The Camelot Project

J. Warnock

This document describes the base technology and ideas behind the project named “Camelot.” This project’s goal is to solve a fundamental problem that confronts today’s companies. The problem is concerned with our ability to communicate visual material between different computer applications and systems. The specific problem is that most programs print to a wide range of printers, but there is no universal way to communicate and view this printed information electronically. The popularity of FAX machines has given us a way to send images around to produce remote paper, but the lack of quality, the high communication bandwidth and the device specific nature of FAX has made the solution less than desirable. What industries badly need is a universal way to communicate documents across a wide variety of machine configurations, operating systems and communication networks. These documents should be viewable on any display and should be printable on any modern printers. If this problem can be solved, then the fundamental way people work will change.

The invention of the PostScript language has gone a long way to solving this problem. PostScript is a device independent page description language. Adobe’s PostScript interpreter has been implemented on over 100 commercially available printer products. These printer products include color machines, high resolution machines, high speed machines and low-cost machines. Over 4000 applications output their printed material to PostScript machines. This support for PostScript as a standard make the PostScript solution a candidate for this electronic document interchange.

Within the PostScript and Display PostScript context the “view and print anywhere” problem has been implemented and solved. Since most applications have PostScript print drivers, documents from a wide variety of applications can be viewed from operating systems that use Display PostScript. PostScript files can be shipped around communication networks and printed remotely. “Encapsulated PostScript” is a type of PostScript file that can be used by many applications to include a PostScript image as part of a page the application builds.

The reason the Display PostScript and PostScript solutions are not a total solution in today's world is that this solution requires powerful desktop machines and PostScript printers. The Display PostScript and PostScript solutions are the correct long-term solution as the power of machines increases over time, but this solution offers little help for the vast majority of today's users with today's machines.

The Camelot Project is an attempt to define technologies and products that will give the value that Display PostScript and PostScript delivers to the vast number of installed machines that exists today. For the purposes of this discussion these machines include 640K Intel 286/386/486 machines (PC compatibles), Apple Macintosh machines, mainframes, and workstations. The displays must include CGA, EGA, VGA and any other higher resolution or color displays supported by the above machines.

Our vision for Camelot is to provide a collection of utilities, applications, and system software so that a corporation can effectively capture documents from any application, send electronic versions of these documents anywhere, and view and print these documents on any machines.

There are at least two technical approaches to the Camelot project. Both solutions depend on the PostScript technology. One approach is to try to make Display PostScript and PostScript implementations smaller and faster so that they can run on the vast majority of today's machines. This approach has been tried and is extremely difficult.

A second approach is to divide the problem into smaller problems. This approach would allow each piece to run independently on the smaller machines while achieving acceptable performance and a solution for the complete problem. This latter approach requires that the problem be divided in a way that is natural for users, and provides a solution for every user. An approach to the Camelot project will now be described that will divide the problem into smaller pieces. This solution depends on a unique property of the PostScript language.

PostScript, as an interpretive language, has some properties that other interpretive languages do not have. In particular, the semantics of operators is not fixed. Operators can be redefined to have any desired behavior. This property of PostScript allows the execution of

a PostScript file to have side effects that are very different from the normal printing of a page. An example might be instructive. Suppose a PostScript file draws 10 sided polygon with the following PostScript procedure:

```
/poly
  {1 0 moveto
   /ang 36 def
   10 {ang cos ang sin lineto
       /ang ang 36 add def
      }repeat
   }def
```

This procedure will build a path that is a ten sided polygon. In this procedure the verbs: "`moveto`" and "`lineto`" have the standard semantics of building a PostScript path within the PostScript Language.

By redefining "`moveto`" and "`lineto`" very different things can happen. For example, if these operators are defined as follows:

```
/moveto
   {exch writenumber writenumber (moveto) writestring}def
/lineto
   {exch writenumber writenumber (lineto) writestring}def
```

then when the "`poly`" procedure is executed a file is written that has the following contents:

```
1.0 0.0 moveto
0.809 0.588 lineto
0.309 0.951 lineto
-0.309 0.951 lineto
-0.809 0.588 lineto
-1.0 0.0 lineto
-0.809 -0.588 lineto
-0.309 -0.951 lineto
0.309 -0.951 lineto
0.809 -0.588 lineto
1.0 0.0 lineto
```

In this example the new redefined "`moveto`" and "`lineto`" definitions don't build a path. Instead they write out the coordinates they have been given and then write out the names of their own operations. The resulting file that is written by these new definitions draws the same polygon as the original file but only uses the "`moveto`" and

"`lineto`" operators. Here, the execution of the PostScript file has allowed a derivative file to be generated. In some sense this derivative file is simpler and uses fewer operators than the original PostScript file but has the same net effect. We will call this operation of processing one PostScript file into another form of PostScript file "rebinding."

