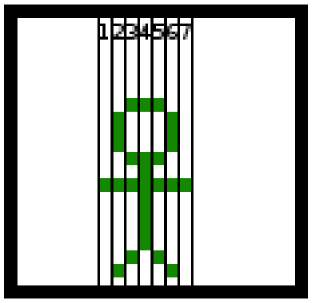
PGF stands for Polygonal Graphics Format. It is a form of raster graphics that does not always form a rectangle. It does this by simply not storing completely transparent pixels. Instead we will use posts (a line of pixels) with a offset from the top. This format is designed to be useful for sprite based video games, web graphics, and other uses for transparent graphics.

PGF implements two forms of compression. The first is the ability to reuse redundant columns. Take the example image diagram on the right. Columns 1 and 7 are the same so we only will need to store column 1 and make the pointer for column 7 point to column 1. This can also be applied to completely transparent columns. (So counting all transparent columns as one column, the image on the right would require 5 columns to be sored in the file). The other means of compression is DEFLATE applied to each individual "chunk".



The format for the header is as follows: (Note: All fields are arranged in the little endian byte order.)

Field Name	Size	Description
Identification	4 Bytes	'P' 'G' 'F' 0x00
Width	Unsigned Short	"Canvas width." More specifically this will define how many columns there are.
Height	Unsigned Short	"Canvas Height." Used as a guide to how much space should be allocated to the image.
Mode	Unsigned Byte	Sets the mode of the format. See the next table.

Graphics Modes:

Mode	Description
0x00	8-bit image with variable length palette[1].
0x01	8-bit gray scale.
0x02	16-bit color (RRRRRGGGGGGBBBBB)
0x03	8-bit alpha 8-bit color (palette). (0-7 = index 8-15 = alpha)
0x04	8-bit alpha 8-bit gray scale. (0-7 = color 8-15 = alpha)
0x05	24-bit (RGB)
0x06	8-bit alpha + 16-bit color (0-15 = color 17-23 = alpha)
0x07	32-bit (RGBA)

1 Variable length palette could contain 0 colors. In which case a global palette should be present. If the implementation does not have a global palette available, then it should use gray scale.

A chunk should follow the following format. Unknown chunks should be ignored by the implementation. If the compressed size is equal to the uncompressed size then it should be treated as raw data.

Field Name	**Size**	**Description**
Identification	4 Bytes	Four character which name the chunk.
Compressed Size	Unsigned Long	The length of the chunk when compressed.
Uncompressed Size	Unsigned Long	The length of the chunk when uncompressed.
Data	*Compressed Size*	The data contained in the chunk.

Chunk: PLTE

Holds the 8-bit palette information. The palette should always default to 256 color gray scale with 0 being black and 255 being white. This chunk will replace the first NumColor indexes with a new color. This chunk, if present, must come before the IDAT chunk.

Field Name	**Size**	**Description**
NumColors	Unsigned Byte	Number of colors in this palette.
Data	*NumColors*3*	RGB palette indexes.

Chunk: IDAT (Required)

This chunk contains the data of the image. It starts with *width* amount of unsigned long pointers that specify the offset to the beginning of that particular column relative to the beginning of the chunk data. Immediately following the columns will start.

A column follows the following form:

Field Name	**Size**	**Description**
Post	*Variable*	*Line of color.*
End of Column	Unsigned Short	0xFFFF – Indicates the end of the column.

A column is composed of multiple posts which repeat until the end of column is reached.

Field Name	**Size**	**Description**
Y-Offset	Unsigned Short	Number of pixels from the top that the post should be begin to draw.
Length	Unsigned Short	Number of pixels to be drawn in this post.
Data	*Length*bbp*	Pixel Data. Format of pixels described above.