Building high level objects in PSTricks

TUG95 — St Petersburg, Florida

July 24, 1995

Denis Girou

Email : **Denis.Girou@idris.fr**

Presented by **Sebastian Rahtz**

Plan

Preliminary remarks

☞ Our goal is mainly **pedagogical**:

- to show the basic building principles of *PSTricks*,
- to try to demonstrate its exceptional power and quality,
- to show the importance of the notion of **object** in *PSTricks*,
- to show how personal objects are built,
- to give ideas and to stimulate the creativity of others...

☞ These extensions are experimental, and linked to the next version of *PSTricks*, so they are not available at the moment...

☞ We found it surprising, even among convinced LATEX users (!), to see how often they do not understand the meaning and power of **computational graphics**[a] versus a "*point and click*" drawing interface which does not exhibit the **logical structure** of such graphics...

☞ This work is extracted from a talk originally given in the GUTenberg 1995 meeting "*Une nouvelle vision des documents électroniques*", La Grande Motte, June 1, 1995.

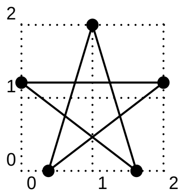
[a] From this point on view, we could also consider in the TEX world other tools like *(Al)DraTEX* from Eitan M. Gurari [Gurari 94], [`XYpic`] from Kristoffer H. Rose and Ross M. Moore or METAPOST from John D. Hobby [`MetaPost`].

A star is born

If we need to draw a **five branched star**, which is a common drawing, we can easily verify that there is no predefined one in *PSTricks*. So, we must do it ourselves; how to do so may not be obvious to everybody, if we want an accurate picture.

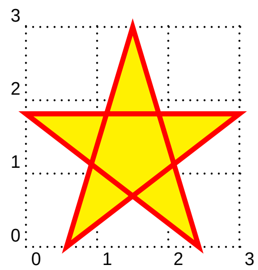
One possible solution is to use the coordinates of a star ... found elsewhere !

```
\psset{showpoints=true,dotsize=0.2}
\pspolygon(0,1.21)(2,1.21)(0.38,0)(1,2)(1.62,0)
```



In this case, we can obviously define a new LATEX command to use our **star** easily. Moreover, we can change many of its attributes, using standard parameters of *PSTricks* :

```
\newcommand{\Star}{%
  \pspolygon(0,1.21)(2,1.21)(0.38,0)(1,2)
            (1.62,0)}
\psset{unit=1.5,linewidth=0.7mm,linecolor=red,
       fillstyle=solid,fillcolor=yellow}
\Star
```

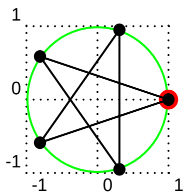


On the road of ... the *polar* star

Nevertheless, even if this result is satisfying, the **genericity** of this method is absolutely null: how are we to obtain a seven or eleven branched star in this way?

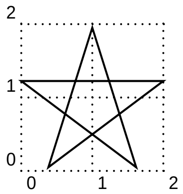
If we consult again the *PSTricks* manual, we see a better idea: using polar coordinates, which is a greatly improved method:

```
\SpecialCoor
\pscircle[linecolor=green](0,0){1}
\psdots[dotsize=0.3,linecolor=red](1;0)
\psset{showpoints=true,dotsize=0.2}
% 360 / 5 * 2 = 144
\pspolygon(1;0)(1;144)(1;288)(1;72)(1;216)
```



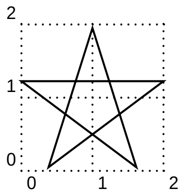
We know how to return to the initial position and scale, with appropriate parameters:

```
\SpecialCoor
% 360 / 5 / 2 / 2 = 18
% 1 / cos(18) = 1.052
\rput{18}(1,0.9){%
  \pspolygon[unit=1.052]
            (1;0)(1;144)(1;288)(1;72)(1;216)}
```



We can simplify it by changing the angle unit and by using the fact that they are calculated *modulo* this:

```
\SpecialCoor
\degrees[5]
% 18 / (360 / 5) = 0.25
\rput{0.25}(1,0.9){%
  \pspolygon[unit=1.052]
           (1;0)(1;2)(1;4)(1;6)(1;8)}
```

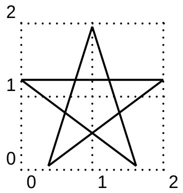


This last formulation is essential, because we can now use the \multido macro, which will open the way of **genericity**.

From particular to general

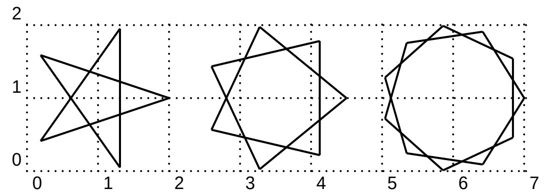
The preceding example gives the same result as the following, where the segments are drawn one after the other with \psline commands:

```
\newcount\IndiceTemp
\psset{unit=1.052}
\SpecialCoor
\degrees[5]
\rput{0.25}(0.95,0.86){%
  \multido{\I=0+2}{5}{%
    % \psline(1;\I)(1;\I+2)
    \IndiceTemp=\I
    \advance\IndiceTemp by 2
    \psline(1;\I)(1;\the\IndiceTemp)}}
```



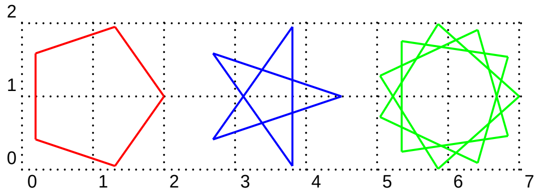
So, now we can build a command to draw a star with any number of branches:

```
\newcount\IndiceTemp
\newcommand{\GenericStar}[1]{%
\SpecialCoor
\degrees[#1]
\IndiceTemp=0
\multido{\I=0+2}{#1}{%
  \advance\IndiceTemp by 2
  \psline(1;\I)(1;\the\IndiceTemp)}}

\rput(1,1){\GenericStar{5}}
\rput(3.5,1){\GenericStar{7}}
\rput(6,1){\GenericStar{11}}
```



Another way of generalizing this is to parameterize the number of vertices that we jump to reach the next vertex used: **2** give the preceding stars that we drawn, but **1** will give the classical regular polygons, and values greater than **2** other types of polygons.

