

PixelArt0*

A package to draw pixel-art pictures.

Louis Paternault
spalax(at)gresille(dot)org

February 20, 2023

Abstract

This package defines macros to draw pixel-art pictures using L^AT_EX.



Warning : This package is an outdated version of pixelart, that works with L^AT_EX (while pixelart requires LuaL^AT_EX). It is kept around for background compatibility.

- If you are a new user:
 - if you are using LuaL^AT_EX, you can use pixelart version 1.0.0 or later;
 - otherwise, you can use the pxpic package, by Jonathan P. Spratte.
- if you did use pixelart before version 1.0.0, you are advised to switch to pixelart version 1.0.0 or later, or to switch to pxpic (see above). Or you can replace your `\requirepackage{pixelart}` by `\requirepackage{pixelart0}` to continue using this outdated version.

Contents

1	Introduction	2
1.1	License	2
1.2	Overview	2
2	Download and Install	2
2.1	GNU/Linux Distribution	2
2.2	LaTeX distribution	2
2.3	Manual installation	3
3	Usage	3
3.1	Package options	3
3.2	Macros	3

*This document corresponds to pixelart0 0.3.0, dated 2022-11-16. Home page, bug requests, etc. at <http://framagit.org/spalax/pixelart>

4 Bugs, Ideas, Undefined behaviours	5
4.1 Missing <code>\pixelart</code> macro	5
4.2 It's insanely sloooooow.	5
4.3 Black and white	5
4.4 Spaces	6
4.5 Uneven lines	7
Change History	7
Index	7

1 Introduction

This document introduces the `pixelart0` package, used to draw pixel-art pictures.

It is an outdated version of `pixelart` (before version 1.0.0), kept as part of `pixelart` for backward compatibility.

1.1 License

This work may be distributed and/or modified under the conditions of the L^AT_EX Project Public License, either version 1.3 of this license or (at your option) any later version.

Further information can be found in the `.dtx` file used to build this document.

1.2 Overview

Installation instructions are given in section 2. Documentation about how to use this package (and examples) is given in section 3. Section 4 lists some known bugs and limitations.

2 Download and Install

2.1 Gnu/Linux Distribution

If applicable, the easiest way to get `pixelart0` working is by installing it by your distribution package. In Debian (and Ubuntu, and surely other distributions that inherit from Debian) it is packaged in `texlive-pictures` since version 2017.20180103-1. So you can install it by running:

```
sudo apt install texlive-pictures
```

2.2 LaTeX distribution

This package is included both in T_EX Live and MiK_TE_X, as part of `pixelart`. It can be installed using their respective package managers.

2.3 Manual installation

- Download the latest archive :

Stable version <https://mirrors.ctan.org/graphics/pgf/contrib/pixelart.zip>

Development version <https://framagit.org/spalax/pixelart/repository/archive.zip?ref=main>

- Unzip the archive.
- If you got the archive from CTAN (stable version), move file `tex/latex/pixelart/pixelart0.sty` in a \LaTeX path.
- If you got the development version, move the `pixelart0.sty` file into a \LaTeX path.

3 Usage

3.1 Package options

This package has a single package option: `draft`. If this option is set (`\usepackage[draft]{pixelart}`), pixel-art pictures are ignored. This can make compilation *way, way* faster¹.

A downside is that since pixel-art pictures are ignored, this can mess up your document layout. A nicer option would be to have option `draft` guess the pixel-art size, and display a dummy picture with the same size².

3.2 Macros

This package defines two macros : `\bwpixelart`³, used to insert a pixel-art picture, and `\tikzbwpixelart`, which has the same purpose, excepted that it is called from within a `tikzpicture` environment.

3.2.1 `\bwpixelart`

`\bwpixelart` To insert a pixel-art picture in your text, use :

$$\bwpixelart[\langle color, raise, scale \rangle]{\langle pixels \rangle}$$

Its optional arguments are:

color=black Foreground color (the background is transparent);

scale=1 Scale. By default, a pixel is the size of a `tikzpicture` default unit, which is probably bigger than what you want.


¹On a document I am writing, containing a lot of pixel-art pictures, option `draft` makes compilation time go from 6 minutes to 22 seconds.

²This has been implemented in `pixelart`.

³`\bwpixelart` stands for *black and white pixel art*, although *color and transparent pixel art* would be more accurate.

raise=0pt Raise the picture. By default, the bottom of the picture is on the baseline. You might want to lower it a little by giving this option a negative argument.

Its mandatory argument is the picture pixels, as 0's and 1's. Line breaks in this argument are interpreted as line breaks in the pixel art pictures. How spaces are interpreted is undefined (see section 4.4 for more information).

For instance, this heart  was drawn using the following code:

```

1 \bwpixelart[color=red, scale=.05, raise=-1ex]{%
2 001101100
3 011111110
4 111111111
5 111111111
6 111111111
7 011111110
8 001111100
9 000111000
10 000010000
11 }
```

3.2.2 `\tikzbpixelart`

`\tikzbpixelart` The second macro, `\tikzbpixelart` is almost identical to the first one, excepted that it is meant to be called from inside a `tikzpicture` environment. Actually, `\bwpixelart{0101}` is more or less equivalent to calling :

```


1 \begin{tikzpicture}
2   \tikzbpixelart{(0, 0)}{0101}
3 \end{tikzpicture}
```

The signature of this macro is :

$$\text{\tikzbpixelart}[\langle color, scale \rangle]{\langle coordinates \rangle}{\langle pixels \rangle}$$

Its optional arguments are `color` and `scale`, used to set the color and scale of the picture.

