LA-UR

94-3378

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36

TITLE:

AUTHORING TOOL EVALUATION

AUTHOR(S):

Arthur Lee Wilson, CIC-3
Kevin S. Klenk, CIC-3
Aaron C. Coday
J. Patrick McGee, CIC-3
Reid R. Rivenburgh, CIC-3
Diane M. Gonzales
Susan M. Mniszewski, CIC-3

SUBMITTED TO:

External Distribution

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

MASTER

By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive royalty-free license to publish or reproduce the published form of this contribution or to allow others to do so, for U.S. Government purposes.

The Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy.

LOS ALAMOS

Los Alamos National Laboratory
Los Alamos New Mexico 87545

ab
DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Authoring Tool Evaluation

Arthur Lee Wilson Aaron C. Coday Diane M. Gonzales
Kevin S. Klenk J. Patrick McGee Susan M. Mniszewski
Reid R. Rivenburgh

September 15, 1994

Abstract

This paper discusses and evaluates a number of authoring tools currently on the market. The tools evaluated are Visix Galaxy, NeuronData Open Interface Elements, Sybase Gain Momentum, XVT Power++, Aimtech IconAuthor, Liant C++/Views, and Inmark Technology zApp. Also discussed is the LIST project and how this evaluation is being used to fit an authoring tool to the project.

Contents

1	Authoring Tool Information	3
1.1	Introduction	4
1.1.1	What is an Authoring Tool?	4
1.1.2	The LIST Project	4
1.2	Authoring Tool Information	5
1.2.1	Authoring Tool Paradigms	5
1.2.2	The Evaluation System	7
2	Authoring Tool Evaluations	9
2.1	Visix Galaxy	10
2.1.1	Overview	10
2.1.2	Evaluation Criteria	10
2.1.3	References	12
2.2	NeuronData Open Interface Elements	13
2.2.1	Overview	13
2.2.2	Evaluation Criteria	13
2.2.3	References	15
2.3	Sybase Gain Momentum	16
2.3.1	Overview	16
2.3.2	Evaluation Criteria	16
2.3.3	References	18
2.4	XVT Power++	19
2.4.1	Overview	19
2.4.2	Evaluation Criteria	19
2.4.3	References	21
2.5	Aimtech IconAuthor	22
2.5.1	Overview	22
2.5.2	Evaluation Criteria	22
2.5.3	References	23
2.6	Liant C++/Views	24
2.6.1	Overview	24
2.6.2	Evaluation Criteria	24
2.6.3	References	25
2.7	Inmark Technology zApp	26

2.7.1	Overview	26
2.7.2	Evaluation Criteria	26
2.7.3	References	27
3	Conclusions	28
3.1	The Closest Fit	29

Chapter 1

Authoring Tool Information

1.1 Introduction

In this paper, we will discuss and evaluate a number of different authoring tools. We will begin by discussing the LIST project and its needs in an authoring tool. Then an evaluation system will be used to judge each product individually. Finally, a tool will be chosen which best reflects the needs of the LIST project.

We expect this paper to be read beyond the scope of the LIST Project and we hope that others will find this information of use. However, we are evaluating these tools based on very specific criteria that fit our project. The ratings that we give may or may not apply to other projects that would like to use one of these authoring tools. When others are using the information in this paper to help evaluate authoring tools, they should keep this in mind. Or, in other words, your mileage may vary.

1.1.1 What is an Authoring Tool?

For the purpose of our project, an authoring tool is something that helps us develop front ends for applications, cuts development time in doing so, and increases programmer efficiency due to ease of use. A more common definition would be a software package that can create multimedia presentations, training courses, on-screen video productions, demo disks, guided tours, interactive kiosks, simulations, and prototypes through scripting languages and intuitive icon-based construction environments.

