PRINTING

From:	
To:	ToolKit Users
Date:	12 May 1984 -- Draft: 00:30
Subject:	ToolKit Printing - Notes

Printing in the ToolKit is organized around the concept of the View. Every panel's View is either "printable" or "not printable". The actual code to image a view on the printed page is the same TView.Draw which is used to image the view on the screen; the ToolKit takes care of differences between the resolution of the printer and that of the screen.formatted

Printing issues for a printable view are handled by its **PrintManager**. Each printable view in an application has its own PrintManager, but usually an application which has more than one printable view will use an instance of the same TPrintManager subclass for each of its printable views.

When creating a view (using TPanel.NewView or, in special cases, TView.CREATE), the application specifies the PrintManager to be used with it. Specifying NIL for the PrintManager indicates that the view is not to be printable.

If a view is to be printable, there are three basic choices for its PrintManager:

(1) The vanilla printManager, obtained by calling TPrintManager.CREATE -- this gives you printing but no facility for specifying headings or margins.

(2) The 'standard' printManager, of class TStdPrintManager, obtained by calling TStdPrintManager.CREATE. This printManager gives the user of the application an interactive Headings and Margins facility. An application wishing to use this PrintManager must USE the Dialog Building Block, UDialog.

(3) If an application is not content with either of the above, it can define its own subclass of TPrintManager (or of TStdPrintManager).

Class:	TPrintManager
SuperClass:	TObject
Defined Subclass:	TStdPrintManager (defined in the Dialog Building Block)

Data Fields:

view:	TView	The view whose printing is managed
pageView:	TView	The view which draws the headings in the margins of the page (the size of this view is the size of *one page*)
breaks:	ARRAY[vhSelect] OF TArray {of LONGINT}	The vertical and horizontal pagebreaks which partition the view up into pieces which will fit onto individual pages. (note: breaks[v] holds information for page breaks which, when drawn on the screen, are represented by vertical lines). The representation is: the absolute value gives the location in the view; the sign tells whether the break is an automatic one (nonnegative) or a manual (user-set) one (negative)
pageMargins:	LRect	The page margins to use when fitting view pieces onto pages; top and left are positive and bottom and right are negative. These

		numbers are in View resolution, hence the LRect even though the actual numbers involved will be very small.
headings:	TList {of THeading}	The headings to be printed on the pages
canEditPages:	BOOLEAN	FALSE by default; the StdPrintManager sets this true, since headings and margins can be edited with it.
layoutDialogBox:	TDialogBox	NIL by default; the StdPrintManager places a reference to a dialog box it creates here.
frameBody:	BOOLEAN	FALSE by default; client can set to TRUE directly, in which the page body will be separated from the page margins on the printed page by a box. Mostly for debugging purposes.
paperLRect:	LRect	The physical bounds of a single page on the printer currently formatted for, in view resolution.
printableLRect:	LRect	The printable area of a single page on the printer currently formatted for, in view resolution.
contentLRect:	LRect	The area on a page into which the body (i.e., chunks of the view) will be stuffed, obtained by taking the paperLRect and insetting it by the amount specified in the pageMargins; in view resolution.
printerMetrics:	TPrinterMetrics	The undigested numbers characterizing the physical properties of the printer currently formatted for; this is a Record whose fields are: paperRect, printableRect (the conterparts of paperLRect and printableLRect, but in device coordinates), and res (the resolutions of the device, packaged into a Point).
pageRiseDirection:	vhSelect	Whether page numbers should be rise fastest left-to-right (h) or top-to-bottom (v)

<u>Methods client may wish to call directly</u>:

PROCEDURE ChangeMargins(newMargins: LRect)	Install a new set of margins into the printManager

<u>Methods client may wish to redefine in a subclass</u>:

PROCEDURE DrawPage	Draw the page whose page-number is indicated by theMarginPad.pageNumber.
PROCEDURE EnterPageEditing	Enter into whatever method the printManager has of allowing the user to edit page headings and margins
PROCEDURE GetPageLimits(pageNumber: LONGINT; VAR viewLRect: LRect)	Given the page number, returns the bounds of the subset of the view which will be mapped into that page.
FUNCTION NewPaginatedView(object: TObject): TPaginatedView	Launch a Paginated View. Default: the ToolKit's standard Page-preview mode is entered into.
FUNCTION NewPageView(object: TObject): TView	Launch the PageView to be used. Default: an object of the ToolKit's standard TPageView class is launched.
FUNCTION PageWith(VAR lPtInView: LPoint; VAR strip: Point): LONGINT	Given a point in the view, returns the page number into which that point falls as the value of the function, and also returns, in 'strip', which vertical and horizontal strip of pages the point falls into.
PROCEDURE Print	Does the actual printing. Client might subclass in order to do something special before or after printing, while deferring to SUPERSELF for the actual printing; clients wishing to do TPrintManager.Print all by themselves must have direct access to the underlying Lisa Printing software, as well as a sense of adventure.
PROCEDURE ReactToPrinterChange	Reacts to change in printer specification; the generic TPrintManager.ReactToPrinterChange gets fresh printer

	metrics, recomputes the printableLRect, paperLRect, and contentLRect, resizes the View if necessary, recomputes the bounds of the PageView, and calls RedoBreaks.
PROCEDURE RedoBreaks;	Recompute page-breaks
PROCEDURE SetDfltHeadings;	Sets the default headings; called by TPrintManager.Init so that the printManager can create desired default headings and install them in its list of headings.
PROCEDURE SkipPage(pageNumber: LONGINT)	Do whatever the application wishes to if it knows that the indicated page-number, at printing time, is not going to be printed (but some large page-number will be); this allows an application to cycle through its data structures to the start of the next page, if it needs to. The generic TPrintManager.SkipPage does nothing.

