PicConv User Manual

Introduction

PicConv is a graphic conversion tool written by Walt of Bonzai. The purpose of this tool is to make it as easy as possible for programmers on the Commodore 64 to work with graphics in pretty much any variation of standard formats. What makes PicConv stand out from the crowd is the ability to chain a number of conversions on a single input file and the ability to script your conversion projects so that they can be included in your own build chain.

PicConv was written during the development of **Unboxed** and we decided to include a lot of the graphics from that demo as examples in the release binaries. We also included some other stuff and hope that the authors are OK with this.

The ability to chain conversions from a single source, will make collaboration between artists and programmers a lot less painful.

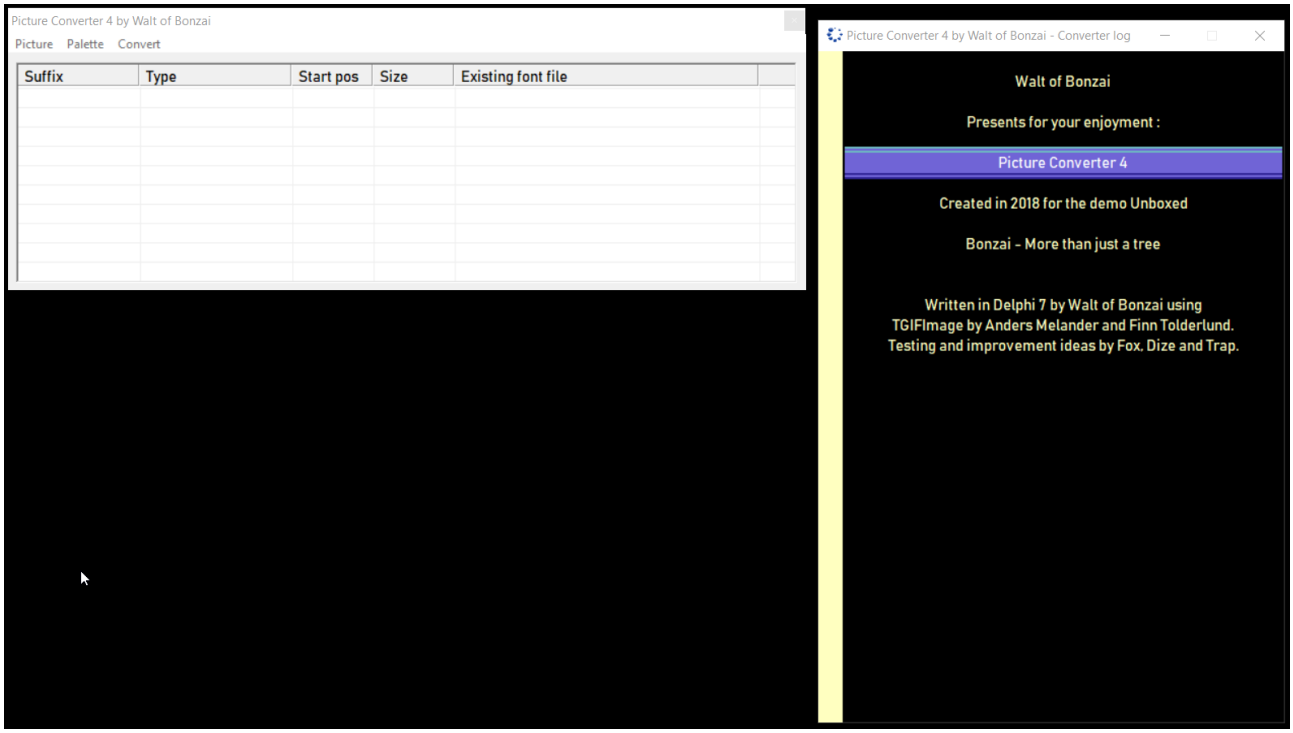
If you require help or have suggestions for improvements, go to the Facebook group “Bonzai's c64 PicConv user group”. Link here: https://www.facebook.com/groups/347360932494631 The group is hosted by Trap with some help from Dize and we will help you as much as we can. Walt is not on social media and he probably never will be so don’t bother asking.

