Computergestuurd Zetten of Electronisch publiceren? Nieuwe trends in wetenschappelijk publiceren*

Philip Taylor

Abstract

W ciągu ostatnich 15 lat skład tekstu wspomagany komputerowo całkowicie wyeliminował stosowanie tradycyjnych technik drukarskich. Obecnie stoimy przed jescze bardziej radykalną rewolucją technologiczną, jaką stanowi pojawienie się wydawnictw electronicznych (electronic publishing - w skrócie e.p.).

W przeciwieństwie do składu komputerowego i tradycyjnego, publicacje electroniczne w ogóle nie wykorzystują papieru, nośnikem informacji staje się ekran komputera.

Potencjalne korzyści płynące z zatosowania tej nowej metody publicacji wydają się oczywiste: znaczna redukcja kosztów i niemal natychmiastowa dystrybucja do zainteresowanych odbiorców. Nie można jednak pomicąć i wad, takich jak: możliwość bezprawnego kopiowania, łatwość plagiatów i nielegalnej redystrybucji.

Obie wspomniane metody publicacji różnią się w sposób zasadniczy: o ile systemy składu tekstu z góry narzucają postać końcową strony, to systemy publicacji electronicznych stwarzają jej bardziej niezależną reprezentację, w której ostateczna forma zależy racjey od programu prezentującego dokument (tzw. viewera, czy po polsku – przeglądarki).

W niniejszej pracy omówione są najkowsze osiągnięcia zarówno w dziedzinie składu komputerowego jak i publicacji electronicznych. Wskazuje się również na różnice tych dwóch metod rozpowszechniania.

Een korte geschiedenis van TEX

Tot ongeveer 15 jaar geleden werd het zetwerk vrijwel volledig genegeerd door de overgrote meerderheid van alle wiskundigen, wetenschappers en geleerden in het algemeen: manuscripten werden voorbereid met een typemachine, de meer vergezochte tekens (voor wiskundigen betekende dat vrijwel alle symbolen) werden later met de hand ingetekend, en het geheel werd gewoon verstuurd naar de uitgever. Een tijdje later kwamen dan de voorproeven terug, verbeteringen werden genoteerd in de kantlijn, en vervolgens werd het geheel weer teruggestuurd naar de uitgever. Eenzelfde maar kortere cyclus volgde vervolgens voor de huisproeven, en uiteindelijk verscheen het in de door de auteur gewenste vorm in het voltooide boek. Op geen enkel punt was er een direct contact tussen de auteur en de zetter, en feitelijk was de eerste persoon zich ook vrijwel onbewust van het bestaan van de tweede.

De zetter was zich echter maar al te goed bewust van het bestaan van de auteur: wiskundig zetwerk werd traditioneel in de zetterswereld altijd aangeduid als 'strafwerk', omdat het berucht moeilijk is om het correct uit te voeren. In de tijd waarin een collega makkelijk in staat was tot het zetten van tien pagina's tekst, was de wiskundige zetter net in staat tot het zetten van een enkele pagina, en zelfs dan wist hij al dat het heel waarschijnlijk was dat het geheel meer dan één keer overnieuw gezet zou moeten worden, aangezien wiskundigen erg gretig zijn in het uitvinden van nieuwe symbolen als de bestaande net niet helemaal lijken te voldoen. Aangezien de zetter dat symbool nog nooit eerder gezien zou hebben, zou hij (heel redelijk) aannemen dat het gewoon een slecht ingetekende versie van een ander symbool betrof, één die hij wel herkende, en vervolgens dat symbool gebruikt hebben ...

Het lag voor de hand dat enkele van de meer oplettende auteurs begonnen te experimenteren met computertechnologie zodra die meer algemeen beschikbaar werd, en voor een tijdje was de academische wereld er schijnbaar van overtuigd dat alle problemen opgelost konden worden als het mogelijk zou zijn om nog een paar extra tekens toe te voegen aan het margrietwiel van de Diablo printer. Er waren zelfs bedrijven die gespecialiseerd waren in het aanpassen van margrietwielen door het vervangen van een kennelijk onnodig teken door één waarvan de eigenaar dacht dat het volstrekt onmisbaar was. Natuurlijk was deze aanpak gedoemd te mislukken: Het is net zo onmogelijk om wiskundig zetwerk te doen met een vaste set van 144 tekens als dat het is met een set van 128 tekens, en ondanks de beste pogingen van de betrokkenen verdween de margrietwiel-printer toch al snel naar de prullenbak.

