**TRS Image**

Kyle Wadsten

**TRS Image** was created as part of my entry into the TRS8BIT **2018 Competition: HI-RES-LO**.

My initial exposure to the TRS80 was in 1980 when my brother bought a Model I. I knew immediately that this was to be my future. And so it was. I have been a professional programmer for 30 years (recently retired). My passion for computers and programming has remained strong ever since that Model I.

A few years back I acquired a Model 4 and more recently, a 4P. Having actual hardware is a little more work (and space) than using an emulator but provides for a much better experience.

TRS Imagewill convert a modern image (jpg, png, etc.) into a BASIC program that will re-create the image on a TRS80. TRS Image is written in Python and runs on a Mac. Due to a number of issues with the wxPython library I am using it does not currently run on a PC.

An older version that uses the PyGame library is available but lacks the File Open and File Save dialog boxes. This older version works on PC or Mac and uses command line arguments for input and output filenames.

Future enhancements will hopefully include a true cross platform version. I have spent endless hours attempting to get TRS Image working on both PC and Mac using the same source code with little success.

With TRS Image you can zoom, pan, and adjust the contrast of an image while it shows in real-time how this image will look on a TRS80. Once you are satisfied with your adjustments, a quick press of the "G" (Generate) key will output a TRS80 BASIC program as a text file.

After the BASIC program is generated you need to transfer it to an actual (or emulated) TRS80. I use Matthew Reed's TRS32 emulator and TRS Tools utility to simplify this process.

Regardless of how you get the BASIC program onto a TRS80 you can then RUN it. The first time you run the program it will build "packed strings" from the DATA statements before displaying the image.

After the program runs the first time it will delete the DATA statements and logic used to build the packed strings\*. This leaves a much smaller program behind. Instructions are displayed describing how to re-save the BASIC program back to diskette in order to run faster and save disk space (7.5K vs 3K).

Running this smaller program significantly speeds up the image display (as you might expect) because it doesn't have to rebuild the packed strings before displaying the image.

The current version of TRS Image targets a **Model III** with **LDOS 5.3.1** and **MISOSYS LBASIC**. I plan on adding the ability to support different versions of BASIC and TRS80 models in a future update.

I have uploaded the python source code and setup instructions for TRS Image to GitHub (<https://github.com/kwadsten/trs-image>)

Kyle Wadsten ([kwadsten@comcast.net](mailto:kwadsten@comcast.net))

**\*String Packing in TRS80 Basic**

One method of drawing images in BASIC on the TRS80 is to use SET/RESET to plot each pixel in the 128x46 screen area. As you might expect, this is extremely slow.

However, the TRS80 character set also contains a set of graphic characters that allow you to plot a 2x3 matrix of pixels at once. These special characters have values between 128-191. The binary representation of these characters are a 8 bit value with the high-order bits set to 10: "10xxxxxx". The remaining bits represent the 2x3 matrix of pixels in reverse order. 0 = off, 1 = on. For example to set the top two pixels of the character to white you would use an ASCII value of 10000011 (131 in decimal, 0x83). Printing this to the screen in basic would be a simple: PRINT (or PRINT@) CHR$(131).

