The importance and futility of device independence in computer graphics

by ANDERS VINBERG
ISSCO (Integrated Software Systems Corporation)
San Diego, California

ABSTRACT

Device independence in computer graphics refers to software that supports all graphics hardware devices. The economic reasons why device independence must be considered mandatory are reviewed in this paper. The ultimate futility of straightforward device independence, because of the widely differing characteristics of different devices, leads to a need for software that adapts intelligently to these characteristics—that has device intelligence. For good results we need not merely technical device intelligence, but also sensitivity to different applications. I coin the phrase layout intelligence for software that thus adapts the graph to the situation and show several examples.
WHAT IS DEVICE INDEPENDENCE?

A device-independent graphics system is one that works with all graphics output devices. Note the stringency here: not “all models of a major vendor,” or “all devices compatible with some popular device,” not “many” or even “most” graphics devices, but all graphics devices.

There are now commercially available graphics systems that offer device independence. It should be borne in mind that software products differ in degrees of support. Some are delivered and guaranteed with device interfaces; others require the user to develop and modify device interfaces. In software, as everywhere else, there is no free lunch; you get what you pay for.

What do we mean when we say that a graphics system “works with” all devices? Realistically, some changes will be needed when a new, previously unheard-of device is brought in. However, we can demand that these changes be kept from the end users. The graphics support staff should be able to modify the system, with minor effort, to support the device. But the instructions from the end users—the graph description, the source code, the English commands, the prompt responses, the touch or voice input—should produce a nearly identical graph on the new device with no change.

WHY IS DEVICE INDEPENDENCE IMPORTANT?

The hardware obsolescence argument

As with the stock market and the weather in England, the only thing that can be said with certainty about graphics hardware technology is that it will change in the future. Scores of hardware vendors bring better mousetraps to the market in a never-ending flow.

It should be remembered that although providing device independence is a technical problem, the motivation for requesting it is an economic one. The economics of the situation indicate that the important effects of a technology change are those experienced by a large number of people.

The effort of the support person is a cost, to be sure, and should be minimized; but it is the effort to adapt by all the end users (hundreds of end users, if graphics is a success) that may become prohibitively expensive and may prevent the organization from taking advantage of the new technology. Thus, by allowing end users to make a large investment in device-specific graph descriptions, an organization may paint itself into a corner and may end by being stuck with obsolete technology for a long time or facing an extremely costly conversion.

This means that when we specify that “the graph description should produce a nearly identical graph with no changes by the user,” the phrase “no changes” is more important than “identical.” If the system modifies the graph slightly, this may be acceptable as long as the meaning is retained. But forcing all users to modify all their old programs or graph descriptions is unacceptable.

This discussion has focused on one important reason for demanding device independence, which we can describe as “hardware obsolescence insurance.” There is another reason, which has nothing to do with the future, but is still economic: previewing.

The previewing argument

Graphics CRTs are used for two kinds of applications: decision support graphics and preview of presentation graphics. Unlike decision support graphics, presentation graphics and report graphics rarely use CRTs as the final output medium; slides, overheads, and above all paper hard copy dominate.

But, because of the low speed and high cost of most hard-copy devices, CRTs are preferred during the design phase, when several graph forms are being tried out to select the most effective.

For this preview work, the most important requirements are that there be absolutely no changes to the graph descriptions and minimal changes to the graph. The preview must be a faithful reproduction of the final result, even if this means that it does not use well the characteristics of the previewing device.

In summary, device independence is of critical importance, because most users will want to use several devices today and all users will want to be able to use new devices tomorrow. Since the software knowledge is now available, and since most graphics software today offers device independence at some level of support, any investment of money, effort, or training in device-dependent software today is indefensible and is surely the most fundamental mistake that can be made when moving into graphics.

WHY IS DEVICE INDEPENDENCE ULTIMATELY FUTILE?

Now for the bad news. Graphics output devices are sufficiently different that a graph that looks good on one device may not look good on another.

Note that we introduced a new concept here: what looks good. Previously we have talked about what can be done—“Can the software system produce the same graph on all devices?” Now we question what should be done.
A basic example is illustrated in Figure 1. Figure 1a shows a very well-designed graph for use as a color slide or a color viewgraph. (Color is represented here as grayscale.) If this chart is copied onto a black-and-white device for reproduction as a report or handout, or if the color viewgraph is simply placed in a monochrome copier, Figure 1b results: a worthless chart, where the two data sets are indistinguishable, but very common. The minimum requirements for a black-and-white paper copy is that shade patterns are used to distinguish the data sets, as in Figure 1c. But the chart, which was designed for projection and hence viewing at a distance, looks amateurish and childlike when copied on an 8½-inch-by-11-inch sheet of paper, viewed at a distance of about 10 inches. Figure 1d shows a better version of the chart—vertical page orientation, different typefaces, smaller annotation, more annotation. It is the same chart, but different. The chart has been tailored for two different applications, embodied by two different output devices.

Note that the term output device may mean more than simply the graph production device. A color viewgraph and a black-and-white report illustration may both be produced on the same device, say one of the many desktop color pen plotters available in the $5,000- to $10,000-range today. The device that differs in this case is not the production device, but the presentation or reproduction device: an overhead projector versus a copying machine.

