Paradigms: The winds and halfwinds—Details matter

Kees van der Laan

1 BLUe's Design V

Hi folks. How to draw lines at 45° in TEX is exercised. This simple-looking and innocent problem touches on fundamental issues, such as should we adhere to the Cartesian picture environment approach,[1] or do we need something else now and then?

This work emerged from studying Gurari's approach to graphics incented by a suggestion in Jos Winnink's review, MAPS 94.2.

The discussion below explains what has been used in BLUe's format system in the turtle graphics[2] macros, especially in the coding of `\N`, `\E`, `\S`, `\W`, `\NE`, `\SE`, `\SW`, `\NW`, next to `\ESE`, and `\WSW`. Examples are included which show what can be attained by these basic functionalities. It is also shown how the obtained pictures can be stored in the picture database `pic.dat` for reuse with different scaling or positioning.

In the Appendix some tree diversions have been given.

2 What is the problem?

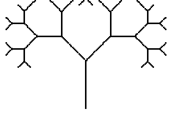
TEX's `\hrule` and `\vrule` primitives are a gem and very powerful. It would be nice to have similar primitives for any direction. In absence of these we can use line pieces provided in fonts. However, the latter approach suffers from the following drawbacks

- for a few discrete directions only
- line lengths are discrete too
- line thickness is inflexible.

3 Why?

The need for general and flexible line elements arose when I faced the problem how to draw classical fractals.[3] The very least is the possibility to draw lines at 45°—the halfwinds.

Example *(Pythagorean tree)*



How to do this in TEX[4]?

Through the picture environment? Too clumsy, and cumbersome when changing the order, for example.

Through POSTSCRIPT? This is possible, especially it provides for lines at all orientations.[5]

Through METAFONT? Definitely, see my METAFONT anthology.

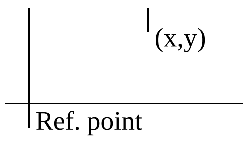
However, why not go for how far we can get just by TEX?

And what about the relevancy? I'm very pleased by the spin-off how to typeset binary trees or charts, even rotated without the use of POSTSCRIPT. See the Appendix or the Publishing with TEX user's guide.

What we need is not the Cartesian picture environment approach, but the good old penplotter TEXniques, better known in the pedagogical world as 'Turtle graphics.'

4 Turtle graphics

The basic idea is that the position of the turtle is maintained in the dimen variables `\x` and `\y`, with the reference point left invariant. The accompanying figure shows the effect of `\N1`: draw the line (`\x`, `\y`)–(`\x`, `\y`+1), i.e. the turtle moves up. After completion `\y` has been increased by `\unitlength`.



The movements—our first steps in the turtle graphics world—can be achieved by the control sequences

- `\N`, `\E`, `\S`, `\W` mean draw north, east, south and west, similar for `\NE`, `\SE`, `\SW`, and `\NW`
- `\whiteN`, `\whiteE`, `\whiteS`, `\whiteW` mean *white*-draw north, east, south and west, i.e., the turtle just moves.[6]

To get the flavour another classical picture, which shows that we don't have to worry about coordinates.

[1]The LATEX picture environment for example.

[2]Knuth already used the 'Turtle' idea in his dragon figures. For those interested in turtle graphics consult for example S. Papert: Mindstorms.

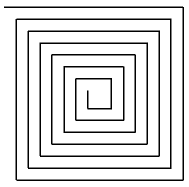
[3]Inspired by Gurari's work on TEX and graphics.

[4]For the coding of `\pythtree` see later. Or Macro writing—with confidence.

[5]J V Romanovsky has provided a concise POSTSCRIPT program.

[6]The halfwinds can be composed from the winds in this case.

Example *(Spiral)*



The spiral is marked up essentially as follows [7]

```
\unitlength.5ex \k30
\loop\E{\the\k}\advance\k-1
     \S{\the\k}\advance\k-1
     \W{\the\k}\advance\k-1
     \N{\the\k}\advance\k-1
\ifnum\k>4 \repeat
```

5 Basics

Essential for the understanding is familiarity with TEX's *boxes of size zero*, to know the effect of `\kern`-s and `\h/vss`-s inside, and of combinations of these boxes. Before reading on peruse the following.

5.1 Kerning indispensible

Kerns and stretchorshrinks within boxes of size zero are at the heart of graphics with TEX alone.

