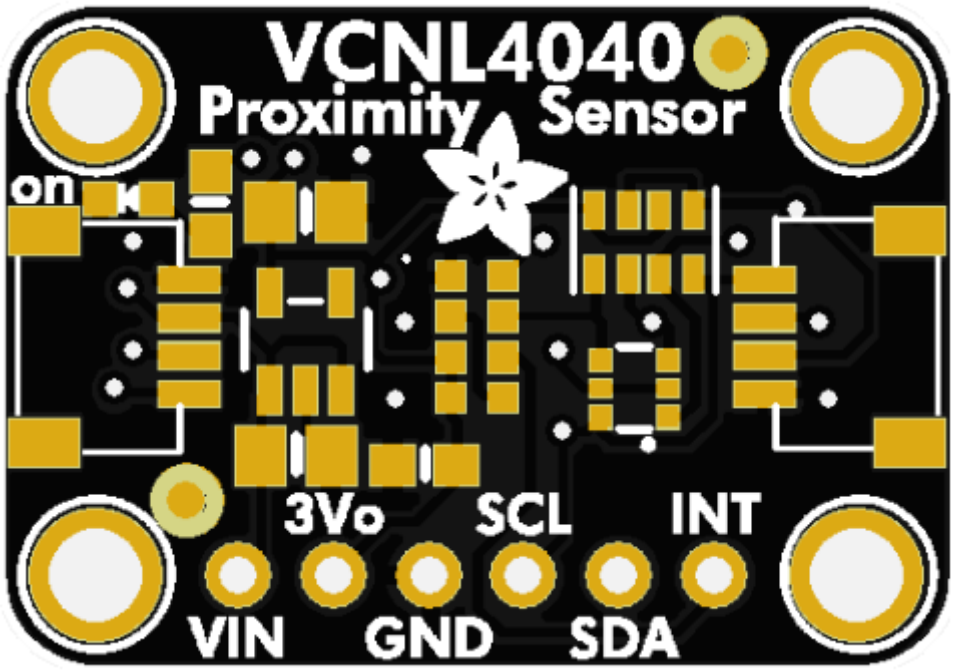
# Adafruit Pinguin for EAGLE CAD

Created by Phillip Burgess

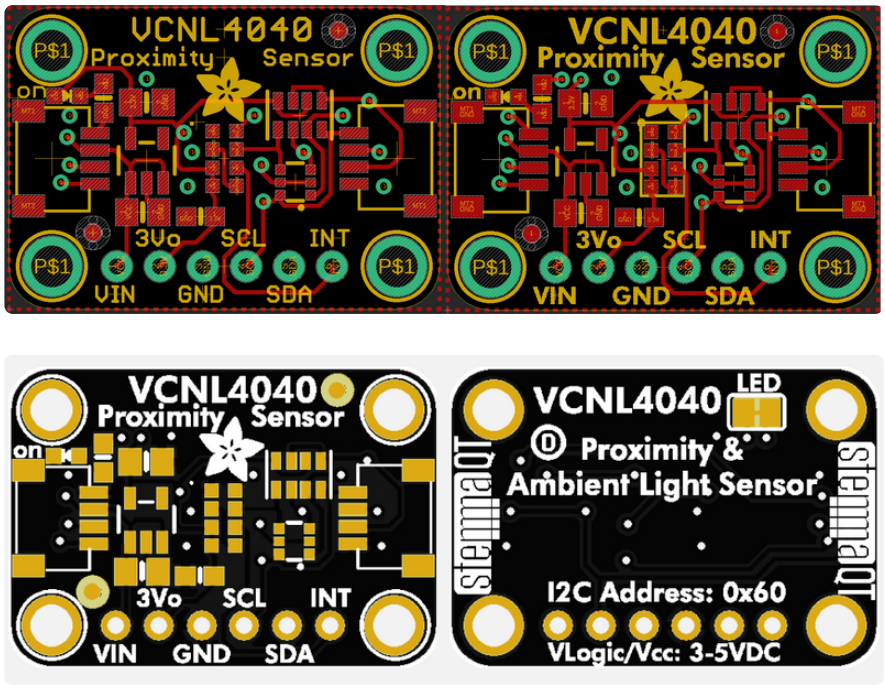



https://learn.adafruit.com/adafruit-pinguin-for-eagle-cad

Last updated on 2024-06-03 03:41:36 PM EDT

# Table of Contents

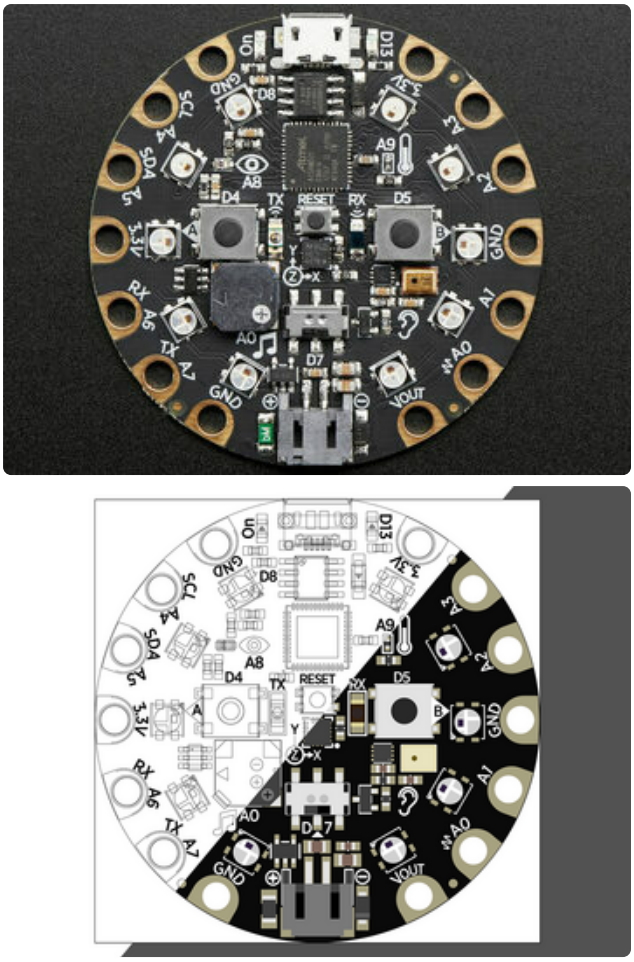
# Overview




Autodesk's **EAGLE** — the PCB design software favored around Adafruit — has a problem: the circuit boards it produces, while perfectly functional, are **ugly**, with vintage plotter-like text and no font support.

One can't really fault the software for this...it's more an artifact of its **age**. When EAGLE was hatched in 1988, circuit boards went inside products, they were rarely products themselves...and if they were, appearances were of minimal concern.