Authoring tools provide true interactivity - not just button-pressing
- and powerful control over media elements, including words, images,
animations, sound and video. [1]

1.1.2 The LIST Project

The Los Alamos Information Systems Technologies project (LIST) is a testbed for telecommunity projects and will serve as a national resource for determining what the National Information Infrastructure (NII) will become. The idea is to have inter-networked community systems that will provide informational and transactional functions to end-users. Some of these functions include electronic mail, internet newsgroup access, national and local directories (online yellow pages), online commerce, online medical consulting, digital libraries, and government information access.

Each community will begin by building their own interface, adding the applications they desire, and then allowing their users to interact with and use the system. These applications will come from a pool of applications that the teams at Los Alamos National Laboratory will have programmed for these specific purposes. They will be interchangeable, extensible, and the choice of which applications are right for each community will be up to the community itself.

In order to facilitate making these applications and allowing community developers to create their own interfaces, this paper examines authoring tools that

can be used to create front ends for the applications we will create and that could be used to make community developer interface builders.

In essence, the authoring tool would be used for the following things:

- Creating the base LIST environment user interface, providing access to tools and applications.
- Creating application user interfaces that can operate as part of LIST or standalone.
- Creating an application user interface builder that can assist community developers in assembling user interfaces for domain-specific applications, e.g. information-only, transactional, personal communication, online classroom, digital commerce, and more.

1.2 Authoring Tool Information

1.2.1 Authoring Tool Paradigms

Scripting/Visual Languages vs. Code Libraries

Research into authoring tools showed that there were two different kinds of Graphical User Interface (GUI) builders that we were interested in: code libraries and visual/scripting languages. There are advantages and disadvantages to each, of course.

A code library is a library of subroutines that can be called and used from a language such as C or C++. In this paper, C++ will be used as the example language. With a code library, most programming is done in C++. To utilize the GUI/multimedia functionality from the authoring software, calls are made to the subroutines in the code library.

A visual language and a scripting language are two different ways of representing the same thing, and for this reason, the term 'visual/scripting language' is sometimes used in this paper. A visual language is simply an iconic representation of a scripting language. A scripting language is more ambitious than a code library in that it tries to cover all the possible functions you would use when creating a GUI. It is essentially like learning a new programming language with a lot of very specific functionality.

A code library is generally harder to use than a scripting language because the developer must know C++ already and then learn to use the code library functions. Significant amounts of time can be spent debugging code that has nothing to do with a GUI. With a scripting language, development is generally a less daunting task since there is a more limited amount to learn about and much of the debugging has been taken care of by the scripting language itself.

However, a code library is usually much more flexible and versatile than a scripting language. A program that uses a code library has the advantage of being able to utilize all the functionality of the underlying programming language

in addition to its own. Working with applications that weren't designed with the code library is easy because you can utilize the programming language itself to do it. Data passing is likewise very easy.

A scripting language can only do what it was made to do. In other words, it is limited in its functionality compared to a code library. Working with applications not made with the scripting language can be difficult and data passing is usually non-trivial.

Comparing authoring tools that use these different paradigms can be like comparing apples and oranges. Depending on the type of project you are doing and how ambitious it is, different authoring tools may be better. Smaller, stand-alone projects usually work best with a scripting language authoring tool because it allows faster prototyping and development. However, for projects that want to work with other applications and do a lot of things that no authoring tool can be expected to do, a code library is usually best.

Authoring Tool Approaches

Another thing that authoring tools handle differently is how they interact with the underlying operating system and graphics packages. The goal of all the authoring tools in this paper is platform independence and they approach that in one of two ways - using a "layered" or "emulated" model.

A product that uses a layered model will use native GUI-building toolkits to get the native look-and-feel on a platform. For example, an authoring tool that uses the layered model on a PC would build its interfaces with direct calls to Windows libraries. This is advantageous because the authoring tool does not have to provide as much software and is therefore generally less expensive. A disadvantage of this is that it is generally slower. In addition, it is impossible to get the look-and-feel of another platform besides the one it is being run on. However, it will always have the most current look-and-feel of that platform.

