PixelArtTikz [en]
PixelArts, with TikZ,
with solution and colors.
Version 0.1.4 – 16/04/2024

Cédric Pierquet
c.pierquet – at – outlook . fr
https://github.com/cpierquet/PixelArtTikz

▶ Commands to display PixelArts.
▶ Environment to complete the PixelArt.

\begin{tikzpicture}

\begin{scope}[local bounding box=pixelart]
\node at (0,0) {\texttt{D\texttt{D}}};
\node at (1,0) {\texttt{D\texttt{A}}};
\node at (2,0) {\texttt{A\texttt{A}}};
\node at (3,0) {\texttt{A\texttt{D}}};
\node at (4,0) {\texttt{D\texttt{D}}};
\node at (0,1) {\texttt{D\texttt{A}}};
\node at (1,1) {\texttt{A\texttt{A}}};
\node at (2,1) {\texttt{A\texttt{A}}};
\node at (3,1) {\texttt{A\texttt{A}}};
\node at (4,1) {\texttt{A\texttt{D}}};
\node at (0,2) {\texttt{D\texttt{A}}};
\node at (1,2) {\texttt{A\texttt{A}}};
\node at (2,2) {\texttt{A\texttt{A}}};
\node at (3,2) {\texttt{A\texttt{A}}};
\node at (4,2) {\texttt{A\texttt{D}}};
\node at (0,3) {\texttt{D\texttt{A}}};
\node at (1,3) {\texttt{A\texttt{A}}};
\node at (2,3) {\texttt{A\texttt{A}}};
\node at (3,3) {\texttt{A\texttt{A}}};
\node at (4,3) {\texttt{A\texttt{D}}};
\node at (0,4) {\texttt{D\texttt{A}}};
\node at (1,4) {\texttt{A\texttt{A}}};
\node at (2,4) {\texttt{A\texttt{A}}};
\node at (3,4) {\texttt{A\texttt{A}}};
\node at (4,4) {\texttt{A\texttt{D}}};
\end{scope}
\end{tikzpicture}
# Contents

## I Introduction

1. The package PixelArtTikz
   1.1 Introduction ........................................... 3
   1.2 Loading of the package, and option .................. 3
   1.3 Used packages ........................................ 3
   1.4 Macros and environment ............................... 4

## II Macros and environment

3 Main macro
   3.1 Example ............................................... 5
   3.2 Options and keys ...................................... 6
   3.3 Starred macro ......................................... 9

4 PixelArt environment
   4.1 Usage .................................................. 10
   4.2 Example ............................................... 10

5 Macro for *mini*-PixelArt
   5.1 Idea .................................................. 11
   5.2 Examples .............................................. 11

6 Macro for cutting PixelArt
   6.1 Idea .................................................. 12
   6.2 Global usage ......................................... 12
   6.3 Example ............................................... 12

7 Anamorphic cylinder PixelArt
   7.1 Idea .................................................. 16
   7.2 Keys and options ...................................... 17
   7.3 Example with inverted data (Yoda) .................. 18
   7.4 Example with classic data (Witch) .................. 20

## III History

21
Part I
Introduction

1 The package PixelArtTikz

1.1 Introduction
The idea is to propose, within a TikZ environment, a macro to generate PixelArt. The data is read from a csv file, already existing in the folder of the tex file, or created on-the-fly by filecontents.

Some advices about the csv file :

- the csv file must use "," as separator;
- empty cells are coded by "-".

\begin{filecontents*}{filename.csv}
A,B,C,D
A,B,D,C
B,A,C,D
B,A,D,C
\end{filecontents*}

While compiling, the file \texttt{filename.csv} will be created, and the option \texttt{[overwrite]} will propagate the modifications!

1.2 Loading of the package, and option
The package \texttt{csvsimple} is necessary in order to read the csv file. The package is available in two versions, one written in \LaTeX\ and the other in \LaTeX3. By default, PixelArtTikz loads the \LaTeX3 version, but an option is available to work with the \LaTeX2\ version.

The option \texttt{[csvii]} forces the usage of the \LaTeX2\ version.

\begin{verbatim}
\usepackage[PixelArtTikz]{\texttt{csvii}} %package latex2
\% which loads
\%
\end{verbatim}

1.3 Used packages
It’s fully compatible with usual \LaTeX\ engines, such as \texttt{latex}, \texttt{pdflatex}, \texttt{lualatex} or \texttt{xelatex}.