Example *(With and without boxes of size zero)*

```
\newdimen\x \x=4ex \newdimen\y \y=2ex
.\hbox to 0pt{\kern\x a\hss}.
\kern10ex and \kern10ex
.\kern\x a.
```

and

```
.\hbox to0pt{\kern\x\vbox to0pt
   {\vss\hbox{a}\kern\y}\hss}.
\kern10ex and \kern10ex
.\kern\x\raise\y\hbox{a}.
```

with result

.. a *and* . *a*.

and

a a

.. *and* . .

By this mechanism we can move to any point on the page and put there what we wish. The essential issue of the box of zero width is that the *reference point is left invariant.* It is the same before and after.

5.2 Putting it together

In vertical mode the `\hbox`-s are aligned on the reference point, and when the heights and depths are zero the `\hbox`-s overprint, and the order of specification is immaterial.

In (restricted) horizontal mode `\hbox`-s of width zero overprint and can be given in any order, The TEXbook 389.

In math mode the zero size boxes overprint. In display math the invariant reference point is centered horizontally.

Coding in TEX simple? Forget it. So unusual, but …consequent, utmost consequent.

5.3 Coding flashbacks

The codes which follow are the fifth or so version. The zeroeth version was there to please me, it made me excited. Recursive programming worked as I expected. Then I had a version which served to try out as many examples as I could think of. During the process of collecting examples I strove after consistency of the codes. Why did I use integer arithmetic anyway? I had no good reason and I decided to use dimensions and do the arithmetic through the dimensions, and widened en-passant the allowed arguments of the winds and halfwinds to expand to 'factor'-s.

And what about the context?

It seemed necessary to abstract from the multitude of boxes and to come up with an enveloping hbox of width zero.

Finally, details were taken care of, such as how do the macros behave in the various contexts. Do the winds and halfwinds behave the same? It turned out that they did not. So I had to go back from where I started. I considered the early versions as prototypes and started anew guided by specs, with simplicity and robustness as yardsticks, stating precisely how the elements should behave. By all means no sinecure, but IMHO with all respect, a must in software engineering.[8] Indeed, a superficially, simple problem.

Is that all? Essentially yes, but …details matter.

6 The winds and halfwinds

The intuitive idea is to compose lines out of elements, from what I call atoms. I used squares, casu quo rectangles, as atoms, and stacked these as follows.[9]

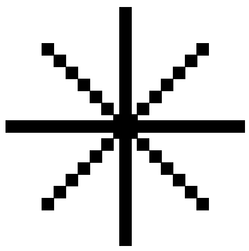
Essential is how lines leave a mathematical point. The accompanying model picture is obtained through the following macros.

```
\let\0\N \let\1\NE \let\2\E \let\3\SE
\let\4\S \let\5\SW \let\6\W \let\7\NW
\linethickness1ex
\setbox\hlfwndelm=\hbox{\vrule
   width\the\linethickness
   height\the\linethickness depth0pt}
\unitlength10ex
\def\draw{\csname\the\dir\endcsname1}
$$\loop\ifnum\dir<8{\draw}\advance\dir1\repeat$$
```

[7]For the placement within context details consult `\spiralpic` in `pic.dat`.

[8]In fact I had this article nearly finished when I realized that the vertical rules induce a vbox of the size of the rules, while other directions when boxed yielded a box of size zero. Not nice and confusing, although for the article and in my hands it was ok. Nevertheless, I decided to go for a uniform approach. When boxed the dimensions of the box should be zero.

[9]In order to make it visible `\linethickness` has been set to 1ex. Experiments with bullets and LATEX's line fonts did not yield the pleasing exact results.



Remarks. `\N` etc. draw in the implied directions with length as prescribed by the argument times `\unitlength`.

7 Pondering aloud

Can we attain compatibility with TEX's rule primitives? I don't think so, alas. On the other hand we have to reconsider the leaving of the rules from a point.

7.1 Thickness

What is meant by thickness if we stack instead of tile? What is the perceived blackness?

I used that stacking with square elements of size `\linethickness`×`\linethickness`—like in the example figure—yields the same blackness as a rule of thickness `\linethickness`.[10]

7.2 Size

Usually the size along the x-axis must be provided. I prefer to have the real size specified independently from the orientation of the line. However, the resulting size is not necessarily #1×`\unitlength`.[11] In general the result differs at most by half the atom size, because it is composed of a multiple of the atomic element. We have to correct by $\sqrt{2}$ to compensate for the direction, when we pace along one of the axes. In the example the required length is 10ex, with as result the stacking of 7 atoms of size 1ex.