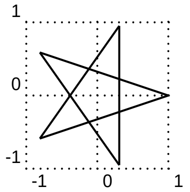
```
\newcount\IndiceTemp
\newcommand{\RegularPolygonA}[2]{%
\SpecialCoor
\degrees[#1]
\IndiceTemp=0
\multido{\I=0+#2}{#1}{%
  \advance\IndiceTemp by #2
  \psline(1;\I)(1;\the\IndiceTemp)}}

\psset{linecolor=red}
\rput(1,1){\RegularPolygonA{5}{1}}
\psset{linecolor=blue}
\rput(3.5,1){\RegularPolygonA{5}{2}}
\psset{linecolor=green}
\rput(6,1){\RegularPolygonA{11}{3}}
```



Filling of areas

In fact, this is a possibility that we have lost!

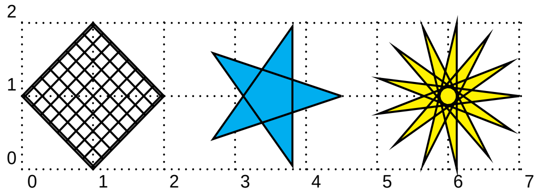
```
\psset{fillstyle=solid,fillcolor=cyan}
\RegularPolygonA{5}{2}
```



It does not work because we moved from the \pspolygon macro to the \psline macro, which, in this case, does not define a closed area that we can fill afterwards.

To solve this difficulty, we must used the **custom graphics** of *PSTricks*.


```
\newcommand{\RegularPolygonB}[2]{%
\SpecialCoor
\degrees[#1]
\pscustom{%
  \moveto(1;0)
  \multido{\I=#2+#2}{#1}{%
    \lineto(1;\I)}}}

\psset{fillstyle=crosshatch}
\rput(1,1){\RegularPolygonB{4}{1}}
\psset{fillstyle=solid,fillcolor=cyan}
\rput(3.5,1){\RegularPolygonB{5}{2}}
\psset{fillstyle=solid,fillcolor=yellow}
\rput(6,1){\RegularPolygonB{13}{6}}
```



Vertex and edges

Now, we must consider the question of the representation of the edges: if we do not want to see the interior lines of the stars, it is clear that we must use a completely different method.

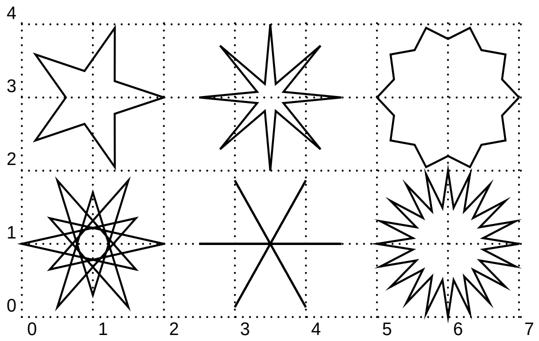
One solution is to introduce some reference points inside the circle (in medium positions), and to use them each time to join one vertex to the other. But in this case, **we will not jump to any vertex to draw a star**.

To do that, we will define a conventional value for this parameter (here 9999) to decide if we will use it or not.

```
\newcount\IndiceTemp
\newcommand{\RegularPolygonC}[3]{%
\SpecialCoor
\degrees[#1]
\IndiceTemp=#2
\divide\IndiceTemp by 2
\pscustom{%
  \moveto(1;0)
  \multido{\I=#2+#2}{#1}{%
    \ifdim#3pt=9999pt
      \lineto(1;\I)
    \else
      \ifodd#2
        \psline(#3;\the\IndiceTemp.5)
               (1;\I)
      \else
        \psline(#3;\the\IndiceTemp)
               (1;\I)
      \fi
      \advance\IndiceTemp by #2
    \fi}}}
```



```
\rput(1,3){\RegularPolygonC{5}{1}{0.38}}
\rput(3.5,3){\RegularPolygonC{8}{1}{0.2}}
\rput(6,3){\RegularPolygonC{10}{1}{0.8}}
\rput(1,1){\RegularPolygonC{6}{7}{0.7}}
\rput(3.5,1){\RegularPolygonC{6}{1}{0}}
\rput(6,1){\RegularPolygonC{20}{1}{0.5}}
```



[a] In the case where we want a special drawing, it is necessary to compute with precision the value of the intermediate point (**0.38** for a star with five branches).

Another interesting idea is to replace the straight lines which join the points by curves.