It loads the following packages and libraries:

- \texttt{tikz}, \texttt{xintexpr} et \texttt{xinttools};
- \texttt{xstring}, \texttt{xparse}, \texttt{simplekv} and \texttt{listofitems}. 
1.4 Macros and environment

There are two ways to create PixelArt:

- with an independent macro;
- with a TikZ environment in order to add code afterwards.

```latex
%Independent macro
\PixlArtTikz[\texttt{keys}][\texttt{options tikz}]{file.csv}

%Semi-independent macro, in a tiks environment
\PixlArtTikz*[\texttt{keys}]{file.csv}

%environment
\begin{EnvPixlArtTikz}[\texttt{keys}][\texttt{options tikz}]{file.csv}
\begin{tikz code}
\end{EnvPixlArtTikz}
```

2 Colors

Concerning colors: the user can use all colors provided by loaded packages!

Without extra packages, the available colors are:

<table>
<thead>
<tr>
<th>magenta</th>
<th>cyan</th>
<th>blue</th>
<th>green</th>
<th>red</th>
<th>darkgray</th>
<th>olive</th>
<th>lime</th>
<th>brown</th>
<th>lightgray</th>
</tr>
</thead>
<tbody>
<tr>
<td>white</td>
<td>gray</td>
<td>black</td>
<td>yellow</td>
<td>violet</td>
<td>teal</td>
<td>purple</td>
<td>pink</td>
<td>orange</td>
<td></td>
</tr>
</tbody>
</table>
Part II
Macros and environment

3 Main macro

3.1 Example

The macro \PixlArtTikz needs:

- the file `csv`;
- the list (by a string) of codes used in the file `csv` (e.g. 234679 or ABCDK...);
- the list of symbols (if needed) to print in the cells, e.g. 25,44,12 or AA,AB,AC;
- the list of colors (for the correction), same order as the codes.

We can begin by creating the file `csv`, directly within the tex code, or with a external file.

\begin{filecontents*}[overwrite]{base.csv}
A,B,C,D
A,B,D,C
B,A,D,C
C,A,B,D
\end{filecontents*}

\begin{center}
\begin{tblr}{colspec={*{4}{Q[1.25cm,c,m]}},hlines,vlines,rows={1.15em}}
\SetCell[c=4]{c} Instructions & & & \\
A & B & C & D \\
45 & 22 & 1 & 7 \\
Black & Green & Yellow & Red \\
\end{tblr}
\end{center}

\PixlArtTikz[Codes=ABCD,Style={\large\sffamily},Unit=0.85]{base.csv}

\PixlArtTikz[Codes=ABCD,Symbols={45,22,1,7},Symb,Style={\large\sffamily},Unit=0.85]{base.csv}

\PixlArtTikz[Codes=ABCD,Colors={black,green,yellow,red},Correction,Unit=0.85]{base.csv}

\PixlArtTikz[Codes=ABCD,Colors={black,green,yellow,red},Correction,Border=false,Unit=0.85]{base.csv}
3.2 Options and keys

The first argument, *optional* and between \[ \ldots \] proposes the keys:

- the key *(Codes)* with the *string* of *simple* codes of the *csv* file;
- the key *(Colors)* with the *list* of colors;
- the key *(Symbols)* with the *optional list* of alt. symbols for the cells;
- the boolean *(Correction)* to color the PixelArt; default *false*
- the boolean *(Symb)* to print the symbols; default *false*
- the boolean *(Border)* to print borders of the cells; default *true*
- the key *(Style)* to specify the style of the text. default \scriptsize

The second argument, *optional* and between <...>, are TikZ options to pass on to the environment which creates the PixelArt.

The third argument, *mandatory*, is the filename of the *csv*.

