Abstract

With \texttt{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 \LaTeX.

Contents

1 Documentation  
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 ...................... 6  
1.4 Other macros .......................................... 6  
1.5 Miscellaneous ....................................... 7  

2 Implementation  
2.1 Options ................................................. 8  
2.2 User macros ........................................... 9  
2.3 Parser ..................................................... 11  
2.4 Modes ...................................................... 12  
2.5 Pixel and Skip ........................................ 13  
2.6 Parser for colours ................................... 13  
2.7 Messages ............................................... 14  

Index  
\footnotesize *jspratte@yahoo.de
1 Documentation

1.1 Drawing pictures

\texttt{\textbackslash pxpic} supports different input modes, all of them have the same basic parsing behaviour. A \texttt{(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 \texttt{(pixel list)} elements are ignored. The different modes are explained in \texttt{subsubsection 1.2.2}. The only disallowed token in the \texttt{(pixel list)} is the control sequence \texttt{\textbackslash pxpic@end} (plus the usual restrictions of \TeX{}, so no unbalanced braces, no macros defined as \texttt{\out}).

There is a small caveat however: \texttt{\textbackslash pxpic} draws each pixel individually, and there is really no space between them, however some \texttt{pdf} viewers fail to display such adjacent lines correctly and leave small gaps (basically the same issue which packages like \texttt{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.

\begin{verbatim}
\texttt{\textbackslash pxpic}(\textit{options})\{\textit{pixel list}\}
\end{verbatim}

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

1.1.1 Examples

Since the above explanation of the \texttt{(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 \texttt{subsection 1.2} and \texttt{subsubsection 1.2.2} to fully understand the examples). Examples in this section will use the following \texttt{\pxpicsetup}:

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

We can draw a small cross rather easily:
\end{verbatim}

\begin{verbatim}
\texttt{\textbackslash pxpic} \{
  \texttt{.k}
  \texttt{kkk}
  \texttt{.k}
\}
\end{verbatim}
A small multicoloured grid:

```
\pxpic
\{{ brgk }\n{ kbrg }\n{ gkbr }\n{ rgkb }\n\}
```

A heart (shamelessly copied example from PixelArt):

```
\pxpic
\{ \[. \[. \rr \[. \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr \rr
```

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} {0,1,1} } } }
```

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.

```
\pxpic \size=2.5 pt, colours = {:=orange, %=black !10, g=black !75, O =orange !80! black}
\{
{{ . . . . . . . . . . . gggg } }
{{ . . . . . . . . . gggkkggg } }
{{ . . . . . . . . ggkkkkkkkg } }
{{ . . . . . . . . gkkkkkkkkg } }
{{ . . . . . . . . kkkkkkkkkk } }
{{ . . . . . . . . gkk . kkkk . kkk } }
{{ . . . . . . . . gk . k . kk . k . kk } }
{{ . . . . . . . gkkkkkkkkkkk } }
{{ . . . . . . . gkkk : : : : kkkk } }
{{ . . . . . . . gk . : kkkk : kkk } }
{{ . . . . . . . gkk . . : : : : kkkk } }
{{ . . . . . . . gkk . . . . ' ' ' kkkk } }
{{ . . . . . . . gkk . . . . . ' ' ' ' kkkk } }
{{ . . . . . . . gk . . . . . . . ' ' ' kkkk } }
{{ . . . . . . . gkk . . . . . . . . ' ' kkk } }
{{ . . . . . . . gkkg . . . . . . . . . ' ' kkk } }
{{ . . . . . . . gkkkk . . . . . . . . . ' ' kkk } }
{{ . . . . . . . OOOOkkk. . . . . . . ' ' ' kkkk } }
{{ . . . . . . . . . . . . . . O O: :O. kkkkkk .O: :O O } }
{{ . . . . . . . . . . . . . . . . . . O: :O O O O O } }
\}
```

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:

```latex
\begin{verbatim}
\pxpic colours = {orange, yellow, black},
skip = \skp
\end{verbatim}
```

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 `expkv`. Options can be set either in the optional argument of `\pxpic` or with `\pxpicsetup{⟨options⟩}`

Sets the ⟨options⟩ locally to the current Tex group.

Package options are not supported.

The available options are:

- **colors=⟨color list⟩**
  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.

- **colours**
  see `colors`.

- **color-list=⟨choice⟩**
  loads a previously through \pxpicnewcolorlist defined colour list. No colour lists are defined by the package.

- **colour-list**
  see `color-list`.

- **gap-hack=⟨dimen⟩**
  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 \hbox. Take a look at my babbling about this issue in subsection 1.5.

- **ht=⟨dimen⟩**
  Set the height of the pixels.

- **mode=⟨choice⟩**
  Set the used mode, see subsubsection 1.2.2 for available modes. Initial value is px.
size=(dimen)
Set both ht and wd. Initial value is 1.0pt.

skip=(dimen)
Define the value to be a skip (an empty space of width wd) in mode=px. No skip definitions are made by the package.

wd=(dimen)
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:

\texttt{colours = \{r=red, x=\{HTML\}{abab0f}\}}

1.2.2 Available modes

\texttt{px}
As already mentioned, \texttt{pxpic} supports different modes of input. The easiest to use mode is px, in which each element of the \texttt{⟨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 \texttt{detokenized}, so (within \TeX's limitations) the name of a pixel can be arbitrary. This is the initial mode \texttt{pxpic} uses. But other options are available as well.

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

\texttt{rgb, cmy, cmyk, hab, Hsb, tHsb, gray, RGB, HTML, HSB, Gray, wave}
The modes rgb, cmy, cmyk, hab, 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 \texttt{⟨pixel list⟩} will be the values in these colour models, so they'll be used like so: \{\texttt{\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 \texttt{mode} option using the macros \texttt{\pxpicnewmode} or \texttt{\pxpicsetmode}.
1.3 Other customisation macros

\texttt{\textbackslash pxpicnewmode}\langle\texttt{name}\rangle\{\texttt{definition}\}\}
\texttt{\textbackslash pxpicsetmode}

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

\texttt{\textbackslash pxpicnewcolorlist}\langle\texttt{name}\rangle\{\texttt{colour list}\}\}
\texttt{\textbackslash pxpicsetcolorlist}
\texttt{\textbackslash pxpicaddcolorlist}

This defines a colour list (to be used with the \texttt{\textbackslash colour-list} option). The syntax of \texttt{\langle colour list\rangle} is the same as for the \texttt{\textbackslash colours} option. The pixels aren't directly defined, but only by the use of \texttt{\textbackslash colour-list=\langle name\rangle}. So
\texttt{\textbackslash pxpicnewcolorlist\langle example\rangle\{r=red,b=blue,g=green,k=black,w=white\}}
\texttt{\textbackslash pxpicsetup\langle\texttt{\textbackslash colour-list=example}\rangle\}}

would have the same effect as
\texttt{\textbackslash pxpicsetup\langle\texttt{\textbackslash colours=\{r=red,b=blue,g=green,k=black,w=white\}\}}

but a \texttt{\textbackslash colour-list} is more efficient if used multiple times. The \texttt{\textbackslash new} variant will only throw an error if the colour list \texttt{(name)} is already defined. The \texttt{\textbackslash set} variant has no such tests, and the \texttt{\textbackslash add} variant will add additional colours to an existing list.

