

**NAME**

pbmmask - create a mask bitmap from a regular bitmap

**SYNOPSIS**

**pbmmask** [-expand] [*pbmfile*]

**DESCRIPTION**

Reads a portable bitmap as input. Creates a corresponding mask bitmap and writes it out.

The color to be interpreted as "background" is determined automatically. Regardless of which color is background, the mask will be white where the background is and black where the figure is.

This lets you do a masked paste like this, for objects with a black background:

```
pbmmask obj > objmask
```

```
pnmpaste < dest -and objmask <x> <y> | pnmpaste -or obj <x> <y>
```

For objects with a white background, you can either invert them or add a step:

```
pbmmask obj > objmask
```

```
pnminvert objmask | pnmpaste -and obj 0 0 > blackback
```

```
pnmpaste < dest -and objmask <x> <y> | pnmpaste -or blackback <x> <y>
```

Note that this three-step version works for objects with black backgrounds too, if you don't care about the wasted time.

You can also use masks with graymaps and pixmaps, using the *pnmarith* tool. For instance:

```
pgmtopgm obj.ppm | pgmtopbm -threshold | pbmmask > objmask.pbm
```

```
pnmarith -multiply dest.ppm objmask.pbm > t1.ppm
```

```
pnminvert objmask.pbm | pnmarith -multiply obj.ppm - > t2.ppm
```

```
pnmarith -add t1.ppm t2.ppm
```

An interesting variation on this is to pipe the mask through the *pnmsmooth* script before using it. This makes the boundary between the two images less sharp.

**OPTIONS****-expand**

Expands the mask by one pixel out from the image. This is useful if you want a little white border around your image. (A better solution might be to turn the *pbmfile* tool into a general cellular automaton tool...)

**SEE ALSO**

**ppmcolormask(1)**, **pnmpaste(1)**, **pnminvert(1)**, **pbm(5)**, **pnmarith(1)**, **pnmsmooth(1)**

**AUTHOR**

Copyright (C) 1988 by Jef Poskanzer.