For large lines one could think of combining LATEX's line10 line elements with the smaller elements. I refrained from that for two reasons. Firstly, the LATEX's NW line element—the `\char'145`—did not fit exactly in the box, and secondly, the inflexibility of the thickness of the font elements.

8 Design specs

With awareness of the above I specified the following for the microscopic level—the wind and halfwind commands proper—and for the macroscopic level—the placement within context.

8.1 Microscopics

The functionality is to draw a line of the specified length in the direction as implied by the control sequence name. The general specifications read as follows.

- as argument a 'factor' is expected, to yield the required length #1×`\unitlength`[12]
- parameterization is by `\linethickness`
- after drawing, the position of the turtle is at the end of each line, the reference point has been left invariant, and all the boxes have zero width, height and depth.

Extra for the four halfwinds the following.

- parameterization is by `\hlfwndelm` and `\linethickness` in there
- draw a line of ≈ the specified size
- the atoms are stacked diagonally, at the corners.

8.2 Macroscopics

The placement within context is the concern of the user. Because of the zero dimensions of the boxes it is a nuisance to skip or kern when using a picture to create the open space, the niche for the picture. Moreover, when the picture does not take dimensions we are in trouble at page breaks. Therefore assistance is badly needed. The picture environment idea combined with databases come to rescue. The *use* of prefab pictures has been simplified in this way, while there is flexibility via `\thispicture` to override the defaults.

Pictures can be stored in BLUe's format pic.dat database. Within each database element the default placement within context can be povided for. Through the use of `\everypicture` and/or `\thispicture` the defaults can be overriden. This approach complies with the general principles adopted in BLUe's format system.

9 Coding the winds and halfwinds

In the fourth version, where emphasis was put on that all boxes have zero dimensions, I also decided to separate to get at the (x, y) position from what is put there. This is much in the spirit of the second `\point` macro of The TEXbook 389, and adheres to the *separation of concerns* adage.[13]

The dimension variables `\x` and `\y` have after completion the values of the coordinates of the end of the line.

```
\newbox\hlfwndelm
\newdimen\linethickness \linethickness1ex
\newdimen\auxdim %linesize
%
\def\xy#1{%Function: place #1 at x, y
   \vbox to0pt{\kern-\y
   \hbox to0pt{\kern\x#1\hss}\vss}}
%
\def\xytxt#1{%Function: place text #1 at x, y
   \xy{\vbox to0pt{\vss
   \hbox to0pt{\strut#1\hss}\kern0pt}}}
%
\def\N#1{\xy{\kern-.5\linethickness
  \vbox to0pt{\vss
  \hrule height#1\unitlength
   width\linethickness}}%
\advance\y#1\unitlength}
```

[10] I also supplied macros—tiled variants—for `\NW` and `\NE`.

[11] To put it in another way: the required length must be a multiple of the atom size.

[12] The idea is that not only integer values can be specified but also decimal fractions.