\texttt{\textbackslash pxpicforget}\langle\texttt{px}\rangle\}
\texttt{\textbackslash pxpicforget\langle\texttt{px}\rangle}\}

Undeﬁnes the \texttt{(px)} deﬁnition for use in \texttt{\textbackslash mode=px} (or skip symbol) added with the \texttt{\textbackslash colours} (or \texttt{\textbackslash skip}) option.

1.4 Other macros

\texttt{\textbackslash px}
\texttt{\textbackslash pxskip}

Inside of a \texttt{\textbackslash pxpic} the macro \texttt{\textbackslash px} draws a pixel (of the currently active colour), and \texttt{\textbackslash pxskip} leaves out a pixel (so this one pixel is fully transparent). Use this in the \texttt{\langle definition\rangle} of a mode in \texttt{\textbackslash pxpicnewmode}.

\texttt{\textbackslash pxpicHT}
\texttt{\textbackslash pxpicWD}

These two are \texttt{\textbackslash dimen} registers storing the height and width of the pixels.

\texttt{\textbackslash pxpiclogo}\langle\texttt{\langle size\rangle}\rangle
\texttt{\textbackslash pxpiclogo\langle\texttt{\langle size\rangle}\rangle}\}

This draws the logo of \texttt{\textbackslash pxpic}. The \texttt{(size)} 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 PixieArt, which, as of writing this, is described as a “working draft” by its author. PixieArt wasn’t intended as a direct competitor (I already started coding PixieArt when I learned about PixiArt’s existence), but I took inspiration from the “Bugs, Ideas, Undefined behaviours” section of PixiArt’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. PixieArt has the gap-hack option to provide some crude hack that might fix the issue, at the cost that the pixels on the far right and bottom are bigger than they were specified to be. 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. Here are examples in which you can compare (the gap-hack is chosen way too big in this example and skips are used close to white pixels on purpose, but it illustrates the effects; the third output, not shown in the code, uses a more reasonable gap-hack=.2pt):

\pxpicsetup  
{  
  colours={k=black, g=green, w=white}  
  ,skip=.
  ,size=10pt  
}  
\pxpic{  
  {kkkkk}  
  {kgggk}  
  {kg. k}  
  {kg . gk}  
  {kgwk}  
  {kkkw}  
}  
raise2pt\pxpic  
{  
  {kkkkk}  
  {kgggk}  
  {kg. k}  
  {kg . gk}  
  {kgwk}  
  {kkkw}  
}  
\pxpic{gap−hack=2pt}  
{  
  {kkkkk}  
  {kgggk}  
  {kg. k}  
  {kg . gk}  
  {kgwk}  
  {kkkw}  
}  
raise2pt\pxpic  
{  
  {kkkkk}  
  {kgggk}  
  {kg. k}  
  {kg . gk}  
  {kgwk}  
  {kkkw}  
}
2 Implementation

Report who we are
\ProvidesPackage{pxpic}[2021-01-16 v1.2 draw pixel pictures]
and load dependencies
\RequirePackage{xcolor}
\RequirePackage{expkv}
\pxpicHT
These two variables store the height and width of a pixel.
\@ifdefinable\pxpicHT{\newdimen\pxpicHT}
\@ifdefinable\pxpicWD{\newdimen\pxpicWD}
\pxpicHT1pt
\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.
\@ifdefinable\pxpic@kern{\newdimen\pxpic@kern}
\pxpic@kern\z@
(End definition for \pxpic@kern.)

2.1 Options

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

The first few options are straight forward. We use \texttt{expkv}'s name space to actually store the skip and px definitions, hence we use \texttt{ekvdefNoVal} in the code of \texttt{skip}.