But as the electronics hobby realm grew over time, the aesthetics of these parts became a distinguishing feature. While not the first, the popularity of the Arduino UNO (2010) really put this idea in the public eye, with its on-brand color and typography.



The thing about custom silkscreens is that the process is laborious, and can take days to get everything right. We'll do that for a popular item like **Circuit Playground**. The custom silk is easier to read and just feels "premium."

But to do that with every esoteric sensor breakout board, there just aren't enough hours in the day. It would be nice, but isn't practical.

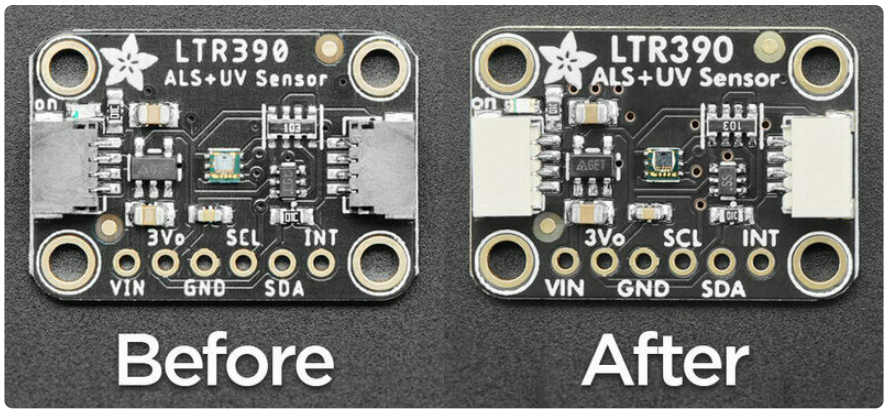
**What if there was a middle ground?** Just a small push that elevates the weakest part of EAGLE's output?