The above example illustrates a capability of the PostScript language that is not frequently used. This "rebinding" of the language, however, is extremely valuable. The Camelot project depends on variations on this idea.

The approach we will take with Camelot is to define a new language of operators and conventions. For the purposes of this discussion we will call this language "Interchange PostScript" or IPS. IPS will primarily contain the graphics and imaging operators of PostScript. The language will be defined so that any IPS file is a valid PostScript file. The file will have the appropriate baggage so that it is a valid EPS file. IPS files will print on PostScript printer and will be able to be used by applications that accept EPS files. IPS will also be structured so that the complete PostScript parser is not necessary to read any file written in IPS. IPS will have an adequate set of operators so that any practical document expressed in PostScript can be represented in IPS. There will be situations in IPS where the IPS file cannot represent visual situations that can be theoretically generated in PostScript. However we believe these situations are extremely rare, and all practical application documents can be represented efficiently in IPS. The right way to think about IPS is as it relates to English. No person in the world knows every English word, but a small subset of the English words, and certain usage patterns enable people to consistently communicate.

Once we have defined IPS, we will build a version of the PostScript interpreter (IPS binder) that will read any PostScript file and rebind that file into an IPS file. The IPS binder can be quite small in that it does not need the graphics, font or device machinery contained in full PostScript interpreter. Another function of the IPS binder will be to include reconstituted fonts into the IPS file. The idea here is to include just the characters of a font that are actually used in the document. A result of including the necessary characters from the fonts used is that an IPS file will be completely self contained. In other words, when I send a file around the country, I don't have to worry about whether the receiving location has all the fonts required

by the document. The current situation is that complex font substitution schemes are used to deal with locations not having the appropriate fonts.

Once IPS is defined and the IPS binder implemented, then users can capture any PostScript file emitted by a PostScript driver, and convert that file to a self contained IPS file. This file can be shipped anywhere around the network and printed on any PostScript machine (management utilities will be written to ease this printing process.)

In addition to the IPS binder, a viewer and browser will be written that will read IPS files, and render those files on displays or to dumb raster printers. It is believed that IPS interpreters can be substantially simpler, and smaller than full PostScript interpreters. It is also believed that an IPS interpreter can have acceptable performance on small machines. The real hope is to make the IPS viewer and browser small enough so that it can co-exist with other applications. It is interesting to think about what those applications can be.

One obvious application for the IPS viewer is in its use in electronic mail systems. Imagine being able to send full text and graphics documents (newspapers, magazine articles, technical manuals etc.) over electronic mail distribution networks. These documents could be viewed on any machine and any selected document could be printed locally. This capability would truly change the way information is managed. Large centrally maintained databases of documents could be accessed remotely and selectively printed remotely. This would save millions of dollars in document inventory costs.

Specific large visual data bases like the value-line stock charts, encyclopedias, atlases, Military maps, Service Manuals, Time-Life Books etc. could be shipped on CD-ROM's with a viewer. This would allow full publication (text, graphics, images and all) to be viewed and printed across a very large base of machines.

Imagine if the IPS viewer is also equipped with text searching capabilities. In this case the user could find all documents that contain a certain word or phrase, and then view that word or phrase in context within the document.

Entire libraries could be archived in electronic form, and since IPS files are self-contained, would be printable at any location.

One of the central requirements of the Camelot Project is that the IPS file format is device independent. This is essential because it is necessary to be able to print the documents on color or black and white machines — on low or high resolution machines. This requirement is also essential in order to visualize the documents at various magnifications on the screen. For example, it is imperative that the user be able to magnify portions of complex maps, so that subportions of the image are easy to read even on low resolution displays.

To accomplish the above requirement it is necessary that consistent font rendering machinery be available to the viewer. For this reason the viewers will need to contain the full ATM implementations as part of each system.

In considering all the requirements of corporations regarding documents, it is important to structure Camelot components so that they can be sold in ways that are useful to the corporations. Several ideas have come to mind.

Components of Camelot are generally not interesting to single users. The exception to this is in the distribution of large generally useful databases. If someone produced a CD-ROM with “maps of the world” on it, then one can imagine selling a retail package with one viewer and the CD-ROM.

In most other applications, the distribution of information is to many people. In these latter cases a corporation would like a copy of the viewer for every PC. One can imagine viewers integrated into mail systems, or as general stand-alone browsing systems. In any event corporations should be interested in site-licensing arrangements. (more to come)