Table of Contents

Getting started

You have downloaded the binaries. Now what?

PicConv is located in the root of the package, when you run it you will get 2 windows, like this:

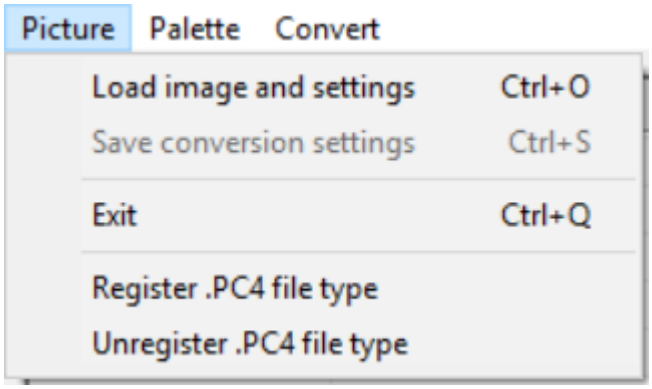


The right window, Converter log, is only for output information. The window on the left is where you build your conversion project.

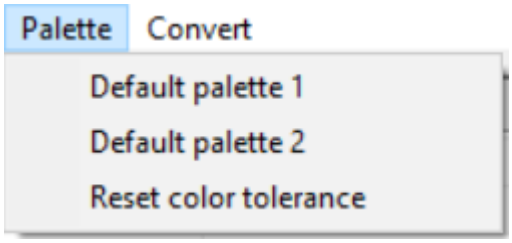
Menus

Let's run through the menus.

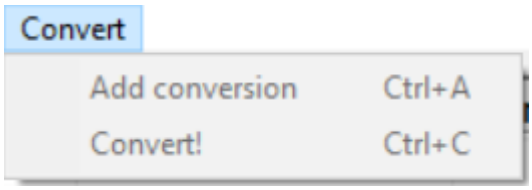
In the "Picture" menu you will find options to load and save your conversion settings (or project). Also there are options to easily register .PC4 (the PicConv project file type) so PicConv automatically loads when you double-click such file.



The “Palette” menu lets you change the palette layout and reset the color tolerance. These things are used when the converter must identify and map the colors of the input to the c64 palette. More about this later.

