

draw pixel pictures

Jonathan P. Spratte*

2021-01-13 V1.1

Abstract

With **pxpic** you draw pictures pixel by pixel. It was inspired by a [lovely post](#) by Paulo Cereda, among other things (most notably a beautiful duck) showcasing the use of characters from the Mario video games by Nintendo in L^AT_EX.

Contents

1	Documentation	2
1.1	Drawing pictures	2
1.1.1	Examples	2
1.2	Setting options	4
1.2.1	Colour syntax	5
1.2.2	Available modes	5
1.3	Other customisation macros	5
1.4	Other macros	6
1.5	Miscellaneous	6
2	Implementation	7
2.1	Options	7
2.2	User macros	8
2.3	Parser	9
2.4	Modes	10
2.5	Pixel and Skip	11
2.6	Parser for colours	11
2.7	Messages	12

Index	13
--------------	-----------

*jspratte@yahoo.de

1 Documentation

1.1 Drawing pictures

pxpic supports different input modes, all of them have the same basic parsing behaviour. A `<pixel list>` contains the pixel colours. The image is built line wise from top left to bottom right. Each row of pixels should be a single \TeX argument (so either just one token, or a group delimited by `{}`), and within each line each pixel in turn should be a single \TeX argument (so either just one token, or a group delimited by `{}`). Spaces and hence single newlines in the sources between `<pixel list>` elements are ignored. The different modes are explained in [subsection 1.2.2](#). The only disallowed token in the `<pixel list>` is the control sequence `\xpic@end` (plus the usual restrictions of \TeX so no unbalanced braces, no macros defined as `\outer`).

There is a small caveat however: **pxpic** draws each pixel individually, and there is really no space between them, however some PDF viewers fail to display such adjacent lines correctly and leave small gaps (basically the same issue which packages like `colortbl` suffer from as well). In print this shouldn't be an issue, but some rasterisation algorithms employed by viewers and conversion tools have this deficit.

`\xpic` `\xpic[<options>]{<pixel list>}`

`<options>` might be any options as listed in [subsection 1.2](#), and `<pixel list>` is a list of pixels as described above. `\xpic` parses the `<pixel list>` and draws the corresponding picture. The result is contained in an `\hbox` and can be used wherever \TeX expects an `\hbox`. As a result, when you're in vertical mode a `\xpic` will form a text line, to prevent this you can use `\leavevmode` before it. The `\xpic` will be bottom aligned, you can change this using `\raisebox` (or, if you want, \TeX 's `\raise` and `\lower` primitives).

1.1.1 Examples

Since the above explanation of the `<pixel list>` syntax might've been a bit cryptic, and a good documentation should contain examples (this doesn't claim this documentation is *good*), well, here are some examples (you might need to take a look at [subsection 1.2](#) and [subsection 1.2.2](#) to fully understand the examples). Examples in this section will use the following `\xpicsetup`:

```
\xpicsetup
{
  mode      = px
  ,colours = {k=black, r=[HTML]{9F393D}, g=green!75!black, b=[rgb]{0,0,1}}
  ,skip     = .
  ,size     = 10pt
}
```

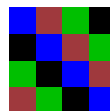
We can draw a small cross rather easily:

```
\xpic
{
  {.k}
  {kkk}
  {.k}
}
```



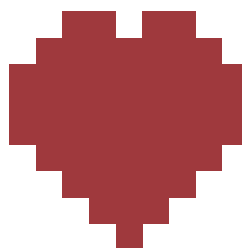
A small multicoloured grid:

```
\pxpic
{
    {b
    {k
    {g
    {rg
}
```



A heart (shamelessly copied example from **PixelArt**):

```
\pxpic
{
    { .
    { .
    { r
    { r
    { r
    { .
    { .
    { .
    { .
}
```



Using `mode=rgb` to draw a short coloured line:

```
\pxpic[mode=rgb]{\{\{1,0,1\}\{1,1,0\}\{0,1,1\}\}}
```



