Paul Huygen

TEXniques

Maak een logo met behulp van literate programming

abstract
In dit artikel wordt beschreven hoe de *literate programming* techniek gebruikt kan worden om complexe taken uit te voeren en te documenteren. Als voorbeeld wordt een logo ontworpen met behulp van de gereedschappen AWK, picTEX en LATEX.

1 Inleiding

Dit artikel is eigenlijk een testimonium paupertatis. Het laat zien hoe iemand die te lui is om Metapost te leren, toch een logo kan maken. Inplaats van vijf regels Metapost code moet er dan een veel ingewikkelder programma komen. Gelukkig kan zo een programma goed gedocumenteerd worden door de *literate programming* techniek te gebruiken.

1.1 Literate Programming *Literate programming* is gebaseerd op een idee van Donald Knuth [3]. Het basis idee is, om bij het programmeren niet uit te gaan van de computer en de computertaal, maar van mensen en mensentaal. In een document legt de programmeur precies uit wat de computer moet doen, en *en passant* voert zij de code er aan toe. Praktisch komt het er op neer dat de programmeur een elektronisch document creëert, waarin stukken programma en uitleg voor de lezer elkaar afwisselen. Een stuk code heet een macro, en heeft een naam (label). In een macro kan een andere macro, die op een heel andere plaats in het document staat, ingevoegd worden. Dit gebeurt door het label van de andere macro tussen de code in te voegen. Hierdoor hoeft de programmeur zich niet te houden aan de volgorde van de code, zoals die door de programmeertaal is vereist. Bovendien is het eenvoudig geworden om top-down te programmeren. Code die bedoeld is om een bepaald deelprobleem op te lossen wordt in een aparte macro gezet met een label dat duidelijk aangeeft wat de code moet doen. Om dit idee toe te passen ontwikkelde Knuth WEB. WEB bestaat uit twee programma's. Het programma WEAVE zet een elektronisch document om in TEX code voor menselijke lezers, en het programma TANGLE zet het document om in Pascal code voor de computer. De bekendste toepassing van WEB is TEX zelf. TEX is het enige programma dat ik ken, dat in boekvorm is uitgegeven [4].

Naderhand zijn er verschillende alternatieve systemen ontwikkeld voor literate programming, voornamelijk omdat gebruikers het systeem voor andere programmeertalen dan Pascal wilden toepassen. Ieder systeem heeft zijn eigen voor- en nadelen. Belangrijke eigenschappen zijn:

- Geschiktheid voor één bepaalde programmeertaal, of voor verscheidene programmeertalen.
- "pretty printing" van de code in de macro's. De keerzijde van pretty printing is meestal, dat het programma met slechts één programmeertaal kan werken.
- Geschiktheid voor één bepaald zetsysteem of tekstverwerker of onafhankelijkheid daarvan. Verreweg de meeste systemen zijn gemaakt voor TEX of een TEX dialect zoals LATEX.
- Geschiktheid om meerdere bestanden met code aan te maken. Het oorspronkelijke

WEB programma zette alle code in één Pascal bestand. Bij de taal C bijvoorbeeld moeten er doorgaans echter meerdere bestanden aangemaakt worden, bijvoorbeeld voor de compiler en voor de precompiler. Ook is het handig als bijvoorbeeld de Makefile in het elektronische document kan worden opgenomen.

- Automatisch aanmaken van indexen. Een systeem dat voor één bepaalde programmeertaal is gemaakt, kan vaak automatisch *identifiers* herkennen en deze in een index opnemen.
- Macro's met argumenten. Het is handig als je met macro's een soort functies kunt maken met argumenten.

Een beschrijving of opsomming van de verschillende systemen valt buiten het bereik van dit artikel. Veel informatie is te vinden in de Literate Programming FAQ (`http://shelob.ce.ttu.edu/daves/lpfaq.txt`)en op `http://www.desy.de/user/projects/LitProg/Start.html`.