[13] IMHO it makes the code more trustworthy and circumvents pitfalls. Especially, the mixing up of the kerns needed to get at (x, y) with the kerns to position what has to be put at (x, y).

```
%
\def\S#1{\advance\y-#1\unitlength{\N{#1}}}
%
\def\E#1{\xy{\vbox to0pt{\vss
   \hrule width#1\unitlength
          height\linethickness
          depth0pt\vss
}}\advance\x#1\unitlength}
%
\def\W#1{\advance\x-#1\unitlength{\E{#1}}}
%
\def\NE#1{\auxdim#1\unitlength\correction
\loop\advance\auxdim-\wd\hlfwndelm
\ifdim\auxdim>-.5\wd\hlfwndelm
   \xy{\vbox to0pt{\vss\copy\hlfwndelm}}%
   \advance\x\wd\hlfwndelm
   \advance\y\ht\hlfwndelm
\repeat}
%
\def\NW#1{\auxdim#1\unitlength\correction
\loop\advance\auxdim-\wd\hlfwndelm
\ifdim\auxdim>-.5\wd\hlfwndelm
   \advance\x-\wd\hlfwndelm
   \xy{\vbox to0pt{\vss\copy\hlfwndelm}}%
   \advance\y\ht\hlfwndelm
\repeat}
%
\def\SW#1{\auxdim#1\unitlength\correction
\loop\advance\auxdim-\wd\hlfwndelm
\ifdim\auxdim>-.5\wd\hlfwndelm
   \advance\x-\wd\hlfwndelm
   \advance\y-\ht\hlfwndelm
   \xy{\vbox to0pt{\vss\copy\hlfwndelm}}%
\repeat}
%
\def\SE#1{\auxdim#1\unitlength\correction
\loop\advance\auxdim-\wd\hlfwndelm
\ifdim\auxdim>-.5\wd\hlfwndelm
   \advance\y-\ht\hlfwndelm
   \xy{\vbox to0pt{\vss\copy\hlfwndelm}}%
   \advance\x\wd\hlfwndelm
\repeat}
%For \NE and \NW the titled variants read
\def\NE#1{\auxdim#1\unitlength\correction
\loop\advance\auxdim-.5\wd\hlfwndelm
\ifdim\auxdim>-.25\wd\hlfwndelm
   \xy{\vbox to0pt{\vss\copy\hlfwndelm}}%
   \advance\x.5\wd\hlfwndelm
   \advance\y.5\ht\hlfwndelm
\repeat}
%
\def\NW#1{\auxdim#1\unitlength\correction
\loop\advance\auxdim-.5\wd\hlfwndelm
\ifdim\auxdim>-.25\wd\hlfwndelm
   \xy{\vbox to0pt{\vss
       \llap{\copy\hlfwndelm}}}%
   \advance\x-.5\wd\hlfwndelm
   \advance\y.5\ht\hlfwndelm
\repeat}
```

10 Coding a database element

As promised in the Macroscopics section the visibility of the picture—that is the size—has to be added when the picture is stored in the database for (re)use. In order to do so I added two extra layers. The structure of a pic.dat database element, and a picture environment on top of for example gkp's environment to allow user flexibility.

If we assume that the picture proper has been created within gkp's

```
\def\gkpbeginpicture(#1,#2)(#3,#4)%
%#1, #2 xsize, ysize (dimensionless)
%#3, #4 xshift, yshift (dimensionless)
{...}
\def\gkpendpicture{...}
```

then the flexibility layer reads

```
\def\beginpicture{\bgroup
   \the\everypicture\the\thispicture
   \gkpbeginpicture(\the\xdim,\the\ydim)%
   (\the\xoffset,\the\yoffset)%
   <pictureproper>}
\def\endpicture{...}
```

A `pic.dat` database entry consists of a triple: `\lst` list element tag, `\<name>pic` the name, and a group. The group contains the defaults preceding the picture proper. The picture proper is enclosed by `\beginpicture` and `\endpicture` which also deliver the box of the size as prescribed by the parameters. Extra `\bgroup` and `\egroup` are there to keep things local.

```
\lst\<name>pic{\bgroup<defaults>
\beginpicture
<pictureproper>
\endpicture\egroup}
```

Example *(Coding the database element bintreepic)*

The `\bintreepic` element of the `pic.dat` database reads as follows.[14]

```
\lst\bintreepic{\bgroup
   \unitlength.5ex\kk32
   \xoffset{-32} \yoffset{-2}%
   \xdim{66}\ydim{5}%
   \def\eertnib##1\bintree{\fi}
\beginpicture\bintree\endpicture\egroup}
%with in the kernel blue.tex
\def\bintree{\S1\ifnum\kk=2 \eertnib\fi
   \divide\kk2 {\W{\the\kk}\bintree}%
   \E{\the\kk}\bintree}
%and accounting for the leaves
\def\eertnib#1\bintree{\fi\global\advance\k1
   \whiteS1\xytxt{ \csname\the\k\endcsname}}
```

11 Coding recursion

Long I believed that tail recursion—as used for example in the `\loop` ...`\repeat`—is not sufficient. We also have to cope with 'Ackermann-like splittings' of the recursion. For the case of the Pythagorean tree for example, this means that at each recursion level we have to take care of the left branch and the right branch, i.e. two invocations must be done. This interferes with how to handle the termination of the recursion. A problem? TeX too difficult? Hang on there are at least two elegant and general solutions.

11.1 Recursion termination

Example *(n!)*

As pedagogical example the calculation of n faculty.