A multicoloured grid using skips and mode=cmy:

```
\pxpic[mode=cmy]
{
    {{1,0,1} {1,
    {{}} {1,
    {{0,1,1} {}
    {{1,1,0} {0,
}
```



Showing the difference between a skipped and a white pixel:

```
\pxpicsetup{colours = {w=white}}
\colorbox{gray}{\pxpic{{bbb}}{b.b}}{bbb}}
\colorbox{gray}{\pxpic{{bbb}}{bwb}}{bbb}}}
```



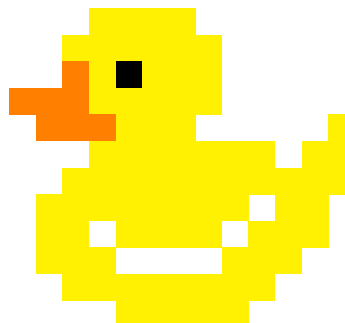
A biggish example: Tux.¹ I put two rows of pixels per code line to reduce the size a bit and the code is displayed tinily.

[illegible]

¹Source: https://www.reddit.com/r/linux/comments/hwpm9j/tux_pixel_art_v10/

Just for Paulo, a duck. Also, showing that the colour definitions in mode=px can be arbitrary tokens or multiple letters:

```
\pxpic[colours = {oo=orange,
                  \ylw=yellow,
                  \blk=black},
        skip    = \skip]
{
  {\skip\skip\skip\ylw\ylw\ylw\ylw}
  {\skip\skip\ylw\ylw\ylw\ylw\ylw\ylw}
  {\skip\skip{oo}\ylw\blk\ylw\ylw\ylw}
  {\oo}{oo}{oo}\ylw\ylw\ylw\ylw\ylw}
  {\skip{oo}{oo}{oo}\ylw\ylw\ylw\skip\skip\skip\ylw}
  {\skip\skip\skip\ylw\ylw\ylw\ylw\ylw\ylw\skip\ylw\ylw}
  {\skip\skip\ylw\ylw\ylw\ylw\ylw\ylw\ylw\ylw\ylw}
  {\skip\ylw\ylw\ylw\ylw\ylw\ylw\ylw\skip\ylw\ylw}
  {\skip\ylw\ylw\skip\ylw\ylw\ylw\skip\ylw\ylw}
  {\skip\ylw\ylw\ylw\skip\skip\skip\skip\ylw\ylw}
  {\skip\skip\ylw\ylw\ylw\ylw\ylw\ylw\ylw\ylw}
  {\skip\skip\skip\skip\ylw\ylw\ylw\ylw\ylw}
}
```



Another example might be the definition of `\pxpiclogo` in [subsection 2.2](#).

Who still needs picture-mode or complicated packages like pstricks or TikZ with such pretty pictures?

1.2 Setting options

To control its behaviour `\pxpic` uses a key=value interface powered by `expl3`. Options can be set either in the optional argument of `\pxpic` or with

`\pxpicsetup`

`\pxpicsetup{<options>}`

Sets the `<options>` locally to the current \TeX group.

Package options are not supported.

The available options are

<code>colors</code>	Define pixel colours for mode=px, see subsubsection 1.2.1 for a description of the value's syntax. No pixel definitions are made by the package.
<code>colours</code>	see <code>colors</code> .
<code>gap-hack</code>	To fix the issues with visible gaps in PDF viewers you can introduce some negative kerns to make the pixels overlap (lines overlap to the top, pixels to the left). This option expects a dimension as its value. A positive value will (maybe) close the gaps, a negative value will introduce real gaps. In any case the outermost pixels' borders still coincide with the borders of the surrounding <code>\hbox</code> . Take a look at my babbling about this issue in subsection 1.5 .
<code>ht</code>	Set the height of the pixels.
<code>mode</code>	Set the used mode, see subsubsection 1.2.2 for available modes. Initial value is px.
<code>size</code>	Set both <code>ht</code> and <code>wd</code> . Initial value is 1.opt.
<code>skip</code>	Define the value to be a skip (an empty space of width <code>wd</code>) in mode=px. No skip definitions are made by the package.
<code>wd</code>	Set the width of the pixels.