Voor de toepassing in dit artikel is Nuweb gebruikt. Nuweb is geschreven door Preston Briggs. Het is te vinden in het CTAN archive in directory `/web/nuweb`. Nuweb heeft de volgende voordelen:

- Bijzonder eenvoudig. Alles kan gedaan worden met een handvol commando's. Door de eenvoud is de vertaling erg snel.
- Kan voor alle programmeertalen gebruikt worden en kan verschillende programma code bestanden voor verschillende programmeertalen tegelijk genereren.
- Semi-automatisch genereren van een index. De programmeur geeft éénmaal aan dat een bepaald woord een identifier is die in de index moet, waarna Nuweb zelf opzoekt in welke macro's de identifier gebruikt wordt.
- De gebruiker formatteert de bestanden met programmeercode zelf. Dit is bijvoorbeeld handig voor debuggen.

Het programma heeft ook nadelen (die voor mij dus niet doorslaggevend zijn):

- Geen *pretty printing*.
- Geen macro's met argumenten.
- Afhankelijk van één typesetter (gebruikt LaTeX voor typesetting)

Het bronbestand voor Nuweb lijkt sterk op een LaTeX bestand. Binnen het bestand kan een instructie gegeven worden om een programma code bestand te openen. In het opgemaakte document ziet dat er bijvoorbeeld zó uit:

```
"xample.pas" 1a ≡
      Progam xample;
        var ⟨variabelen voor xample 1b⟩;
        begin
          ⟨open het invoer bestand 1c⟩
          ⟨Lees het bestand en verwerk de gegevens 1d⟩
          ⟨Schrijf het resultaat op het beeldscherm 1e⟩
        end.
      ◇
```

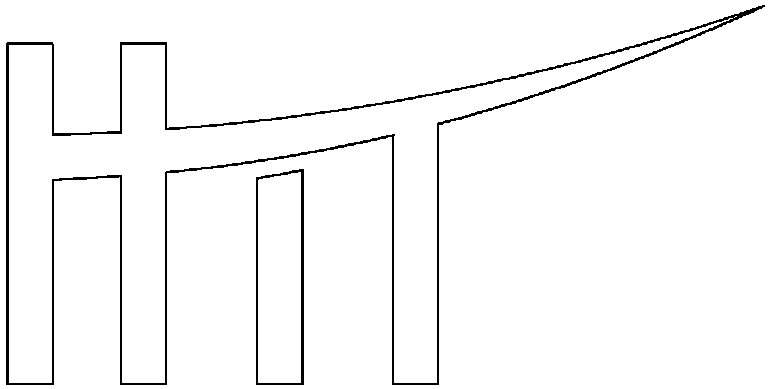
In dit codefragment wordt het bestand `xample.pas` geopend, en gevuld met programmacode. De programmacode bevat verwijzingen naar macro's die geëxpandeerd moeten worden (dat wil zeggen, dat de verwijzing vervangen moet worden door de code die in de macro staat). De macro kan op een heel andere plaats gedefinieerd worden. Dat ziet er bijvoorbeeld zo uit:

⟨variabelen voor xample 1b⟩ ≡
```
aap, noot, mies: real;
maan zaag, fien, vier:integer;
```
◇
Macro referenced in scrap 1a.

Een macro kan op verschillende plaatsen in het document "gedefinieerd" worden. De inhoud van de verschillende definities wordt gewoon aan elkaar geplakt. De macro's worden automatisch genummerd (in het voorbeeld worden de nummers 1a t/m. 1e gebruikt). Het nummer is gebaseerd op het nummer van de bladzijde waarop de macro staat (bijvoorbeeld macro 6a staat op bladzijde 6).

