Adobe 'Marked Objects' plugin for WaRMreader

Wendy McKay
Control and Dynamical Systems, CalTech, Pasadena, USA
wgm@cds.caltech.edu
http://www.cds.caltech.edu/ wgm/

Ross Moore
Mathematics Department, Macquarie University, Sydney
ross@maths.mq.edu.au
http://www.maths.mq.edu.au/r̃oss/

Thomas Ruark
Adobe Systems Inc., San Jose
truark@adobe.com

Abstract

The WaRMreader [4] method of using XY-pic for placing labels over imported graphics was presented at TUG'99* in Vancouver [3]. Central to this method is the use of a `.bb` file, which contains information concerning "marked points" within the graphic, as well as specifying the bounding box.

A plugin module has been developed, for use with Adobe's *Illustrator* [2] (vers. 9, and later), which makes it easy to specify the desired marked points, and store the corresponding information within a `.bb` file. This information is valuable markup which could be used also for different purposes, with other software packages.

In this talk we demonstrate some possible work-flows for using the new plugin tool. Also, we describe the work done to convince the *Illustrator* Development Team, at Adobe Systems Inc., that such a tool is a simple and useful addition to their software.

Using the WaRMreader macros

The WaRMreader [4] method is an extension of XY-pic's `\xyimport` command, that facilitates placement of labels over imported graphics. It works by having knowledge of the location of interesting places in the imported graphic, stored within a separate file in an easily readable text-based format. Since this file must also contain the bounding box information for the graphic, we refer to it as the `.bb` file, even though filename extensions other than `.bb` can (and should) be used, according to the format of the information within the file.

Back in 1999, when WaRMreader was presented at TUG'99, there was no convenient way for automatically generating a `.bb` file from graphics software on most computing platforms. In this paper we describe a new plugin module for Adobe's *Illustrator* [2] software that provides an intuitive windowed interface for the easy creation of such a file, by inserting 'marked points' and other associated information. Before showing how this works, within the *Illustrator* program, we first revisit how the information is used within a (LA)TEX document

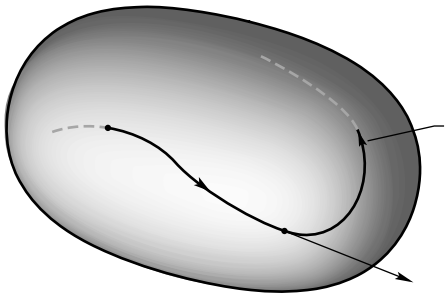

Figure 1: Image created with Adobe's *Illustrator* program, with no visible labels. Such labels will be added later, when the image is included within a LATEX document.

```
%%Creator:Adobe Illustrator 9.0, Macintosh 2000
%%Title: Fig8_2_1.eps
%%Date: 7/6/2001 7:16:6 PM
%%BoundingBox: 0 0 165 108
%%Coordinates: LL
%%StartMarkedPoints
%%MarkedPoint:(144,75) : point(0,0) : M %M, manifold
%%MarkedPoint:(164,62) : point(0,0) : traj %trajectory of\\fluid particle\\moving through\\$x$ at time $t=t$
%%MarkedPoint:(151,4) : point(0,0) : u %u(\varphi_t(x),t)
%%MarkedPoint:(105,22) : point(0,0) : phi %\varphi_t(x)
%%MarkedPoint:(38,61) : point(0,0) : x %x
%%EndMarkedPoints
```

Figure 2: Contents of the file `exampl.bb` defining the ‘marked points’ for `exampl.eps`, as shown in figure 3.

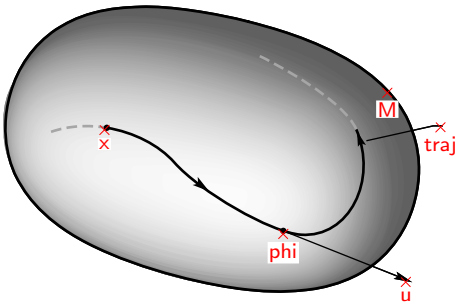

Figure 3: This is the image in figure 1, now showing the locations of ‘marked points’ whose coordinates and ID names have been read from a `.bb` file(`exampl.bb`).