1.2.1 Colour syntax

In the value of the `colours` option you'll have to use the following syntax. Use a comma separated key=value list in which each key corresponds to a new pixel name for `mode=px`, and each value to the used colour. If the colour starts with an opening bracket use the complete value as is behind `\color`, else use the whole value as the first mandatory argument to `\color` with a set of braces added. For example to define `r` as the named colour red, and `x` as the colour `#abab0f` (in the `HTML` colour model) use:

```
colours = {r=red, x=[HTML]{abab0f}}
```

1.2.2 Available modes

`px` As already mentioned, `\xpic` supports different modes of input. The easiest to use mode is `px`, in which each element of the `\pixel list` has been previously defined as either a coloured pixel (using the `colour` option) or as a skipped pixel (using the `skip` option, resulting in a fully transparent pixel). Each element will be `\detokenized`, so (within `TEX`'s limitations) the name of a pixel can be arbitrary. This is the initial mode `\xpic` uses. But other options are available as well.

`named` Another mode is `named`, in which each element of the `\pixel list` should be a named colour (or colour expression) known to `xcolor`. Each element will be used like so: `\color{<element>}\px`. An exception is an element which is empty (`{}`), which will be a skipped pixel.

`rgb, cmy, cmyk, hsb, Hsb, tHsb, gray, RGB, HTML, HSB, Gray, wave`
The modes `rgb, cmy, cmyk, hsb, Hsb, tHsb, gray, RGB, HTML, HSB, Gray, and wave` correspond to the different colour models supported by `xcolor`. With these modes each element of the `\pixel list` will be the values in these colour models, so they'll be used like so: `\color[<mode>]{<element>}\px`. An exception is an element which is empty (`{}`), which will be a skipped pixel.

You can define additional modes selectable with the `mode` option using the macros `\xpicnewmode` or `\xpicsetmode`.

1.3 Other customisation macros

<code>\xpicnewmode</code>	<code>\xpicnewmode{<name>}{<definition>}</code>
---------------------------	---

<code>\xpicsetmode</code>

You can define your own modes with `\xpicnewmode`. Inside `<definition>` #1 is the currently parsed item in the `\xpic \pixel list`. You can output a pixel using `\px`, and skip a pixel using `\pxskip`. The pixel will use the currently active colour (so if you want to draw a red pixel you could use `\color{red}\px`). `\xpicnewmode` will throw an error if you try to define a mode which already exists, `\xpicsetmode` has no checks on the name.

<code>\xpicforget</code>	<code>\xpicforget{<px>}</code>
--------------------------	--------------------------------------

Undefines the `<px>` definition for use in `mode=px` (or `skip` symbol) added with the `colours` (or `skip`) option.

1.4 Other macros

<hr/> <code>\px</code> <hr/> <code>\pxskip</code> <hr/>	Inside of a <code>\pxpic</code> the macro <code>\px</code> draws a pixel (of the currently active colour), and <code>\pxskip</code> leaves out a pixel (so this one pixel is fully transparent). Use this in the $\langle definition \rangle$ of a mode in <code>\pxpicnewmode</code> .
<hr/> <code>\pxpicHT</code> <hr/> <code>\pxpicWD</code> <hr/>	These two are <code>dimen</code> registers storing the height and width of the pixels.
<hr/> <code>\pxpiclogo</code> <hr/>	<code>\pxpiclogo[\langle size \rangle]</code> This draws the logo of pxpic . The $\langle size \rangle$ controls the pixel size.

1.5 Miscellaneous

If you find bugs or have suggestions I'll be glad to hear about it, you can either open a ticket on Github (https://github.com/Skillmon/ltx_pxpic) or email me (see the first page).