Gelijktijdig met deze pogingen, begonnen de producenten van matrix-printers een zekere invloed te krijgen. Met een 7×5 matrix, is er een potentieel aantal verschillende tekens beschikbaar van

$$\sum_{i=0}^{35}\binom{35}{i} = 2^{35}$$

*Dit artikel werd oorspronkelijk gepresenteerd als een lezing voor een gecombineerde bijeenkomst van de Poolse Wiskundige Vereniging en het EmNet Projekt (Euromath Trust) in Torun, Polen, 1995. Sindsdien is het gepubliceerd in het engels in GUST (Grupa Użytkowników Systemu TEX); Biuletyn Zeszyt 6, pp. 12–27, een poolse vertaling verscheen in *Pro Dialog*, Augustus 1996, en de engelse versie werd wederom gedrukt in TUGboat 17 (1996), pp. 367–381.

Het artikel is geherdrukt met toestemming van de auteur, editors en de organisatie van de oorspronkelijke bijeenkomst. Deze vertaling in het nederlands is gemaakt door Taco Hoekwater, eventuele manco's in het taalgebruik zijn zijn verantwoordelijkheid.

(inderdaad een enorm aantal). Helaas zijn er daarvan een aantal vrijwel onmogelijk van elkaar te onderscheiden: een enkele stip op de coördinaten (4,3) lijkt verbazend veel op een andere stip op de coördinaten (4,4), zelfs voor de meest oplettende lezer (Ik geloof dat er een totaal van 33 034 338 305 *verschillende* tekens mogelijk is, in tegenstelling tot een totaal van 34 359 738 368 totale tekens. Daarbij wordt een tekens als *verschillend* aangeduid indien het niet eenvoudigweg wordt verkregen door het horizontaal, verticaal of in beide richtingen verschuiven van een ander teken. Deze getallen zijn gebaseerd op een analyse uitgevoerd door Dr. Warren Dicks van de Autonome Universiteit van Barcelona). Daarbij komt dat de kwaliteit van de uitvoer van een 7 × 5 matrix printer zo ongelooflijk slecht is dat *geen enkele* poging gedaan zou moeten worden om er boeken mee te maken – een goed advies dat toendertijd helaas zelden werd gevolgd.

Uiteraard moest er geschikte software worden geschreven om die nieuwe technologische uitvindingen te kunnen gebruiken, en met name de Unix wereld besloot te standaardiseren rond het programma ROFF en zijn afgeleiden: NROFF, TROFF en uiteindelijk DITROFF kwamen allemaal aan bod. Helaas bood geen één van de uit ROFF afgeleide programma's de mogelijkheid om direct wiskunde te zetten, zodat aparte programma's gebruikt moesten worden zoals EQN en TBL om de wiskundige functionaliteit toe te voegen. Er waren ook wel commerciële systemen, die bijvoorbeeld gebruikt werden voor het zetten van publicaties zoals de *Transactions of the American Mathematical Society*, maar die waren zowel duur als ondoorgrondelijk, met gebruikmaking van een non-mnemonische syntax voor het beschrijven van de mogelijke wiskunde constructies.

Gelukkig (zoals terugkijkend overduidelijk zichtbaar is), was er op zijn minst één uitmundende wiskundige die geloofde dat iets beters niet alleen gemaakt kon, maar ook gemaakt *moest* worden; en omdat hij niet alleen een wiskundige was maar ook een computer-wetenschapper, besloot hij het zelf te schrijven. Zijn naam was Knuth, en zijn programma was TEX.

En toch, we danken het alleen aan een gelukkig toeval dat TEX geboren werd. In die tijd werkte Knuth aan zijn *opus magnum*, een serie van 7 boeken getiteld *The Art of Computer Programming*, en in 1977 bleek dat de populariteit van de eerste delen zo groot was dat deel 2 al aan een herdruk toe was. De timing hiervan viel helaas zó dat hoewel de eerste editie nog gedrukt was met de traditionele loodmethode, de tweede editie gedrukt werd met een van de eerste fotozetters [aan de lezer: door het hele artikel gebruik ik het woord *zetter* voor zowel de *persoon* die het zetwerk uitvoert, als voor de *apparatuur* die daarvoor gebruikt wordt. Ik hoop dat het altijd duidelijk is welke van de twee betekenissen bedoeld wordt, aangezien er geen ander woord is dat eenvoudig en eenduidig gebruikt zou kunnen worden om één van de twee betekenissen te vervangen]. En hoewel de nieuwe fotozetter *in theorie* in staat was tot het leveren van gelijkwaardige, of zelfs betere, uitvoer dan de traditionele loodmachine die eerder gebruikt werd, liet het resultaat veel te wensen over. Knuth, als wiskundige en wetenschapper, was ervan overtuigd dat de fout niet in het apparaat zelf lag, maar in de software die gebruikt werd om het te besturen. En liever dan zijn levenswerk te zien verschijnen in een tweede-keus formaat, besloot hij een klein gedeelte van zijn professionele leven te besteden aan het schrijven van een verzameling programma's die *wel* in staat zouden zijn het volledige potentieel van de fotozetter te gebruiken. Toen hij deze moedige beslissing nam kon hij nog niet vermoeden dat het uiteindelijk minstens tien jaar zou gaan duren, in plaats van het ene jaar dat hij ervoor geraamd had, dit ondanks het feit dat hij binnen dat ene jaar toch beslist een demostreerbare werkende versie had.