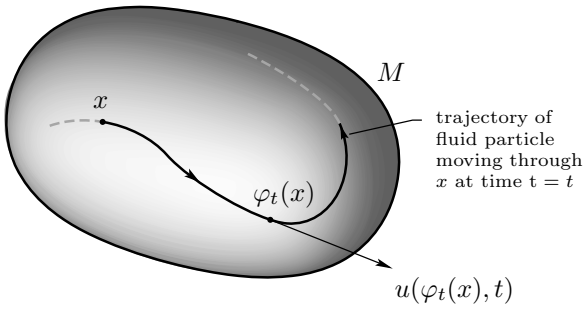

Figure 4: Imported image, fully annotated with meaningful labels, typeset by TEX and positioned using the intuitive XY-pic notation, as in figure 5.

```
\begin{center}
\def\Fname{exampl}%
\renewcommand{\xyWARMinclude}[1]{%
  \includegraphics[scale=.95]{#1}}
\begin{xy}
 \xyWARMprocessMoEPS{\Fname}{pdf}
 \xyMarkedImport{\Fname}% repeat name for pdfTeX
 \xyMarkedPos{x}*++!D{x}
 \xyMarkedPos{phi}*++!D!L(.3){\varphi_t(x)}
 \xyMarkedPos{M}*+!LD{M}
 \xyMarkedPos{u}*+!UL{u(\varphi_t(x),t)}
 \xyMarkedPos{traj}*++!L!U(.5){\parbox{6em}%
 {\scriptsize  trajectory of\\fluid particle\\
  moving through\\$x$ at time $\mathrm{t}=t$}}
\end{xy}
\end{center}
```

Figure 5: This is the LATEX coding for the placement of labels, as in figure 4, over the imported graphic of figure 1.

for placing nicely typeset labels over an imported graphic image. This will help in understanding the nature of the information that needs to be provided within the `.bb` file, and consequently gives a motivation for the operations that can be performed with the new plugin tool and its associated windows and dialogs.

Figure 1 shows a mathematical diagram created using the *Illustrator* software. The need for explanatory labels is clear, but just which parts of the image should be labelled, and what these labels should say, is dependent upon the context in which the image is to be used. In figure 3 several important features of the diagram are marked with small crosses, each with an identifying string. These crosses and identifying names correspond to the information in the `.bb` file, listed in figure 2.

A possible set of labels is shown in figure 4, using the XY-pic environment and coding shown in figure 5.[2] Essentially the same coding will work in Plain TEX, or other TEX format, by using macros `\xy ...\endxy` as delimiters, instead of the LATEX `xy` environment. Also another macro for importing the image could replace `\includegraphics`, provided that this macro creates a box of the correct size to exactly contain the image.

Placement of the labels is best done using a `\xyMarkedPos` command, defined in `warmread.sty`,

[2] The 2nd argument to `\xyWARMprocessMoEPS` is ‘`pdf`’ when the processor is pdf-LATEX and `exampl.pdf` has been created from `exampl.eps`. When the processor is LATEX+dvips (or other driver) this would be ‘`eps`’ with the imported graphic named `exampl.eps`.

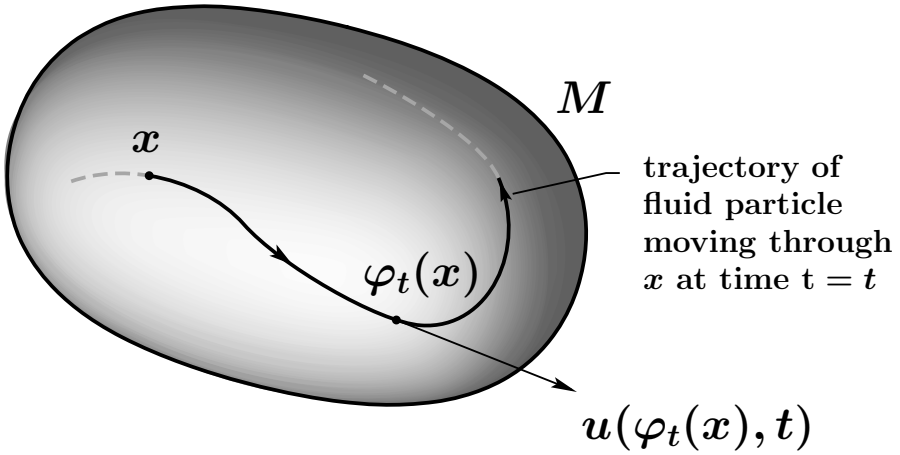

Figure 6: The imported graphic has been resized and the styles used for the labels has been made larger and heavier. Yet the coding to position the labels over the graphic is essentially unchanged.

