(Al)DraTEX*

A package for doing (portable) graphics in TEX

Jos Winnink

jos.winnink@cpb.nl

1 Introduction

Aren't there enough graphic systems that you can use in combination with (LA)TEX? You may ask. It depends on what you need and prefer. Some systems use picture formats (e.g. different type of bitmap formats) that are generated outside of TEX[1]. This mechanism is inherent not portable[2]. The same argument can be given to the use of PostScript. That leaves two ways to implement graphics in TEX in a portable way.

One solution is to implement the graphic system in TEX (like PICTEX[3], and (Al)DraTEX). The other route that can be taken is to use METAFONT in combination with TEX. In this article I will concentrate on the (Al)DraTEX-package. This package is a representative of the TEX approach.

Because (Al)DraTEX only depends on TEX, it can be used on any system on which you want to process your TEX documents. It is possible to use (Al)DraTEX in a non portable fashion. Gurari describes in his book how to use PostScript, but that is not the subject of this article.

In the rest of this article I will give a rough overview of the system and present some examples with the corresponding the source code. The examples are taken from Gurari's book.

2 Structure of (Al)DraTEX

Gurari used a two level design when creating (Al)DraTEX. There is a lower level which consists off so called *low-level drawing facilities*. The file `dratex.sty` implements this level.

The file `aldratex.sty` implements the high-level facilities, which are modular designed. This means that it is possible to only load the module(s) you need.

The (Al)DraTEX package can be used without modifications with *plain* TEX, LATEX2ε and LATEX.

2.1 The low-level facilities (DraTEX)

Of course there are the basic drawing facilities:

- lines
- moving from one location to another
- marked locations
- rectangles
- circles and ovals
- bézier curves

Then there are *add-on* features like:

- painting
- clipping
- putting text into pictures
- dimensions
- backtracking features
- the placement of figures

DraTEX offers different coordinate systems:

- two dimensional
- three dimensional
- polar coordinates

The axes can be rotated during the drawing process.

Furthermore DraTEX has knowledge of intersection points of lines and circles. You can do repetitons on simple variables (Do-loops), on paths and with data. It is possible and usefull to define your own objects and commands. It speaks for itself that doing calculations is also possible.

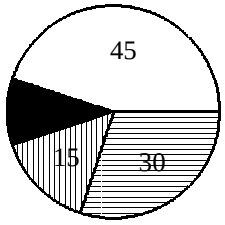
2.2 The high-level facilities (AlDraTEX)

Based on the low-level facilities offered by DraTEX, charts and diagrams are implemented in AlDraTEX.

Chart types

The following types of charts are offered.

- pie charts



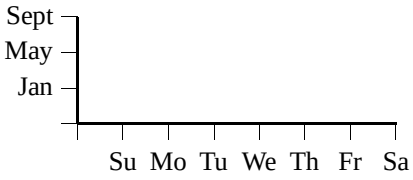
*Eitan M. Gurari, *TEX & LATEX, Drawing & Literate Programming*, McGraw Hill, 1994.

[1] TEX means any system like LATEX, LATEX2ε and *plain*-TEX.

[2] Portable to me means in this case only depending on *standard* TEX tools.

[3] Michael J. Wichura, *The PICTEX manual*.

- xy charts



- bar charts

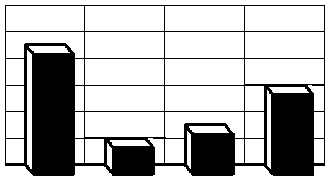
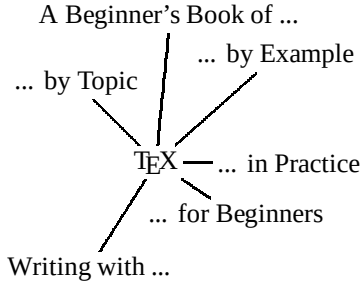


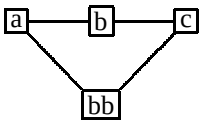
Diagram types

The following diagram types are implemented.

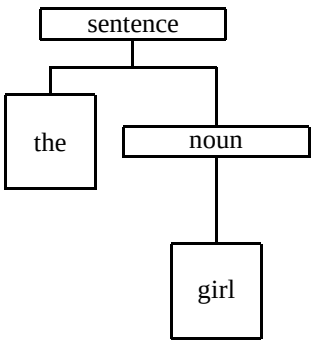
- spead diagrams, like e.g.



- grid diagrams



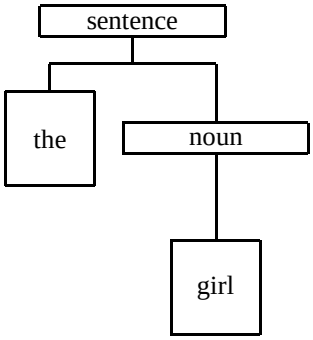
- tree diagrams



3 Examples made with (Al)DraTEX

Example 1

This picture

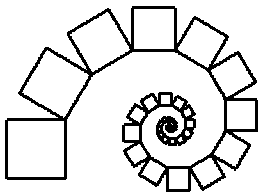


was produced by the following code

```
\Draw
\TreeSpec(\NodeA)(\NodeB)()
\SaveDrawSize(a){\RectNode(x)(--XXXXXXXXX--)}
\SaveDrawSize(b){\RectNode(x)(--XXXX~~X~~X--)}
\Define\NodeA{\MinNodeSize(a) \RectNode}
\Define\NodeB{\MinNodeSize(b) \RectNode}
\Tree()(2,sentence//
        0,the & 1,noun//
        0,girl// )
\EndDraw
```

Example 2

Making art like:



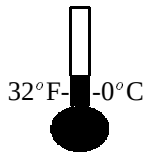
is done by executing:

```
\Draw
\RotateTo(90)
\Do(1,50) {
        \Do(1,4){
                \LineF(22)
                \Rotate(90)
        }
        \MoveF(22)\Rotate(-30)
        \Scale(0.9,0.9)
}
\EndDraw
```

Example 3

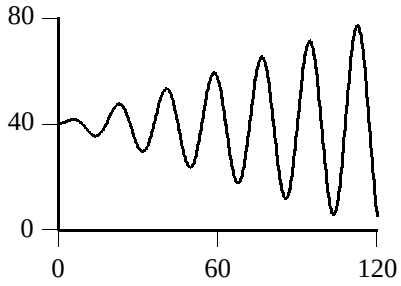
To illustrate your own weather forecast column in the newspaper you could use the following program fragment.

```
\Draw(0.05in,1mm)
\DrawRectAt(-1,7,1,16)
\PaintRectAt(-1,3,1,7)
\PaintCircle(3) \Move(-0.04,5)
\Text(--32$^o$F- - -0$^o$C --)
\EndDraw
```



Example 4

It is even possible to compute mathematical functions like the *sine*. Do some computation with them and plot the results.

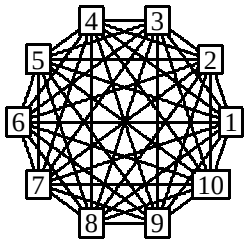


Is created bij the following piece of code

```
\Draw
\MarkLoc(a) \MoveTo(120,80) \MarkLoc(b)
\Axis(a,b)(S-1,0&60&120)
\Axis(a,b)(W-1,0&40&80)
\Define\Sin(1){
        \SaveUnits
        \Units(1pt,1pt) \MarkLoc(t)
        \MoveTo(0,0)    \MarkLoc(o)
        \RotateTo(#1)   \MoveF(1)
        \MarkLoc(p)             \CSeg\Y(o,p)
        \MoveToLoc(t)   \RecallUnits
}
\Define\Y(2){\Q=#2;}
\MoveTo(0,40)
\Do(0,120){
        \I=\DoReg; \I*20;
        \Sin(\Val\I) \Q*\DoReg;
        \Q/3; \Q+40;
        \LineTo(\DoReg,\Val\Q)
}
\EndDraw
```

Example 5

Drawing a fully connected network with 10 nodes

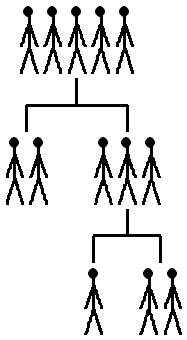


needs the following instructions:

```
\Draw
\Do(1,10){
        \MoveF(40)
        \RectNode(\DoReg)(--\DoReg--)
        \MoveTo(0,0)
        \Rotate(36)
}
\Do(1,9){
        \I=\DoReg; \K=\I; \K+1;
        \Do(\Val\K,10){
                \Edge(\Val\I,\DoReg)
        }
}
\EndDraw
```

Example 6

An example tree



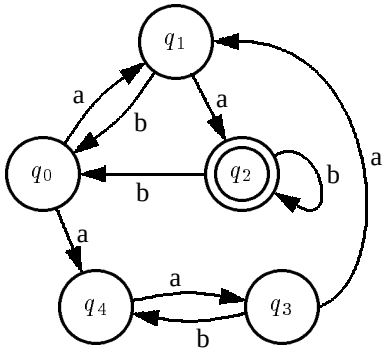
is produced by

```
\Draw
\PictNode(1){
   \Do(1,#1) {
      {\Line(3,10) {\Line(3,-10)}
        {\Move(0,9){\Line(-3,-9)} {\Line(3,-9)}}
        \Line(0,13)
        \Text(--$\bullet$--)
      }
      \Move(9,0)
   }
}
\Tree()( 2,5 //
   0,2 & 2,3 //
   0,1 & 0,2 //
)
\EndDraw
```

Example 7

A transition diagram describing a *finite state automata* is quite easy generated using the following program.

```
\Draw
\NewCIRCNode(\StateNode,106,)
\NewCIRCNode(\AStateNode,106,103)
\Define\StateAt(3)%
  {\MoveTo(#2,#3) \StateNode(#1)(--$q_{#1}$--)}
\Define\AStateAt(3)%
  {\MoveTo(#2,#3) \AStateNode(#1)(--$q_{#1}$--)}
\DiagramSpec(\StateAt & \AStateAt & \TransEdge)
\ArrowHeads(1)
\Diagram (0,0,0 & 1,50,50 & 3,90,-50 & 4,20,-
50)%
  (2,75,0)%
  (0,1,a,b & 1,2,a, & 0,2, ,b & 2,2,0,b & 0,4,a,
                                    & 4,3,a,b)
\CurvedEdgeAt(3,1,0,1,1,0)(20,0.3,0,0.5)
\EdgeLabel[+](--a--)
\EndDraw
```



4 Conclusions

In my opinion Gurari succeeded in implementing a powerfull graphics facility for TEX with TEX. The drawback of his approach is the time it takes to process the graphics. An advantage is that you get a system that has a one-pass approach and can be used on any platform for which there exists a TEX implementation.

The system itself is much more versatile than e.g. the *picture* environment that comes with LATEX2ε.

I myself experienced a problem with the software that was included in the book. This was version 1.0 that gave some problems with LATEX. On the *FTP-site*[4] mentioned in the book I found version 1.1 and this version has not shown any problems up to now.

I like to suggest to the people involved in the LATEX3 project to look at this package before implemting a graphics subsystem.

[4] ftp-address: `ftp.cis.ohio-state.edu` in `/pub/tex/osu/gurari`.