# Introducing Pinguin

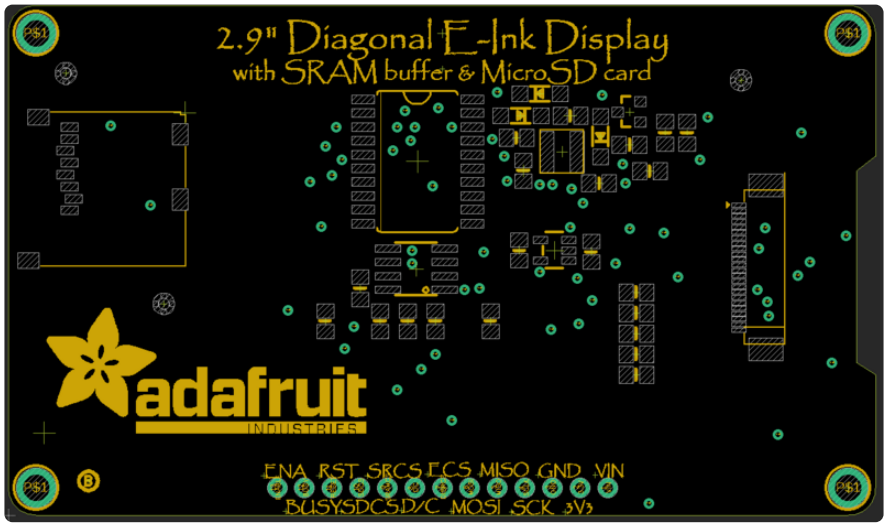
**Pinguin** is a Python script that substitutes **TrueType fonts** for EAGLE's ugly plotter-stroke text. EAGLE currently can't do this on its own...in fact, if you try even using the built-in "proportional" font (intended only for schematics) on a PCB, those labels are converted to the ugly font on output.

Not a typo, the name's a portmanteau of **pin** (since it's mostly I/O pins that get labeled) and **penguin** (keeping with the avian motif of EAGLE and other silkscreen-improving projects like SparkFun's Buzzard (https://adafru.it/11bd)).

Pinguin has limitations, as we'll explain, and probably bugs...it's far from a perfect solution...but it's quick and often good enough to dress up boards for better readability.




Yes, you can now make PCBs where the silkscreen labeling is in Papyrus (https://adafru.it/11bf)!




---

# Installation

Pinguin has the following prerequisites:

- **Autodesk EAGLE** PCB design software. If reading this guide, you're likely already a user... but if not, a limited free version is available for Windows, Mac and Linux (https://adafru.it/zBt).
- A recent (**3.X**) version of **Python**...a standard feature on many systems now (e.g. macOS includes Python 3.7). Type `python -V` from a command prompt and see what you get. If not present, visit Python.org (https://adafru.it/11b6) for installation help.
- The **Pillow** graphics library. This might already be installed with Python, or you can type:

```
pip install pillow
```

(may require “pip3” or capitalized “Pillow”, but that’s the basic idea.)

The Pinguin software (https://adafru.it/11b7) can be retrieved either via git if you have that installed:

```
git clone https://github.com/adafruit/Adafruit_Pinguin
```

...or by downloading and uncompressing the ZIP archive from GitHub:

**Download Adafruit_Pinguin ZIP Archive**

https://adafru.it/11b8

You’ll have a folder, **Adafruit_Pinguin**, containing the **pinguin.py** Python script and a **fonts** subdirectory.

---

# Using Pinguin

Pinguin is invoked from the command line. It’s an esoteric tool that won’t see widespread use, so please forgive that error handling is minimal. If you encounter trouble that can’t be puzzled out from Python’s traceback messages, ask for help in the Adafruit Forums (https://adafru.it/jlf), or open an issue (https://adafru.it/11bb) on GitHub if it’s clearly a bug.

At its simplest, Pinguin accepts an EAGLE **.brd** file for input:

```
python pinguin.py /path/to/file/board.brd
```

(You might need “`python3`” on a few systems that keep Python 2.X around for vintage compatibility.)

The output will be a **new .brd file** (the original remains untouched) with “_out” inserted in the name; e.g. if `board.brd` is the input file, the output will be `board_out.brd`.