A similar package is **PixelArt**, which, as of writing this, is described as a “working draft” by its author. **pxpic** wasn't intended as a direct competitor (I already started coding **pxpic** when I learned about **PixelArt**'s existence), but I took inspiration from the “Bugs, Ideas, Undefined behaviours” section of **PixelArt**'s documentation for the syntax of `mode=px`.

Regarding the gap issue: The pixels are output touching each other with no real gap, however some PDF viewers and tools will display such a gap. To make things even worse, the effect depends on the viewers current magnification. **pxpic** has the `gap-hack` option to provide some crude hack that might fix the issue, at the cost that the pixels are smaller than they were specified to be, except for the pixels at the right and bottom border. Also pixels next to skipped pixels have a different size (skipped pixels don't cover pixels to their left or top as they are transparent). You'll want to find a good trade-off value if you want to use `gap-hack`, that mitigates the effect but isn't too big (to make the errors less obvious). You can play with the value and decide for yourself what's the lesser evil. Or you do like me, don't use `gap-hack` and blame the viewers.

2 Implementation

Report who we are

```
1 \ProvidesPackage{pxpic}[2021-01-13 v1.1 draw pixel pictures]
```

and load dependencies

```
2 \RequirePackage{xcolor}
3 \RequirePackage{expkv}
```

`\pxpicHT` These two variables store the height and width of a pixel.
`\pxpicWD`

```
4 \@ifdefinable\pxpicHT{\newdimen\pxpicHT}
5 \@ifdefinable\pxpicWD{\newdimen\pxpicWD}
6 \pxpicHT1pt
7 \pxpicWD\pxpicHT
```

(End definition for \pxpicHT and \pxpicWD. These variables are documented on page 6.)

`\pxpic@kern` To fix the visible gaps in some PDF viewers if the user chooses so with the gap-hack option we introduce some \kerns of the length stored in this register.

```
8 \@ifdefinable\pxpic@kern{\newdimen\pxpic@kern}
9 \pxpic@kern\z@
```

(End definition for \pxpic@kern.)

2.1 Options

We define the options using `expkv` directly (no fancy options are involved and these are just a few anyway).

The first few options are straight forward. We use `expkv`'s name space to actually store the skip and px definitions, hence we use `\ekvdefNoVal` in the code of skip.

```
10 \protected\ekvdef{pxpic}{size}
11   {\pxpicHT\dimexpr#1\relax\pxpicWD\dimexpr#1\relax}
12 \protected\ekvdef{pxpic}{ht}{\pxpicHT\dimexpr#1\relax}
13 \protected\ekvdef{pxpic}{wd}{\pxpicWD\dimexpr#1\relax}
14 \protected\ekvdef{pxpic}{gap-hack}{\pxpic@kern\dimexpr#1\relax}
15 \protected\ekvdef{pxpic}{skip}{\ekvdefNoVal{pxpic@px}{#1}{\pxskip}}
```

The colours option is parsed using `\ekvparse` and `\pxpic@setcolor`.

```
16 \protected\ekvdef{pxpic}{colors}{\ekvparse\pxpic@noval\pxpic@setcolor{#1}}
17 \ekvletkv{pxpic}{colours}{pxpic}{colors}
```

And the mode just checks whether the mode macro is defined and lets the auxiliary macro `\pxpic@parse@px` to the defined mode.

```
18 \protected\ekvdef{pxpic}{mode}
19   {%
20     \@ifundefined{pxpic@parse@px@#1}%
21     {\pxpic@unknown@mode{#1}}%
22     {%
23       \expandafter\let\expandafter\pxpic@parse@px
24       \csname pxpic@parse@px@#1\endcsname
25     }%
26   }
```