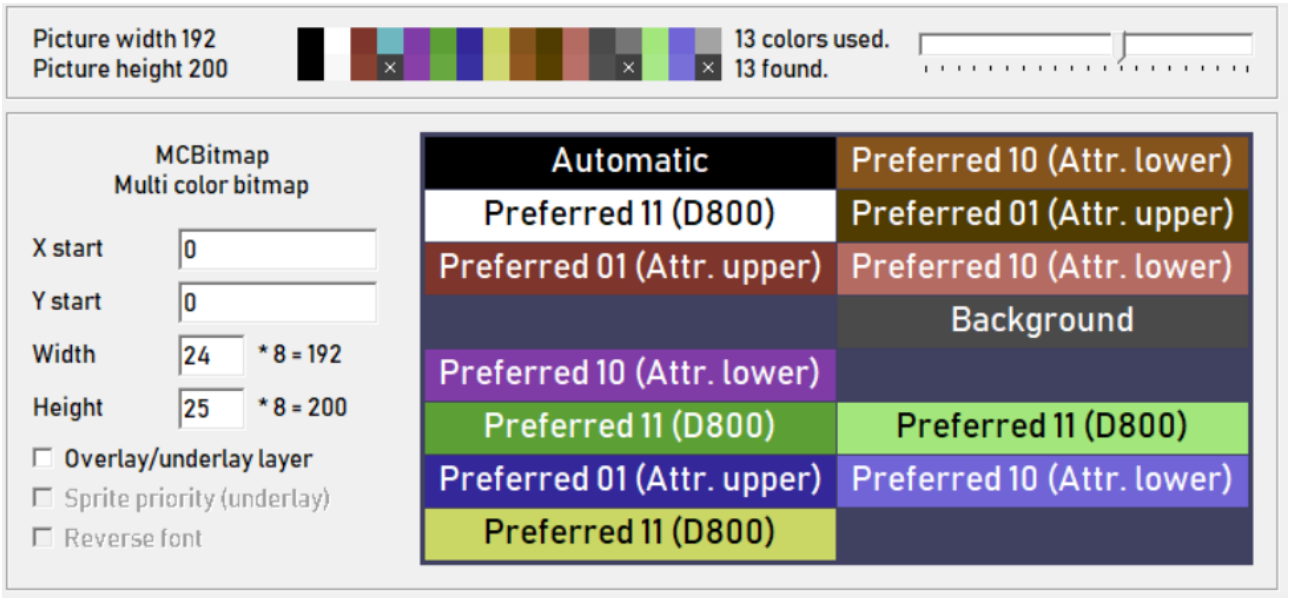


Finally there is the “Convert” menu. This is where all the magic happens. Since we haven’t loaded any graphics yet, the options are greyed out.



Understanding the palette

When you load a graphic file into the converter and add a conversion layer (see the examples), it will show you the palette window.



PicConv will try to match the colors in the graphic file with the palette. If you are still not able to get all matches, try to fiddle with the tolerance bar (that's the one on the top right of this screenshot).

At the top middle you will see the standard c64 palette on top and the lower shows how the colors are mapped. An **x** indicates the color is not used or was not found.

If you have problems hitting the colors, you can change the palette in the **Palette** menu or assign colors in the picture window by right clicking and selecting "Assign color to palette".

Colors are usually assigned as **Automatic**. For example if you have colors 5,2 and 3 in a field, the colors will be assigned the following masks: 2=01, 3=10 and 5=11 (Programmers love this shit).

If you set **Preferred**, that color will be mapped to the mask of your choice. This can give you the benefit of ordering the output binary thus giving you an advantage when you compress the output. If you are clever enough, this will also make some code easier to write.

Fixed works a bit like **Preferred**, only this will force the mapping (**Preferred** tries to fix it for you, but in some cases you want complete control of the conversion proces). If you use this, PicConv will generate an error in the output log if a conversion cannot be made.

Bitmap conversion example

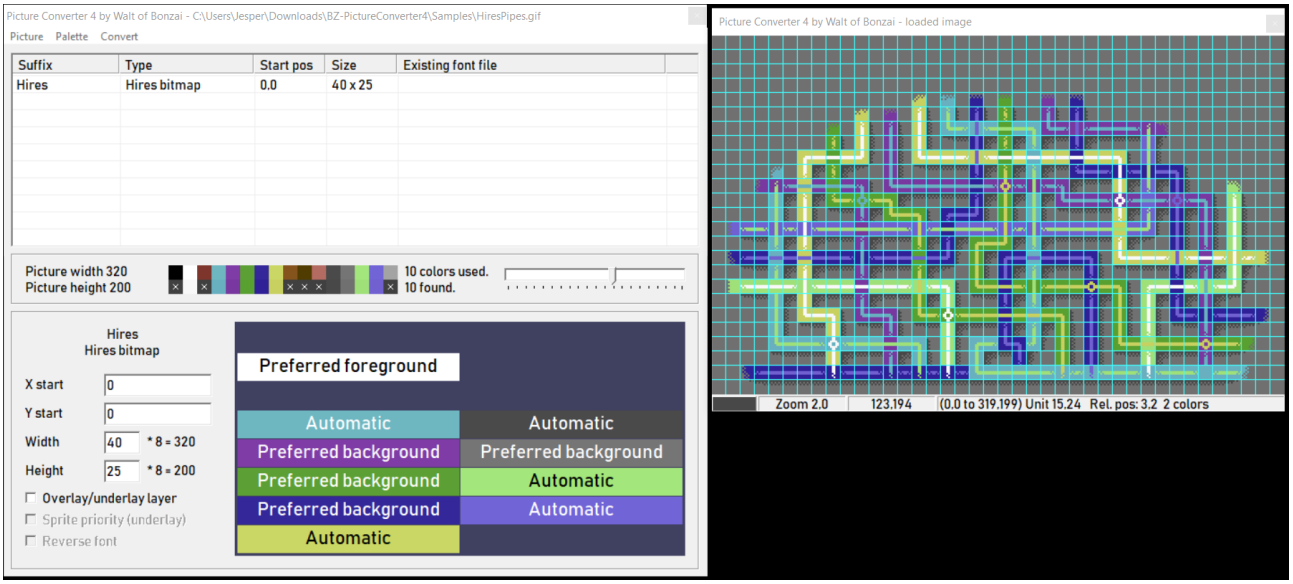
Done with all the boring stuff, let's get cracking.

