PicComment notes

126476_PixelRule.tiff ¬

Convert PICT 1.1 now recognizes all the PicComments documented in Inside Macintosh and the Macintosh Technical Notes. For these PicComments, Convert PICT writes out any parameters to the comments as structured information to the eps file (if it doesn't recognize a PicComment, the parameters are written as a hex string).

Thus, Convert PICT does the first step in converting these PicComments: it gets the information into an easily usable form. Many of the PicComments have no meaning in a NeXT eps file, and so are ignored. For others, Convert PICT writes some PostScript code to support them. While I've tried to provide support for some PicComments, none of my support in PostScript is exact, and I know that some is quite incomplete. This is partially because I don't have easy access to a LaserWriter for testing).

In any case, this provides, at least, approximations of some of the PicComment behaviors. Further, if you need one of these PicComments to behave better, it's just an issue of fiddling with the PostScript files in Convert PICT (See the `Technical Gabble' topic ;Techie stuff.rtfd;;¬).

The table below lists all the documented PicComments, and my level of support for them. There are three levels of support listed:

57736_paste.tiff ¬ *none* means that the piccomment and any parameters is ignored and no further processing is done in the eps file. In many cases, this is appropriate support, since the piccomment may not have any real applicability in an eps file under NeXTSTEP

709369_paste.tiff ¬ *minimal* means that there is some processing of the piccomment and any parameters, but it does

not fully implement the effects of the PicComment.

965645_paste.tiff ¬ *mostly* As far as the author can tell without comparative testing, these PicComments are completely implemented.

PicComment name	Support	Notes
picLParen	none	Not applicable here (?)
picRParen	none	Not applicable here (?)
TextBegin	minimal	While these do provide rotatation to the text, I'm very sure the effect is not 100% correct. However, I have no examples to study to determine precise correctness. Thus, I'm leaving at this intermediate state.
TextCenter	minimal	See above
TextEnd	minimal	See above
StringBegin	none	Complex and probably not applicable to this environment
StringEnd	none	See above
LineLayoutOff	none	Not applicable here
LineLayoutOn	none	Not applicable here
ClientLineLayout	none	Too complex to implement now
PolyBegin	none	Too complex to implement without output examples.
PolyEnd	none	See above
Polynone	none	See above
PolySmooth	none	See above
PolyVerb	none	See above
PolyClose	none	See above
DashedLine	minimal	1-pixel thick straight lines are dashed properly, but may not dash much of anything else.
DashedStop	minimal	See above.

SetLineWidth	mostly	Appears complete
PostScriptBegin	none	Not applicable here
PostScriptEnd	none	Not applicable here
PostScriptHandle	none	Not applicable here
PostScriptFile	none	Not implementable here
TextIsPostScript	minimal	This is implemented insofar as it causes all following text strings to be ignored. We have to ignore them because they may be referencing Apple's PS dictionary which isn't here.
ResourcePS	none	Not implementable here
PSBeginNoSave	none	Not applicable here
SetGrayLevel	mostly	Appears complete
RotateCenter	minimal	While these seem to work better than the text rotation PicComments above, I suspect they're still skewed. Like the text rotation comments, at least this will tell you that things were rotated. it's just that the rotation placement may be wrong.
RotateBegin	minimal	See above
RotateEnd	minimal	See above
FormsPrinting	none	Not applicable to an eps file
EndFormsPrinting	none	Not applicable to an eps file
BitMapThinningOff	none	Not applicable here
BitMapThinningOn	none	Not applicable here