[14]Note that there is flexibility here. `\bintreepic` consists of an invoke of `\bintree` preceded scaling and positioning parameters, assigned with default values. The flexibility comes down to the possibility to override the defaults via the use of `\thispicture`{...}.

```
\n=...  %counter var
\nfac=1 %default
\def\fac{\ifnum\n=1 \caf\fi
   \multiply\nfac\n \advance\n-1 \fac}
\def\caf#1\fac{\fi}
```

The termination is obtained because `\fac` is used as end terminator in the definition of `\caf`. `\fac` is implicitly gobbled, and the process backtracks, resulting finally in termination.

Example *(Pythagorean tree)*

Without explanation the Pythagorean tree recursive macro is given below.[15] At each level we have two invocations—the branching—to take care of the left and right branch.

```
\def\pythtree{\ifnum\level=1 \eerthtyp\fi
    \advance\level-1
    \multiply\kk23\divide\kk32%\sqrt2
    {\leftbranch\draw\pythtree}%
     \rightbranch\draw\pythtree}
\def\eerthtyp#1\pythtree{\fi}
```

No build up of `\fi`-s, no use of `\expandafter` nor `\let`. The `\let` mechanism has been used throughout in The TEXbook.

11.2 A la Knuth

In the coding of `\squine`—a quadratic spline—Knuth showed how to terminate (tail) recursions in TEX. For `\fac` this results in the following.

```
\def\fac{\ifnum\n>1 \expandafter\dofac\fi}
\def\dofac{\multiply\nfac\n
   \advance\n-1 \fac}
```

For the Pythagorean tree the result reads as follows.

```
\def\pythtree{\ifnum\level>1
   \expandafter\dopyth\fi}
\def\dopyth{\advance\level-1
    \multiply\kk23\divide\kk32
    {\leftbranch\draw\pythtree}%
     \rightbranch\draw\pythtree}
```

Earlier I noticed the elegance of introducing an extra level, especially to circumvent too many `\expandafter`-s in a row.

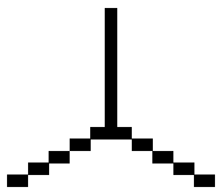
As an aside the use reads

```
\let\0\N \let\1\NE \let\2\E \let\3\SE
\let\4\S \let\5\SW \let\6\W \let\7\NW
\def\leftbranch{\advance\dir7
 \ifnum\dir>7 \advance\dir-8 \fi}%mod 8
\def\rightbranch{\advance\dir1
 \ifnum\dir>7 \advance\dir-8 \fi}%mod 8
\def\draw{\csname\the\dir\endcsname{\the\kk}}
\def\pythtree{\ifnum\level>1
   \expandafter\dopyth\fi}
\def\dopyth{\advance\level-1
    \multiply\kk23\divide\kk32
    {\leftbranch\draw\pythtree}%
     \rightbranch\draw\pythtree}
$$\unitlength0.1pt\kk128 %Size
                  \level5%Order
  \N{\the\kk}            %Trunk
  \pythtree$$
\tracingstats1
\bye
```

12 Trinaries

For 45° lines I used square elements. Why not use a rectangular element for 30° lines in conformance to the direction?

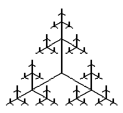
Example *(Lines at 30°)*



This model is obtained by

```
$$\x0pt\y0pt
  {\N{10}}{\ESE{10}}{\WSW{10}}$$
%with initializations
\linethickness1ex
\setbox\trielm=\hbox{\vrule
   width1.74\linethickness
   height\linethickness\relax}
%To account for element in 30 degrees direction
\unitlengthy\ht\trielm %default.2pt
\unitlengthx\wd\trielm %default.3482pt
\unitlength\unitlengthy%default.2pt
%and the macros
\def\WSW#1{\auxdim#1\unitlength\divide\auxdim2
   \loop\advance\auxdim-\unitlengthy
   \ifdim\auxdim>-.5\unitlengthy
      \advance\x-\unitlengthx
      \advance\y-\unitlengthy
      \xy{\vbox to0pt{\vss\copy\trielm}}%
   \repeat}
%
\def\ESE#1{\auxdim#1\unitlength\divide\auxdim2
   \loop\advance\auxdim-\unitlengthy
   \ifdim\auxdim>-.5\unitlengthy
      \advance\y-\unitlengthy
      \xy{\vbox to0pt{\vss\copy\trielm}}%
      \advance\x\unitlengthx
   \repeat}
```