**HERE’S WHAT HAPPENS:**

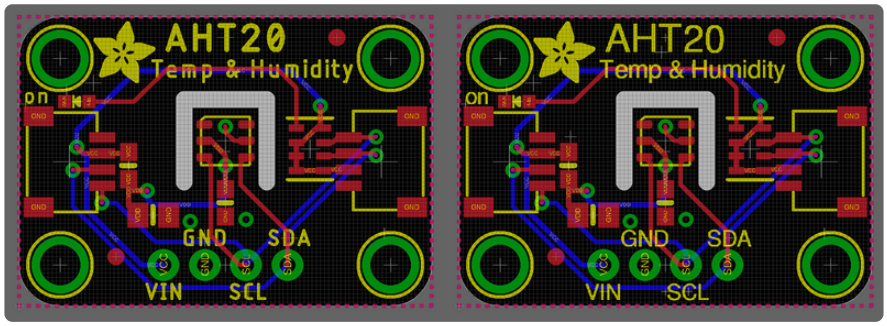
1. The script looks for any **text objects** in layers **21** (tPlace) and **22** (bPlace) — these are normally the top and bottom **silkscreen** layers.
2. A **TrueType font** is chosen based on the object’s **font** property: **vector**, **proportional** or **fixed**. EAGLE allows selecting “proportional” even though it can’t handle this on output.

3. A high-resolution **raster** equivalent is generated and placed at roughly the same position, either in layer **170** (for top elements) or **171** (bottom).
4. The **original** text object is moved to layer **172** (top) or **173** (bottom) for safekeeping.

Layers **170–173** were chosen as they're **not normally dedicated to anything** in EAGLE. If you do use any of those layers for something in your own workflow, you'll want to modify the script (there are global settings near the top).

The new layers **170** & **171** should be included as **silk** when generating EAGLE's **CAM output**. The backup layers **172** & **173** should be **hidden**.

Here's what a board might look like before, using the "vector" font, and the output file after processing:




Because the text elements have been rasterized, they're **no longer directly editable in EAGLE**. You can move and rotate these rasters, but to change the text you'll need to find the original item in backup layers 172 or 173, change properties to move it back to layer 21 or 22, edit the text and re-run Pinguin.

This is why Pinguin outputs to a new file. The raster text...especially from "vector" font text...isn't a perfect match, and you may be iterating a few times to find an ideal size and appearance. Then tweak the positions in EAGLE in the output file if needed.

## Selecting and Scaling Fonts

A few command line options allow changing TrueType font selections and relative sizes.

`-vfont`, `-pfont` and `-ffont` select different TrueType fonts for the vector, proportional and fixed text elements. They can be set individually, or in combination. Each accepts a TrueType font filename, for example:

```
python pinguin.py board.brd -vfont fonts/GNU/FreeSans.ttf
```

The default proportional and fixed fonts match what EAGLE uses for display (but not output), so it's somewhat WYSIWYG-like in that regard. The default vector font isn't a close match so you'll probably want to experiment. The fonts folder contains a few items with permissive licensing, and there's a ton more at fonts.google.com (https://adafru.it/MEh).

`-vscale`, `-pscale` and `-fscale` adjust the size of the rasterized vector, proportional or fixed text. Each accepts a floating-point value, empirically derived, larger value = larger font. For example:

```
python pinguin.py board.brd -vscale 1.33
```

Default scales for vector, proportional and fixed fonts are around 1.33, 1.41 and 1.41, respectively. Combined with the default fonts, these are a near perfect match for proportional and fixed text elements (vector, not so much).

## Other Settings, and Limitations

Pinguin's raster resolution can be configured with the `-dpi` setting:

```
python pinguin.py board.brd -dpi 600
```

Default is 1200 dots per inch, which is probably excessive, but no harm done. Too much is better than too little, especially with text at an angle. You can try lower values if concerned about output file sizes.