\protected\ekvdef{pxpic}{size}{\pxpicHT\dimexpr#1\relax\pxpicWD\dimexpr#1\relax}
\protected\ekvdef{pxpic}{ht}{\pxpicHT\dimexpr#1\relax}
\protected\ekvdef{pxpic}{wd}{\pxpicWD\dimexpr#1\relax}
\protected\ekvdef{pxpic}{gap-hack}{\pxpic@kern\dimexpr#1\relax}
\protected\ekvdef{pxpic}{skip}{\ekvdefNoVal{pxpic@px}{#1}{\pxskip}}
The colours option is parsed using \texttt{ekvparse} and \texttt{pxpic@setcolor}.
\protected\ekvdef{pxpic}{colors}{\ekvparse\pxpic@err@noval\pxpic@setcolor{#1}}
\ekvletkv{pxpic}{colours}{pxpic}{colors}
And the mode just checks whether the mode macro is defined and lets the auxiliary macro \texttt{pxpic@parse@px} to the defined mode.
\protected\ekvdef{pxpic}{mode}{
\%\@ifundefined{pxpic@parse@px@#1}{\%\pxpic@err@unknown@mode{#1}}\%\%
\expandafter\let\expandafter\pxpic@parse@px
\csname pxpic@parse@px@#1\endcsname
\%}
}
A similar check is done for the colour-list option.

\protected\ekvdef{pxpic}{color-list}
{\@ifundefined{pxpic@colorlist@#1}\{\pxpic@err@unknown@colorlist(#1)\}\{\csname pxpic@colorlist@#1@endcsname\}}
\ekvletkv{pxpic}{colour-list}{pxpic}{color-list}

\section{User macros}
\texttt{\pxpic} expands directly to an opened \hbox, the auxiliary \texttt{\pxpic@} checks for the optional argument and inserts the rest of the code. We need to set \baselineskip to \pxpicHT so that the pixels are stacked vertically without gaps. \texttt{\pxpic@parse} will parse the \langle pixel list \rangle until \texttt{\pxpic@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 \texttt{\pxpic@parse} so that the first line isn’t moved.

\@ifdefinable\pxpic{\protected\def\pxpic{\hbox{\bgroup\pxpic@}}}
\newcommand\pxpic@[2][{}]{\vbox{\pxpicsetup{#1} \let\px\pxpic@px \let\pxskip\pxpic@skip \advance\pxpicHT\pxpic@kern \advance\pxpicWD\pxpic@kern \baselineskip\pxpicHT \kern\pxpic@kern \pxpic@end}}
(End definition for \pxpic and \pxpic@. These functions are documented on page 2.)

\texttt{\pxpicsetup} Just directly defined to call \texttt{exPkv}'s parser for the \texttt{\pxpic} set.
\ekvsetdef{pxpicsetup}{pxpic}
(End definition for \texttt{\pxpicsetup}. This function is documented on page 4.)

\texttt{\pxpiclogo} The logo is just a biggish pixel picture. The \texttt{\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.
\newcommand*{\pxpiclogo}[1][.13ex]{\begingroup \pxpicHT\dimexpr#1\relax \pxpicWD\pxpicHT \pxpic@kern#2 \leavevmode \lower3.2\pxpicHT\pxpic}
\pxpicforget  Straight forward, just let the px macro to an undefined macro.
\begin{verbatim}
\newcommand\pxpicforget[1]{\expandafter\let\csname pxpic@px@#1\endcsname\pxpic@undef}
\end{verbatim}

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

\pxpicnewmode \pxpicsetmode  These are pretty simple as well, the new variant will use \newcommand which will do the testing for us, the set variant uses \def.
\begin{verbatim}
\protected\long\def\pxpicnewmode#1\@\#2\{%\@ifundefined{pxpic@parse@px@#1\@\#1}{\pxpicsetmode{#1}}{\pxpic@err@defined@colorlist{#1}\@gobble}\}
\end{verbatim}

(End definition for \pxpicnewmode and \pxpicsetmode. These functions are documented on page 6.)

