Apple II
Technical Notes

Developer Technical Support

Apple IIGS
#46: DrawPicture Data Format

Written by: Jeff Erickson & Keith Rollin
November 1988

This Technical Note describes the internal format of the QuickDraw II picture data structure.

This Technical Note presents the internal format of the QuickDraw II picture data structure for informational purposes only. You should **not** use this information to write your own bottleneck procedures; the only routines which should create and read `PICT` format files are those provided in QuickDraw II. If we added new objects to the picture definition, your program would not operate on new pictures. This Note documents this information for **debugging purposes only**.

Picture Data Structure Definition

Pictures are stored in memory in the following format:

They begin with a `WORD` which indicates the mode of the port which was used to record when the picture was created. This information is useful when the picture is played back, possibly in a different graphics mode.

Following the `WORD` is a `RECT` which indicates the frame of the picture and is used for scaling when you redraw the picture. Following the `RECT` is the version number of this `PICT` format, then a series of word-sized opcodes which describe the sequences of QuickDraw II commands that were used to create the picture.

Name	Description	Size (bytes)
`pictSCB`	picture’s scan line control byte	2 (high byte = 0)
`picFrame`	picture’s boundary rectangle	8
`version`	picture version	2 (Currently $8211)
`opcode`	operation code	2
`<data>`	operation data	variable, depending on opcode
:		
`opcode`	operation code	2
`<data>`	operation data	variable, depending on opcode

Opcodes

As mentioned above, pictures are described by a series of opcodes which are used to record the QuickDraw II commands that created the picture. These opcodes are two bytes long and are usually followed by a number of parameters.

All currently defined opcodes and their parameters are listed below. Any opcodes not listed here are reserved.

Opcode	**Name**	**Description**	**Parm Bytes**	**Parameter Description**
$0000	`NOP`	no operation	0	none
$0001	`ClipRgn`	clip to a region	[region size]	region
$0002	`BkPat`	background pattern	32	background pattern (8x8 pixels)
$0003	`TxFont`	text font	4	Font Manager font ID (long)
$0004	`TxFace`	text face	2	text face (word)
$0005	`TxMode`	text mode	2	text mode (word)
$0006	`SpExtra`	space extra	4	space extra (fixed)
$0007	`PnSize`	pen size	4	pen size (point)
$0008	`PnMode`	pen mode	2	pen mode (word)
$0009	`PnPat`	pen pattern	32	pen pattern (8x8 pixels)
$000A	`FillPat`	fill pattern	32	fill pattern (8x8 pixels)
$000B	`OvSize`	oval size	4	oval size (point)
$000C	`Origin`	origin	4	origin (point)
$000D	`TxSize`	text size	2	text size (word)
$000E	`FGColor`	foreground color	2	color (word)
$000F	`BGColor`	background color	2	color (word)
$XX11	`Version`	version	0	none: high byte = version (currently $82)
$0012	`ChExtra`	character extra	4	char. extra (fixed)

$0013	`PnMask`	pen mask	8	mask (8 bytes)
$0014	`ArcRot`	arc rot	2	Reserved (related to things drawn w/patterns). (word)
$0015	`FontFlags`	font flags	2	font flags (word)
$0020	`Line`	line	8	`pnLoc` (point), `newPt` (point)
$0021	`LineFrom`	line from pen loc.	4	`newPt` (point)
$0022	`ShortLine`	short line	6	`pnLoc` (point), `dv`, `dh` (signed bytes)
$0023	`ShortLFrom`	ditto from pen loc	2	`dv`, `dh` (signed bytes)
$0028	`LongText`	long text	5+text	`txLoc` (point), `count` (byte), text
$0029	`DHText`	hor. offset text	2+text	`dh` (unsigned byte), `count` (byte), text
$002A	`DVText`	vert. offset text	2+text	`dv` (unsigned byte), `count` (byte), text
$002B	`DHDVText`	offset text	3+text	`dv`, `dh` (unsigned bytes), `count` (byte), text
$002C	`RealLongText`	very long text	6+text	`txLoc` (point), `count` (word), text

Opcodes between $0030 and $008C are a combination of a graphic verb and a graphic object, as listed below (where “V” stands for the graphic verb, and “X” is a stands for the graphic object). For example, $0069 means `PaintSameArc`, and is followed by two one-word parameters.