### MULTI-LINE TEXT

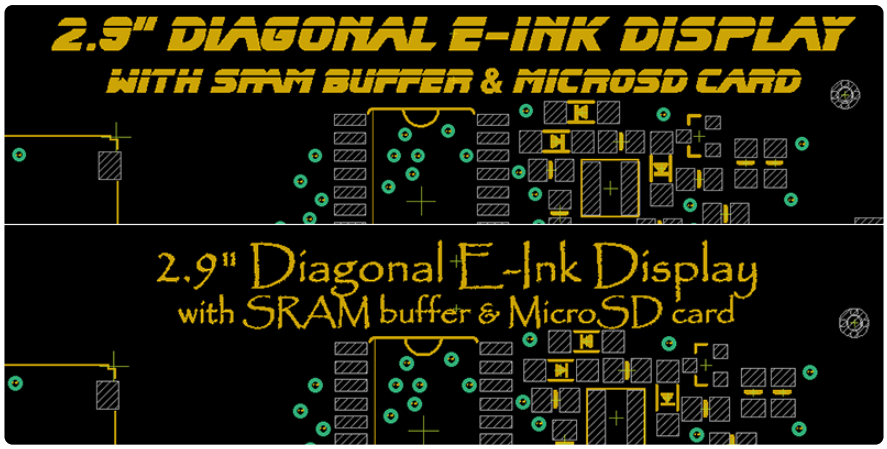
EAGLE allows entering multi-line text, but Pinguin might not render this with the same line spacing, or might get the vertical position slightly wrong. A workaround for now is to enter each line as a separate text object.

### NO INVERSE

Pinguin only generates "positives" — the text is whatever your silk color is. If you need inverse (e.g. black text inside a white box), SparkFun's Buzzard (https://adafru.it/11be) has you covered!

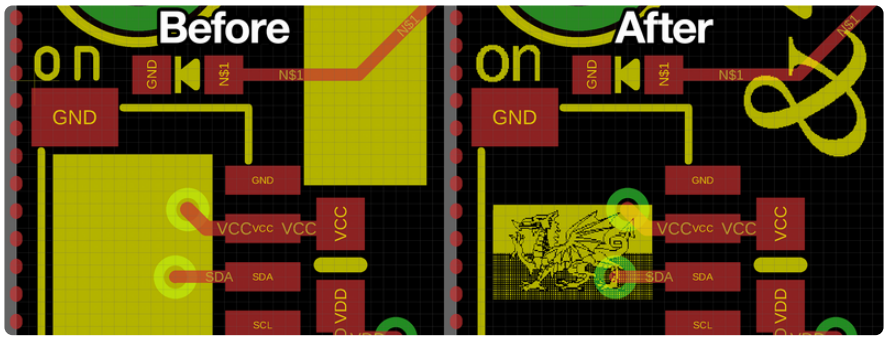
### REALITY LOOKS BETTER THAN EAGLE

Something about the way EAGLE renders these raster objects makes them look grainy on-screen. Pinguin's (and Buzzard's) rasters are actually pretty sharp, and you'll see this in a Gerber viewer app.

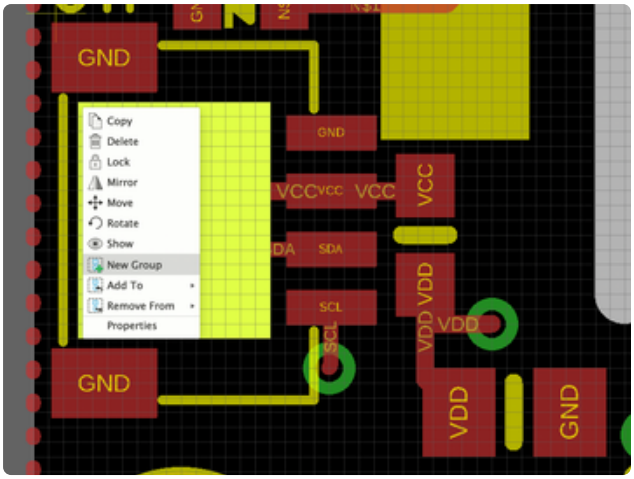


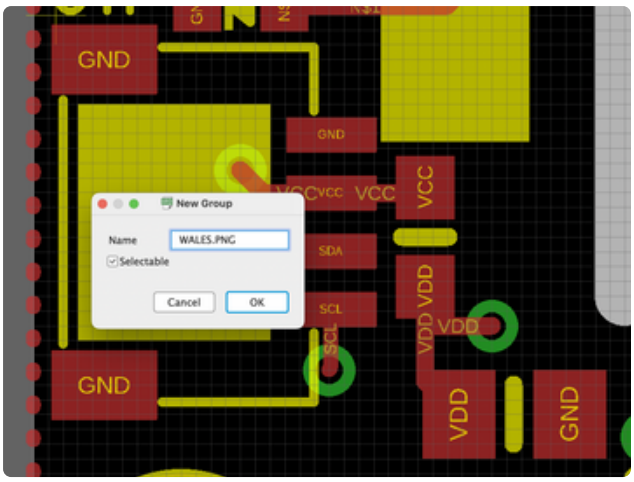

## Inserting Logos/Symbols

Not just for text! If you have certain silkscreen graphics that you tend to use again and again — logos or other design elements — Pinguin can drop these in over placeholder rectangles.