\pxpicnewcolorlist \pxpicsetcolorlist \pxpicaddcolorlist \pxpicsetcolorlist@ \pxpicaddcolorlist@  The colour list is first parsed with \ekvparse inside an \def. \ekvparse will prevent the parsed list from further expanding, leaving each list element and \pxpic@experr@noval or \pxpic@setcolor@colorlist before it. In a second \def these will be expanded, \pxpic@experr@noval throwing an error for each element missing a colour definition, and \pxpic@setcolor@colorlist testing for an opening bracket (which we do expandably) and leaving the correct definition protected against further expansion. The add variant uses a temporary macro for the parsing part and adds the result to the list holding macro. The second expansion step in set and both in add are done inside a group to revert any definition (also those letting tokens to \relax by \csname) made at this point except for the list macro itself.
\begin{verbatim}
\protected\def\pxpicnewcolorlist#1\@\{\%
\@ifundefined{pxpic@colorlist@#1\@}{\pxpicsetcolorlist@\@\#1\@}{\pxpic@err@defined@colorlist@\@\#1\@}\@gobble\}
\end{verbatim}

10
\protected\def\pxpicsetcolorlist#1\%
{\expandafter\pxpicsetcolorlist@\csname pxpic@colorlist@#1\endcsname}

\protected\long\def\pxpicsetcolorlist@#1#2\%
{\edef#1{\ekvparse\pxpic@experr@noval\pxpic@setcolor@colorlist{#2}}% 
  \begingroup\edef#1{\endgroup\protected\def\unexpanded{#1}{#1}}%
  #1%}
\protected\def\pxpicaddcolorlist#1\%
{\@ifundefined{pxpic@colorlist@#1}{\pxpic@err@unknown@colorlist{#1}\@gobble}{\expandafter\pxpicaddcolorlist@\csname pxpic@colorlist@#1\endcsname}}

\protected\long\def\pxpicaddcolorlist@#1#2\%
{\begingroup
  \edef\pxpic@tmp{\ekvparse\pxpic@experr@noval\pxpic@setcolor@colorlist{#2}}%
  \edef\pxpic@tmp{\endgroup\protected\def\unexpanded{#1}{\unexpanded\expandafter{#1}\pxpic@tmp}}%
}\pxpic@tmp

(End definition for \pxpicnewcolorlist and others. These functions are documented on page 6.)

2.3 Parser

\pxpic@ifend\pxpic@ifempty\pxpic@ifbracket These are three helper macros. The first just gobbles everything until the next \pxpic@end, and we borrow a fast test for an empty argument from \expkv. The last can be used to check for an opening bracket if used like \pxpic@ifbracket\pxpic@end #1.\pxpic@end[\pxpic@end.
\long\def\pxpic@ifend#1\pxpic@end{}
\let\pxpic@ifempty\ekv@ifempty
\long\def\pxpic@ifbracket#1\pxpic@end[#2]\pxpic@end{\pxpic@ifempty[#2]}

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

\pxpic@openbrace For some weirder \TeX programming it is sometimes necessary to insert an unmatched opening brace. This code does exactly that if it’s expanded twice. It is put into a single macro so that one can \expandafter it easier.
\newcommand*{\pxpic@openbrace{\expandafter{\iffalse}\fi}}

(End definition for \pxpic@openbrace.)

\pxpic@parse\pxpic@done 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 \pxpic@) 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 \pxpic@parseline.
The line parsing loop also checks whether we’re done, if not we place a pixel using the current definition of \pxpic@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 \pxpic@parse. Before each pixel we introduce a negative \kern to maybe fix the gap issue by letting the pixels overlap a bit.

2.4 Modes

The modes define how a single element of the ⟨pixel list⟩ is parsed.

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

\pxpic@parse@px@px

\pxpic@parse@px@named named just checks whether the skip is empty. If so skip, else call \color with the element and output a pixel.
The colour model modes are all the same in principle. They test for an empty element to introduce a skip, else they call \color with the respective colour model and output a pixel. We use the auxiliary \pxpic@tmp to do all those definitions and undefine it afterwards.

