PDFTEX, een eerste indruk

Hans Hagen

April 1 1996

De aanleiding

Enkele weken terug vroeg Erik Frambach me of ik bereid was de web2c TEX omgeving te testen. Deze omgeving zal mogelijk de kern vormen van 4TEX voor Windows (of hoe dat ook mag heten). Het installeren van deze hi–TEX omgeving was vrij eenvoudig, met name omdat er slechts één installatiefile is. We kunnen op dit moment binnen onze organisatie dus kiezen uit HugeEmTEX (dat DOS ondersteunt), YandYTEX (dat, dankzij het dynamisch geheugenbeheer, het gelijktijdig draaien van meerdere jobs mogelijk maakt) en Web2cTEX (waarvan als enige ook de source beschikbaar is).

In eerste instantie installeerde ik alleen de meest relevante delen van deze distributie: het programma zelf, de pool file en wat DLL–files. Aangezien wij vrij veel PDF output produceren, drukte ik ook wat documentatie van de variant PDFTEX af. Bestudering hiervan was reden genoeg om ook dit programma maar eens te installeren.

PDFTEX, wat is dat?

Knuth heeft, sinds hij TEX en METAFONT ontwikkelde, regelmatig geschreven en gezegd dat zijn versie weliswaar bevroren is, maar dat anderen er naar hartelust aan mogen sleutelen. De afgelopen jaren hebben slechts weinigen deze uitdaging aanvaard.

John Hobby heeft bijvoorbeeld METAFONT zodanig aangepast en uitgebreid, dat geavanceerde POSTSCRIPT output mogelijk is. Vanuit het oogpunt van de gebruiker is dit wellicht de meest belangwekkende aanpassing van een van Knuth's programma's. Ik zou zeggen, werp eens een blik op MetaPost.

Op het TEX front hebben de ontwikkelingen zich vooral beperkt tot uitbreidingen die eerder format–schrijvers dan gebruikers zullen plezieren. Zo is er al enige tijd de multi–linguale duizendpoot Omega en is er sinds kort e-TEX, waarmee ik binnenkort aan de slag hoop te gaan.

Het door Han The Thanh, Petr Sojka & Jiří Zlatuška ontwikkelde PDFTEX is voor TEX, wat MetaPost is voor METAFONT. Waar MetaPost aansluiting realiseerde bij het (inmiddels tot standaard verheven) POSTSCRIPT, zorgt PDFTEX voor aansluiting op het (wellicht op korte termijn tot standaard te verheffen) PDF formaat.

Het traject om van een in TEX gecodeerde brontekst tot PDF te komen, loopt via DVI en POSTSCRIPT. Met de komst van PDFTEX komen twee slagen te vervallen. Dit betekent niet alleen een winst in termen van verwerkingstijd, het scheelt ook schijfruimte. Bovendien neemt de kans af dat problemen optreden tijdens het vertalen naar PDF. Dit laatste aspect moet niet worden verwaarloosd. Zo zitten er in de recente versie van Acrobat Distiller een paar storende fouten, die op deze manier goed te omzeilen zijn. Een nog storender fout in Acrobat Reader dwingt me echter definitieve versie te verwerken met Distiller (level 2.1 code).

De functionaliteit

Samengevat komt het er dus op neer dat PDFTEX in plaats van DVI files PDF files genereert. Nu zal een kritische lezer zich inmiddels al hebben afgevraagd of alles wat we tot nu toe kon met TEX ook met PDFTEX kan. Immers, veel macropakketten gebruiken de TEX primitieve `\special` om effecten te bewerkstelligen die TEX niet in zich heeft:

- het integreren van illustraties (bitmap en vector)
- het weergeven van (tekst in) kleur
- roteren, ronde hoeken en andere grafische geintjes
- het ondersteunen van hypertekstmogelijkheden

We kunnen potentiële gebruikers geruststellen: veel is mogelijk.

Men kan bijvoorbeeld figuren integreren met behulp van de primitieve `\pdfimage`. De reikwijdte van deze primitieve is echter beperkt tot files in het bitmap PNG formaat. Andere bitmap formaten kunnen echter vooraf worden omgezet in dit formaat.