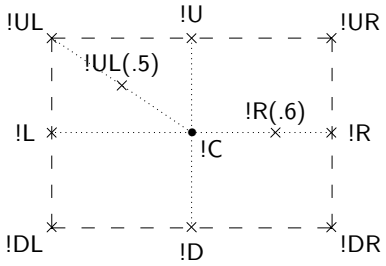

Figure 7: XY-pic modifiers corresponding to the center (!C), the edges (!L, !R, !U, !D), all 4 corners (!UL, !UR, etc.), and arbitrary locations (!R(.6) etc.), to be the anchor point for positioning of an object over a marked point.

for each marked-point using its identifier. There is one line of coding for each label placed, with the TEX coding for the label itself as the `{....}` on each such line, assuming math mode. Detailed positioning of the labels is achieved using short XY-pic 'modifier" strings. For example, with the `u` point, the `!UL` means that the Upper-Left corner of the rectangle containing $u(\varphi_t(x), t)$ should be at the location of the marked point. In fact the preceding `+` (within `*+!UL`) has "grown" the rectangle to include a margin of 3pt on all sides; it is the upper-left corner of this enlarged rectangle that is positioned exactly over the marked point. For more details, consult the XY-pic Reference Manual [5].

Although the WaRMreader [4] macros provide some alternative ways to position labels, the above method using XY-pic modifiers is the most flexible for ensuring that labels can be easily positioned over parts of the image which are both near to the feature being labelled, and are relatively clear of other features prominent within the graphic. For example, the larger image with bold-faced labels, shown in figure 6, uses exactly the same coding as in figure 5 to position all the labels. That heavier style of labelling is more suited for display as a lecture slide, whereas the lighter style of figure 4 is better suited to a journal or book. (Indeed the image in figure 4 was scaled to 95%, so that after adding labels the whole figure would still fit snugly into the 2-column format of this journal.) The difference in coding is simply to precede the `xy` environment with LATEX and XY-pic style-changing commands, as follows:

```
\def\Fname{exampl}%
\mathversion{bold}\LARGE\bfseries
\renewcommand{\xyWARMinclude}[1]{%
  \includegraphics[scale=1.3]{#1}}%
\objectmargin{5pt}%
\begin{xy}
 ...
```

and to change the `\scriptsize` within the `\parbox` to `\large`, for one of the labels.

The primary advantages of using this method to label graphics are as follows.

- The style, size, and font-face of labels placed over imported images necessarily matches that used in the surrounding document.
- Since labels are added within the LATEX source, the wording of labels can be changed without need to alter the underlying graphic in any way.
- The same image can be used in many settings, with different sizes, styles or text for the labels;

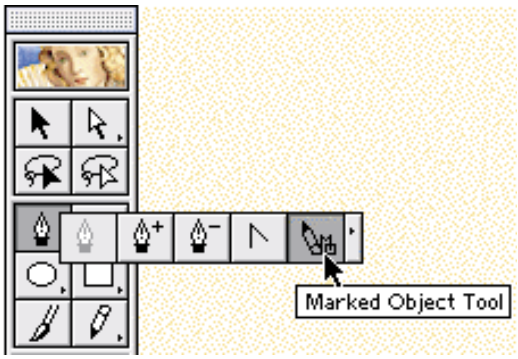

Figure 8: The Pen tool pop-out has a new icon, corresponding to the ‘Marked Objects’ point selector.

there is no need to make any edits within the image file itself.

A subsidiary advantage is that the method of labelling is unaffected by compatibility problems in font technologies within the software used to create the graphic images. For example, there are known problems with the use of Computer Modern fonts in artwork created with earlier versions of *Illustrator*. When such images are viewed in recent versions of Adobe software, either font substitutions occur, or some glyphs may fail to appear at all. With the WaRMreader method such problems become irrelevant, since no text fonts are needed at all within the imported images.

Adobe ‘Marked Objects’ plugin tool

In response to a request from Wendy and Ross, Tom Ruark has been working on a plugin module for Adobe *Illustrator* which greatly simplifies the task of constructing a `.bb` file while composing artwork using that program. This work is not yet complete, but a beta version of the plugin is available at Wendy’s site.[3] There is one variant for Macintosh and another for Windows-based platforms.

When the plugin is installed (in the sub-folder named `Plug-ins/tools/` within Illustrator’s installation folder) an icon for an extra tool (MO-pen) appears within the pop-out ‘Pen Tool’ menu, see figure 8. When selected, the cursor changes; any click within the artwork canvas creates a ‘Marked Object’ within a new non-printing layer. Selecting ‘Show Marked Objects window’ from the ‘windows’ menu, reveals a window that allows properties of marked objects to be seen and edited. Figure 9 shows a view of this dialog, along with the ‘Layers window’ showing the presence of the special layer.