```latex
\def\pxpic@tmp#1{%
  \pxpicnewmode{#1}{%}
  \% \pxpic@ifempty{##1}{\pxskip}{{\@undeclaredcolor[#1]{##1}\px}}%
}%
\pxpic@tmp{rgb}
\pxpic@tmp{cmy}
\pxpic@tmp{cmyk}
\pxpic@tmp{hsb}
\pxpic@tmp{Hsb}
\pxpic@tmp{tHsb}
\pxpic@tmp{gray}
\pxpic@tmp{RGB}
\pxpic@tmp{HTML}
\pxpic@tmp{Gray}
\pxpic@tmp{wave}
\let\pxpic@tmp\pxpic@undef
```

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

### 2.5 Pixel and Skip

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

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

### 2.6 Parser for colours

```latex
\newcommand\pxpic@setcolor[2]{
  \if\catcode`\[=12
    \pxpic@setcolor@b{\>[0\xcolor]#1}{\>[0\xcolor]#2}\
  \else
    \pxpic@setcolor@a{\>[0\xcolor]#1}{\>[0\xcolor]#2}\
  \fi
}
```

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

```latex
\newcommand\pxpic@setcolor[2]{
  \if\catcode`\[=12
    \pxpic@setcolor@b{\>[0\xcolor]#1}{\>[0\xcolor]#2}\
  \else
    \pxpic@setcolor@a{\>[0\xcolor]#1}{\>[0\xcolor]#2}\
  \fi
}
```

\newcommand\pxpic@setcolor[2]{
  \if\catcode`\[=12
    \pxpic@setcolor@b{\>[0\xcolor]#1}{\>[0\xcolor]#2}\
  \else
    \pxpic@setcolor@a{\>[0\xcolor]#1}{\>[0\xcolor]#2}\
  \fi
}
This macro should leave the correct code in the input stream to define a single pixel. It is to be used inside of \edef, hence using \unexpanded, which doesn’t have an opening brace directly after it so that the \pxpic@ifbracket test is fully expanded. Next we expand \pxpic@setcolor@a/b twice (which will expand the \csname contained in it) and then leave the opening bracket for \unexpanded in the input stream. The code should be used inside a group so that all the implicit definitions to \relax done by \csname are reverted.

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

\pxpic@setcolor@colorlist This macro should leave the correct code in the input stream to define a single pixel. It is to be used inside of \edef, hence using \unexpanded, which doesn’t have an opening brace directly after it so that the \pxpic@ifbracket test is fully expanded. Next we expand \pxpic@setcolor@a/b twice (which will expand the \csname contained in it) and then leave the opening bracket for \unexpanded in the input stream. The code should be used inside a group so that all the implicit definitions to \relax done by \csname are reverted.

(End definition for \pxpic@setcolor@colorlist.)

2.7 Messages
These are just some macros throwing errors, nothing special here.

(End definition for \pxpic@err@noval and others.)
\textbf{\texttt{pxpic@experr}} This macro can be used to throw an error expandably. For this an undefined control sequence \texttt{pxpic\_Error} is used. The group containing \texttt{expandafter} keeps the definition of \texttt{pxpic\_Error}: local (it is \texttt{\relax} after the \texttt{\csname}) so that it is undefined when it's used. The \texttt{\@firstofone} is needed to get the readable output (now the undefined macro and actual message are always the same argument).

\begin{verbatim}
def\pxpic@experr#1\%
  {\long\def\pxpic@experr##1\%
    {\expandafter\expandafter\expandafter\pxpic@ifend\@firstofone{#1##1}\
      \pxpic@end}}
\begingroup\expandafter\endgroup\expandafter\pxpic@experr\csname pxpic Error:\endcsname
\end{verbatim}

(End definition for \texttt{\pxpic@experr}.)

\textbf{\texttt{pxpic@experr@noval}} With the expandable error throwing mechanism out of the way, the following is straightforward again.

\begin{verbatim}
\newcommand\pxpic@experr@noval[1]{\pxpic@experr{Missing colour definition for '#1'}}
\end{verbatim}

(End definition for \texttt{\pxpic@experr@noval}.)
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
\texttt{\textbackslash px} ........................... 6, 40, 153, 161, 188, 193
\texttt{\textbackslash pxpic} ........................... 2, 34, 58
\texttt{\textbackslash pxpicaddcolorlist} ........................... 6, 85