%creation of the csv
\begin{filecontents*}[overwrite]{test1.csv}
-,-,-,-,-,-,4,4,4,4,,-,-,-,-,-
-,-,-,-,4,4,1,1,1,4,4,,-,-,-,-
-,-,-,4,1,1,1,1,1,1,4,,-,-,-,-
-,-,4,1,1,1,1,1,1,1,4,,-,-,-,-
-,-,4,1,9,1,1,1,1,9,9,1,4,-,-
-,-,4,9,9,9,4,4,4,9,9,9,4,-,-
-,-,4,9,4,9,4,4,4,9,9,9,4,-,-
-,-,4,1,9,9,4,4,4,9,9,1,4,-,-
-,-,4,1,1,9,4,4,4,9,1,1,4,-,-
-,-,4,1,1,1,4,4,4,1,1,1,4,-,-
-,-,4,1,1,1,4,1,1,1,1,4,-,-,-
-,-,4,3,1,1,1,1,1,3,4,-,-,-,-
-,-,4,6,3,1,1,1,1,3,6,4,-,-,-
-,-,4,6,1,1,1,1,1,6,6,4,-,-,-
-,-,4,6,1,1,1,1,1,1,6,6,4,-,-
-,-,4,6,4,1,1,4,4,1,1,4,6,4,-
-,-,2,2,4,2,4,4,2,4,2,4,2,2
-,-,2,2,2,2,2,4,2,2,2,2,2,2
-,-,2,2,2,2,2,2,2,2,2,2,2,2
-,-,-,-,-,-,4,1,1,1,4,-,-,-,-,-
-,-,-,-,-,-,4,1,1,4,-,-,-,-,-
-,-,-,-,-,-,4,4,-,-,-,-,-,-,-
\end{filecontents*}
% simple codes
% empty case with -
\PixlArtTikz[Codes=123469,Style=\ttfamily,Unit=0.35]{test1.csv}
--
\PixlArtTikz[Codes=123469,Colors={red,brown,yellow,black,blue,white},Correction,Unit=0.35]{test1.csv}
--
\PixlArtTikz[Codes=123469,Colors={red,brown,yellow,black,blue,white},Correction,Unit=0.35,Border=false]{test1.csv}
In the following example, the *symbols* to print can’t be used for the *codes*, so we can use the keys *(Symbols)* and *(Symb)* to bypass this limitation.
3.3 Starred macro

The starred macro `\PixlArtTikz*` is to be used within an already created environment. It can be useful for adding code after the PixelArt.

In this case:

- the optional argument between `<...>` is discarded;
- the key `<Unit>` is discarded too (units can be configured in the environment!)

\begin{center}
\begin{tikzpicture}[scale=0.5]
%grid to show positioning
\draw[very thin,gray,xstep=1,ystep=1] (0,0) grid (17,-24);
\foreach \x in {0,1,...,17} \draw[very thin,gray] (\x,-3pt)--(\x,3pt) node[above,font=\textsize\sffamily] {\x};
\foreach \y in {0,-1,...,-24} \draw[very thin,gray] (3pt,\y)--(-3pt,\y) node[left,font=\textsize\sffamily] {\y};
%the PixelArt
\PixlArtTikz*[Codes=123469,Colors={red,brown,yellow,black,blue,white},Correction]{test1.csv}
%added code
\filldraw[blue] (14,-1) circle[radius=1];
\filldraw[yellow] (14,-1) circle[radius=0.8];
\draw[green,very thick,<-,>=latex] (15,-1) to[bend left=30] (18,-2) node[right,font=\textsize\sffamily] {Code Ti\textit{k}Z};
\end{tikzpicture}
\end{center}
4 PixelArt environment

4.1 Usage
The package PixelArtTikz provides an environment to create a PixelArt and add code afterwards.

- The environment is created within TikZ and additional code is passed on to the TikZ environment!
- The additional code will be printed on top of the PixelArt!

\begin{EnvPixlArtTikz}\[\text{keys\}<\text{options tikz}\}{\text{filename.csv}}\]
%tikz code(s)
\end{EnvPixlArtTikz}

The first argument, \textit{optional} and between \texttt{[\ldots]}\texttt{],} proposes the keys:

- the key \texttt{(Codes)} with the \textit{string} of \textit{simple} codes of the \texttt{csv} file;
- the key \texttt{(Colors)} with the \textit{list} of colors;
- the key \texttt{(Symbols)} with the \textit{optional list} of alt. symbols for the cells;
- the boolean \texttt{(Correction)} to color the PixelArt; \texttt{default false}
- the boolean \texttt{(Symb)} to print the symbols; \texttt{default false}
- the boolean \texttt{(Border)} to print borders of the cells; \texttt{default true}
- the key \texttt{(Style)} to specify the style of the text. \texttt{default \scriptsize}

The second argument, \textit{optional} and between \texttt{<\ldots>}, is for TikZ options to be passed on to the environment which creates the PixelArt.