As a new marked point is created, by a mouse-click with the MO-pen tool, it is allocated a default ID which is a count of the number of marked objects that have been created. (This count also includes any that have been subsequently deleted.) Using the Marked Objects window and selecting the list entry for the marked object, this ID can be adjusted to something that is more meaningful within the particular artwork. Also a text-string may be associated with the marked point. This allows a short meaningful description of the place being marked to be included with the artwork. These strings are located in the special layer, so will normally be non-printing, however the layer can be made printing if desired; e.g., for editing purposes. Furthermore, by click and drag with the usual selection tool, these string objects can be shifted around within the artwork for clarity. The font and other attributes of these strings can be adjusted to taste; this includes altering the default position for the location of these strings relative to the marked point. See figure 10 for an example showing the appearance of some artwork, both when the marked objects are hidden and when visible. In future it may become possible to specify also a shape and size for marked objects, other than just points.

Saving the marked point information in a `.bb` file occurs automatically when the artwork is saved.[4] However a variant of the `.bb` file can be saved at any time, with an arbitrary filename prefix, using a file dialog accessed from the Marked Objects window, as shown in figure 11.

Getting help from Adobe

When Wendy first approached Adobe with the proposal for a plugin to create `.bb` files, Tom requested that she provide a work-flow for how the module would be used. Figure 12 shows such a work-flow. As well as this, Wendy and Ross designed a possible interface to the plugin tool, based on extending existing interfaces within *Illustrator*. The realisation was that the kind of coordinate information required in the `.bb` file is similar to what is needed for creating image-maps for graphics on web pages. That is, it was recognised that all the information that might be needed for marked objects should be already available (at least internally) for artwork being edited with *Illustrator*. To emphasise this, the original proposal suggested that the ‘Marked

[3] http://www.cds.caltech.edu/~wgm/WARM/adobe/

[4] The current version of the plugin is slightly buggy in that it always saves a `.bb` file, even when there are no marked objects.

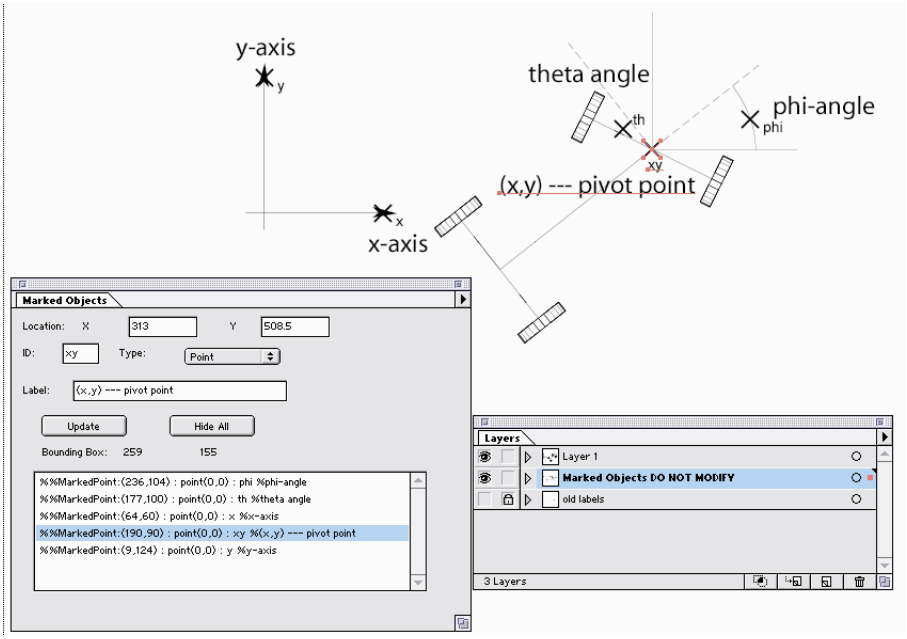

Figure 9: The 'Marked Objects' window lists all the marked points and allows their locations and other properties to be adjusted. Marked objects are created within a separate layer which should be 'hidden' and made non-printing before the artwork is saved into an `.eps` file.