1.2 AWK Om het logo te maken moeten er veel berekeningen gemaakt worden. Noch LaTeX noch Nuweb kunnen goed rekenen. Daarom gebruiken we een andere computertaal om de LaTeX code te genereren. Deze computertaal is AWK [1]. AWK, genoemd naar de eerste letters van de achternamen van de ontwikkelaars ervan (Aho, Weinberger en Kernighan) is een geïnterpreteerde, C-achtige taal, die vooral geschikt is om tekst strings te bewerken.

1.3 Vorm van het logo en ontwerp beslissingen Het logo bestaat uit de letters H, I en T, die samen een baan dragen die horizontaal begint en geleidelijk opstijgt. De baan bevat de horizontale balk van de H, de punt van de I en de horizontale balk van de T. Uiteindelijk ziet het er uit als in figuur 1. Het ontwerp wordt geparametriseerd, dat



Figuur 1. Het logo zoals dat met dit document gemaakt kan worden. Dit document maakt alleen de uitlijning van het logo. Het werkelijke logo is massief geel.

wil zeggen dat van de onderdelen zo min mogelijk de dimensies zelf aangegeven zullen worden, maar zoveel mogelijk de verhoudingen van de dimensies onderling met behulp van parameters. Op die manier kan, door het veranderen van een paar parameters, het karakter van het logo veranderd worden. Deze techniek is afgekeken van Knuth [2]. Het hier beschreven progamma maakt alleen maar een uitlijning van het logo. Om vanuit deze uitlijning tot figuur 1 te komen, heb ik nog veel werk met behulp van een grafisch programma moeten verrichten.

2 Maatvoering

2.1 Breedte, hoogte en indeling van de ruimte. Definieer de breedte en hoogte van het logo in willekeurige eenheden.

⟨definieer dimensies van het logo 5a⟩ ≡
```
breedte=1000
hoogte=500
```
◇

Macro defined by scraps 5abc, 6abcd.
Macro referenced in scrap 12a.

Bij het uitlijnen van tekst wordt de breedte van de letter `X` in het gebruikte font vaak als maat gebruikt. Ik doe dit analoog, maar gebruik de eerste letter van het logo, de `H`, als maat.

⟨definieer dimensies van het logo 5b⟩ ≡
```
hwijdte=0.15*breedte
letafst=1.2*hwijdte
```
◇

Macro defined by scraps 5abc, 6abcd.
Macro referenced in scrap 12a.

De letters en de baan hebben een niet-verwaarloosbare dikte. *hdikte* is de helft van de dikte.

⟨definieer dimensies van het logo 5c⟩ ≡
```
dikte=0.06*breedte
hdikte=0.5*dikte
```
◇

Macro defined by scraps 5abc, 6abcd.
Macro referenced in scrap 12a.

Zet desgewenst een box om het plaatje ter oriëntatie. Zoals we later zullen zien moeten we daarvoor AWK instrueren om een LaTeX regel af te drukken.

⟨put alles in het picture 5d⟩ ≡
```
# printf("\\put(0,0){\\framebox(%d,%d){~}}\n", breedte, hoogte)
```
◇

Macro defined by scraps 5d, 9bcd, 10ab, 11ab.
Macro referenced in scrap 12c.

2.2 Positionering van de elementen in het logo De letter `H` staat helemaal links, dan komt de `I` en dan de `T`. De afstanden worden door *hwijdte* en *letafst* bepaald. Let erop dat de letters dikte hebben.

⟨definieer dimensies van het logo 6a⟩ ≡
```
ipos=hdikte+hwijdte+letafst
tpos=ipos+letafst
```
◇

Macro defined by scraps 5abc, 6abcd.
Macro referenced in scrap 12a.

De baan begint links op een fractie *baanbhoogfrac* van de hoogte van het logo en eindigt in de rechter bovenhoek.

⟨definieer dimensies van het logo 6b⟩ ≡
```
baanbhoogfrac=0.60
baanbhoogte=baanbhoogfrac*hoogte
```
◇

Macro defined by scraps 5abc, 6abcd.
Macro referenced in scrap 12a.