The third argument, \textit{mandatory}, is the filename of the \texttt{csv}.

4.2 Example
The symbols are at the nodes \((c; -l)\) where \(l\) and \(c\) are the row and column of the data in the \texttt{csv} file.

\begin{center}
\begin{EnvPixlArtTikz}
[\text{Codes=123469,Colors=\{red,brown,yellow,black,blue,white\},Correction,Unit=0.25}]
\texttt{\{test1.csv\}}
\filldraw[\texttt{blue}] (14,-1) circle[\texttt{radius=1}] ;
\filldraw[\texttt{yellow}] (14,-1) circle[\texttt{radius=0.8}] ;% \
\draw[\texttt{green,very thick,\texttt{<,-}>latex}] (15,-1) to[\texttt{bend left=30}] (18,-2)\% %
\texttt{node[right,font=\texttt{\scriptsize\texttt{sfamily}}} \{\texttt{\texttt{TikZ} code}\} ;
\end{EnvPixlArtTikz}
\end{center}
5 Macro for *mini*-PixelArt

5.1 Idea

The idea is to propose a macro to insert, without csv file, a small PixelArt with small colors list.

\MiniPixlArt[\text{name}]{\text{list of colors}}

The first argument, \textit{optional} and between [...] proposes the \textbf{keys}:

- the key \textit{(Unit)} for dimension of the cells; default 0.25em,
- the boolean \textit{(Border)} to print a small border for the cells. default false

The second argument, \textit{mandatory} and between {...}, is the colors of the cells:

- each color is \textit{coded} by a letter:
  - R : red
  - C : blue
  - B : black
  - . : white
  - O : orange
  - G : green
  - Y : yellow
  - L : gray
  - M : maroon
  - P : purple
  - each linebreak is done by , ;
  - the thickness of the borders are 10\% of the unit.

The last argument, \textit{optional} and between <...>, proposes options for the \texttt{tikz} environment.

5.2 Examples

\MiniPixlArt{%
...RR....RR..
.RRRRRRRRR.,
.RRRRRRRRR,
.RRRRRRRRRR,
.RRRRRRRRRR,
.RRRRRRRRRR,
.RRRRRRRRRR,
.RRRRRRRRRR.,
...RRRR..., 
...RRRR...,
....RR....
}%

Inline, we can give \MiniPixlArt[\text{Unit=5mm,Border}]{\text{BCGYOGR,YLP.BR}}<baseline=(current bounding box.center)> this miniPA.

Inline, we can give this miniPA.
6 Macro for cutting PixelArt

6.1 Idea

The idea is to offer commands to create collaborative PixelArts, to form a large image from several small ones (of the same size).

For practical reasons (related to the use of filecontents) the file csv must end with an empty line (it is created automatically via filecontents, and the code uses this specificity).

The available (keys) are exactly the same as those in classic displays.

6.2 Global usage

The argument cutting can be given in the form:

- \texttt{\langle number of block length\rangle x \langle number of block col\rangle} to specify the size of the blocks;
- \texttt{\langle number of V blocks\rangle + \langle number of H blocks\rangle} to specify the number of blocks.

\begin{verbatim}
\CutPixlArtTikz(*\[keys\]<tikz options>{file.csv}{cutting}
\%(*):= change layout formatting (A1 or 1.1)
\%1 := keys
\%2 := tikz options
\%3 := csv file
\%4 := block size (LxC) or block number (L+C)
\end{verbatim}

\begin{verbatim}
\HelpGridPixelArtTikz(*\[scale\]{file.csv}{cutting}
\%(*):= change layout formatting (A1 or 1.1)
\%1 := scale of notice
\%2 := csv file
\%4 := block size (LxC) or block number (L+C)
\end{verbatim}