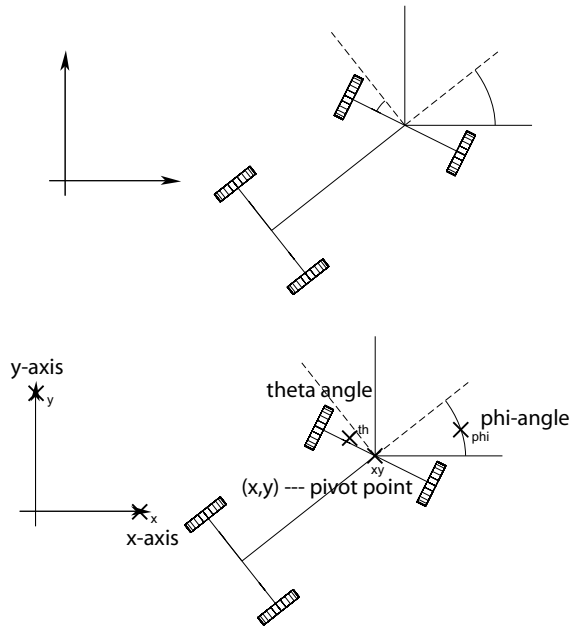

Figure 10: The upper image shows some artwork with the Marked Objects layer hidden; i.e., as it would be seen when imported into a LaTeX document. The lower image shows also the marked objects with their IDs and associated strings.

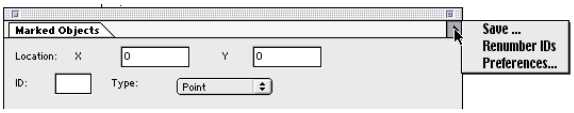

Figure 11: Saving a `.bb` file happens automatically, or can be done at any time from the 'Marked Objects' window, using the menu item shown here.

Object' interface could be developed as extensions to the existing interfaces showing 'Document Info' and other 'Attributes'. These suggestions are illustrated in figures 13 and 14, which accompanied the workflow (in figure 12) for the proposal.

In fact Adobe's *Illustrator* team of developers decided that a separate interface would be best. Programming this should not present great difficulty, as indeed all the APIs necessary to collect the required information were already available within the existing code-base for the *Illustrator* program. Sure enough, Tom was able to produce the first attempt at a working plugin within a very short time. Subsequent revisions have been to improve the interface and to fix bugs that have emerged during testing.