Say you have a bitmap picture in one of the supported input formats, GIF or BMP. The source in this example will be a hires bitmap.

1. Select **Picture / Load image and settings**
2. A prompt will open, go to the samples folder and double-click the "hirespipes.gif"
3. A third window will now appear showing you the graphics
4. You will notice that the main window now shows a line:

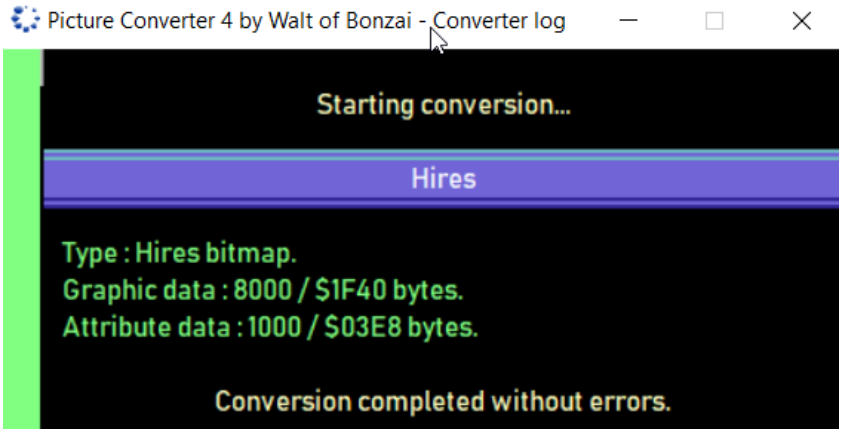
Suffix	Type	Start pos	Size	Existing font file
Hires	Hires bitmap	0,0	40 x 25	

   Since PicConv discovered an existing PC4 project file, it loaded the conversion settings that are associated to this picture for us.
5. Click the line in the main window. You will notice that the graphics window now added an overlay matrix. This matrix shows you what area of the graphics that are targeted for conversion.



6. In the lower left part of the main window you will see the X and Y start coordinates and the size of the area to convert. The X and Y values can be negative and the width and height can be larger than the graphics area, if you need a specific layout of the output.
7. The lower right part of the main window shows you how the colors from the source graphics are mapped to the different nibbles. You can change any of these to meet your requirements by simply clicking one of the colors, a menu will let you select what nibble you want to use for that specific color.
8. We are not going to make any changes, but simply run the conversion project. Click **"Convert" / "Convert!"**