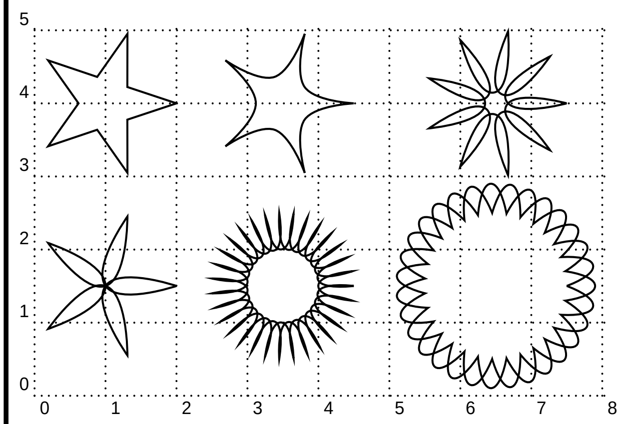
To do that, we have just to introduce a switch with a boolean value defined by a `\newif` command, and to draw lines or curves according to this value.

```
\newcount\IndiceTemp
\newif\ifCurves
\newcommand{\RegularPolygonD}[3]{%
\SpecialCoor
\degrees[#1]
\ifodd#2
  \def\DecimalPart{.5}%
\else
  \def\DecimalPart{}%
\fi
```



```
\IndiceTemp=#2
\divide\IndiceTemp by 2
\pscustom{%
  \moveto(1;0)
  \multido{\I=#2+#2}{#1}{%
    \ifdim#3pt=9999pt
      \lineto(1;\I)
    \else
      \ifCurves
        \pscurve(#3;\the\IndiceTemp\DecimalPart)
                (1;\I)
      \else
        \psline(#3;\the\IndiceTempB\DecimalPart)
               (1;\I)
      \fi
      \advance\IndiceTemp by #2
    \fi}}}
```

```
\rput(1,4){\RegularPolygonD{5}{1}{0.38}}
\Curvestrue
\rput(3.5,4){\RegularPolygonD{5}{1}{0.38}}
\rput(6.5,4){\RegularPolygonD{9}{1}{0.15}}
\rput(1,1.5){\RegularPolygonD{5}{4}{0}}
\rput(3.5,1.5){\RegularPolygonD{31}{2}{0.5}}
\rput(6.5,1.5){\RegularPolygonD{31}{2}{1.4}}
```

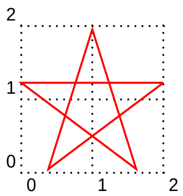


The problem of the management of parameters

Before we go any further, and improve our command with some new functionality, we must first notice that the management of the parameters is beginning to be heavy and uneasy.

To do the same thing as our first drawing, we must now write:

```
\Curvesfalse
\psset{unit=1.052,linecolor=red}
\rput{18}(0.95,0.86){%
  \RegularPolygonD{5}{2}{9999}}
```



The inconvenience comes from the necessity to define:

- **external** parameters which are not always used (\Curvesfalse),
- **internal** parameters which are not always used (the 9999 in third position),
- standard parameters of *PSTricks* necessarily as **global**, so before the usage of the command (linecolor=red).

$\Longrightarrow$ **So, it is clear that this method is not acceptable for complex cases.**

The solution?

To define our own PSTricks *objects*!

Definition of new *objects*

- We have simply to follow the same method as the one used for predefined objects in *PSTricks* itself.
- Some basic techniques in *PSTricks*, which we will use, have been previously described [Zandt (Van) et Girou 94].

Management of parameters

The analysis and management of parameters uses the [keyval] extension written by David CARLISLE[a].

We must consider three types of parameters:

❶ **integer** value parameters,

❷ **real** value parameters,

❸ **boolean** value parameters.

[a] In fact, this will be true only starting with the 0.94 version of *PSTricks*. Up to now, *PSTricks* used its own mechanism to manage the parameters. For the moment, we have written a *quick and dirty* interface between the 0.93 version of *PSTricks* and [keyval] to be able to use the new system.

❶ **Integer** value parameters, where we define a macro-command, using a temporary counter:

```
% Definition of the parameters
% ---------------------------

% Number of sides of the polygon
\define@key{PST}{RPolyNbSides}{%
\pst@cntg=#1\relax
\edef\psk@RPolyNbSides{\the\pst@cntg}}

% Offset to calculate angle
\define@key{PST}{RPolyOffset}{%
\pst@cntg=#1\relax
\edef\psk@RPolyOffset{\the\pst@cntg}}

% Rotation of the polygon (within 360 deg.)
\define@key{PST}{RPolyRotation}{%
\pst@cntg=#1\relax
\edef\psk@RPolyRotation{\the\pst@cntg}}
```


❷ **Real** value parameters, where we use a **register dimension**:

```
% Position of the intermediate point
% (default=9999pt i.e. not used)
\newdimen\psl@RPolyIntermediatePoint
\define@key{PST}{RPolyIntermediatePoint}{%
\pssetlength\psl@RPolyIntermediatePoint{#1}}
```

❸ **Boolean** value parameters, where we use a \newif command:

```
% Nodes joined by lines or curves
\newif\ifPst@RPolyCurves
\define@key{PST}{RPolyCurves}[true]{%
\@nameuse{Pst@RPolyCurves#1}}
```


Default values for parameters

The \setkeys command of *keyval* allows us to initialize the defined parameters:

```
% Defaults values:
% ---------------
% 5 sides, no offset, 0 degrees rotation,
% no intermediate point,
% points joined by lines
\setkeys{PST}{%
RPolyNbSides=5,RPolyOffset=1,
RPolyRotation=0,
RPolyIntermediatePoint=9999pt,
RPolyCurves=false}
```