6.3 Example

\begin{verbatim}
\def\listcoulkoala{%
\{VertForet,Noir,GrisClair,GrisFonce,Beige,Blanc,BleuClair,VertClair,Marron\}
\end{verbatim}

\begin{verbatim}
%block with 8x8 size
\HelpGridPixelArtTikz(PAkoala.csv){8x8}
\end{verbatim}

\begin{verbatim}
\PixlArtTikz[Correction,Unit=0.125,Codes=ABCDEFGHI,Colors={\listcoulkoala},Grid=8x8]{PAkoala.csv}
\end{verbatim}
\begin{table}
\centering
\begin{tabular}{|c|c|c|c|}
\hline
1.1 & 1.2 & 1.3 & 1.4 \\
\hline
2.1 & 2.2 & 2.3 & 2.4 \\
\hline
3.1 & 3.2 & 3.3 & 3.4 \\
\hline
4.1 & 4.2 & 4.3 & 4.4 \\
\hline
\end{tabular}
\end{table}

\begin{tikzpicture}[scale=0.5]
\draw[step=1,black,very thin] (0,0) grid (4,4);
\filldraw[fill=white,draw=black] (0,0) rectangle (4,4);
\end{tikzpicture}
<table>
<thead>
<tr>
<th>Sample</th>
<th>Sample</th>
<th>Sample</th>
<th>Sample</th>
<th>Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAAAA</td>
<td>HAAAAA</td>
<td>HAAAAA</td>
<td>AAAAAA</td>
<td>HAAAAA</td>
</tr>
<tr>
<td>AAAA</td>
<td>HAAAAA</td>
<td>HAAAAA</td>
<td>AAAA</td>
<td>HAAAAA</td>
</tr>
<tr>
<td>AAAA</td>
<td>HAAAAA</td>
<td>HAAAAA</td>
<td>AAAA</td>
<td>HAAAAA</td>
</tr>
<tr>
<td>AAAA</td>
<td>HAAAAA</td>
<td>HAAAAA</td>
<td>AAAA</td>
<td>HAAAAA</td>
</tr>
<tr>
<td>AAAA</td>
<td>HAAAAA</td>
<td>HAAAAA</td>
<td>AAAA</td>
<td>HAAAAA</td>
</tr>
<tr>
<td>AAAA</td>
<td>HAAAAA</td>
<td>HAAAAA</td>
<td>AAAA</td>
<td>HAAAAA</td>
</tr>
<tr>
<td>AAAA</td>
<td>HAAAAA</td>
<td>HAAAAA</td>
<td>AAAA</td>
<td>HAAAAA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sample A1</th>
<th>Sample A2</th>
<th>Sample A3</th>
<th>Sample A4</th>
<th>Sample A5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample B1</td>
<td>Sample B2</td>
<td>Sample B3</td>
<td>Sample B4</td>
<td>Sample B5</td>
</tr>
<tr>
<td>Sample C1</td>
<td>Sample C2</td>
<td>Sample C3</td>
<td>Sample C4</td>
<td>Sample C5</td>
</tr>
<tr>
<td>Sample D1</td>
<td>Sample D2</td>
<td>Sample D3</td>
<td>Sample D4</td>
<td>Sample D5</td>
</tr>
<tr>
<td>Sample E1</td>
<td>Sample E2</td>
<td>Sample E3</td>
<td>Sample E4</td>
<td>Sample E5</td>
</tr>
</tbody>
</table>
7 Anamorphic cylinder PixelArt

7.1 Idea

The idea is to provide something to create PixelArt with the aim of using a cylindrical anamorphosis. On https://www.youtube.com/watch?v=PT8KUozBgiI, there is a video demonstration, proposed by Jean-Yves Labouche. The overall operation is similar to that of the main command, however there are some adjustments:

- the possibility of giving the file csv in normal or inverted mode;
- the dimensions (width & middle) must be specified to produce the PixelArt;
- the command is standalone (for the moment) so no addition(s) later.