Anders ligt het met vector graphics. Aangezien de eerste processlag meteen de laatste is, kunnen we deze niet zoals gebruikelijk afhandelen met specials. Specials hebben bij afwezigheid van postprocessing eigenlijk hun functie verloren. De primitieve `\special` is dan ook gereduceerd tot een die alleen tussenvoegen van PDF code toestaat. Aangezien met GhostView figuren in EPS formaat kunnen worden omgezet in PDF formaat, kunnen toch dergelijke figuren worden tussengevoegd. Hier blijkt het nut van een discussielijst, waarop Tanmoy Bhattacharya uit de doeken deed hoe een door GhostView geproduceerde file kon worden omgezet in een PDF file die door een utility wordt omgezet in een PDFTEX file, die op haar beurt weer met `\input` kon worden tussengevoegd (een doordenkertje). Wat experimenteren leerde me dat de tweede tussenstap niet echt nodig is en dat TEX het heel goed zelf afkan, en nog snel ook. Echter, niet alle door GhostView geproduceerde output is op dit moment geschikt. Voor hen die

dat nog niet weten: GhostView is (en wordt) een uitstekend alternatief voor Acrobat Reader!

De tot PDF special gereduceerde \special kan ook worden gebruikt om kleuren in te stellen. En aangezien TEX uitstekend in staat is zelf het kleurmanagement te voeren, is ondersteuning van kleur dus ook geen probleem.

Wat lastiger ligt het met specifieke grafische mogelijkheden. Waar POSTSCRIPT een volwassen programmeertaal is, is PDF slechts een weergavetaal. Dit betekent dat we in PDF bijvoorbeeld niet kunnen rekenen. Het PDF producerende programma moet dus het rekenwerk doen, en dat is in dit geval dus TEX. Het implementeren van bijvoorbeeld kaders met ronde hoeken is nauwelijks een probleem, evenals roteren. Geavanceerde macropakketten als PSTricks zullen echter grotendeels moeten worden herschreven, tenzij men natuurlijk de plaatjes afzonderlijk produceert en omzet in PDF. Het aanpassen van MetaPost danwel het omzetten van door dit programma geproduceerde output zou nauwelijk sproblemen moeten opleveren, maar op dit moment expandeert 5K POSTSCRIPT nog tot 633K PDF, dat door PDFTEX weer wordt gecomprimeerd tot 120K.

PDFTEX gaat ver in de ondersteuning van hypertekst mogelijkheden. Er zijn daartoe een aantal nieuwe primitieven beschikbaar —de PDFTEX primitieven beginnen allemaal met \pdf— die een curieuze mix vormen van TEX en verbatim PDF. Het gebruik van de primitieven \pdfdest en \pdfannotlink, iets dat de meeste macropakketten verbergen voor hun gebruikers, vergt dus wel wat kennis van PDF.

Een zeer groot voordeel van PDFTEX is dat het programma nog steeds DVI–code kan produceren. Persoonlijk houd ik graag dit achterdeurtje open, niet in de laatste plaats omdat het uit DVI te produceren POSTSCRIPT vaak veel beter overdraagbaar is dan PDF, wat vooral een gevolg is van tekortkomingen in viewers. Bovendien is DVI output vaak prettig compact en zijn DVI viewers veel beter geschikt om te testen: ze starten sneller op en zijn in weergave superieur.

De keuze tussen PDF en DVI wordt bepaald door de primitieve \pdfoutput. Aanzienlijk compactere files worden verkregen met behulp van \pdfcompresslevel en \pdffontsubset. Dit laatste benadrukt nog eens een essentieel verschil met DVI: de fonts zitten in de file!

Het aanpassen van macropakketten

De vraag zal wellicht rijzen hoe eenvoudig een beetje macropakket is aan te passen aan PDFTEX. Ik kan zeggen: dat valt reuze mee. Het heeft me precies een dag gekost om een interface te schrijven die de functionaliteit van ConTEXt ondersteunt. We praten dan over: kleur, kaders met ronde hoeken, het tussenvoegen van plaatjes, hyperlinks in alle toonaarden en wat opsmuk voor interactieve teksten, zoals full screen opstarten (iets dat onder Distiller 3 geheid tot problemen leidt). De documentatie was helaas zeer gebrekkig, maar de WEB–source bood gelukkig uitkomst.