Macro-command to define the *PSTricks object* \PstRegularPolygon

Now, we can define a new object \PstRegularPolygon, being careful to perform some verification of the validity of received values:

```
% The macro \PstRegularPolygon
% ----------------------------

\def\PstRegularPolygon{%
\@ifnextchar[\@PstRegularPolygon{%
\@PstRegularPolygon[]}}
\def\@PstRegularPolygon[#1]{{%
\setkeys{PST}{#1}
```



```
% Validation of the parameters
% ----------------------------
\ifnum\psk@RPolyNbSides<3
{\@pstrickserr{RPolyNbSides must be greater than
   2 (and not `\psk@RPolyNbSides')}\@eha}%
\fi
\ifnum\psk@RPolyNbSides>100
{\@pstrickserr{RPolyNbSides must be less than
   101 (and not `\psk@RPolyNbSides')}\@eha}%
\fi
\ifnum\psk@RPolyOffset<1
{\@pstrickserr{RPolyOffset must be greater than
   0 (and not `\psk@RPolyOffset')}\@eha}%
\fi
\ifnum\psk@RPolyOffset>100
{\@pstrickserr{RPolyOffset must be less than
   101 (and not `\psk@RPolyOffset')}\@eha}%
\fi
```



```
\ifodd\psk@RPolyOffset
  \def\Pst@RPolyDecimalPart{.5}%
\else
  \def\Pst@RPolyDecimalPart{}%
\fi
\pst@cnta=\psk@RPolyOffset
\divide\pst@cnta\tw@
\begin@ClosedObj
\SpecialCoor
\rput{\psk@RPolyRotation}(0,0){%
  \degrees[\psk@RPolyNbSides]
  \pscustom{%
    \moveto(1;0)
    \multido{\i=\psk@RPolyOffset+%
                \psk@RPolyOffset}
            {\psk@RPolyNbSides}{%
      \ifdim\psl@RPolyIntermediatePoint=9999pt
        \lineto(1;\i)
      \else
```



```
        \ifPst@RPolyCurves
          \pscurve%
            (\psl@RPolyIntermediatePoint;%
             \the\pst@cnta\Pst@RPolyDecimalPart)
            (1;\i)
        \else
          \psline%
            (\psl@RPolyIntermediatePoint;%
             \the\pst@cnta\Pst@RPolyDecimalPart)
            (1;\i)
        \fi
        \advance\pst@cnta\psk@RPolyOffset
      \fi}}} % End \multido, \pscustom and \rput
\end@ClosedObj}}
```

Note: In fact, for performance reasons, the *real* implementation puts the tests outside the loop, and repeats the \multido commands.

Personalizable functions

Now that we have a *PSTricks object*, we can add some new functionality to it.

We begin with the definition of a function to execute on each vertex of the polygon.

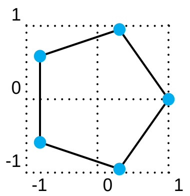
To do that, we will use a very simple but powerful technique, one which also has very great **flexibility**: if a function with a specific name has not been defined, nothing happens, otherwise we execute it, which can produce very different effects.

The function must know the name of the index variable (here `\i`).

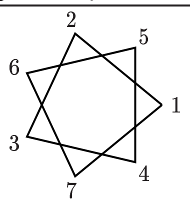

```
  \fi}}    % End \multido and \pscustom
\ifx\PstRegularPolygonNode\@undefined
\else
  \multido{\i=0+\psk@RPolyOffset}
          {\psk@RPolyNbSides}{%
    \PstRegularPolygonNode}
\fi}       % End \rput
```



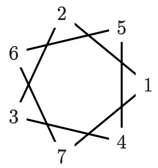
```
\providecommand{\PstRegularPolygonNode}{%
  \psdots[dotsize=0.2,linecolor=cyan](1;\i)}
\PstRegularPolygon
```



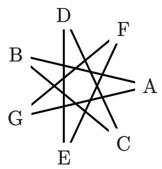

```
\providecommand{\PstRegularPolygonNode}{%
  \rput(1.2;\i){\small\the\multidocount}}
\PstRegularPolygon[RPolyNbSides=7,RPolyOffset=2]
```



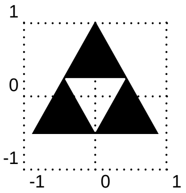

```
\providecommand{\PstRegularPolygonNode}{%
  \rput*(1;\i){\footnotesize\the\multidocount}}
\PstRegularPolygon[RPolyNbSides=7,RPolyOffset=2]
```



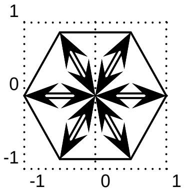

```
\newcounter{Letter}
\providecommand{\PstRegularPolygonNode}{%
  \setcounter{Letter}{\the\multidocount}
  \rput*(1;\i){\footnotesize\Alph{Letter}}}
\PstRegularPolygon[RPolyNbSides=7,RPolyOffset=3]
```



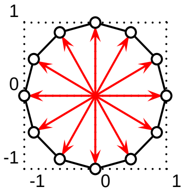
```
\providecommand{\PstRegularPolygonNode}{%
  \SpecialCoor
  \degrees[3]
  \rput{0.5}(0.5;\i){%
    \pspolygon*(0.5;0.5)(0.5;1.5)(0.5;2.5)}}
\PstRegularPolygon
  [RPolyNbSides=3,RPolyRotation=90]
```




```
\providecommand{\PstRegularPolygonNode}{%
  \psline[doubleline=true]{<->}(0;0)(1;\i)}
\PstRegularPolygon[RPolyNbSides=6]
```




```
\providecommand{\PstRegularPolygonNode}{%
  \psline[linecolor=red]{->}(0;0)(0.95;\i)
  \psdots[dotstyle=o,dotsize=0.2](1;\i)}
\PstRegularPolygon[RPolyNbSides=12]
```