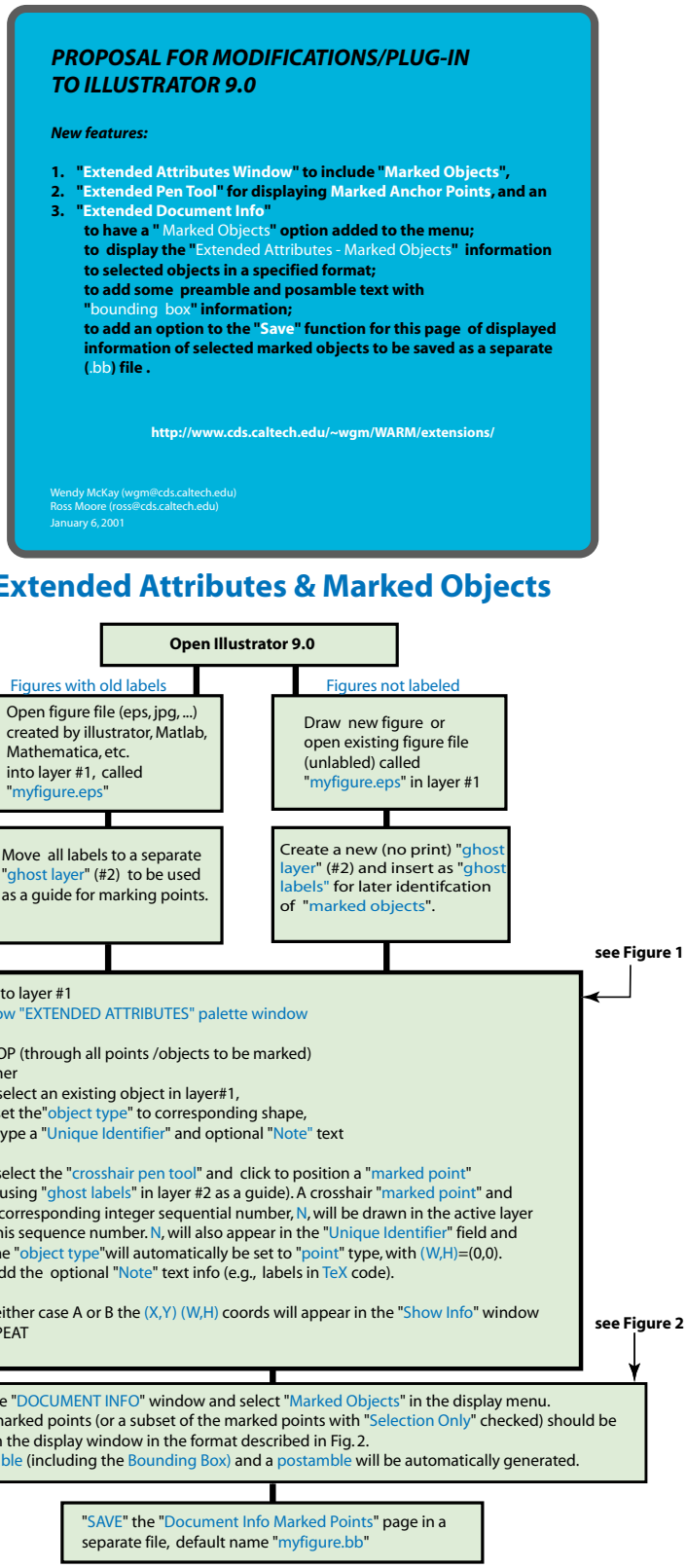

Figure 12: Reproduction of the proposal to produce a plugin module for Adobe's *Illustrator* software. Top: front-page with brief summary. Bottom: workflow, for how the plugin tool might typically be used.

