



Contents

- [1 General](#)
- [2 Alpha Transparency](#)
- [3 PNGOUT](#)
- [4 Extreme optimization](#)
- [5 Multiple PNGs](#)
- [6 Tools](#)

General

Because of the low bandwidth of [GPRS](#) many operators put a restriction on the size of jars. This limit varies from country to country. Typically in Europe it is around 200k - 300k, but in developing markets it could be much lower. For this reason it's important to reduce the size of the resources in your jar.

[Java ME](#) supports two types of the [PNG](#) format. The most common type is PNG-24 which supports 16.7 million colors. There is also what's known as indexed PNG (or PNG-8) which only has 256 colors.

If you convert a PNG-24 to PNG-8 you usually get around a five time size reduction, but you have a much smaller amount of colors. In most cases the reduction of colors is not even discernable, in fact most commercial games use images with a lot less than 256 colors.

You can create PNG-8 using the "save for web" feature in Photoshop or Export feature in Fireworks. This is a very nice feature. It shows you what the image looks like before and after the conversion. Normally you will see no noticeable degradation in quality.

The new version of the Gimp (2.4) also can create PNG-8. It's in the menu image/mode/Indexed.

Alpha Transparency

Unfortunately, neither Photoshop nor the Gimp support alpha transparency (they only support single bit transparency). This means if you have irregular shaped images (i.e. non-rectangular) they will have a pixelized looking border. You can compensate for this effect by selecting the "matte" colour to be the same as the background colour on which the images are going to be displayed. Of course that is only effective if you have a homogenous background colour.

There is also a program called Web Image Guru which can create indexed PNGs with full alpha transparency.

There are two open source tools that can convert PNG-24 to PNG-8 with alpha transparency, PNGQuant and PNGNQ. Unfortunately, both of these tools have a bad palette reduction algorithm (if your original image has 600 colours, you are going to have to reduce the colours to at least 256 to make it PNG-8). This results in a noticeable loss of quality. Hopefully, in the future these tools will be improved.

Unfortunately, only Nokia and Sony Ericsson provide good support for support alpha transparency, so most people stick with single bit transparent PNGs and leave it at that.

PNGOUT

Another tool that is essential in reducing your PNG's is PNGOUT <http://advsys.net/ken/utills.htm>. This reduces the size by removing unneeded headers and recompressing the PNG using more efficient algorithms. Usually you gain around a 10% reduction.

The default parameters are usually the best, however if you plan to port your application I recommend using the /MINCODES2 option. This is a little know feature that solves problems with some phones that have buggy zip decompression implementations.

You can create a batch file to run PNGOUT on all PNGs within a directory using this code:

```
for %i in (*.png) do pngout "%i" /mincodes2
```

This tool is constantly being improved, so make sure that you get the latest version.

Extreme optimization

If you really need that extra few bytes reduction you can use PNGSlim <http://people.bath.ac.uk/ea2aced/tech/png/pngslim.zip>. This is a script that uses a combination of pngout, advdef, DeflOpt and pngrewrite. It yields another 0.5% reduction to pngs that already have been optimised using pngout, although it runs rather slow.

Multiple PNGs

If you have multiple PNG files in your project, you can reduce about 200 bytes per PNG (header information), joining all the images in one only large PNG file. Then, you can use some [Sprite](#) design pattern to extract the little images from the big one

Tools

- [PNGSlim](#)
- [PNGOut](#)
- [PNGOptimizer](#)
- [OptiPNG Advanced Optimizer](#)