (finns liksom övriga program som nämns nedan även för PC-miljö) låter oss inspektera en postscript-fil (glöm inte showpage) innan den skrivs ut:


```
> gs filnamn.ps
```

Det finns ett alternativ, som är en påbyggnad till GhostScript, som låter oss flytta oss godtyckligt mellan sidorna och dessutom skriva ut enstaka sidor (kommandot gv är samma sak):

```
> ghostview filnamn.ps
```

Programmet GhostScript kan arbeta i en interaktiv mod, vilket gör att man kan experimentera litet med PostScript vid datorn. Vi visar en liten sådan session.

```
> gs
Initializing... done.
GS>newpath
GS>0 0 moveto
GS>100 100 lineto
GS>stroke    %Nu ritas ett streck
GS>newpath
GS>200 100 lineto
Error: /nocurrentpoint in --lineto--

Operand stack:
 200 100
Execution stack:
 %interp_exit --nostringval-- --nostringval-
- %loop_continue --nostringval-- --nostring
val-- false --nostringval-- --nostringval-- --nostringval--
Dictionary stack:
 461/479 5/200
GS<2>100 100 moveto
GS<2>200 100 lineto
GS<2>stroke
GS<2>quit
```

Anm. <2> anger att det finns 2 element på stacken som är obehandlade. Borde tagits bort med *pop*.

6.0 Vill du veta mer? Praktiskt

PostScript Language Reference Manual (Second Edition, Addison-Wesley 1990) är en präktigt tjock bok som innehåller allt och bitvis är riktigt läsvärd. Hittas numera på nätet.

Om du vill skriva ut en postscript-fil så använd `lpr -Pskrivare fil.ps`. Första raden skall inledas med %!, annars skrivs bara postscript-koden ut. Man kan skapa en postscriptfil med innehållet (som ett raster) i ett fönster eller för en del av skärmen på flera olika sätt. Lämpligen betittar man filen med *gs* före utskrift på papper.

Många program, t ex MATLAB, låter oss spara figurer som postscriptfiler, så att de senare kan fogas in i andra dokument. Man måste se till att filen är av EPS-typ (Encapsulated PostScript) och helst EPSI (EPS with Independant bitmap preview), så att dokumentprogrammet som normalt inte kan PostScript förmår reservera tillräckligt utrymme på sidan och visa upp figuren. Det finns program, t ex *ps2epsi*, som gör om en vanlig postscriptfil till detta format.

7.0 PDF

Har man sagt A får man säga B. PostScript fungerade länge som standard vid utbyte av dokument med modern formatering, men numera har PDF (Portable Document Format, också från Adobe och kom 1993) övertagit den rollen (någon vill säkert säga att MS Words doc-format är en konkurrent), för vilken det finns en fri läsare *Acrobat Reader* (också den ovan nämnda *ghostview* duger). *Framemaker* kan producera PDF.

Några egenskaper hos PDF:

- PDF bygger på PostScript men är inte ett programspråk (det finns inga variabler eller funktioner/procedurer). Fortfarande utrustnings- och plattformsoberoende.
- PDF är ett strukturerat format, dvs en PDF-fil består av från varandra oberoende objekt, t ex sidor. Det gör att man snabbare kan komma åt en viss sida och gör det också möjligt att inkrementellt ändra i en fil. Strukturen hålles samman i bl a en korsreferenstabell.
- PDF har en massa utvidgningar (vissa bara för interaktivt utnyttjande): länkar, knappar, ljud och rörliga bilder.
- PDF komprimerar bilder med JPEG, LZW eller CCITT och text med bl a LZW. Har dessutom kortare kommandonamn, t ex m i st f moveto.
- PDF har en mekanism som sägs ge bättre typsnitt om ett visst typsnitt saknas i betraktarens miljö.
- Det finns programvara för konvertering från PS till PDF (fri: ps2pdf, kommersiell t ex: Adobe Distiller). Vid utskrift görs konvertering i andra riktningen.