Drawing of epicycloïdes

These specific curves known as **epicycloïdes** are very easy to draw using our regular polygons:

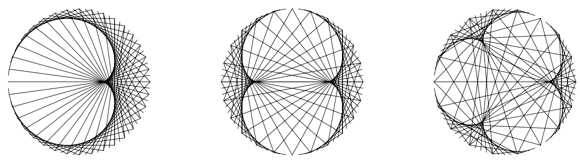
```
% Regular polygon or epicycloide
\newif\ifPst@RPolyEpicycloide
\define@key{PST}{RPolyEpicycloide}[true]{%
\@nameuse{Pst@RPolyEpicycloide#1}}

% Default: no epicycloide
\setkeys{PST}{RPolyEpicycloide=false}
```



```
  \pscustom{%
    \ifRPolyEpicycloide
      \pst@cnta=\psk@RPolyNbSides
      \divide\pst@cnta\tw@
      \multido{\i=0+1}{\psk@RPolyNbSides}{%
        \moveto(1;\i)
        \lineto(1;\the\pst@cnta)
        \advance\pst@cnta\psk@RPolyOffset}
    \else
```

```
        \fi}        % End \multido
    \fi}            % End \pscustom
```

```
\psset{linewidth=0.001,RPolyNbSides=72,
       RPolyEpicycloide}
% Cardioide (epicycloïde of order 2)
\rput(1,1){\PstRegularPolygon[RPolyOffset=2]}
% Nephroïde (epicycloïde of order 3)
\rput(4,1){\PstRegularPolygon[RPolyOffset=3]}
% Epicycloïde of order 4
\rput(7,1){\PstRegularPolygon[RPolyOffset=4]}
```



Of what is done a calendar?

Even if multiple embellishments in designing calendars may seem without serious interest[a], it's a very exemplary case for illustrating the **power** and **flexibility** of the conjunction of TEX + *PSTricks*.

There are in fact two distinct parts:

- the computations to do[b], which are not completely simple, but very well known, and requiring only integers, which is specially convenient for TEX,
- the presentation style, in both its **structural** and **aesthetic** aspects.

[a] Nevertheless, we began this package to realize the calendar of the (numerous) training courses of our institution, and we soon discovered the interest in being able to use very high level macros!

[b] We limit ourself here to **gregorian** calendars, even though it would not be difficult to implement some of other types.

Historic

The convenience of TEX for this kind of application is so obvious that there are already rather a lot of implementations.

A simple research on **CTAN** give (at least) six answers:

- [`kalender`], by Andreas Schrell, in 1991 ;
- [`calendar`], by Jürgen Stuber, in 1991 ;
- [`yearcal`], by Michael Barr, in 1992 ;
- some macros by Stephan von Bechtolsheim, in *TEX in Practice*, in 1993 [Bechtolsheim (von) 93] ;
- *calend0* and its presentation styles, by Jordi Saludes, in 1993 [Saludes 93] ;
- and the anonymous (!) *calendar.tex*.

In fact, their functionality and their level of sophistication are greatly variable (*calend0* and [`calendar`] are the only ones which allow the definition of **events**, some print only an annual calendar, etc.)[a].

Nevertheless, if all the packages generate calendars in one or more formats, the great weakness of all of them is their presentation, which is very poor and not attractive (usually looking like a `tabular` environment), due of course to the great limitations of TEX in this field. Moreover, the size of these calendars is fixed most of the time, which gives us no adaptability at all.

We can also change some very inconvenient characteristics of previous implementations (month and week names only in english or german[b], and weeks beginning only on Sunday or Monday).

[a] There is no doubt that *calend0* is the most evolved, even if it does not have all the options of the others.

[b] We introduce a general multilingual management.

```
% With the macros
% of Stephan von Bechtolsheim
% in TeX in Practice
\scriptsize
\ThreeMonthlyCalendars{1995}{6}{2.3cm}
```

June 1995

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

July 1995

Su	Mo	Tu	We	Th	Fr	Sa
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

August 1995

Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

But here *PSTricks* can show all its power!

Implementation choices

We choose to re-use the computation part of the macros of [Saludes 93], because they obviously answer our needs and there is no improvement possible here.

But on the other hand, we have redefined all the presentation interface, using *PSTricks*[a].

[a] For the moment, the management of **events** is not satisfying, and must be re-written to use **external** files for both generic events (like Christmas, etc.) and personal events.

For each language, we specify its own characteristics using special macros, **with names suffixed by the name of the language**.

The choice of the language will determine automatically if the calendars are of **european** style (with weeks beginning on Monday) or of **american** style (with weeks beginning on Sunday).