2.2 User macros

\xpic \xpic expands directly to an opened \hbox, the auxiliary \xpic@ checks for the optional argument and inserts the rest of the code. We need to set \baselineskip to \xpicHT so that the pixels are stacked vertically without gaps. \xpic@parse will parse the *<pixel list>* until \xpic@end is hit. The final \egroup closes the \hbox. The row-wise output is done via a \vbox in which each pixel row will be wrapped inside an \hbox. The \kern negates a negative \kern in \xpic@parse so that the first line isn't moved.

```

27 \newcommand*\xpic{\hbox\bgroup\xpic@}
28 \newcommand\xpic@[2] []
29   {%
30     \vbox
31     {%
32       \xpicsetup{#1}%
33       \let\px\xpic@px
34       \let\pxskip\xpic@skip
35       \baselineskip\xpicHT
36       \kern\xpic@kern
37       \xpic@parse#2\xpic@end
38     }%
39   \egroup
40 }

```

(End definition for \xpic and \xpic@. These functions are documented on page 2.)

\xpicsetup Just directly defined to call **expkv**'s parser for the **\xpic** set.

```

41 \ekvsetdef\xpicsetup{\xpic}

```

(End definition for \xpicsetup. This function is documented on page 4.)

\xpiclogo The logo is just a biggish pixel picture. The \lower will move it down a bit so that it appears correctly aligned on the baseline. Since the logo should be part of a normal sentence in most usages we put \leavevmode before it. Also we make sure that the mode and px definitions are correct.

```

42 \newcommand*\xpiclogo[1] [.13ex]
43   {%
44     \begingroup
45     \xpicHT\dimexpr#1\relax
46     \xpicWD\xpicHT
47     \leavevmode
48     \lower3.2\xpicHT\xpic
49     [mode=px,colours={o=[HTML]{9F393D},g=black!75},skip=.]
50     {
51       {.....g}
52       {.....gggg}
53       {.oooo.....gggg.....ggg}
54       {.ooooo...oo.....oo...oo...ggggg...gg...g.....g}
55       {.oooooooooooo...ooooo...oooo...ggggggggggg...ggggg...gggggggg}
56       {..ooooo...ooooo.ooooo.ooooo...ggggg...gggg...ggggggg.ggggggggg}
57       {..ooooo...ooooo...ooooo...ggggg...gggg...gggg...gggg.ggg}
58       {..ooooo...ooooo...ooooo...ggggg...gggg...gggg...gggg}
59       {oooooooooooo...ooooo.ooooo...ggggggggggg...gggg...ggggggggg}
60       {o.ooooo.ooooo.ooooo.ooooo.g.gggggggg...ggggg.gggggggg}
61     }

```



```

62      {...ooo.o.....o.oo...oo.....ggg.g.....gg.....ggg}
63      {...ooo.....ggg}
64      {...ooo.....ggg}
65      {...o.....g}
66    }%
67  \endgroup
68 }

```

(End definition for `\xpiclogo`. This function is documented on page 6.)

`\xpicforget` Straight forward, just let the `px` macro to an undefined macro.

```

69 \newcommand\xpicforget[1]
70   {\expandafter\let\csname\ekv@name\xpic@px\}{#1}\endcsname\xpic@undef}

```

(End definition for `\xpicforget`. This function is documented on page 5.)

`\xpicnewmode` These are pretty simple as well, the new variant will use `\newcommand` which will do the testing for us, the set variant uses `\def`.

`\xpicsetmode`

```

71 \protected\def\xpicnewmode#1#2%
72   {\expandafter\newcommand\csname xpic@parse@px@#1\endcsname[1]{#2}}
73 \protected\def\xpicsetmode#1#2%
74   {\long\expandafter\def\csname xpic@parse@px@#1\endcsname##1{#2}}

```

(End definition for `\xpicnewmode` and `\xpicsetmode`. These functions are documented on page 5.)