2.3 Overige maatvoering Sommige elementen worden met bogen aan elkaar verbonden. In ieder geval wordt de `T` met de baan verbonden via een boog. De grootte van de boog wordt bepaald door de volgende parameter:

⟨definieer dimensies van het logo 6c⟩ ≡
```
boogparameter=0.04+breedte
```
◇

Macro defined by scraps 5abc, 6abcd.
Macro referenced in scrap 12a.

De hoogte van de `H` en de ruimte tussen de baan en de `i` moeten worden gedefinieerd.

⟨definieer dimensies van het logo 6d⟩ ≡
```
hhoogte=0.90*hoogte
ispleet=0.05*hoogte
```
◇

Macro defined by scraps 5abc, 6abcd.
Macro referenced in scrap 12a.

3 Maak de baan

De baan wordt beschreven door een combinatie van Bézier curven en tweedegraads polynomen. De bovenkant van de baan is een segment van een parabool met minimum op (0, baanhoogte + hdikte), die loopt door de rechterbovenhoek van het logo, het punt (breedte, hoogte). De onderkant van de baan heeft een minimum op (0, baanhoogte − hdikte) en loopt ook door de rechterbovenhoek van het logo. Om de berekeningen te vereenvoudigen gaan we over op andere coördinatenstelsels. Noem het oorspronkelijke coördinatenstelsel (x, y), dan gaan we voor de bovenkant van de baan over op (f, y_h) en voor de onderkant over op (f, y_l), waarbij:

$$f = \frac{x}{\text{breedte}} \tag{1}$$

$$y_l = \frac{y - (\text{baanbhoogte} - \text{hdikte})}{\text{hoogte} - (\text{baanbhoogte} - \text{hdikte})} \tag{2}$$

$$y_h = \frac{y - (\text{baanbhoogte} + \text{hdikte})}{\text{hoogte} - (\text{baanbhoogte} + \text{hdikte})} \tag{3}$$

$$\tag{4}$$

y_l en y_h zijn gelijk aan nul aan de linkerkant van het logo en 1 aan de rechterkant. Maak AWK functies om heen en weer te gaan van het ene naar het andere coördinatenstelsel.

⟨definieer AWK functies 7⟩ ≡
```
function f(x){
    return x/breedte
 }
 function x(f){
    return f*breedte
 }
 function yh(y){
    minim=baanbhoogte+hdikte
    return (y-minim)/(hoogte-minim)
 }
 function yl(y){
    minim=baanbhoogte-hdikte
    return (y-minim)/(hoogte-minim)
 }
 function yfromh(yo){
    minim=baanbhoogte+hdikte
    return yo*(hoogte-minim)+minim
 }
 function yfroml(yo){
    minim=baanbhoogte-hdikte
    return yo*(hoogte-minim)+minim
 }
```
◇

Macro defined by scraps 7, 8ab, 9a, 12b.
Macro referenced in scrap 12a.

In de alternatieve coördinatenstelsels wordt de baan beschreven als $y = x^2$. De afgeleide functie is dan $y' = 2x$. De raaklijn door de parabool in een punt (x_i, y_i) is dan $y = 2x_i x - x_i^2$. De raaklijnen door de parabool in de punten (x_1, y_1) en (x_2, y_2) snijden elkaar in (x_s, y_s), waarbij:

$$x_s = \frac{x_1^2 - x_2^2}{2\,(x_1 - x_2)} \tag{5}$$

$$y_s = -x_1^2 + \frac{x_1\left(x_1^2 - x_2^2\right)}{x_1 - x_2} \tag{6}$$

In de LaTeX picture mode kunnen we het paraboolsegment dus beschrijven als een Bézier curve door de drie punten (x_1, x_1^2), (x_s, y_s) en (x_2, y_2). We moeten wel nog de punten terugschalen naar het oorspronkelijke coördinatenstelsel.