9. In the converter log window you will see the results:



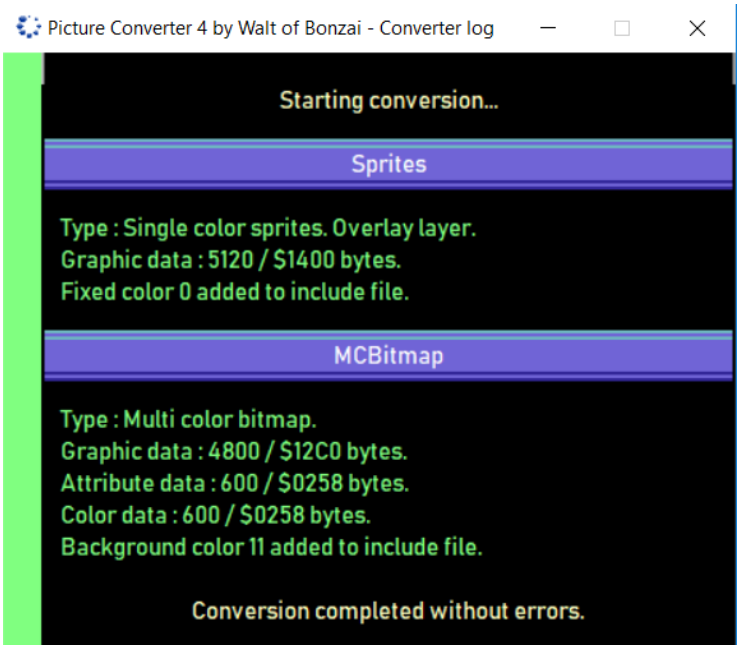
10. As you can see, the output for this conversion was graphic data and attribute data.
11. Browse to the location where the source graphic is located. Here you will find 3 new files:
    HiresPipes_Hires.INC includes the constants of the conversion if you require this for your project.
    HiresPipes_Hires.prg is the graphic data.
    HiresPipes_Hires_Attr.prg is the attribute data.
    The 2 PRG files are ready for your to use. If you use KickAssembler, it's a simple matter of using
    .import c64 "HiresPipes_Hires.prg"
    .import c64 "HiresPipes_Hires_Attr.prg"
12. You are done with your first conversion

Multi-layer conversion example

Let's explore an advanced example where we start to use the real powers of PicConv. This time we will convert a multicolor bitmap that includes some overlay sprites.

1. Open the project Samples\Unboxed\FourPics\P4.PC4
2. As you can see in the main window, there are now 2 conversion runs.
   MCBitmap converts the multicolor bitmap
   Sprites converts the overlay sprites
3. Click the first line "MCBitmap". You will notice the grid from earlier appears. This time we are not using the normal full screen area, but a smaller area.
4. Click the second line "Sprites". Notice that the grid changes to the size of sprites (24x21). This line will convert all black pixels to singlecolor sprites. Also notice that we use a negative offset for the Y position to get things aligned to our requirements.