De eerste gepubliceerde referentie naar TEX is waarschijnlijk *Mathetical Typography*, uitgegeven als rapport *STAN-CS-78-648* door de computer wetenschappelijke afdeling van de Stanford Universiteit; in de bibliografie daarvan geeft Knuth de uiteindelijke referentie op als: *Tau Epsilon Chi, a system for technical text.* Op dat moment was die nog 'in bewerking' en nu helaas uit de handel. Voor lezers die geïnteresseerd zijn in het onderwerp, is het genoemde artikel beslist het lezen waard, en de bibliografie alleen al maakt het een waardevolle aanwinst. Het werd heruitgegeven in het *Bulletin of the American Mathematical Society*, en in die vorm zou het nog steeds beschikbaar moeten zijn.

TEX was zowel typisch voor als afwijkend van de programma's uit die tijd: het was typisch aangezien het volledig script-georienteerd was, een gevolg van het feit dat het programma ontstond voordat er een wijd-gebruikte grafische werkomgeving beschikbaar was; het was afwijkend in die zin dat het een volledige programmeerbare *macro* programmeertaal bevatte, waarin er geen gereserveerde woorden waren, en waarin zelfs eenvoudige tekens in staat waren dynamisch van betekenis te veranderen. Zodoende bestond een TEX document uit zowel de tekst die gezet moest worden als uit de commando's die daarvoor zorgden, en alleen TEX zelf was eenduidig in staat om te beslissen of iets beschouwd moest worden als 'programma' of als 'tekst'.

Ondankt het feit dat het feitelijk werd geschreven voor het bereiken van één enkel doel – het zetten van Deel 2 van *The Art of Computer Programming* – ontwikkelde TEX al snel een eigen leven, en al vlug werd het de *de facto* standaard voor zetten binnen een groot gedeelte van Stanford University. Al snel was TEX ook daarbuiten beroemd, en in 1980 onstond de TEX Users Group, met bestuursleden van ver buiten de grenzen van de faculteit van Stanford. De American Mathematical Society was gerepresenteerd binnen dat bestuur, en de verbindingen tussen Knuth en het AMS waren hecht: Knuth droeg het TEX logo over aan het AMS die daarna een trademark aanvroegen als bescherming tegen niet-officiele wijzigingen aan het programma – helaas werd het verzoek afgewezen wegens een eerdere toewijzing van TEX aan Honeywell, maar ondanks het gebrek aan een officiële registratie zorgen de hoge standing en hoge waardering voor Knuth ervoor dat

het TEX logo (en zijn niet-gezette vervanger, TeX) alom erkend en gerespecteerd wordt.

Binnen een paar jaar werd duidelijk dat de originele implementatie van TEX nog een paar dingen te wensen overliet, zowel in termen van functionaliteit als in termen van portabiliteit, en Knuth besloot beide bij te werken door TEX compleet te herschrijven. Deze keer besloot hij SAIL ('Stanford Artificial Intelligence Language') te vermijden, en in plaats daarvan gebruik te maken van de veel algemener gebruikte programmeertaal Pascal. Om de portabiliteit nog groter te maken, adopteerde hij alleen een precieze subset van Pascal, met gebruikmaking van alleen die mogelijkheden waarvan hij erop vertrouwde dat ze in elke Pascal implementatie aanwezig (of eenvoudig na te maken) zouden zijn. Tevens besloot hij daarbij meteen gebruik te maken van deze kans om het programma te schrijven in een vorm die hij 'beletterd' noemde: hij wilde dat mensen in staat zouden zijn om de broncode van TEX te kunnen *lezen* op dezelfde manier waarop ze een boek zouden lezen, zodat ze daarbij in staat zouden zijn om te profiteren van een groot produkt van software-ontwikkeling, gepresenteerd in een leesbare vorm. Wederom besloot Knuth dat hiervoor geen geschikte hulpmiddelen beschikbaar waren, en wederom boog hij daarom af van het hoofdproject voor het ontwikkelen en implementeren van het WEB concept, met de twee daarbij behorende programma's TANGLE en WEAVE.