Binnen ConTEXt zijn alle specials ondergebracht in aparte modules. Ik hoefde dus alleen een nieuwe module toe te voegen. Om een indruk te krijgen van de complexiteit van zo'n module en de aard van de aanpassingen, geven we hieronder een deel van de betreffende module weer:

```
\startspecials[tpd]

\definespecial\dostartgraymode#1%
  {\special{PDF: #1 g}}

\definespecial\dostopgraymode%
  {\special{PDF: 0 g}}

\definespecial\dostartcolormode#1#2#3%
  {\special{PDF: #1 #2 #3 rg}}

\definespecial\dostopcolormode
  {\special{PDF: 0 0 0 rg}}

\definespecial\dostartrotation#1%
  {\processaction
     [#1]
     [ 90=>\special{PDF:q  0  1 -1  0 0 0 cm},
      180=>\special{PDF:q -1  0  0 -1 0 0 cm},
      270=>\special{PDF:q  0 -1  1  0 0 0 cm},
      360=>\special{PDF:q  1  0  0  1 0 0 cm}]}

\definespecial\dostoprotation%
  {\special{PDF:Q}}

\definespecial\dosetupinteraction%
  {\pdfoutput=1\relax
   \pdfcompresslevel=9\relax
   \pdffontsubset=1\relax}

\definespecial\dostartthisislocation#1%
  {\pdfdest name {#1} fitp}}

\definespecial\dostartgotolocation#1#2#3#4#5%
  {\doifelse{#3}{\jobname}
     {\!!doneafalse}
     {\doifelse{#3}{}
        {\!!doneafalse}
        {\!!doneatrue}}%
   \pdfannotlink
     width #1sp
     height #2sp
     depth 0pt
     attr{/Border [ 16 16 0 ]}
     goto \if!!donea file {#3.pdf} \fi name {#4}%
   \pdfendlink}

\definespecial\dosetupscreen#1#2#3#4#5%
  {\pdfpagewidth =#3sp
   \pdfpageheight=#4sp
   \ifcase#5\else
     \pdfcatalog {/PageMode /FullScreen}%
   \fi}
```

```
\definespecial\dosetupidentity#1#2#3#4#5%
  {\pdfinfo
     {/Title (#1)
      /Subject (#2)
      /Author (#3)
      /Creator (#4)
      /ModificationDate (#5)}}

\stopspecials
```

De niet getoonde macros betreffen het tussenvoegen van externe files (figuren), het tekenen van kaders met ronde hoeken en wat additionele interactiviteit.

Binnen ConTEXt kunnen specials exclusief of parallel worden uitgevoerd. De PDFTEX specials moeten per se exclusief zijn, dat wil zeggen dat er niet gelijktijdig specials voor andere drivers mogen worden weggeschreven. De switch van DVI naar PDF output wordt dan ook gemaakt met:

```
\gebruikspecials[reset,tpd]
```

Ik kan van deze plaats niet zeggen hoe eenvoudig het is om een willekeurig macropakket aan te passen. Veel hangt af van de wijze waarop specials worden ondersteund. Her en der door de source verspreidde `\specials` maken het er in ieder geval niet eenvoudiger op.

Enkele kanttekeningen

Is hiermee DVI overbodig geworden? Zoals ik al aangaf, blijven we natuurlijk DVI gebruiken. Tijdens het schrijven van deze tekst maak ik gebruik van DVI previewers (van E.Mattes en YandY) die ik geen van beiden zou willen missen. Hetzelfde geldt voor de hoogwaardige POSTSCRIPT output.

Om TIFF en EPS figuren te kunnen opnemen moet men eerst de figuren converteren met GhostView. Op dit moment is die conversie niet optimaal, zowel in termen van kwaliteit als in termen van omvang van de files. Hoe lang zal het echter nog duren voordat alle tekenpakketten files ook in PDF formaat kunnen exporteren? Hoe dan ook: het hoort niet tot de taken van TEX dit te doen. In die zin is de primitieve `\pdfimage` dan ook eigenlijk overbodig. De kunst is nu nog de bounding box informatie uit de tussen te voegen PDF file te destilleren.

Het afhandelen van verbatim POSTSCRIPT valt natuurlijk buiten de scope van PDFTEX. Specials zijn niet voor niets specials. In die zin hadden de ontwerpers beter een nieuwe primitieve `\pdfspecial` kunnen implementeren en `\special` kunnen gebruiken in de traditionele zin, namelijk het tussenvoegen van verbatim code in de vorm van commentaar.

Ik ben hierboven voorbij gegaan aan enkele ander mogelijkheden. Zo ondersteunt PDFTEX het aanmaken van outlines. Dit lijken me voor TEX gebruikers nauwelijks nodig, omdat de meeste wat omvangrijker documenten zelf zorg dragen voor hun inhoudsopgave. Waarom zouden we hoogwaardige typografie inleveren voor een door een viewer gegenereerde inhoudsopgave, die bovendien nog een deel van het scherm in beslag neemt.

PDFTEX ondersteunt ook het hergebruik van informatie (meestal graphics), of in PDF termen: form objects. Daarnaast kunnen article treads worden aangemaakt. Dit laatste is vooral handig wanneer we lezers door een tekst willen loodsen. Tot slot noemen we nog de mogelijkheid om annotaties op te nemen. Natuurlijk heb ik nog wel een verlanglijstje.