De volgende AWK functie tekent een stuk van de bovenkant van de baan, tussen $x =$ `xb` en $x =$ `xe`. De getransformeerde coördinaten worden met een `t` aangegeven (bijvoorbeeld `xtb` is de getransformeerde van `xb`).

⟨definieer AWK functies 8a⟩ ≡
```
function hparaboolsegment(xb, xe){
  xtb=f(xb); xte=f(xe); ytb=xtb*xtb; yte=xte*xte;
  xtm=(xtb*xtb-xte*xte)/(2*(xtb-xte));
  ytm=0-xtb*xtb+xtb*(xtb*xtb-xte*xte)/(xtb-xte);
  xm=x(xtm); ym=yfromh(ytm)
  printf("\\qbezier(%d,%d)(%d,%d)(%d,%d)\n",
         xb, yfromh(ytb), xm, ym, xe, yfromh(yte))
 }
```
◇

Macro defined by scraps 7, 8ab, 9a, 12b.
Macro referenced in scrap 12a.

Nu komt een codefragment om een stuk van de onderkant van de baan te tekenen.

⟨definieer AWK functies 8b⟩ ≡
```
function lparaboolsegment(xb, xe){
  xtb=f(xb); xte=f(xe); ytb=xtb*xtb; yte=xte*xte;
  xtm=(xtb*xtb-xte*xte)/(2*(xtb-xte));
  ytm=0-xtb*xtb+xtb*(xtb*xtb-xte*xte)/(xtb-xte);
  xm=x(xtm); ym=yfroml(ytm)
  printf("\\qbezier(%d,%d)(%d,%d)(%d,%d)\n",
          xb, yfroml(ytb), xm, ym, xe, yfroml(yte))
 }
◇
```
Macro defined by scraps 7, 8ab, 9a, 12b.
Macro referenced in scrap 12a.

De bovenkant van de letter I moet gelijkvormig zijn aan de onderkant van de baan. Daarom definieer ik hier analoge functies voor deze letter.

⟨definieer AWK functies 9a⟩ ≡
```
function yfromi(yo){
  minim=baanbhoogte-hdikte-ispleet
  return yo*(hoogte-minim)+minim
 }
 function iparaboolsegment(xb, xe){
  xtb=f(xb); xte=f(xe); ytb=xtb*xtb; yte=xte*xte;
  xtm=(xtb*xtb-xte*xte)/(2*(xtb-xte));
  ytm=0-xtb*xtb+xtb*(xtb*xtb-xte*xte)/(xtb-xte);
  xm=x(xtm); ym=yfromi(ytm)
  printf("\\qbezier(%d,%d)(%d,%d)(%d,%d)\n",
         xb, yfromi(ytb), xm, ym, xe, yfromi(yte))
 }
◇
```
Macro defined by scraps 7, 8ab, 9a, 12b.
Macro referenced in scrap 12a.

3.1 Doe het! Probeer maar eens een baan te maken. Eerst tussen de poten van de H. De linkerpoot van de H neemt *dikte* in beslag en de rechterpoot begint bij hwijdte + 0.5dikte.

⟨put alles in het picture 9b⟩ ≡
```
hparaboolsegment(dikte, hwijdte);
 lparaboolsegment(dikte, hwijdte);
◇
```
Macro defined by scraps 5d, 9bcd, 10ab, 11ab.
Macro referenced in scrap 12c.

Nu de bovenkant van de baan tussen de H en de rechter bovenhoek:

⟨put alles in het picture 9c⟩ ≡
```
hparaboolsegment(dikte+hwijdte, breedte);
◇
```
Macro defined by scraps 5d, 9bcd, 10ab, 11ab.
Macro referenced in scrap 12c.

Nu de onderkant van de baan tussen de H en de rechter bovenhoek. Er moet een gat komen waar de verticale stok van de T in komt.