\texttt{\textbackslash pxpicforget} ........................... 6, 79
\texttt{\textbackslash pxpicHT} ........................... 4, 6, 11, 12, 42, 44, 54, 55, 58, 177
\texttt{\textbackslash pxpiclogo} ........................... 6, 51
\texttt{\textbackslash pxpicnewcolorlist} ........................... 6, 85
\texttt{\textbackslash pxpicnewmode} ........................... 6, 81, 157
\texttt{\textbackslash pxpicsetcolorlist} ........................... 6, 85
\texttt{\textbackslash pxpicsetmode} ........................... 6, 81
\texttt{\textbackslash pxpicsetup} ........................... 4, 39, 50
\texttt{\textbackslash pxpicWD} ........................... 4, 6, 11, 13, 43, 55, 177, 178
\texttt{\textbackslash pxskip} ........................... 6, 15, 41, 145, 152, 160
\texttt{\textbackslash pxpic\textbackslash linedone} ........................... 129
\texttt{\textbackslash pxpic\textbackslash openbrace} ........................... 120, 199, 200
\texttt{\textbackslash pxpic\textbackslash parse} ........................... 46, 121
\texttt{\textbackslash pxpic\textbackslash parse\textbackslash px} ........................... 23, 133, 137, 139
\texttt{\textbackslash pxpic\textbackslash parse\textbackslash px\textbackslash cmy} ........................... 155
\texttt{\textbackslash pxpic\textbackslash parse\textbackslash px\textbackslash cmyk} ........................... 155
\texttt{\textbackslash pxpic\textbackslash parse\textbackslash px\textbackslash Gray} ........................... 155
\texttt{\textbackslash pxpic\textbackslash parse\textbackslash px\textbackslash gray} ........................... 155
\texttt{\textbackslash pxpic\textbackslash parse\textbackslash px\textbackslash HSB} ........................... 155
\texttt{\textbackslash pxpic\textbackslash parse\textbackslash px\textbackslash HSB} ........................... 155
\texttt{\textbackslash pxpic\textbackslash parse\textbackslash px\textbackslash HTML} ........................... 155
\texttt{\textbackslash pxpic\textbackslash parse\textbackslash px\textbackslash named} ........................... 149
\texttt{\textbackslash pxpic\textbackslash parse\textbackslash px\textbackslash px} ........................... 139
\texttt{\textbackslash pxpic\textbackslash parse\textbackslash px\textbackslash RGB} ........................... 155
\texttt{\textbackslash pxpic\textbackslash parse\textbackslash px\textbackslash rgb} ........................... 155
\texttt{\textbackslash pxpic\textbackslash parse\textbackslash px\textbackslash t\textbackslash HSB} ........................... 155
\texttt{\textbackslash pxpic\textbackslash parse\textbackslash px\textbackslash t\textbackslash wave} ........................... 155
\texttt{\textbackslash pxpic\textbackslash parse\textbackslash px\textbackslash t\textbackslash line} ........................... 125, 129
\texttt{\textbackslash pxpic\textbackslash px} ........................... 40, 177
\texttt{\textbackslash pxpic\textbackslash setcolor} ........................... 16, 179
\texttt{\textbackslash pxpic\textbackslash setcolor\textbackslash a} ........................... 179, 199
\texttt{\textbackslash pxpic\textbackslash setcolor\textbackslash b} ........................... 179, 200
\texttt{\textbackslash pxpic\textbackslash setcolor\textbackslash colorlist} ........................... 95, 109, 195
\texttt{\textbackslash pxpic\textbackslash skip} ........................... 41, 177
\texttt{\textbackslash pxpic\textbackslash tmp} ........................... 108, 110, 113,
\hspace{1em}115, 155, 164, 165, 166, 167, 168,
\hspace{1em}169, 170, 171, 172, 173, 174, 175, 176
\texttt{\textbackslash pxpic\textbackslash undef} ........................... 80, 176
\texttt{\textbackslash pxpic\textbackslash addcolorlist\textbackslash a} ........................... 85
\texttt{\textbackslash pxpic\textbackslash setcolorlist\textbackslash a} ........................... 85