---

From the collection of the Computer History Museum (www.computerhistory.org)
The differing requirements of color and monochrome are obvious; the differing requirements of hard copy and projection are equally important but less generally recognized. Many of today's output devices have idiosyncrasies that pose other difficulties, some obvious, others deeply technical.

The low-resolution CRT devices so common today require simple, large annotation without frills to be legible. Some are also cell-oriented, able to place text only in some locations on the screen; for these devices, graphics elements must be adjusted to fit the text. Text may be available only in some sizes or orientations. All these need adaptation by the software.

Other differences to be accommodated concern what happens when two graphics items occupy the same location: does one hide the other, do both shine through, do colors mix, and if so how—or do you get a smear?

All this, and much more, means that device independence is not the answer. What is needed is what I call device intelligence.
GRAPHICAL VERSUS TECHNICAL DEVICE INTELLIGENCE

Device intelligence means that the software adapts to the many peculiarities of the output device.

The phrase is occasionally used in a narrow technological sense, referring merely to using the varying capabilities of the device. If the device can draw higher-level constructs, such as a circle, rectangle, dashed line, character, conic segment, or axis, or if it can fill in an area of certain shape, these functions can be offloaded from the host computer, and, more important, from the communications line. This means that if the software knows and adapts to the capabilities of the device, the graph is drawn faster. However, device intelligence in this technical sense does not mean that the graph is good or even meaningful; it just means that the same ugly graph is drawn quicker.

Speed is certainly important, and technical device intelligence is a valuable first step (and one not trivially achieved: the complexities of the software needed to use fully all device functions, and emulate them fully when not present, are significant). But to achieve good results we need graphical device intelligence. This means that the graph layout is adapted to the device characteristics: page orientation, annotation style, amount and size, and data set identification, are all affected. Therefore I will refer to it as layout intelligence.

We remember from the discussion of the many possible uses for the common pen plotter that the choice of layout must not be determined only by the choice of production device. The intended application or intended reproduction device also affect the ideal layout choice. When using a CRT device to design and preview a graph for eventual production and use as
a slide, we certainly want to see the slide layout faithfully rendered, even if the limitations of the CRT device may make some details of the chart ugly or even illegible. Thus, in this situation, "improvements" of the graph to make it fit the CRT would be detrimental. The layout intelligence must be driven both by device choice and explicit specification of intended use, desired use of colors, and other relevant factors.

In an upcoming version of ISSCO's TELL-A-GRAF system, for example, these choices may all be made automatically with minimal specification. Let us look at an example.

We have prepared a format description to allow TELL-A-GRAF to read data from the COBOL files of the accounting programs and to select interesting data from it. We specify what information we want, and how we want to see it:

**DATA FILE IS** "ACCIQ82".
**DATA FORMAT IS** "SALES BY OFFICE".
**GENERATE A DATE AREA CHART.**

If we now specify

**LAYOUT IS SLIDE.**
**DEVICE IS DICOMED MODEL D148C.**

we get the slide shown in Figure 2a.

If we specify

**LAYOUT IS VUGRAPH.**
**DEVICE IS HP MODEL 7221.**

we get the viewgraph shown in Figure 2b.

If we specify

**LAYOUT IS REPORT.**
**DEVICE IS COMP80.**

we get the chart shown in Figure 2c. If we prefer a horizontal ("landscape") page orientation for our report illustration, we can specify

**LAYOUT IS REPORT.**
**DEVICE IS COMP80.**
**ORIENTATION IS LANDSCAPE.**

we get the chart shown in Figure 2d.
we would get the graph illustrated in Figure 2f, since the system knows that the 4025 model has no color.

All the variants shown represent the same data and are basically the same chart. Each variant is optimized for its intended use.

This application intelligence allows even the casual user to get a really good graph by simply making five choices:

1. Which data file to use (“DATA FILE” statement)
2. Which information to retrieve from this data file (“DATA FORMAT” statement)
3. Which chart type to use (“GENERATE” statement)
4. Which layout to use (“LAYOUT” statement)
5. Which device to use (“DEVICE” statement)

FLEXIBILITY RETAINED

An important point must be added: all this automation is achieved without sacrificing flexibility and control over details. For example, assume that the organization has its own graphics standards that state that all area charts must have the following:

- Tick marks pointing inward on the horizontal axis
- Horizontal grid lines, no tick marks on the vertical axis, and a double vertical axis
- The words COMPANY CONFIDENTIAL in the lower left corner

These local standards can be entered into the system once and for all and will then apply to all subsequent area charts generated (unless explicitly overridden, of course). To specify these local standards, one enters:

**GENERATE AN AREA CHART**
**X TICKS REVERSED.**
**Y GRID, NO TICKS, DOUBLE AXIS.**
**COMMENT “COMPANY CONFIDENTIAL”.**
**STORE DEFAULTS.**

If these standards had been stored, the six charts shown would have looked like the ones shown in Figure 3a through 3f. Even so, the individual user can still customize the chart to make a special point, as shown in Figure 4.

Thus, while preserving the basic flexibility of the software system, the layout intelligence gives the casual user the benefit of the accumulated experience of the graphics experts involved in the design of these application-specific layouts. If you are an expert in something else, with no desire to become a graphics expert, this layout intelligence should prove a great boon through making the use of graphics more effective without requiring you to reinvent the wheel of good graphics.