⟨put alles in het picture 9d⟩ ≡
```
lparaboolsegment(dikte+hwijdte, tpos-hdikte);
  lparaboolsegment(tpos+hdikte, breedte);
```
◇

Macro defined by scraps 5d, 9bcd, 10ab, 11ab.
Macro referenced in scrap 12c.

4 De letter H

Van de letter `H` is de dwarsbalk al in 3.1 neergezet. Nu de poten nog. Er is geen scheiding tussen de poten en de dwarsbalk. De linkerpoot:

⟨put alles in het picture 10a⟩ ≡
```
printf("\\put(0,0){\\line(0,1){%d}}\n", hhoogte);
  printf("\\put(%d,0){\\line(0,1){%d}}\n", dikte, yfroml(f(dikte)*f(dikte)));
  printf("\\put(%d,%d){\\line(0,1){%d}}\n", dikte, yfromh(f(dikte)*f(dikte)),
           hhoogte-yfromh(f(dikte)*f(dikte)));
  printf("\\put(0,0){\\line(1,0){%d}}\n", dikte);
  printf("\\put(0,%d){\\line(1,0){%d}}\n", hhoogte, dikte);
```
◇

Macro defined by scraps 5d, 9bcd, 10ab, 11ab.
Macro referenced in scrap 12c.

Nu de rechter poot:

⟨put alles in het picture 10b⟩ ≡
```
printf("\\put(%d,0){\\line(0,1){%d}}\n",
         hwijdte, yfroml(f(hwijdte)*f(hwijdte)));
  printf("\\put(%d,%d){\\line(0,1){%d}}\n",
         hwijdte, yfromh(f(hwijdte)*f(hwijdte)),
           hhoogte-yfromh(f(hwijdte)*f(hwijdte)));
  xcoor=hwijdte+dikte
printf("\\put(%d,0){\\line(0,1){%d}}\n",
       xcoor, yfroml(f(xcoor)*f(xcoor)));
  printf("\\put(%d,%d){\\line(0,1){%d}}\n",
         xcoor, yfromh(f(xcoor)*f(xcoor)),
         hhoogte-yfromh(f(xcoor)*f(xcoor)));
  printf("\\put(%d,0){\\line(1,0){%d}}\n", hwijdte, dikte);
  printf("\\put(%d,%d){\\line(1,0){%d}}\n",hwijdte, hhoogte, dikte);

```
◇

Macro defined by scraps 5d, 9bcd, 10ab, 11ab.
Macro referenced in scrap 12c.

5 De letter I

De letter `I` is een verticale “balk”, echter met een schuine bovenkant, die gelijkvormig is aan de onderkant van de baan. De functies *yfromi* en *iparaboolsegment* om de coördinaten van de bovenste hoeken van de `I` te berekenen resp. de bovenrand te tekenen zij al gedefinieerd in hoofdstuk 3.

⟨put alles in het picture 11a⟩ ≡
```
xli=ipos-hdikte; xre=ipos+hdikte;
printf("\\put(%d,0){\\line(0,1){%d}}\n", xli, yfromi(f(xli)*f(xli)));
printf("\\put(%d,0){\\line(0,1){%d}}\n", xre, yfromi(f(xre)*f(xre)));
printf("\\put(%d,0){\\line(1,0){%d}}\n", xli, dikte);
iparaboolsegment(xli, xre);
```
◇

Macro defined by scraps 5d, 9bcd, 10ab, 11ab.
Macro referenced in scrap 12c.

6 De letter T

De letter T is een verticale stok die een open verbinding met de baan heeft. De baan heeft al een "gat" waar de stok van de T tegenaan moet lopen.

⟨put alles in het picture 11b⟩ ≡
```
xli=tpos-hdikte; xre=tpos+hdikte;
printf("\\put(%d,0){\\line(0,1){%d}}\n", xli, yfroml(f(xli)*f(xli)));
printf("\\put(%d,0){\\line(0,1){%d}}\n", xre, yfroml(f(xre)*f(xre)));
printf("\\put(%d,0){\\line(1,0){%d}}\n", xli, dikte);

```
◇

Macro defined by scraps 5d, 9bcd, 10ab, 11ab.
Macro referenced in scrap 12c.