Here are some examples:

```
% European calendar (for all Europe?!)
\def\PstCL@europe{%
  \ifPst@CalendarMondayFirsttrue
  \def\PstCalendarWEaDefault{6}
  \def\PstCalendarWEbDefault{7}}
```



```
% French language
\def\PstCL@french{\PstCL@europe}
\def\PstMonthName@french#1{%
  \ifcase#1 \or Janvier\or F\'evrier\or
                Mars\or Avril\or Mai\or
                Juin\or Juillet\or
                Ao\^ut\or Septembre\or
                Octobre\or Novembre\or
                D\'ecembre\fi}
\def\PstWeekDayLong@french#1{%
  \ifcase#1 Lundi\or Mardi\or Mercredi\or
            Jeudi\or Vendredi\or Samedi\or
            Dimanche\fi}
\def\PstWeekDayShort@french#1{%
  \ifcase#1 Lun\or Mar\or Mer\or Jeu\or
            Ven\or Sam\or Dim\fi}
\def\PstWeekDayVeryShort@french#1{%
  \ifcase#1 L\or M\or M\or J\or V\or S\or
            D\fi}
```

```
% US language
\def\PstCL@US{%
  \psset@CalendarMondayFirst{false}
  \def\PstCalendarWEaDefault{1}
  \def\PstCalendarWEbDefault{7}}
\def\PstMonthName@US#1{%
  \ifcase#1 \or January\or February\or
                March\or April\or May\or
                June\or July\or August\or
                September\or October\or
                November\or December\fi}
\def\PstWeekDayLong@US#1{%
  \ifcase#1 Sunday\or Monday\or Tueday\or
            Wednesday\or Thursday\or
            Friday\or Saturday\fi}
\def\PstWeekDayShort@US#1{%
  \ifcase#1 Sun\or Mon\or Tue\or Wed\or
            Thu\or Fri\or Sat\fi}
\def\PstWeekDayVeryShort@US#1{%
  \ifcase#1 S\or M\or T\or W\or T\or F\or
            S\fi}
```



```
% Macro to set up the specific macros
% for various language (default = US)
\define@key{PST}{CalendarLanguage}{%
\@ifundefined{PstCL@#1}
{\@pstrickserr{%
Undefined calendar language: ‘#1’}\@eha}
\edef\Pst@CalendarLanguage{#1}
\@nameuse{PstCL@#1}}

% To specify the type of the day names
\define@key{PST}{CalendarDayNames}{%
\@ifundefined{PstCDN@#1}
{\@pstrickserr{%
Undefined calendar day names type: ‘#1’}
\@eha}
\@nameuse{PstCDN@#1}}
```


So, for instance, the macro which defines the way to represent the day names is a two-level one:

```
% Definition of the type of the day names
% used: long, short or very short
\def\PstCDN@long{%
\def\PstWeekDay##1{{\@nameuse{%
PstWeekDayLong@\Pst@CalendarLanguage}
{##1}}}}

\def\PstCDN@short{%
\def\PstWeekDay##1{{\@nameuse{%
PstWeekDayShort@\Pst@CalendarLanguage}
{##1}}}}

\def\PstCDN@veryshort{%
\def\PstWeekDay##1{{\@nameuse{%
PstWeekDayVeryShort@\Pst@CalendarLanguage}
{##1}}}}
```


It is the same for the macro which defines the way to represent the month and the year:

```
% Default definition used to print
% month name and year
\def\PstCalendarMonthYear#1#2{%
\PstMonthName{#1} #2}

% Definition of the type of month names used
\def\PstMonthName#1{%
{\@nameuse{%
PstMonthName@\Pst@CalendarLanguage}{#1}}}
```

```
\PstCalendarMonth{7}{1995}
```

July 1995

S	M	T	W	T	F	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					


```
\PstCalendarMonth
  [CalendarLanguage=french,
   CalendarDayNames=short]{7}{1995}
```

Juillet 1995

Lun	Mar	Mer	Jeu	Ven	Sam	Dim
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						


```
\renewcommand{\PstCalendarAttrMonthYear}{%
  \Large\itshape}
\renewcommand{\PstCalendarAttrDays}{%
  \tiny\bfseries\white
  \rput[r](-0.5,0){\psframe*(0.5,0.5)}
  \rput(-0.25,0.25)}
\PstCalendarMonth
  [unit=0.5,CalendarLanguage=german]{8}{1995}
```

August 1995

M	D	M	D	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			


```
\renewcommand{\PstCalendarMonthYear}[2]{%
  #2, \PstMonthName{#1}}
\renewcommand{\PstCalendarAttrDayNames}{%
  \small\bfseries}
\renewcommand{\PstCalendarAttrWE}{%
  \small\bfseries\blue}
\newpsstyle{StyleCalendarDays}{linestyle=none}
\newpsstyle{StyleCalendarWE}{linestyle=none}
\renewcommand{\PstCalendarAttrWhatever}{%
  \rput(0,2){\psframe[linestyle=none,
                      fillstyle=solid,
                      fillcolor=LightBlue](7,1)}}
\PstCalendarMonth
  [yunit=0.4,gridcolor=white,
   CalendarLanguage=spanish,
   CalendarDayNames=short]{12}{1995}
```

1995, Diciembre

Lun	Mar	Mié	Jue	Vie	Sáb	Dom
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

```
\PstCalendarAttrMonthsDefaults
\renewcommand{\PstCalendarAttrMonthYear}{%
  \rput(0,7){%
    \psframe[fillstyle=solid,fillcolor=black](7,1)}
  \tiny\bfseries\white}
\renewcommand{\PstCalendarAttrWE}{\tiny\red}
\newpsstyle{StyleCalendarDays}{linestyle=none}
\newpsstyle{StyleCalendarWE}{linestyle=none}
\PstCalendarYear
  [xunit=2.6mm,yunit=2mm,CalendarLanguage=german,
   CalendarNbMonthsRow=4,CalendarCompleteGrid,
   CalendarHSpace=1mm]{1995}
```