5. Run the conversion and you will get this output:

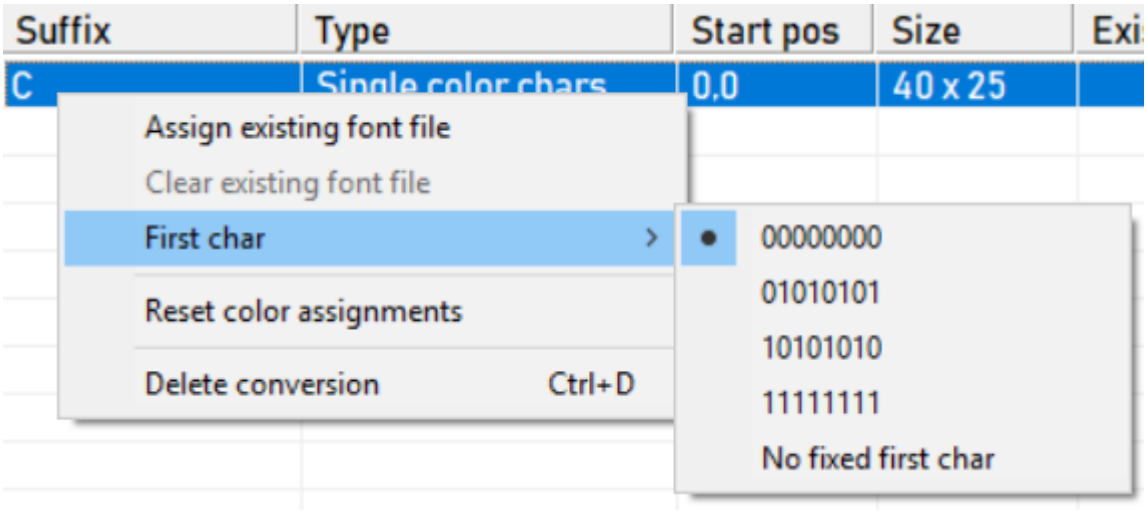


6. The output has put the multicolor bitmap, attributes and colors in separate binaries and the sprites in its own binaries as well. Ready to put it to work in your project like before.

Charset conversion example

Converting charsets has some extra features.

1. Open the project “GaryByMermaid.pc4” in the samples folder
   Right click the conversion line. You will see this menu:



   “Assign existing font file” allows you to share one charset between multiple screens.
   “First char” let’s you decide how to deal with your fill char – if you require one. This gives you added control as you can guess.
2. Let’s not change anything just yet. Try to run convert and you will get a lot of errors. For the sake of learning we left it this way. Can you figure out what’s wrong? If not, remember that charmode only allows you to have one color per char and a shared background color. In this example, you will notice the palette is all messed up. To fix it, set all colors to foreground except for the black – leave that as background. Now you can successfully convert the picture.
3. If you open the output binary GaryByMermaid_C.prg (this is the font file), you will see something like this:
   0020 0000 0000 0000 0000 0000 0000 f0f0
   0020 is the file header (location information). Ignore that. Look at the next 8 bytes – notice they are all zeroes.
4. Now, try to change the first char to 11111111 and run convert. The output this time will be:
   0020 ffff ffff ffff ffff 0000 0000 0000
   Notice first char is now using your newly selected mask.
5.

Shared font files

Adding to the above information. There may be cases where you have different input files that you want to put in one shared charset. PicConv does that for you too. For the sake of example, we will use the output from the previous example.

1. Rename the file **GaryByMermaid_C.prg** to **GaryByMermaid_C_fontfile0.prg**. Could be any binary, this is just for example
2. Right-click your conversion line in PicConv. The one called “C” in the Suffix column and select **Assign existing font file**. A file explorer will appear. Now select the file we just renamed and click **Open**.
3. You will notice that the file is now shown in the **Existing font file** column. What this means is that all output from this conversion step will be appended to the existing font file and your screen map binary file will of course match the layout.

Other stuff

This is a small chapter explaining some of the features I didn't cover in the examples.

You can right-click on any conversion line in your project. Here you can delete the line and reset your color assignments.

Overlay/underlay: This option can be selected per conversion line. When you select it, that line will be processed before any other conversion line. Colors that are used in this layer will be marked as already used and thus you can convert other layers in different formats without getting errors. An example is the FourPics in the sample folder **\Samples\Unboxed\FourPics**. If you open the P4 project, you will notice that the sprite layer has this option selected. So here the sprites will be converted first, removing the black color from the equation enabling us to convert the multicolor bitmap without color clashes.

Reverse font: This will EOR the charset with $FF.

Commandline usage

You can use PicConv with the following parameters to simply open the program with the selected graphic file as source:

PicConv.exe %graphic file%

An example:

PicConv.exe PCEnd.gif

If you want to include the conversion in your own tool chain or other automation, you can use the following commandline, which will search for an associated PicConv project file in the same folder as the graphic file. If found, it will convert it and exit provided the conversion did not get any errors.

PicConv.exe %graphic file% %Anything really%

An example:

PicConv.exe PCEnd.gif BonzaiRule

As long as there is any second parameter, this is the behavior you will get.

Document revision history

Date	Revision	Author	Description
2018.11.21	1.0	Trap	Initial version