Een WEB programma bestaat uit een hoogst gestyleerd dialect van Pascal, met grote tussengevoegde commentaren die de bedoeling en functie van de elementen en modules van het programma uitleggen (Ik vermoed dat Knuth dit zou ontkennen, en in plaats daarvan zou zeggen dat een WEB programma bestaat uit een hoogst uitgebreide beschrijving van de werking van het programma, hier en daar onderbroken met stukjes Pascal die die functionaliteit implementeren; en ik vermoed dat hij vrijwel zeker gelijk zou hebben!). Door het toe te staan de stukken van het Pascal programma in een willekeurige opeenvolging te presenteren (in tegenstelling tot de strenge regels over de volgorde die worden gesteld in de Pascal standaard), geeft WEB de programmeur de mogelijkheid om de onderdelen van een programma te presenteren in een natuurlijke en logische volgorde, in plaats van de kunstmatige opeenvolging die wordt afgedwongen door de Pascal ontwerp-eis van 'efficiënte compileerbaarheid'. Het is de taak van het programma TANGLE om ervoor te zorgen dat de brokstukken worden samengevoegd op de manier die geëist wordt door Pascal, en het is de taak van WEAVE om ervoor te zorgen dat de commentaren samen met de brokjes code omgewerkt worden in een vorm die geschikt is om direct door TEX gezet te worden.

Zodoende werd TEX voor de eerst keer zelf-verwijzend: om de Pascal code te krijgen uit de WEB broncode, was een werkende versie van TANGLE nodig, en om een leesbare versie van de code te krijgen was een werkende versie van WEAVE nodig, maar beide programma's zijn zelf geschreven in WEB, dus om een werkende versie van TANGLE te maken is een werkende versie van TANGLE nodig, *ad infinitum*. Natuurlijk is 'bootstrapping' (zoals de technische term luidt) al lang bekend en begrepen binnen de computerwereld, en de verwachting was dat 'met de hand compileren' van TANGLE uit de WEB broncode daarvan ruim binnen de grenzen van de mogelijkheden van de 'gemiddelde implementator' zou liggen. Ik herinner me echter maar al te goed het trauma dat een collega meemaakte toen hij probeerde dat 'bootstrappen' zelf te doen...

Tijdens de her-implementatie herschreef Knuth vrijwel het gehele TEX programma: hij had veel geleerd over de grenzen van TEX tijdens die eerste jaren van gebruik, en in 1982 was er een volledige herschreven TEX onstaan. Deze versie van TEX (vaak aangeduid als TEX82 om het verschil duidelijk te maken met de eerdere versie die analoog hieraan werd aangegeven als TEX78) werd snel overgedragen naar een wijde verzameling machines, en is misschien wel het wijdst verspreidde programma in de wereld op dit moment, beschikbaar als het is voor elk type systeem vanaf de kleinste PC tot de grootste super-computer. De vrijwel algemene acceptatie van TEX als *het* standaardpakket voor computer-zetten is vrijwel zeker het resultaat van een grote verzameling van positieve eigenschappen: De broncode van het programma, en van de meeste implementaties, is gratis beschikbaar danwel voor een minimaal bedrag dat uitsluitend dient ter compensatie van de kosten van het materiaal waarop verspreiding plaatsvindt; het programma is vrijwel bug-vrij, een claim die tot-voorkort ondersteund werd door Knuth zelf met een cheque die uitgereikt werd voor iedere bug die gemeld werd, waarbij de waarde van de cheque elk jaar verdubbelde sinds het systeem werd ingevoerd (hij biedt nog steeds een cheque aan, maar de waarde verdubbelt niet meer elk jaar, omdat hij heeft geschat dat anders over niet al te lang het een groter bedrag zal zijn dan de federale reserves...); het programma is erg stabiel (er waren vrijwel geen belangrijke veranderingen gedurende de periode 1982–1990, en evenzo zijn er vrijwel geheel geen veranderingen geweest sinds 1990, en die zullen er ook niet meer komen in de toekomst); en er is een enorme hoeveelheid gebruikers over de hele wereld, waarvan de meeste maar al te bereid zijn om hun kennis over te dragen aan wie die nodig heeft. Zodoende kunnen problemen die een gevolg zijn van een gebrek aan ervaring met TEX snel worden opgelost door een bericht te sturen naar één van de TEX-verbonden maillijsten en nieuwsgroepen (zelfs diegenen zonder netwerk-toegang zijn niet afgesloten, aangezien het TUG (TEX Users Group) kantoor zelfs telefonische ondersteuning levert van 3 uur 's morgens tot laat in de avond – een service die *niet* alleen beschikbaar is voor leden van TUG).

Kortom, tijdens de jaren tachtig kwan TEX boven als *het* standaard pakket voor computer-zetten: het was beschikbaar voor bijna elk beschikbaar systeem, stuurprogramma's werden geschreven voor alles van dot-matrix printers tot 2400 dpi fotozetters (maar *niet* voor margrietwiel printers!), en een steeds groeiende hoeveelheid aan publicaties verscheen die gezet waren met TEX, danwel

TEX als onderwerp hadden, of beide. Veel wetenschappelijke journals adopteerden TEX (of een van de afgeleiden zoals LATEX, wat gezien kan worden als een wat beperkter maar meer gebruikersvriendelijk 'front-end' voor TEX) als het standaard formaat waarin artikelen aangeleverd moesten worden. Aangezien een auteur heel eenvoudig een proef van een artikel kon maken op zijn locale implementatie van TEX, en aangezien TEX *garandeerde* dezelfde resultaten op te leveren ongeacht op welk systeem het draaide, werd het aantal contacten tussen auteur en uitgever geminimaliseerd, en iedereen had daar voordeel van. Bovendien, aangezien TEX was ontworpen door een wiskundige, en aangezien een groot gedeelte van de *raison d'être* was geweest het mogelijk te maken om wiskunde vrijwel even eenvoudig te zetten als broodtekst, verliep de acceptatie van TEX binnen de wiskundige gemeenschap zo mogelijk nog sneller dan binnen de algemene wetenschappelijke en academische wereld.