2.3 Parser

`\xpic@ifend` These are three helper macros. The first just gobbles everything until the next **`\xpic@ifempty`** `\xpic@end`, and we borrow a fast test for an empty argument from **`explx`**. The last can be used to check for an opening bracket if used like **`\xpic@ifbracket`** `\xpic@end` **`#1.\xpic@end[]\xpic@end`**.

```

75 \long\def\xpic@ifend#1\xpic@end{}
76 \let\xpic@ifempty\ekv@ifempty
77 \long\def\xpic@ifbracket#1\xpic@end[#2]\xpic@end{\xpic@ifempty{#2}}

```

(End definition for `\xpic@ifend`, `\xpic@ifempty`, and `\xpic@ifbracket`.)

`\xpic@parse` The parsing loop is pretty simple, first check whether we're done, else open a new `\hbox` (which will form a row in the `\vbox` placed by `\xpic@`) in which the inner parsing loop is run. Then call the next iteration. If we're done just gobble the remainder of the current iteration. First we introduce our `\kern` which might fix the gap issue. Another `\kern` is done at the start of each `\hbox` to compensate the unnecessary `\kern` done by the first `\xpic@parseline`.

`\xpic@done`

```

78 \newcommand\xpic@parse[1]
79   {%
80     \xpic@ifend#1\xpic@done\xpic@end
81     \kern-\xpic@kern
82     \hbox{\kern\xpic@kern\xpic@parseline#1\xpic@end}%
83     \xpic@parse
84   }
85 \long\def\xpic@done\xpic@end\kern-\xpic@kern\hbox#1\xpic@parse{}

```

(End definition for `\xpic@parse` and `\xpic@done`.)

`\xpic@parseline` The line parsing loop also checks whether we're done, if not we place a pixel using
`\xpic@linedone` the current definition of `\xpic@parse@px` (which will be set by the current mode) and afterwards call the next iteration. If we're done we gobble the remainder of the current iteration and control goes back to `\xpic@parse`. Before each pixel we introduce a negative `\kern` to maybe fix the gap issue by letting the pixels overlap a bit.

```

86 \newcommand\xpic@parseline[1]
87 {%
88   \xpic@ifend#1\xpic@linedone\xpic@end
89   \kern-\xpic@kern
90   \xpic@parse@px{#1}%
91   \xpic@parseline
92 }
93 \long\def\xpic@linedone
94   \xpic@end\kern-\xpic@kern\xpic@parse@px#1\xpic@parseline
95 {}

```

(End definition for `\xpic@parseline` and `\xpic@linedone`.)

2.4 Modes

The modes define how a single element of the `<pixel list>` is parsed.

`\xpic@parse@px@px` In the `px` mode we check whether the pixel is defined (using the name space of `expkv`),
`\xpic@parse@px` if so call it, else throw an error and skip. Since this is also the initial mode we `\let` the auxiliary macro `\xpic@parse@px` to this mode here.

```

96 \newcommand\xpic@parse@px@px[1]
97 {%
98   \ekvifdefinedNoVal{xpic@px}{#1}
99   {\csname\ekv@name{xpic@px}{#1}N\endcsname}%
100   {%
101     \xpic@unknown@px{#1}%
102     \pxskip
103   }%
104 }
105 \let\xpic@parse@px\xpic@parse@px@px

```

(End definition for `\xpic@parse@px@px` and `\xpic@parse@px`.)

`\xpic@parse@px@named` `named` just checks whether the skip is empty. If so skip, else call `\color` with the element and output a pixel.

```

106 \newcommand\xpic@parse@px@named[1]
107 {%
108   \xpic@ifempty{#1}
109   {\pxskip}
110   {\color{#1}\px}%
111 }

```

(End definition for `\xpic@parse@px@named`.)

`\xpic@parse@px@rgb` The colour model modes are all the same in principle. They test for an empty element
`\xpic@parse@px@cmv` to introduce a skip, else they call `\color` with the respective colour model and output
`\xpic@parse@px@cmv` a pixel. We use the auxiliary `\xpic@tmp` to do all those definitions and undefine it
`\xpic@parse@px@hsb` afterwards.