Example *(Trinary tree)*



```
$$\x0pt\y0pt\level6 \kk128\tritree}$$
%with trinary tree macro
%Pretext
$$\def\tritree{\ifnum1=\level \eertirt\fi
    \advance\level-1 \divide\kk2
    {\N{\the\kk}\tritree}%
    {\ESE{\the\kk}\tritree}%
     \WSW{\the\kk}\tritree}
\def\eertirt#1\tritree{\fi}
\x0pt\y0pt\level5 \kk128
\vbox to2cm{\hsize2cm
    \offinterlineskip
    \vss\tritree\vss}$$
%Posttext
```

Remark. The `\unitlength`-s are default equal to the sides of the elementary rectangular block. The size of the tree can be controlled by `\kk`.

[15]Hopefully, it is self-explanatory, at least with respect to the main flow. By the way in what order is the tree drawn?

13 Epilogue

The lines at 45° have little compatibility with TEX's rules, alas, especially with non-neglible thickness. I was surprised to realize that TEX's defaults for rules are not symmetric around their axes in relation to the reference point.

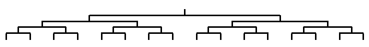
Have fun, and all the best.

14 Appendix: Binary tree and chart

Example *(Binary tree)*

```
\bluepictures\bintreepic\par
$$\bintreepic$$
```

with result



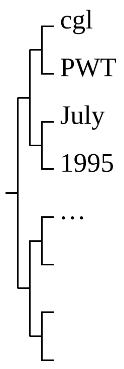
How can we add leaves[16]?

14.1 Rotated tree

Once we understand turtle graphics, rotating a tree can be done easily by shifting the meaning of the directions, and adjusting the positioning of the leaves.[17]

Example *(Rotated tree)*

```
\thispicture{\def\1{cgl}
  \def\2{PWT}\def\3{July}
  \def\4{1995}\def\5{\dots}
  \yoffset{-16}\ydim{28}}
$$\rotatedbintreepic$$
```



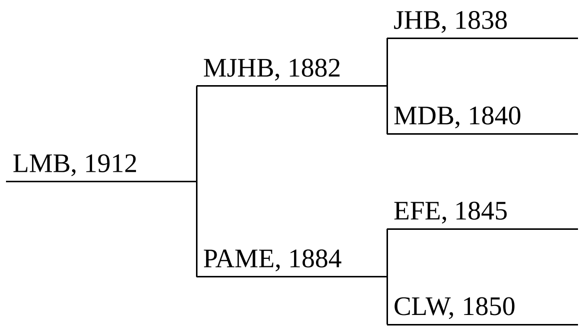
`\bintree` and `\eertnib` come with `blue.tex`, and `\rotatedtreepic` is included in `pic.dat`. The `\rotatedtreepic` entry reads as follows.

```
\lst\rotatedbintreepic{%July 1995, cgl
\bgroup\unitlength1ex%
  \let\W\N \let\exchange\E
  \let\E\S \let\S\exchange
  \def\1{x}\def\2{y}\def\3{a}\def\4{b}%
  \def\5{piet}\def\6{hans}\def\7{etc.}%
  \k0\kk16\xdim{10}\ydim{30}%
\beginpicture\bintree\endpicture\egroup
\thispicture{}}
```

14.2 Chart

Through the `\bintree` macro we can also obtain charts elegantly.

Example *(Chart, The TEXbook ex22.14 248)*



obtained via

```
%labels in preorder (default in \chartpic)
\def\1{LMB, 1912}
\def\2{MJHB, 1882}\def\5{PAME, 1884}
\def\3{JHB, 1838} \def\4{MDB, 1840}
\def\6{EFE, 1845} \def\7{CLW, 1850}
\ekk8
\k0\unitlength2ex\x0pt\y0pt\kk8
\hbox{\modbintree}
%with auxiliaries
\let\Eold\E
\def\E{\global\advance\k1
  \xytxt{ \csname\the\k\endcsname}\Eold}
```

Remark. An aid in finding the numbers of the branches is to delete `\csname` and `\endcsname` in `\E` . The way of traversal at hand is called preorder.

When using `\chartpic` from `pic.dat` the texts along the branches —`\def\1{...} etc.`—have to be supplied within a `\thispicture`, to override the defaults.

[16]See the PWT user guide for the answer.

[17]A white lie. The tree is actually mirrored because I like the leaves to be numbered from the top. In general we can rotate via POSTSCRIPT.