Om een eenvoudig voorbeeld te geven van het waarom TEX ideaal geschikt is voor het zetten van wiskundig werk, overwegen we de hieronder staande groep vergelijkingen:

$$\begin{aligned}\left(\int_{-\infty}^{\infty} e^{-x^2}\,dx\right)^2 &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-(x^2+y^2)}\,dx\,dy \\ &= \int_0^{2\pi}\int_0^{\infty} e^{-2r} r\,dr\,d\theta \\ &= \int_0^{2\pi}\left(-\frac{e^{-r^2}}{2}\Big|_{r=0}^{r=\infty}\right)d\theta \\ &= \pi. \qquad (11)\end{aligned}$$

Een wiskundige die dit met de hand schrijft zal vrijwel zeker beginnen met het meest linkse element van de eerste regel, om vervolgens van links naar rechts te werken, ondertussen afwisselend tussen de basisregel en de super- en subscripten als de logica dat dicteert. Een pure WYSIWYG ('What You See Is What You Get') tekstverwerker daarentegen zou van de typist eisen dat die elke rij van de formules zou verdelen is horizontale banen (dus de bovenste baan zou bijvoorbeeld alleen de tekens ∞, 2, ∞ en ∞ bevatten), en deze dan laag voor laag in te voeren. Omdat in het algemeen WYSIWYG systemen voorgaande of volgende lijnen niet automatische aanpassen als een tussenliggende regel verkort of verlengd wordt, is het corrigeren van zulke formules extreem lastig en foutgevoelig. Nieuwere WYSIWIG-achtige systemen eisen weer een andere aanpak waarbij de auteur de formule moet invoeren in de volgorde die wordt gedicteerd door de ontleed-boom van het programma. Het hoeft niet eens vermeld te worden dat ook deze aanpak onredelijke eisen stelt aan de auteur.

TEX staat het de wiskundige toe formules in the voeren in de meest natuurlijke manier, links beginnen en rechts eindigend; de uitlijning wordt automatisch bijgehouden als er toevoegingen of verwijderingen plaatvinden, en zelfs het horizontaal lijnen van de vier = tekens wordt automatisch uitgevoerd, vrijwel volledig onafhankelijk van de lengte van de linker- en rechter-delen. Om dit duidelijker te maken volgt hieronder de exacte code die gebruikt werd om de tabel te zetten:

```
$$
\eqalignno
 {\biggl(
    \int_{-\infty}^\infty e^{-x^2}\,dx
  \biggl)^2
  &=\int_{-\infty}^\infty
    \int_{-\infty}^\infty
        e^{-(x^2+y^2)}\,dx\,dy \cr
  &=\int_0^{2\pi}\int_0^\infty
        e^{-2r}r\,dr\,d\theta \cr
  &=\int_0^{2\pi}
    \biggl(
        -{e^{-r^2}\over 2}
            \bigg \vert_{r=0}^{r=\infty}\,
    \biggr)
    \,d\theta \cr
  &=\pi.&(11)\cr
}
$$
```

Het is de moeite waard om op te merken dat TEX de spaties in de wiskundige tekst volledig negeert. Dit is omdat de regels voor het toevoegen van witruimte binnen wiskundig zetwerk ingewikkeld zijn, en niet verwacht kan worden dat die regels correct begrepen worden door een simpele wiskundige! Daardoor is de layout van de bovenstaande formules enkel en alleen voor het gemak van de auteur, en wordt volledig genegeerd door TEX, die veel meer geïnteresseerd is in speciale tekens zoals dollartekens, backslashes, accolades, streepjes, dakjes en ampersands. En terwijl elk van deze tekens een speciale betekenis heeft voor TEX (een dollarteken bijvoorbeeld begint en eindigt een stuk wiskundige tekst) kan toch elk van deze betekenissen eenvoudig worden aangepast, toegewezen aan een ander teken, of zelfs volledig uitgeschakeld worden indien de functionaliteit niet nodig is. Bijvoorbeeld, als een toetsenbord geen backslash kende, zou het eenvoudig zijn om de betekenis daarvan te koppelen aan een andere toets (zoals bijvoorbeeld de yen, als het een japans toetsenbord betrof).