Its first mandatory argument is the coordinate of the top left corner of the picture; the second one is the list of pixels (using the same syntax as the `\bwpixelart` macro).

For instance, this heart  was drawn using the following code:

```

1 \begin{tikzpicture}[scale=.05, baseline=-1em]
2   \fill[red] (5, -4) circle (6.5);
3   \tikzbpixelart{(0, 0)}{%
4 0011001100
5 011111110
6 111111111
7 111111111
8 111111111
9 011111110
10 001111100
11 000111000
12 000010000
```

```

13 }
14 \end{tikzpicture}

```

4 Bugs, Ideas, Undefined behaviours

Note that most of the stuff in this section have been fixed in the version 1.0.0 of `pixelart`.

I have great ideas about what this package could do, but:

- I do not need them;
- I am not sure there is a huge *need* for some pixel-art package;
- I have a full-time job, my wife has a far-more-than-full-time job, my daughter *is* a full-time job⁴, so I have very little time to hack...

So, I am listing here some known bugs, undefined behaviours, limitations.

4.1 Missing `\pixelart` macro

There is no `\pixelart` macro. This is on purpose: given that this package is more or less a working draft, I did not want to register a badly designed `\pixelart` macro. This means that some folk wanting to improve this package can extend the `\bwpixelart` macro and use the name `\pixelart` to fix my design mistakes.

4.2 It's insanely sloooooow.

That's it. It takes almost 30 seconds to compile a document containing only a 128×128 picture (about 16 000 pixels). I have no idea how to fix it. Good luck.

4.3 Black and white

Right now, it is black and white only (or, to be more accurate, any single color on a transparent background).

One *could* produce colored pixel-art pictures, but... it's complicated. For instance, this heart (borrowed from the Django project⁵):



could be produced using the following code. Basically (given that colors `violet1` to `violet5` have been correctly defined), we stack up several single-color pixel-art pictures.

```

1 \begin{tikzpicture}[scale=.1]
2   \tikzbpixelart[color=violet1]{(0, 0)}{%
3     0000000
4     0000010
5   }
6   \tikzbpixelart[color=violet2]{(0, 0)}{%
7     0000000

```

⁴She has grown up, and I can get more sleep and more free time now, which explains `pixelart` version 1.0.0!

⁵<https://www.djangoproject.com/>

```

8   0110100
9   0000000
10  0010100
11  0000000
12  0001000
13  }
14  \tikzbpixelart[color=violet3]{(0, 0)}{%
15  0000010
16  0000000
17  1000010
18  0000000
19  0001000
20  }
21  \tikzbpixelart[color=violet4]{(0, 0)}{%
22  0010100
23  1001000
24  0110100
25  0001010
26  0010100
27  }
28  \tikzbpixelart[color=violet5]{(0, 0)}{%
29  0100000
30  0000001
31  0001001
32  0100000
33  }
34  \end{tikzpicture}

```

One could imagine a simpler syntax: we assign several colors to characters, and we use 1, 2, 3, etc. as the pixels to define the picture. This would give the following code.

```

1  \begin{tikzpicture}[scale=.1]
2  \tikzpixelart[colors={
3    1=violet1,
4    2=violet2,
5    3=violet3,
6    4=violet4,
7    5=violet5,
8  }]{(0, 0)}{%
9  0540430
10 4224215
11 3445435
12 0524240
13 0043400
14 0002000
15  }
16 \end{tikzpicture}

```

4.4 Spaces

Spaces are interpreted as line breaks. For instance, this heart ♥ could be written as :

```

1  \bwpixelart[scale=.03, raise=-1pt]{%

```

```

2 0011001100 0111111110 1111111111
3 1111111111 1111111111 0111111110
4 0011111100 0001111000 0000110000
5 }

```

This will work (right now), but is an undefined behaviour, and might change in a later version without prior notice.

4.5 Uneven lines

Right now, lines do not *have* to have the same number of characters. For instance, the following heart ♥ could be written as :

```

1 \bwpixelart[scale=.03, raise=-1pt]{%
2 00110011
3 011111111
4 1111111111
5 1111111111
6 1111111111
7 011111111
8 00111111
9 0001111
10 000011
11 }

```

This is an undefined behaviour and might raise an error in the future.

Change History

v0.1.0		v0.2.0	
General: First published version. . .	7	General: Add package option	
v0.1.2		draft.	3
General: First line-break of		v0.3.0	
pixelart argument is now		General: Rename pixelart to	
automatically ignored.	3	pixelart0.	1

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in **roman** refer to the code lines where the entry is used.

B	T
\bwpixelart	\tikzbwpixelart
<u>62</u>	<u>68</u> , <u>75</u>