Principle is similar to text. Place **rectangle** objects in the **tPlace (21)** or **bPlace (22)** layers and run the board file through Pinguin. But first...





Associate an **image filename** with a rectangle by setting the object's **group** property.

You can do this by right-clicking over a rectangle and selecting "**New Group**" from the pop-up menu, then entering an **image filename** (located in Pinguin's **"symbols" sub-folder**). Or you can **change** the image (group) via the **Properties** dialog.

The group thing is admittedly weird and clunky but it's the only place we found to easily attach this extra information to a rectangle.

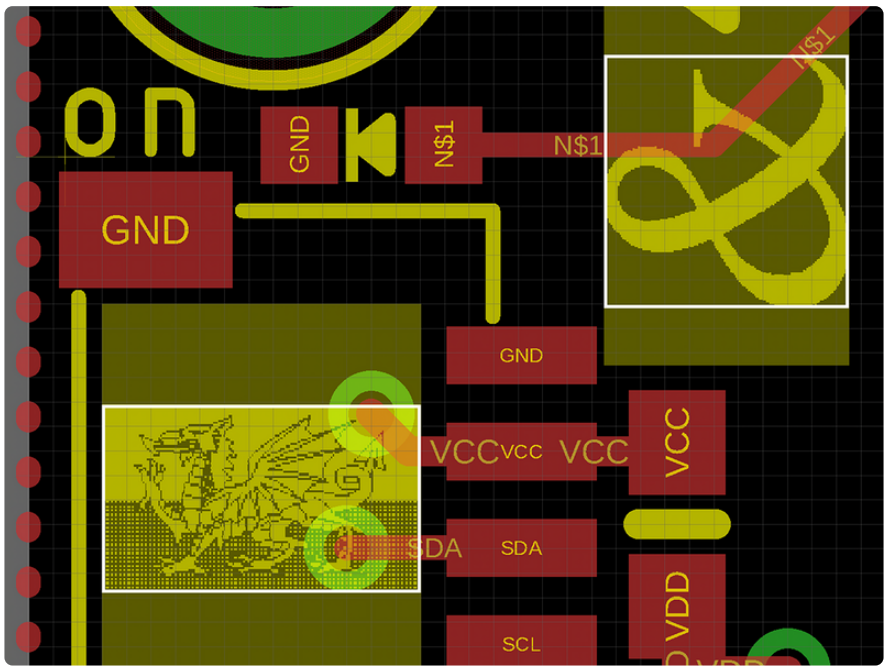
Images can be in **BMP** or **PNG** format. BMP is ancient but long-time EAGLE users might already have a lot of silkscreen elements in that format (it's what the old school import-bmp.ulp script uses; how we did this before Pinguin).

Pinguin will observe a rectangle's **rotation** setting when placing images. For example, if you need a logo rotated 90 degrees, set the rectangle's **angle** property. You'll get no visual feedback in EAGLE (it still looks like a rectangle), but you'll notice the

difference when running it through the Pinguin script. This might require a couple tries to get right. As with text, the original rectangles are saved in layers 172–173 if you need them back.

Unlike text which is always rasterized at 1200 dots per inch (or whatever `-dpi` setting was specified), images don't observe a hard number here; the source bitmap is **scaled** to the rectangle's dimensions (letterboxing or pillarboxing as necessary to keep image pixels square, see below), **whatever that works out to be**.

Let's suppose one has a logo they anticipate using at various sizes from 1/2 to 1 inch wide. You could produce two different images, one for each size, at 1200 DPI (or your preferred working resolution). These would then be 600 and 1200 pixels wide. But it's likely sufficient to split the difference and make a single image...900 pixels in this case...which works out to 1800 DPI at the 1/2 inch size and 900 DPI at the 1 inch size. Both are more than ample and won't appear jaggy on the finished board; the actual silkscreen resolution will be something less than that.




As explained on the "Using" page, EAGLE's bitmap rendering makes things appear more rough than the finished product. Try using a Gerber PCB viewer for a better preview of the finished product!