Bovendien kunt u zien dat TEX gemakkelijk te onthouden opdrachtreeksen ('commando's', vooraf gegaan door een backslash) gebruikt. Om een paar voorbeelden te noemen: `\int` stelt een integraalteken voor, `\infty` staat voor oneindig, `\exp` de exponent-operatie (het exponentieelteken *e*) en zo verder. Samengestelde subscripts en superscripts worden gepresenteerd in de logische volgorde in plaats van volgens de verticale volgorde op de pagina, en de mogelijkheid is open om TEX hints te geven over de logische structuur van de expressie, zodat (bijvoorbeeld) `\,` gebruikt wordt om wat extra wit toe te voegen voor een differentiaal zoals $d\theta$, waardoor zowel de verschijning als de leesbaarheid van de formule verbeterd wordt.

De aantrekkingskracht van TEX voor wiskundigen is duidelijk: een hoogst logische taal voor het markeren, die ingegeven kan worden door middel van elk willekeurig toetsenbord; de mogelijkheid om een grote verzameling wiskundige symbolen te gebruiken; de uitvoer is volgens professionele standaards; ondersteuning door een grote verzameling journals; en de mogelijkheid om een

proef uit te draaien op alles vanaf een dot-matrix printer tot een 600 dpi laser printer. Voeg daaraan toe de nu alom aanwezige mogelijkheid om het resultaat te bekijken op een computerscherm (iets waarvan de eerste TEX-promotors slechts konden dromen), en het wordt moeilijk uit te leggen waarom een willekeurige wiskundige met de beschikking over een computer TEX *niet* voor zijn artikelen zou gebruiken!

Maar het gebruik van TEX is niet alleen beperkt tot wiskundigen en Noord-Amerikanen, en tijdens de TEX User Group conferentie in 1989 drong een invloedrijke en welbespraakte groep europese TEX gebruikers zich op aan Knuth en slaagde erin hem ervan te overtuigen dat, ondanks zijn stelling van de vorige dag dat de ontwikkeling van TEX afgelopen was, er toch zaken ontbraken die ervoor zorgden dat TEX geheel nutteloos was voor het grootste deel van de wereld, aangezien hoewel het zich perfect gedroeg in talen zonder accenten, TEX vreselijk tekort schoot bij het zetten van talen waarin meer accenten voorkwamen dan slechts af en toe een enkele diacriet. Knuth, die onderkende dat het argument geldig was, gaf toe dat er iets gedaan moest worden.

Het resultaat van dit alles was TEX 3: TEX 82 werd bekend simpelweg TEX 2, en TEX 3 werd de Enige Echte TEX. In de praktijk gebeurde dit niet zomaar: degenen die geen behoefte hadden aan de uitgebreide diacrietenondersteuning van TEX 3 bleven gewoon TEX 2 gebruiken, een een tijdlang werden TEX macro schrijvers gedwongen hoogst defensieve code te schrijven die eerst de omgeving moest controleren voordat er aannames gedaan konden worden over (bijvoorbeeld) het aantal verschillende tekens waarmee TEX intern om kon gaan (dat was 128 vóór TEX 3, en 256 daarna). Met de verschijning van TEX 3 maakte Knuth *absoluut* duidelijk dat dit echt het einde was van de ontwikkeling van TEX: hij had betere dingen te doen met zijn tijd, en TEX was nu bevroren (uitgezonderd het oplossen van essentiële bugs, die hij bleef uitvoeren, maar alleen dan als aangetoond kon worden dat het repareren essentieel was). Bovendien maakte hij het even duidelijk dat TEX *niet* verder ontwikkeld mocht worden door iemand anders: hij wilde TEX als *zijn* schepping overlaten voor zijn nageslacht, tot in alle eeuwigheid, en niet als zijn-schepping-zoals-aangepast-door-iemand-anders.

In het algemeen werd dit goed opgevat door de TEX wereld: Knuth staat in enorm hoog aanzien bij diegenen die TEX gebruiken, en er waren er slechts heel weinigen die het idee voorstonden om zijn wensen te negeren en bereid waren veranderingen aan TEX voor te stellen. Maar er was een redelijk grote groep TEX gebruikers, waaronder de schrijver dezes, die van mening waren dat als TEX *niet* veranderde, het eenvoudig zou sterven. Niet omdat er fundamentele gebreken in TEX waren – algemeen wordt aanvaard dat dat er daarvan slechts heel weinig zijn – maar omdat de wereld niet heeft stilgezeten sinds 1978, en hoewel een script-taal toen state-of-the-art geweest mag zijn, is dat zeker nu niet meer het geval. Daarbij komt dat hoewel het aantal interne tekens was verhoogd van 128 naar 256, Knuth maar heel weinig tot niets had gedaan aan ondersteuning van asiatische talen, waarin het aantal verschillende tekens moet worden uitgedrukt in duizenden of tienduizenden. En als laatste waren er mensen die ervan overtuigd waren dat op *sommige* gebieden flinke vooruitgang kon worden geboekt (vooral vanuit het standpunt van de macro programmeur, ook bekend als de 'format writer' als het pakket macros een compleet functioneel systeem betreft) met een relatief kleine investering met betrekking tot het veranderen van TEX.