`\xpic@parse@px@Hsb`
`\xpic@parse@px@tHsb`
`\xpic@parse@px@gray`
`\xpic@parse@px@RGB`
`\xpic@parse@px@HTML`
`\xpic@parse@px@HSB`
`\xpic@parse@px@Gray`
`\xpic@parse@px@wave`

```

112 \def\pxpic@tmp#1%
113   {%
114     \pxpicnewmode{#1}%
115     {%
116       \pxpic@ifempty{##1}
117       {\pxskip}
118       {\color{#1}{##1}\px}}%
119     }%
120   }
121 \pxpic@tmp{rgb}
122 \pxpic@tmp{cmy}
123 \pxpic@tmp{cmyk}
124 \pxpic@tmp{hsb}
125 \pxpic@tmp{Hsb}
126 \pxpic@tmp{tHsb}
127 \pxpic@tmp{gray}
128 \pxpic@tmp{RGB}
129 \pxpic@tmp{HTML}
130 \pxpic@tmp{HSB}
131 \pxpic@tmp{Gray}
132 \pxpic@tmp{wave}
133 \let\pxpic@tmp\pxpic@undef

```

(End definition for \pxpic@parse@px@rgb and others.)

2.5 Pixel and Skip

\pxpic@px The actual definition of pixels and skips is stored in macros to which the frontend macros
\pxpic@skip \px and \pxskip will be let inside of \pxpic.

```

134 \newcommand\pxpic@px{\vrule\@height\pxpicHT\@width\pxpicWD\@depth\z@}
135 \newcommand\pxpic@skip{\hskip\pxpicWD}

```

(End definition for \pxpic@px and \pxpic@skip.)

2.6 Parser for colours

\pxpic@setcolor First we test whether the colour starts with an opening bracket or not. Depending on
\pxpic@setcolor@a that we either just put the colour after \color, or put braces around it (as it then is a
\pxpic@setcolor@b colour expression for xcolor and just a single argument). \pxpic@setcolor defines a px
in the name space of **exp_{kv}** (this has a slight overhead during definition, but **exp_{kv}** is fast
in checking whether one of its keys is defined or not, and reduces the amount of code in
this package).

```

136 \newcommand\pxpic@setcolor[2]
137   {%
138     \pxpic@ifbracket\pxpic@end#2.\pxpic@end[]\pxpic@end
139     \pxpic@setcolor@a\pxpic@setcolor@b
140     {#1}{#2}%
141   }
142 \newcommand\pxpic@setcolor@a[2]
143   {\ekvdefNoVal\pxpic@px{#1}{\color{#2}\px}}
144 \newcommand\pxpic@setcolor@b[2]
145   {\ekvdefNoVal\pxpic@px{#1}{\color{#2}\px}}

```

(End definition for \pxpic@setcolor, \pxpic@setcolor@a, and \pxpic@setcolor@b.)

2.7 Messages

`\xpic@noval` These are just some macros throwing errors, nothing special here.

```
\xpic@unknown@px 146 \newcommand\xpic@noval[1]
\xpic@unknown@mode 147 {\PackageError{xpic}{Missing colour definition for name '\detokenize{#1}'}{}}
148 \newcommand\xpic@unknown@px[1]
149 {\PackageError{xpic}{Unknown pixel '\detokenize{#1}'. Skipping}{}}
150 \newcommand\xpic@unknown@mode[1]
151 {\PackageError{xpic}{Unknown mode '#1'}{}}
```

(End definition for `\xpic@noval`, `\xpic@unknown@px`, and `\xpic@unknown@mode`.)