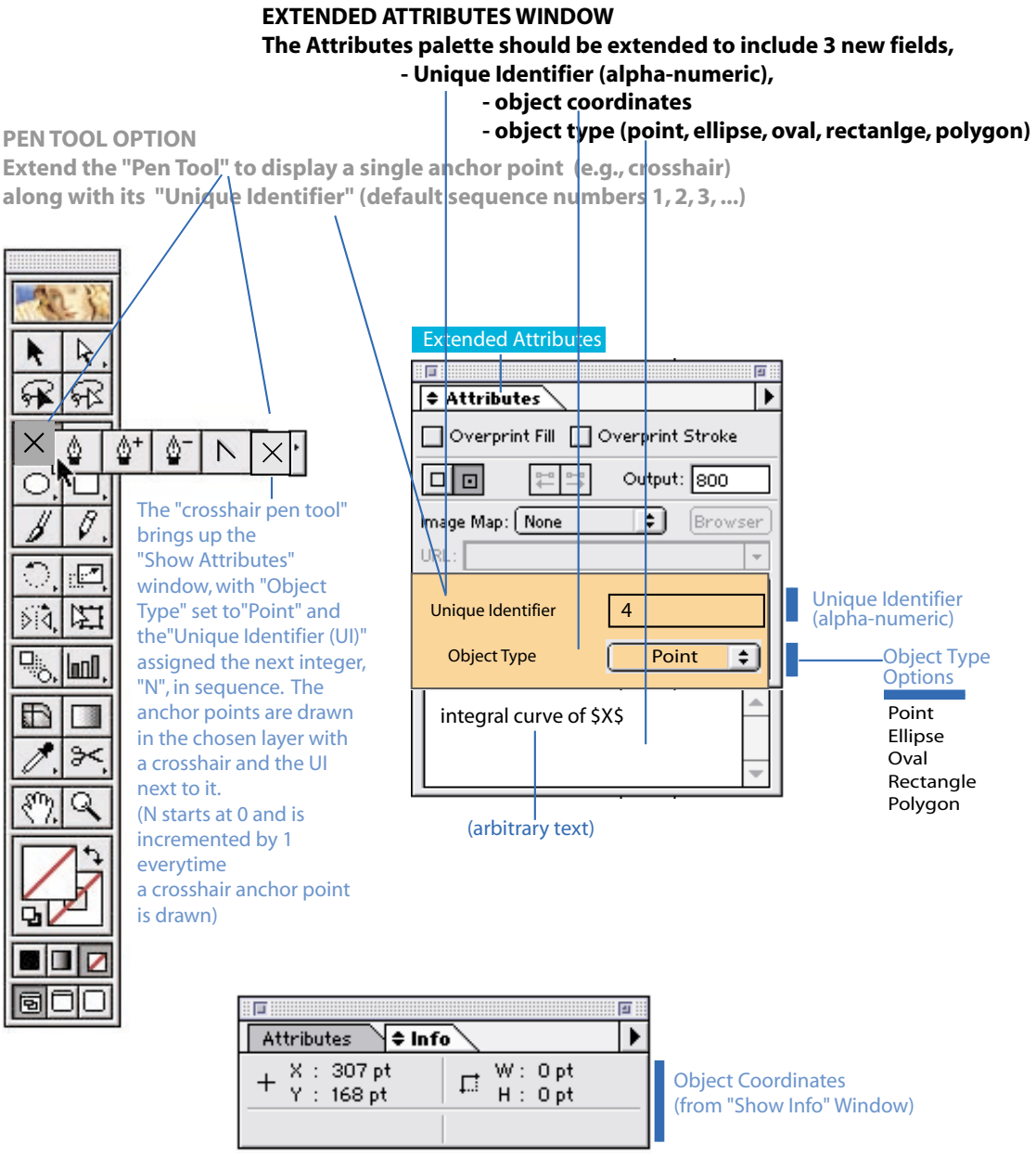

Figure 13: Reproduction of Figure 1 from the proposal to produce a plugin module: extensions to the pen-tool and 'Attributes' window. In the actual module the MO-pen icon was redesigned. These extensions to the windows, and that to the 'Document Info' window shown in figure 14, were combined into a single new 'Marked Objects' window; see figure 9.