De uitvoering van deze ideeën is waarschijnlijk de speerpunt van de TEX technologie vandaag de dag: bedrijven zoals Blue Sky hebben ogenblikkelijke / toenemende TEX interpreters geproduceerd, die in staat zijn om veranderingen in de broncode van een TEX document direct te laten zien; Advent Publishing maakten 3B2, dat zowel een grafische als een tekstuele specificatie van de layout toelaat, waarbij de veranderingen in de één automatisch worden doorgevoerd in de ander; John Plaice en Yannis Haramboulos hebben een 64-bits versie van TEX ontwikkeld die intern gebruik maakt van unicode; en de groep waarmee ik het meest verbonden ben (de NTS groep, waarbij NTS staat voor 'New Typesetting System') heeft een volledig compatibele opvolger van TEX gemaakt met de naam e-TEX, die functionaliteit toevoegt zonder de compatibiliteit in gevaar te brengen (de NTS groep heeft ook het voornemen TEX te her-implementeren vanaf niets in één van de moderne snelle-typering talen zoals Prolog of CLOS. Het idee is het toestaan van snelle experimenten met alternatieve zet-algoritmes of paradigma's). Of één van deze ideeën zal aanslaan valt nog te bezien, hoewel de Apple Macintosh *aficionados* Classic Textures (het eerder genoemde produkt van Blue Sky) erg hoog aanslaan. Een fundamentele vraag is die van stabiliteit: aangezien een van de belangrijkste sterke punten van TEX zijn stabiliteit is, kan alleen de toekomst uitwijzen wat de wereld zal vinden van een systeem dat specifiek bedoeld is om te blijven reageren en evolueren, in plaats van star en versteend te zijn.

Wat misschien de moeite van het vermelden waard is, is dat al deze projecten ervoor gezorgd hebben dat Knuth's wensen gehonoreerd werden, niet alleen naar de letter maar ook naar de geest: niet één ervan streeft ernaar zichzelf TEX te noemen (Het project van John Plaice en Yannis Haramboulos noemt zich zelfs Omega, wat absoluut nooit verward kan worden met TEX), terwijl ze toch allemaal toegeven veel verschuldigd te zijn aan Knuth en TEX: zonder hen zou geen van deze andere projecten ooit het licht van de dag gezien hebben.

Parallele Ontwikkelingen

Natuurlijk stond de rest van de wereld niet stil terwijl TEX aan het evolueren was: de computerwetenschap bleef zichzelf ontwikkelen, en computer-netwerken verspreidden zich van het laboratorium naar het leger en de universiteiten en uiteindelijk naar de hele wereld. Regel-georiënteerde editors vielen in onmin en werden vervangen door volledig-scherm editors (behalve in

de wat achterlopende wereld van MS/DOS, die tot relatief kortgeleden alleen EDLIN bleef aanbieden). Script-georiënteerde markup talen zoals de ROFF familie die eerder genoemd werd werden aangevallen door steeds slimmer wordende tekstverwerkers, en WYSIWYG ('What You See Is What You Get'), GUI ('Graphical User Interface') en WIMP ('Windows, Icons, Menus and Pull-down lists') werden de orde van de dag.

Rond dezelfde tijd dat Knuth begon met het werk aan TEX, her-implementeerden John Warnock en Martin Newell, werkzaam bij Xerox PARC, een eerdere taal ('the Design System') als JaM ('John and Martin'!). Vanuit dit geclusterde begin ontwikkelden zich uiteindelijk zowel de Interpress (Xerox print protocol) als de PostScript taal. Terwijl Interpress relatief onbekend bleef, sloeg Adobe PostScript in als een bom: voor de eerste keer was er een *de facto* pagina-beschrijvingstaal, die het toestond ingewikkelde pagina's te beschrijven als een algoritme (en dus erg efficient). Hoewel Hewlett Packard's Printer Control Language (PCL) wijd ondersteund en nagemaakt bleef (en is), vestigde PostScript zichzelf als *de* standaard voor hogekwaliteit printers (waarmee ik laser-printers en beter bedoel), en al vrij snel streefden printer-producerende bedrijven ernaar om PostScript interpreters of PostScript emulators te leveren voor hun high-end produkten. Helaas (voor de schrijvers van de emulators) is PostScript een complexe taal, en veel van de eerdere emulators waren gebrekig op één of meer manieren. Adobe, zijnde de ontwikkelaars van de taal, hadden natuurlijk veel minder last van deze problemen, maar zelfs zij maakten verbeterde versies van hun interpreter beschikbaar naarmate de tijd vorderde.