Miscellaneous notes and clarifications

Printable Views and the "Format For Printer..." dialog

The Lisa User Interface Standard specifies that a document be formatted for one particular printer at any one time. The record which contains information about the printer formatted for is called a "Print Preference", and such an object is stored in TDocManager objects, under the alias type 'TPrReserve'. A default specification is provided by the generic TDocManager.CREATE method. In a client's own DocManager.CREATE, client after calling SUPERSELF.CREATE may want to change details of this default print preference, but this requires sinful knowledge of the innards of this record, coerced from type TPrReserve to TPrPrf.

More commonly, an application accepts the default print preference, and its user changes the specification through the 'Format For Printer...' dialog. (Unusually, the specification can be changed by the user at print time, if he has chosen to defer print arbitration until then).

No matter how many printable view an application may have in its window, the same Print Preference applies to all of them. Consequently, whenever the Print Preference changes, every view needs to be informed of the change, and the channel for this is the method TView.ReactToPrinterChange. Any printable view will, when it receives this message, call its own printManager's own method of the same name. TPrintManager.ReactToPrinterChange is responsible for taking all actions necessary when the Print Preference changes. These actions typically include recomputing page breaks and possibly resizing the view so that it fits perfectly in an exact number of pages.

About Page Breaks

Page breaks are imaginary lines which divide a view up into a checkerboard pattern, which each square of the checkerboard allocated to a different printed page.

The print metrics of the current Print Preference determine the physical size of the printed page, as well as the printable subarea (some printers cannot image to the edge of the page).

TPrinterMetrics, in which a printer's metrics are stuffed, contains the resolution of the printer device (packaged into a Point, called res), as well as two Rectangles: paperRect and printableRect,.which give, in device resolution, the physical and printable areas of a page on the printer.

More useful usually will be the same rectangles transformed for convenience into View coordintes (which are usually the same as screen coordinates). These will be found as fields TPrintManager.paperLRect and TPrintManager.printableLRect.

The printManager's printableLRect and printerMetrics' printableRect are provided for the information of the application, but play no direct role in computation of pagebreaks. The application may wish to take into account the printable limits of the page when determining which page Margins to use -- if margins are made too small, an attempt may be made at print time to image in a portion of the paper which the printer is incapable of laying down ink.

The printManager has another rectangle, called its **contentLRect**, obtained by subtracting it's pageMargins from its paperLRect. This is the 'body' of the page.

The default algorithm for determining page breaks is to allocate as much of the view to the page as will possibly fit within its contentLRect.

For a graphical application, the default algorithm will probably be entirely adequate. For other applications, the nature of the view may dictate that the boundary between what goes on one page and what goes on another be determined by using special information. For example, a Word Processor will normally not want to have the split from one page to the next fall in the midst of a line. Similarly, a Spreadsheet application will normally want to have a new column or row of data start on a fresh page, rather than having a column or row split between two adjacent pages.

The mechanism for the Application to communicate such special pagebreak location information is through the function TView.ForceBreakAt, whose interface is:

```
    FUNCTION TView.ForceBreak(vhs: VHSelect; precedingLocation: LONGINT; proposedLocation: LONGINT):
        LONGINT;
```

The idea here is that the ToolKit, with due consideration of the contentLRect, proposes that the pagebreak in the **vhs** direction which follows the one at **precedingLocation** be located at **proposedLocation**. The generic TView.ForceBreakAt returns proposedLocation as the function value. This means that the default is to set the pagebreak just where the ToolKit proposed to locate it.

Note that for pagebreaks, the vhs field gives the orientation of the actual line drawn on the screen to represent the pagebreak. Hence, breaks[h] is a the array of pagebreaks which are drawn with Horizontal lines, which hence serve to separate vertically adjacent pages.

The Application's view is expected to redefine ForceBreakAt if it is not satisfied with the generic ForceBreakAt. Note that unless it is really looking for trouble, an application's ForceBreakAt should return a value which is greater than precedingLocation and which does not exceed proposedLocation. Giving a value less than precedingLocation is palpable nonsense, whereas giving a value greater than proposedLocation would lead to trying to stuff more onto a page than the page has room for.

Resizing Views — a Caution

A view's original size is determined by the value given to its extentLRect when calling TView.CREATE or TPanel.NewView; its size can be changed at any time by calling TView.Resize, giving the new extentLRect as an argument. But applications are cautioned that TView.Resize does NOT take care of some of the printing—related issues that it might think it does. In particular, it deletes old superfluous breaks which now fall outside the new extentLRect, and it changes the end—of—view sentinels which form the final elements of each of the printManager's breaks arrays. But it does NOT recompute page breaks. If you wish pagebreaks to be recomputed, be sure to call view.RedoBreaks after calling view.Resize. Better still, for a more comprehensive effect, you can call view.ReactToPrinterChange, which, even though the printer may not just have changed, will assure that a complete rethinking of all printer—related issues takes place.