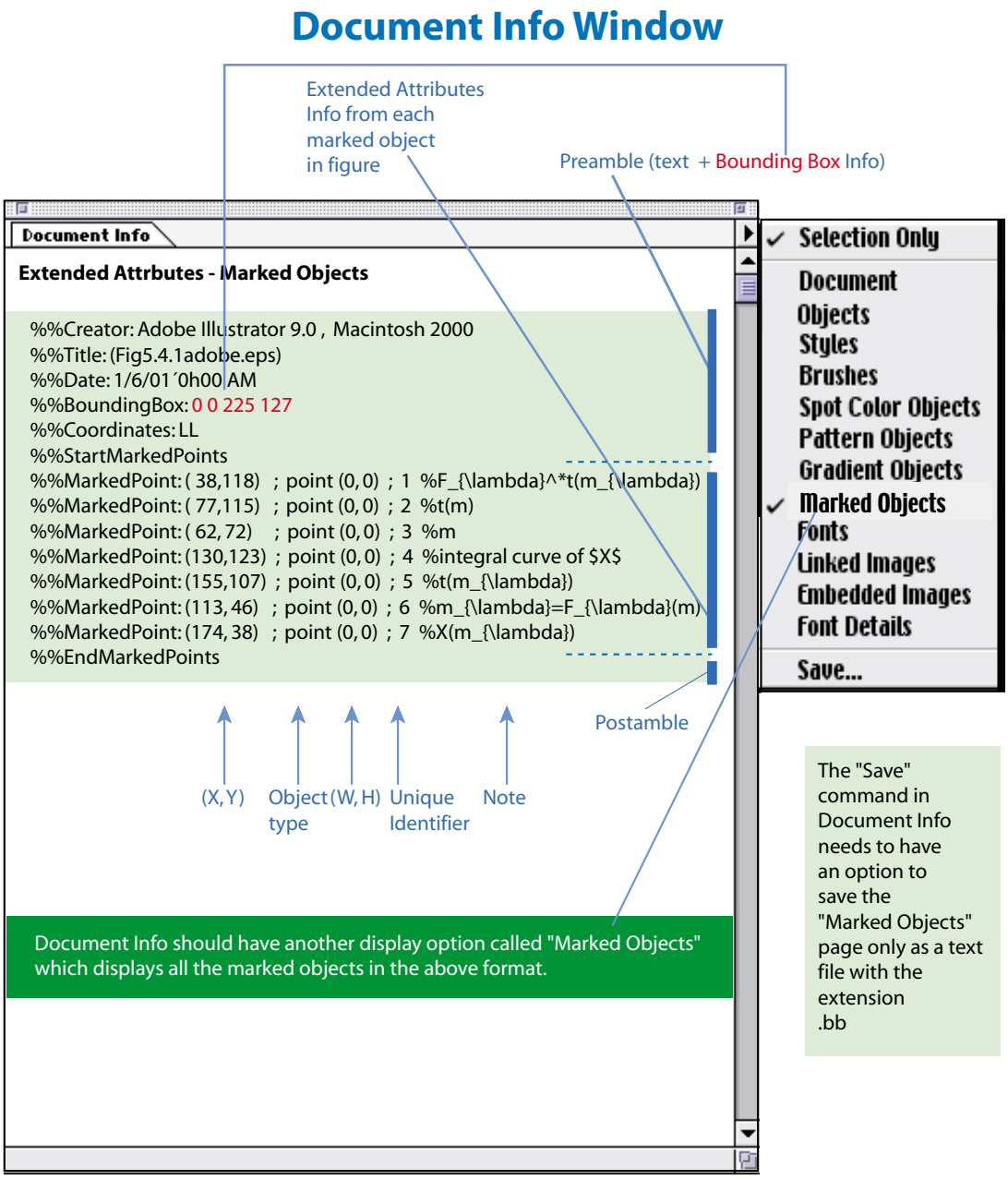

Figure 14: Reproduction of Figure 2 from the proposal to produce a plugin module: proposed extensions to the 'Document Info' window, to display information about marked points. These extensions were not implemented as shown, but were combined with the coordinate data and presented in the 'Marked Objects' window, as shown in figure 9.

Known bugs and workarounds

As mentioned earlier, the plugin module is still under development, mainly to ensure that the interface is reliable and easy to use. Some 'bugs' are known to exist. Mostly these can be worked around, with advance knowledge of what can go wrong.
The following problems have been encountered while working with *Illustrator* and the Marked Objects plugin.

bounding-box wrong, by large amount:
This occurs when marking points in graphics originally created using older versions of *Illustrator*. These can have a bounding-box whose bottom-left corner is not at the origin of coordinates $(0, 0)$, which is usual for `.eps` files created in *Illustrator* version 9.0 and later.

One 'fix' is to open the corresponding `.eps` file in a text editor, copy its %%BoundingBox, then paste this into the `.bb` file to replace the incorrect information there.

A better 'fix' is to create a new document for artwork in *Illustrator* 9, then 'Select All' objects in the older artwork. Copy and paste these into the new document. Do all marking of points in the new version only. (Note that it is *not* sufficient to do a 'Save As...' version 9.0 update of the older artwork. This will *not* make any difference to its %%BoundingBox, so the problem will persist.)

bounding-box wrong, by small amount:
This occurs when there are hidden layers in the artwork, other than the special Marked Objects layer, in which there is an object that lies (at least partially) outside the bounding-box of the unhidden layers. The %%BoundingBox recorded in the `.bb` file encloses the hidden objects, whereas that for the `.eps` file only covers objects in the unhidden layers.

The fix is to copy the %%BoundingBox comment from the `.eps` file into the `.bb` file, as in the first 'fix' above.

absurd coordinates for a marked point:
4- or 5-digit numbers, perhaps negative, lying clearly outside the boundary of the artwork, have been observed to occur as coordinates for marked-points. This appears to happen only with artwork created with earlier versions of *Illustrator*, after an ID or attached string is moved manually.

The above solution of copying all the artwork into a new *Illustrator* 9 document should ensure that this problem does not occur.

badly sorted numerical IDs:
This occurs when the default numerical IDs for marked points are never changed. Then the order will be lexicographic; e.g., 1, 10, 11, 12, 2, 3, 4, 5, ...

It is better to use alphanumeric identifiers, which have names that are meaningful to the places being marked.

Acknowledgement

Two of the authors are deeply indebted to Professor Jerrold Marsden for continued support in the development of WaRMreader and other TEX-related software projects. The images used as examples for labelling graphics have come from one [1] of the many mathematical texts which Jerry has either written or co-authored.

References

[1] Abraham, R., J.E. Marsden and T. Ratiu; "Manifolds, Tensor Analysis and Applications". Applied Mathematical Sciences Series, Vol. 75; Springer–Verlag, New York, (1988) 654pp.

[2] Adobe Systems Inc.; Adobe *Illustrator* 9 (latest version 9.02); purchase online from http://www.adobe.com/products/illustrator/main.html.

[3] Moore, Ross and Wendy McKay; "Convenient Labelling of Graphics, the WARMreader Way". TUGboat Vol. 20 (3), 1999; pp. 262–271.

[4] Moore, Ross; "What is WaRMreader?", examples, documentation and macros for `warmread.sty`; available online at http://www-texdev.mpce.mq.edu.au/WARM/ and http://cds.caltech.edu/ wgm/WARM/reader2001.html.

[5] Rose K. and R. Moore; XY-pic Reference Manual; accompanies the XY-pic diagram macros for TEX; version 3.7 available from http://www.ctan.org/ctan/tex-archive/CTAN::/macros/generic/diagrams/xypic/.