Januar 1995

M	D	M	D	F	S	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Februar 1995

M	D	M	D	F	S	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28					

März 1995

M	D	M	D	F	S	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

April 1995

M	D	M	D	F	S	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Mai 1995

M	D	M	D	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Juni 1995

M	D	M	D	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

Juli 1995

M	D	M	D	F	S	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

August 1995

M	D	M	D	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

September 1995

M	D	M	D	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

Oktober 1995

M	D	M	D	F	S	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

November 1995

M	D	M	D	F	S	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

Dezember 1995

M	D	M	D	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Management of events[a]

```
\begin{pspicture}(7,8)
  \psset{CalendarLanguage=french}
  \renewcommand{\PstCalendarAttrEvents}{%
    \small\bfseries\red}
  \PstCalendarMonthEvent{11}{1994}
  \PstCalendarEvent{11}{1994}{1}{1}
    {\tiny\bfseries Toussaint}
  \PstCalendarEvent{11}{1994}{11}{1}
    {\tiny\bfseries
     \begin{tabular}{c}
       Armistice\\
       14-18
     \end{tabular}}
\end{pspicture}
```

[a] This implementation is provisional and clearly not satisfying...

Novembre 1994

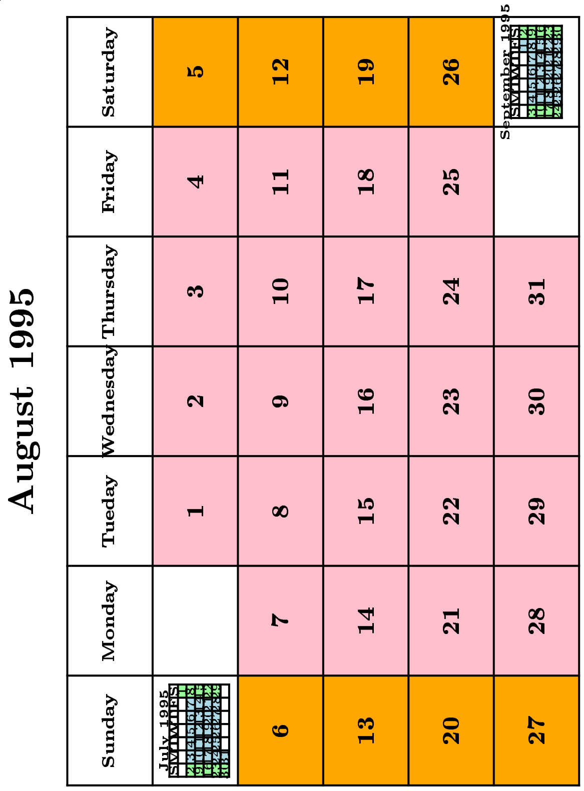
L	M	M	J	V	S	D
	1 Toussaint	2	3	4	5	6
7	8	9	10	11 Armistice 14-18	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				


```
\renewcommand{\PstCalendarAttrEvents}{%
  \large\bfseries\white}
\definecolor{MyGray}{gray}{0.65}
\modpsstyle{StyleCalendarDays}{fillcolor=MyGray}
\modpsstyle{StyleCalendarEvents}{fillcolor=black}
\psset{CalendarLanguage=french,
       CalendarEventsColorLine=white}
\begin{pspicture}(7,8)
  \PstCalendarMonthEvent
    [CalendarDayNames=short]{1}{1995}
  \PstCalendarEvent{1}{1995}{9}{5}
                 {\tiny\white CRAY}
  \PstCalendarEvent{1}{1995}{16}{3}
                 {\tiny\white F 90}
  \PstCalendarEvent{1}{1995}{23}{2}
                 {\tiny\white Grappe}
  \PstCalendarEvent{1}{1995}{30}{2}
                 {\tiny\white Graph.}
  % White separator lines
  \psline[linecolor=white](0,3)(3,3)
  \psline[linecolor=white](0,2)(2,2)
  \psline[linecolor=white](0,1)(2,1)
\end{pspicture}
```

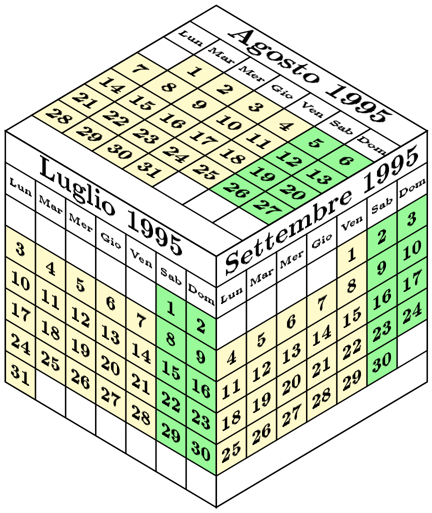
Janvier 1995

Lun	Mar	Mer	Jeu	Ven	Sam	Dim
						1
2	3	4	5	6	7	8
9 CRAY	10 CRAY	11 CRAY	12 CRAY	13 CRAY	14	15
16 F 90	17 F 90	18 F 90	19	20	21	22
23 Grappe	24 Grappe	25	26	27	28	29
30 Graph.	31 Graph.					