Gedurende een hele tijd bleven gedeelten van PostScript een goed bewaard geheim: de mysterieuze `eexec` operator was niet gedocumenteerd, en hoewel het PostScript handboek informatie gaf over het formaat van de zogenaamde 'Type 3' fonts, bleven de net zo mysterieuze 'Type 1' fonts ongedocumenteerd. Natuurlijk is reverse engineering een goed begrepen gereedschap en eindelijk werden de grenzen doorbroken: beschrijvingen van `eexec` begonnen op te duiken in de pers en ten lange leste liet Adobe zich vermurwen en stond de volledige documentatie van zowel `eexec` als hun Type 1 fonts af.

Niet veel later vestigden Type 1 fonts zich net zo als de standaard voor fonts als PostScript dat was voor pagina-beschrijvingstalen; bedrijven zoals Corel begonnen hun eigen Type 1 fonts te produceren, die nauw aansloten bij de industriële standaardfonts, maar die net genoeg afweken (op zijn minst qua naam) om beschuldigingen van font-piraterij te kunnen ontlopen (hoewel dit laatste probleem tot aan de dag van vandaag een zorg blijft voor de top van de font ontwerpers, zoals Hermann Zapf). Alle belangrijke font-gieterijen begonnen hun fonts aan te bieden in Type 1 formaat, en velen beloofden *al* hun fonts binnen afzienbare tijd beschikbaar te hebben in Type 1 formaat. De zogenaamde 'font magic' die het vroege Adobe fonts toestond redelijke resultaten op te leveren op lage-resolutie apparaten zoals 300 dpi laserprinters werd hernoemd naar 'font hinting', en ook dit werd uiteindelijk gedocumenteerd door Adobe. Nieuwe mogelijkheden bleven toegevoegd worden aan de PostScript taal, en in 1990 kondigde Adobe een volledig nieuwe versie van de taal aan, 'PostScript Level 2'. Deze nieuwe versie bevatte alle eerdere toevoegingen aan de taal, en introduceerde tevens vele nieuwe mogelijkheden zoals de mogelijkheid om compacte (binaire) representaties van een PostScript document te gebruiken, zowel als de eerdere (ASCII) representatie; nieuwe kleur-modellen werden toegevoegd, en ondersteuning werd beschikbaar voor composiet-fonts.

PostScript was oorspronkelijk ontstaan als een interne taal voor printers, maar het werd al snel duidelijk dat een versie van PostScript die in staat was een computer-scherm te besturen extreem nuttig zou zijn. Adobe creëerde een eigen versie hiervan met de naam 'Display PostScript', maar ondertussen was L. Peter Deutsch al begonnen met het werk aan een eigen PostScript interpreter met de naam 'GhostScript', en een van de funtamentele functionele eigenschappen hiervan was dat het in staat was om het scherm te besturen van elke computer waarop het gebruikt werd (het bevatte ook een ruime verzameling stuurprogramma's voor non-PostScript printers, en pseudo-stuurprogramma's voor een paar van de meer populaire grafische uitwisselformaten). Gedurende 1995 kondigde Peter eindelijk GhostScript versie 3 aan, welke een vrijwel complete PostScript Level 2 emulatie besloeg. En hoewel de officiële Adobe interpreter een gelicenseerd (en relatief duur) produkt bleef, was en bleef GhostScript een gratis programma voor diegenen die het niet gebruiken om er geld mee te verdienen. De computerwereld is een grote dank verschuldigd aan L. Peter Deutsch, zowel voor zijn kunde in het schrijven van GhostScript als voor zijn vrijgevigheid in het gratis beschikbaar stellen van het programma, en ook is grote dank verschuldigd aan al de verschillende personen die hun privé stuurprogramma's en/of verbeteringen aan het GhostScript project hebben gedoneerd (PS-View van Bogusław Jackowski en Piotr Pianowski verdient extra vermelding).

Van het ARPAnet naar het Web

Een paar jaar voordat Knuth begon aan het werk aan TEX, was het amerikaanse leger in de personificatie van [D]ARPA (het [Defence] Advanced Research Projects Agency) een piloot-project begonnen met het doel computers met elkaar te verbinden over heel lange afstanden. Hoewel verbindingen tussen lokale computers niet ongewoon waren, waren verbindingen over duizenden kilometers toen nog onbekend, maar [D]ARPA onderkende de potentiële militaire belangrijkheid van zulke verbindingen en startte daarom een hele serie onderzoeksprojecten met als doelstelling zulke verbindingen mogelijk te maken. Deze projecten begonnen oorspronkelijk geïsoleerd van elkaar, maar zodra het test-netwerk beschikbaar was kreeg het project een voorwaartse duw – feitelijk een eigen bestaan – uit zichzelf, en de gehele ontwikkelings-strategie vanaf dat moment vond plaats door discussie *via* zowel als *over* het netwerk. Dit netwerk, dat om voor-de-hand-liggende rede-