# TRS80 String Packing

#

# 2x3 "text semigraphics"

#

# b0 b1

# b2 b3

# b4 b5

#

# 1 0 b5 b4 b3 b2 b1 b0 = character value

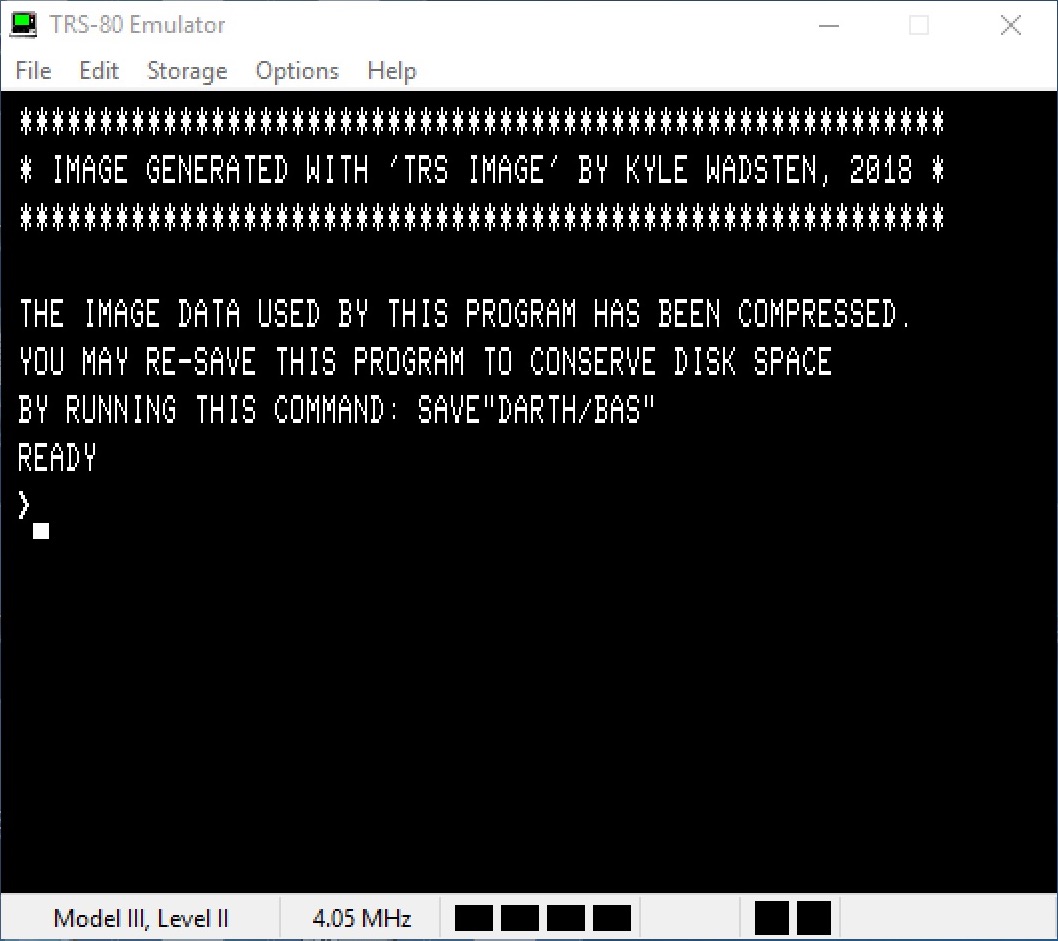
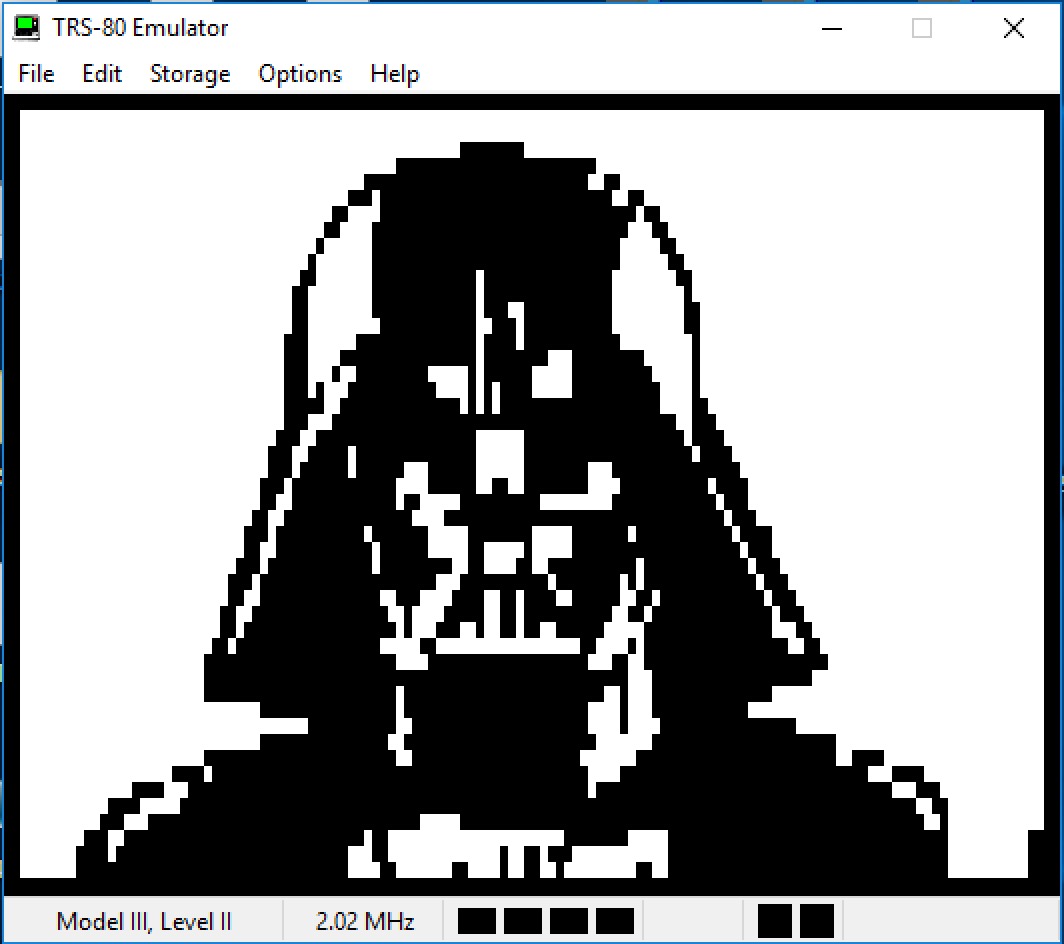