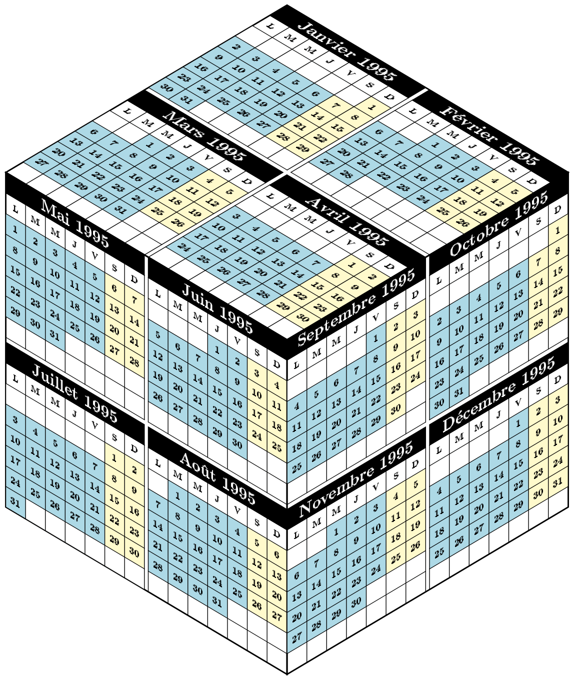

```
\rotateleft{%
  \begin{pspicture}(10.5,9.1)
    \psset{xunit=1.5cm,yunit=1.2cm}
    \renewcommand{\PstCalendarAttrDayNames}{%
      \scriptsize\bfseries}
    \modpsstyle{StyleCalendarDays}{fillcolor=Pink}
    \modpsstyle{StyleCalendarWE}{fillcolor=Orange}
    \PstCalendarMonthEvent
      [CalendarDayNames=long]{8}{1995}
  \renewcommand{\PstCalendarAttrMonthYear}{%
    \tiny\bfseries}
  \renewcommand{\PstCalendarAttrDayNames}{%
    \tiny\bfseries}
  \renewcommand{\PstCalendarAttrDays}{\tiny}
  \modpsstyle{StyleCalendarDays}{fillcolor=LightBlue}
  \modpsstyle{StyleCalendarWE}{fillcolor=PaleGreen}
  \modpsstyle{StyleCalendarEvents}{fillstyle=none}
  \PstCalendarEvent{8}{1995}{-8}{1}
    {\PstCalendarMonth
      [xunit=1.8mm,yunit=1.2mm,
       CalendarDayNames=veryshort]{7}{1995}}
  \PstCalendarEvent{8}{1995}{-35}{1}
    {\PstCalendarMonth
      [xunit=1.8mm,yunit=1.2mm,
       CalendarDayNames=veryshort]{9}{1995}}
\end{pspicture}}
```


```
\renewcommand{\PstCalendarAttrDayNames}{%
  \tiny\bfseries}
\modpsstyle{StyleCalendarDays}{%
  fillcolor=LemonChiffon}
\modpsstyle{StyleCalendarWE}{fillcolor=PaleGreen}
\PstCalendarQuarterOnCube
  [unit=0.6,CalendarLanguage=italian,
   CalendarDayNames=short]{7}{1995}
```



```
\renewcommand{\PstCalendarAttrMonthYear}{%
  \rput(0,7){%
    \psframe[fillstyle=solid,fillcolor=black]
      (7,1)}
  \LARGE\bfseries\white}
\modpsstyle{StyleCalendarDays}{
  fillcolor=LightBlue}
\modpsstyle{StyleCalendarWE}{%
  fillcolor=LemonChiffon}
\PstCalendarYearOnCube
  [unit=0.8,CalendarLanguage=french,
   CalendarCompleteGrid]{1995}
```

☞ Timothy Van Zandt, of course, for his splendid developments *PSTricks* and `Seminar`, and for his constant help and support since our first experiments with *PSTricks*.

And also:

☞ Philippe Esperet, *Lycée Henri IV*, Paris, `<100060.2106@compuserve.com>`, for his helpful comments and ideas.

☞ Sebastian Rahtz, for his help concerning the translation of this talk and for presenting it.

Bibliography

[Bechtolsheim (von) 93] Stephan von Bechtolsheim, *TEX in Practice*, four volumes, Springer-Verlag, New York, 1993.

[*calendar*] by Jürgen Stuber, Germany, 1991.

[Girou 94] Denis Girou, *Présentation de PSTricks*, Cahiers GUTenberg, Numéro 16, février 1994, pages 21–70.

[Gurari 94] Eitan M. Gurari, *TEX and LATEX: Drawing and Literate Programming*, McGraw-Hill, New-York, 1994, `<gurari@cis.ohio-state.edu>`.

[*kalender*] by Andreas Schrell, Wuppertal, Germany, 1991.

[*keyval*] by David Carlisle, Great Britain, documentation electronically distributed with the software, `<carlisle@cs.man.ac.uk>`.

[*MetaPost*] by John D. Hobby, ATT Bell Laboratories, USA.

Bibliography

[*PSTricks*] by Timothy Van ZANDT, Princeton University, USA, documentation electronically distributed with the software, Version 0.93a, March 1993, `<tvz@Princeton.EDU>`.

[Saludes 93] Jordi SALUDES, *Doing astronomical computations with TEX: Making agendas*, TUGboat, Volume 14, Number 1, April 1993, pages 40–53, `<saludes@ma2.upc.es>`.

[*XYpic*] by Kristoffer H. ROSE, DIKU, University of Copenhagen, Denmark, `<kris@diku.dk>` and Ross M. MOORE, Macquarie University, Sydney, Australia, `<ross@mpce.mq.edu.au>`.

[*yearcal*] by Michael BARR, Department of Mathematics and Statistics, McGill University, Montreal, Canada, 1992, `<barr@math.mcgill.ca>`.

[Zandt (Van) and Girou 94] Timothy Van ZANDT and Denis GIROU, *Inside PSTricks*, TUGboat, Volume 15, Number 3, September 1994, pages 239–246.