7 Implementatie

Het document dat u nu leest (`logo.w`) maakt een LaTeX bestand aan, `logow.tex` waarin het logo getekend wordt. Het document maakt ook een AWK bestand aan, `logow.awk`, waarmee de picTeX instructies gemaakt worden. De picTeX instructies worden met een `\input` instructie in `logow.tex` opgenomen. Met LaTeX en een printer driver kan het logo tenslotte op papier gezet worden.

Schrijf het LaTeX script. Het begint met een standaard preambule, waarin `unitlength` wordt ingesteld. Vervolgens wordt het te maken picture in een `center` omgeving opgenomen.

"logow.tex" 11c ≡
```
\documentclass[a5paper,landscape]{artikel3}
\pagestyle{empty}
\setlength{\unitlength}{0.1mm}
\begin{document}
\begin{center}
\mbox{\input{logopic.tex}}
\end{center}
\end{document}
```
◇

Schrijf het AWK script.

"logow.awk" 12a ≡
```
BEGIN{
  ⟨definieer dimensies van het logo 5a, … ⟩
  ⟨maak het picTeX bestand 12c⟩
  }
  ⟨definieer AWK functies 7, … ⟩
```
◇

Het AWK script moet vaak hele regels LaTeX tekst schrijven. Om dit te vereenvoudigen is er de volgende AWK functie:

⟨definieer AWK functies 12b⟩ ≡

```
function pl(regel){
  printf("%s\n", regel)
}
```

◇

Macro defined by scraps 7, 8ab, 9a, 12b.
Macro referenced in scrap 12a.

7.1 De structuur van het LaTeX bestand Een LaTeX bestand heeft een vaste structuur. Maak in de "body" van het document een gecentreerd "picture" met het logo.

⟨maak het picTeX bestand 12c⟩ ≡

⟨open het picture 12d⟩
⟨put alles in het picture 5d, … ⟩
⟨sluit het picture 12e⟩

◇

Macro referenced in scrap 12a.

Maak een gecentreerd picture van breedte *breedte* en hoogte *hoogte* eenheden.

⟨open het picture 12d⟩ ≡

```
printf("\\begin{picture}(%d,%d)\n", breedte, hoogte)
```

◇

Macro referenced in scrap 12c.

⟨sluit het picture 12e⟩ ≡

```
pl("\\end{picture}")
```

◇

Macro referenced in scrap 12c.

7.2 Zet alles in beweging Het voordeel van de Nuweb aanpak is, dat ook het script om alles in beweging te zetten in het document gevoegd kan worden. Hier komt een eenvoudige Makefile.

"Makefile" 13a ≡
⟨suffix regels 13b, … ⟩
⟨specifieke regels 14a, … ⟩
⟨regel om dit document af te drukken 14d⟩
⟨regel om het logo te bekijken 15b⟩
⟨regel om het document te previewen 15a⟩
◇

Suffix regels binnen een Makefile geven aan hoe bestanden met een bepaald achtervoegsel gemaakt moeten worden. Bijvoorbeeld: Als je een bestand met achtervoegsel `.dvi` nodig hebt, en je hebt een bestand met achtervoegsel `.tex`, dan moet je (in ons geval) LaTeX draaien met het `.tex` bestand als invoer. De suffix regel die dit recept opgeeft aan het make programma gaat als volgt:

⟨suffix regels 13b⟩ ≡
```
.tex.dvi:
        latex $*.tex
```
◇
Macro defined by scraps 13bc.
Macro referenced in scrap 13a.