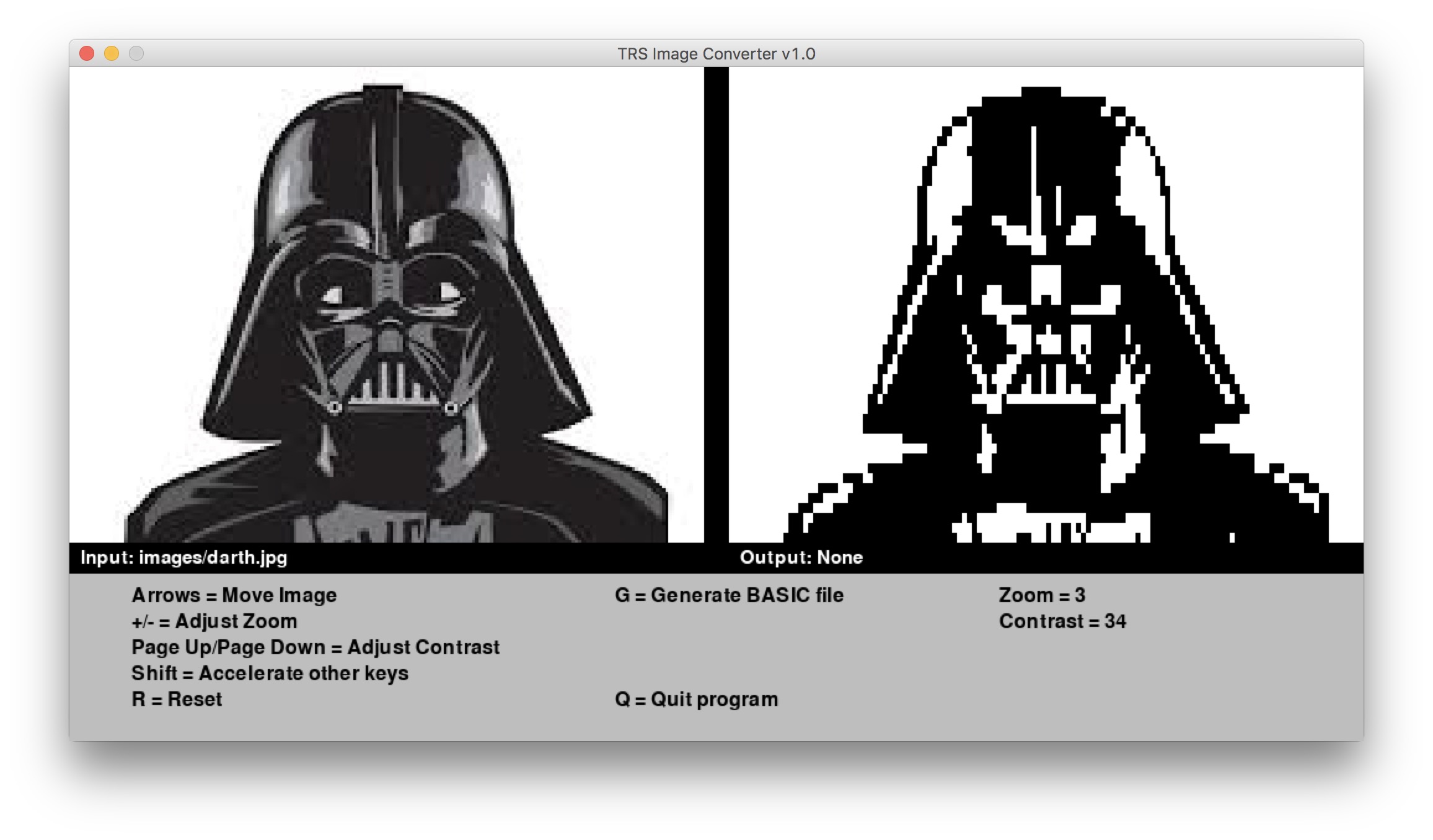
#

# 0 = off (black)

# 1 = on (white)

#

See <http://www.trs-80.com/wordpress/zaps-patches-pokes-tips/graphics/> for more information.

**Screens****hots**