\PixlArtTikzCylindric\{file.csv\}

The files illustrating this paragraph are given below.

\begin{filecontents*}[overwrite]{PAyoda.csv}
E,E,E,A,A,E,A,E,A,E,D
E,E,E,F,F,F,F,E,D
E,E,E,F,C,F,F,E,D
E,A,F,C,C,F,F,F,A
E,A,F,C,C,F,F,F,A
E,E,E,A,A,A,A,E,E
E,E,E,A,A,A,A,E,E
E,E,A,E,B,A,B,E,A,E
E,E,E,A,A,A,A,E,E
\end{filecontents*}

\PixlArtTikz[%
Codes=ABCDEF,
Colors={green,black,brown,white,brown!75!black,lightgray},
Correction,Unit=0.35\%
]{PAyoda.csv}
7.2 Keys and options

The first argument, \textit{optional} and between \texttt{[...]} offers Keys necessary for the command to function properly:

\begin{itemize}
\item the key \texttt{Width} which defines the width (radius in cm) of the rendering; \hspace{1cm} default 6
\item the key \texttt{Center} which defines the width (radius in cm) of the \textit{middle}; \hspace{1cm} default 1.25
\item the key \texttt{Codes} contains the \textit{string} of the \textit{simple} codes of the file \texttt{csv};
\item the key \texttt{Colors} which contains the \textit{list} of associated colors;
\item the key \texttt{Symbols} which contains the \textit{possible list} of alternative characters to display in the boxes;
\item the key \texttt{Style} which allows you to specify the style of characters. \hspace{1cm} default \texttt{normalsize}
\item the Boolean key \texttt{Correction} which allows you to color the PixelArt; \hspace{1cm} default \texttt{false}
\item the Boolean key \texttt{Symb} which allows you to display the \textit{alternative} characters; \hspace{1cm} default \texttt{false}
\item the Boolean key \texttt{Solution} which displays the solution (with \textit{mirror} effect); \hspace{1cm} default \texttt{false}
\item the Boolean key \texttt{Swap} which allows you to specify the data type (\texttt{true} := normal; \texttt{false} := inverted). \hspace{1cm} default \texttt{false}
\end{itemize}

The second argument, \textit{mandatory}, is the name of the \texttt{csv} file to use.
7.3 Example with inverted data (Yoda)

In this paragraph, we use the data PAyoda, which corresponds to the inverted layout, so the key $\text{(Swap)}$ is not necessary.

```latex
%classic version
\PixlArtTikzCylindric[Codes=ABCDEF,Style=\small\ttfamily]{PAyoda.csv}

%version with 'symbols'
\PixlArtTikzCylindric[Codes=ABCDEF,Symbols={1,2,3,4,5,6},Symb,Style=\small\ttfamily]{PAyoda.csv}
```
\begin{tabular}{c}
\PixlArtTikzCylindric[\%
  Codes=ABCDEFGHIJKLMNOPQRSTUVWXYZ,
  Colors={green,black,brown,white,brown!75!black,lightgray},
  Solution]\%
  \{PAyoda.csv\}
\]
\PixlArtTikzCylindric[\%
  Codes=ABCDEFGHIJKLMNOPQRSTUVWXYZ,
  Colors={green,black,brown,white,brown!75!black,lightgray},
  Correction]\%
  \{PAyoda.csv\}
\end{tabular}
Example with classic data (Witch)

In this paragraph, we use the data \texttt{PAsorciere}, which corresponds to the normal layout, so the key \texttt{Swap} is necessary.

\begin{verbatim}
\%classic version
\PixlArtTikzCylindric[\
Width=4,Center=1,Codes=VNGOJ,\
Colors={purple!50,black,green,orange,yellow},\
Swap,Style=\ttfamily\scriptsize]\
\{PAsorciere.csv\}
\end{verbatim}

\begin{verbatim}
\%Correction and solution
\begin{tabular}{c}
\PixlArtTikzCylindric[\
Width=4,Center=0.75,Codes=VNGOJ,\
Colors={purple!50,black,green,orange,yellow},\
Swap,Solution]\
\{PAsorciere.csv\}
\\
\PixlArtTikzCylindric[\
Width=4,Center=0.75,Codes=VNGOJ,\
Colors={purple!50,black,green,orange,yellow},\
Swap,Correction]\
\{PAsorciere.csv\}
\end{tabular}
\end{verbatim}
Part III

History

v0.1.4: Anamorphic cylinder PixelArts
v0.1.3: Cut PixelArts within several PixelArts
v0.1.2: mini-PixelArts
v0.1.1: Bugfix with color
v0.1.0: Initial version