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

P

<code>\px</code>	<u>6</u> , <u>33</u> , <u>110</u> , <u>118</u> , <u>143</u> , <u>145</u>
<code>\xpic</code>	<u>2</u> , <u>27</u> , <u>48</u>
<code>\xpicforget</code>	<u>5</u> , <u>69</u>
<code>\xpicHT</code>	<u>4</u> , <u>6</u> , <u>11</u> , <u>12</u> , <u>35</u> , <u>45</u> , <u>46</u> , <u>48</u> , <u>134</u>
<code>\xpiclogo</code>	<u>6</u> , <u>42</u>
<code>\xpicnewmode</code>	<u>5</u> , <u>71</u> , <u>114</u>
<code>\xpicsetmode</code>	<u>5</u> , <u>71</u>
<code>\xpicsetup</code>	<u>4</u> , <u>32</u> , <u>41</u>
<code>\xpicWD</code>	<u>4</u> , <u>6</u> , <u>11</u> , <u>13</u> , <u>46</u> , <u>134</u> , <u>135</u>
<code>\pxskip</code>	<u>6</u> , <u>15</u> , <u>34</u> , <u>102</u> , <u>109</u> , <u>117</u>

T

TEX and L^AT_EX 2_ε commands:

<code>\xpic@</code>	<u>27</u>
<code>\xpic@done</code>	<u>78</u>
<code>\xpic@end</code>	<u>37</u> , <u>75</u> , <u>77</u> , <u>80</u> , <u>82</u> , <u>85</u> , <u>88</u> , <u>94</u> , <u>138</u>
<code>\xpic@ifbracket</code>	<u>75</u> , <u>138</u>
<code>\xpic@ifempty</code>	<u>75</u> , <u>108</u> , <u>116</u>
<code>\xpic@ifend</code>	<u>75</u> , <u>80</u> , <u>88</u>
<code>\xpic@kern</code> .	<u>8</u> , <u>14</u> , <u>36</u> , <u>81</u> , <u>82</u> , <u>85</u> , <u>89</u> , <u>94</u>
<code>\xpic@linedone</code>	<u>86</u>
<code>\xpic@noval</code>	<u>16</u> , <u>146</u>
<code>\xpic@parse</code>	<u>37</u> , <u>78</u>
<code>\xpic@parse@px</code>	<u>23</u> , <u>90</u> , <u>94</u> , <u>96</u>

<code>\xpic@parse@px@cmy</code>	<u>112</u>
<code>\xpic@parse@px@cmyk</code>	<u>112</u>
<code>\xpic@parse@px@Gray</code>	<u>112</u>
<code>\xpic@parse@px@gray</code>	<u>112</u>
<code>\xpic@parse@px@HSB</code>	<u>112</u>
<code>\xpic@parse@px@Hsb</code>	<u>112</u>
<code>\xpic@parse@px@hsb</code>	<u>112</u>
<code>\xpic@parse@px@HTML</code>	<u>112</u>
<code>\xpic@parse@px@named</code>	<u>106</u>
<code>\xpic@parse@px@px</code>	<u>96</u>
<code>\xpic@parse@px@RGB</code>	<u>112</u>
<code>\xpic@parse@px@rgb</code>	<u>112</u>
<code>\xpic@parse@px@tHsb</code>	<u>112</u>
<code>\xpic@parse@px@wave</code>	<u>112</u>
<code>\xpic@parseline</code>	<u>82</u> , <u>86</u>
<code>\xpic@px</code>	<u>33</u> , <u>134</u>
<code>\xpic@setcolor</code>	<u>16</u> , <u>136</u>
<code>\xpic@setcolor@a</code>	<u>136</u>
<code>\xpic@setcolor@b</code>	<u>136</u>
<code>\xpic@skip</code>	<u>34</u> , <u>134</u>
<code>\xpic@tmp</code>	<u>112</u> , <u>121</u> , <u>122</u> , <u>123</u> , <u>124</u> , <u>125</u> , <u>126</u> , <u>127</u> , <u>128</u> , <u>129</u> , <u>130</u> , <u>131</u> , <u>132</u> , <u>133</u>
<code>\xpic@undef</code>	<u>70</u> , <u>133</u>
<code>\xpic@unknown@mode</code>	<u>21</u> , <u>146</u>
<code>\xpic@unknown@px</code>	<u>101</u> , <u>146</u>