Zo kunnen we ook suffix regels maken om een `.tex` bestand uit een `.w` bestand te maken (door Nuweb te draaien)

⟨suffix regels 13c⟩ ≡
```
.w.tex:
        nuweb $*.w
```
◇
Macro defined by scraps 13bc.
Macro referenced in scrap 13a.

Het LaTeX bestand met de picTeX code voor het logo, `logopic.tex` moet worden aangemaakt met behulp van het AWK script `logow.awk` dat op zijn beurt weer uit `logo.w` ontstaat.

⟨specifieke regels 14a⟩ ≡
```
logow.awk: logo.w
        nuweb logo

logopic.tex: logow.awk
        gawk -f logow.awk >logopic.tex
```
◇
Macro defined by scraps 14abc.
Macro referenced in scrap 13a.

Om het logo te kunnen afdrukken of met een previewer te bekijken, moeten we een `.dvi` bestand hebben.

⟨specifieke regels 14b⟩ ≡
```
logow.tex: logo.w
        nuweb logo

logow.dvi:logow.tex logopic.tex
        latex logow
```
◇
Macro defined by scraps 14abc.
Macro referenced in scrap 13a.

Maak de `.dvi` versie van dit `.w` document. LaTeX wordt verscheidene malen herhaald om er zeker van te zijn dat alle referenties goed zijn.

⟨specifieke regels 14c⟩ ≡

```
logo.dvi: logo.tex logopic.tex
        latex logo
        bibtex logo
        latex logo
        nuweb logo
        latex logo
```

◇

Macro defined by scraps 14abc.
Macro referenced in scrap 13a.

Druk het Nuweb document af. Hier gebeurt dit met dvips en het Unix programma `lpr`. Voor andere operating systems moet deze regel waarschijnlijk worden aangepast.

⟨regel om dit document af te drukken 14d⟩ ≡

```
printnu: logo.dvi
        dvips -f logo.dvi |lpr
```

◇

Macro referenced in scrap 13a.

⟨regel om het document te previewen 15a⟩ ≡

```
viewnu: logo.dvi
        xdvi logo.dvi
```

◇

Macro referenced in scrap 13a.

Bekijk het logo op het beeldscherm. Ook deze regel moet waarschijnlijk aangepast worden op andere computers.

⟨regel om het logo te bekijken 15b⟩ ≡

```
view: logow.dvi
        xdvi logow.dvi
```

◇

Macro referenced in scrap 13a.

8 Conclusies

Het beginsel van parametriseren zoals dat door Knuth is beschreven, wordt hier nog maar beperkt toegepast. De parameters in deze toepassing beschrijven alleen de verhoudingen tussen lengtes en breedtes. Je zou ook parameters kunnen verzinnen als *rondheid* waarmee je regelt of en hoe de verschillende onderdelen met rondingen aan elkaar veronden zijn inplaats van rechte hoeken.

De *literate programming* techniek is geschikt om complexe programmeertaken, waarbij meerdere programmeertalen door elkaar gebruikt worden, uit te voeren. Het "programma" bestaat uit één ordelijk document dat goed te raadplegen en aan anderen over te dragen is, terwijl toch de sterke eigenschappen van de verschillende programmeertalen gebruikt kunnen worden. In dit voorbeeld zijn de goede grafische eigenschappen van picTEX gecombineerd met de goede rekenvaardigheid van AWK.

Tenslotte laat dit document zien dat eigenwijsheid lonend is. Door geen Metapost te leren heb ik een eenvoudig probleem ingewikkeld kunnen maken en daar publiciteit mee gegenereerd.

Referenties

1. Alfred V. Aho, Brian W. Kernighan, and Peter J Weinberger. *The AWK Programming Language*. Addison-Wesley Publishing Company, 1988.
2. Donald Knuth. The concept of a meta-font. *Visible Language*, 16(1):3–27, 1982.
3. Donald E. Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, May 1984.
4. Donald E. Knuth. *TEX: The Program*. Computers & Typesetting. Addison-Wesley, 1986.

9 Index