Graphic Verbs:

$00X0	`Frame…`	frame something [Specific to object type: see below.]
$00X1	`Paint…`	paint something
$00X2	`Erase…`	erase something
$00X3	`Invert…`	invert something
$00X4	`Fill…`	fill something
$00XV+8	`…Same…`	draw same thing somehow [See below; underlined parms do not appear.]

Graphic Objects:

$003V	`…Rect`	draw a rectangle somehow 8 (0 if – SameRect) rect (2 points)
$004V	`…RRect`	draw a round rect somehow 8 (0) rect (2 points)
$005V	`…Oval`	draw an oval somehow 8 (0) rect (2 points)
$006V	`…Arc`	draw an arc somehow 12 (4) rect (2 points), start, arc angle (words)
$007V	`…Poly`	draw a polygon somehow [polygon size] (0) polygon
$008V	`…Rgn`	draw a region somehow [region size] (0) region
$0090	`BitsRect`	copybits, rect clipped variable† (see below, but without `maskRgn`)
$0091	`BitsRgn`	copybits, rgn clipped variable† (see below)
$00A1	`LongComment`	long comment 4+data kind (word), size (word), data

†Bits… data:

`origSCB`	original scan line control byte	2	SCB (word — high byte = 0)
`BWvsColor`	black and white vs. color	2	reserved (word)

`width`	width of pixel image in bytes	2	width (word)
`boundsRect`	bounds rectangle	8	rect (2 points)
`srcRect`	source rectangle	8	rect (2 points)
`destRect`	destination rectangle	8	rect (2 points)
`mode`	transfer mode	2	pen mode (word)
`maskRgn`	mask region (BitsRgn ONLY!)	[region size]	region
`pixData`	pixel image	[pixdata size]	width*(bounds.bottom-bounds.top)

Differences Between IIGS Pictures and Macintosh Pictures

1. QuickDraw II pictures are modeled after `PICT2` on the Macintosh, which use two bytes for its opcodes and data (the exception to this is the $11 (version) opcode, which is followed by a one-byte parameter). Macintosh `PICT` 1.0 formats, which use one-byte opcodes, would have to undergo extensive modifications to be displayed on the IIGS.
2. There is no `EndOfPicture` opcode on the IIGS as there is on the Macintosh. Also, the first word of the picture is a `pictSCB`, not the length of the picture. The picture size is determined solely by the size of the handle on the IIGS. There is also no picture header on the IIGS as on the Macintosh.
3. The number sex of the Macintosh is opposite that of the Apple IIGS. The Macintosh stores the high bytes of words and long words first, whereas the IIGS stores the low byte first.

4. The following Macintosh picture opcodes are not available on the IIGS: `txRatio`, `PackBitsRect`, `PackBitsRgn`, `shortComment`, `EndOfPicture`.
5. QuickDraw II defines the following opcodes that the Macintosh does not: `ChExtra` ($12), `PnMask` ($13), `ArcRot` ($14), `FontFlags` ($15), and `RealLongText` ($2C).

Notes on the Interpretation of IIGS Pictures

- The state of the pen, the clip region, various patterns and colors, and the origin of the current port is saved before a picture is drawn, and restored afterwards. The current port is set up in a default state equivalent to that of a newly created port just before drawing begins. Picture opcodes act just like their QuickDraw II tool counterparts, with a few exceptions.
- Two pen locations are tracked as the picture is drawn, one for lines and one for text. Thus, `LineFrom` always draws from the end of the last line, regardless of any intermediate text opcodes.
- Text calls do not change the position of the "text pen," as do normal QuickDraw II text calls. Thus, if a picture contains two lines of text, the second one directly below the first, the second will be stored using a `DVtext` opcode.
- `DrawPicture` performs considerable setup before it draws pictures. Among other things, it calls `InstallFont`, which is a Font Manager call. If you are going to support pictures in your application, you should load and start the Font Manager.

Further Reference

- *Apple IIGS Toolbox Reference*, Volume 2