An authoring tool that uses the emulated model will avoid native toolkits and provide all the functionality itself. This is a more complicated way of doing things, but has several advantages. The advantages to this approach are that it doesn't require developers to buy extra software, it can emulate the look-and-feel of other platforms besides the current one, and it is faster because it can eliminate much of the code in the native toolkit that is not relevant to the authoring tool (e.g. in Unix, it would make calls directly to Xt/Xlib, removing the overhead of the motif layer, etc). However, because the vendor has to create a lot more software, the authoring tool is usually more expensive. Also, if the underlying system software changes it's look-and-feel, the authoring tool will have to come out with an update to keep current since it does not make high level calls to the current platform's API.

Which approach is used by the authoring tool chosen for any project should again be based on the needs of the project. Different approaches have different advantages and disadvantages. The layered approach is better for more economical projects that don't care as much about getting the same look-and-feel on different platforms and where speed is not as important. The emulated approach is good for

projects where user customization is desired and speed is important but economy is not.

1.2.2 The Evaluation System

To evaluate the authoring tools in this paper, the following categories of criteria were used. They were picked because they represent what the LIST project needs out of the authoring tools.

- **Multi-Platform Support:** The LIST project prototype software needs to run on several platforms, including Unix (SunOS, Irix, etc.) and IBM PC's (Windows, Windows NT). If the tool supports other platforms, such as the Macintosh, it is considered a bonus.
- **Widgets / Customizeable Widgets:** Any tool chosen has to have a base set of widgets such as lists, scrollable windows, buttons, flashing buttons, etc. The best tool in this category would have the largest set of easy to use widgets. The toolkit must also be able to let the developers make their own widgets and integrate them into the application. An object-oriented approach with inheritance would be best.
- **Interactive Builder:** This is a major requirement since it is one of the main purposes of using an authoring tool. An interactive GUI builder that allows a user to "paint" the interface and then attach functionality saves time and therefore development money. It can also be used for rapid prototyping and experimentation with interfaces.
- **C++/dynamic-link/fork-exec/etc.:** Most authoring tools have very specific functionality and to do anything beyond that, more programming must be done in another language and then utilized and interacted with by the GUI. The LIST project has a lot of functionality having nothing to do with GUI building, and therefore needs a toolset that allows the use of external programs. In addition, data passing back and forth between the applications and the external programs would be good. If this cannot be done through straight C++ function calls, then it should at least be accomplishable through dynamic-link libraries or something similar. In addition, the tool should allow us to coexist with independent software libraries without interaction problems or conflicts.
- **Multimedia:** Since the LIST project will use a lot of multimedia from all sorts of different sources (CD-ROM, laserdisk, etc.), the authoring tool must have the capability of handling a wide array of multimedia devices and techniques. Examples of capabilities desired are the ability to play analog (e.g. NTSC) or digital (e.g. QuickTime) video at a good speed ¹

¹the optimum would be 30+ frames/sec, but we recognize that this is a lofty goal for current technology)

from direct input or some analog or digital storage device (e.g. laserdisk or CD-ROM); synchronizable sound; and the ability to handle a wide array of image, sound, and video formats.

- **Static Graphics:** Any good tool needs to let the users put in their own graphics. Static graphics, in this case, means anything that is being drawn from data. For example, a map drawn from a set of stored data points would be static graphics. There need not be any user-interaction with the graphic once it is drawn (see dynamic graphics, below). In order to do this, the toolset needs to have a good set of drawing primitives and low-level graphics calls.
- **Hypertext:** Hypertext can be an excellent addition to a user-interface. Without it, UI's are limited to buttons, etc. HTML format capability is preferred, but any system that would allow writing a text file with hypertext links in it and dynamically loading and using it would be acceptable. Using the authoring tool to statically create the page and attach functionality to the links would be the worst way. In order to change anything, the "hot spots" must be manually changed with the authoring tool - this severely limits long-run functionality.
- **International Character Sets:** Since this project is meant to be used by people using many different languages, it needs to have the capability to use international character sets. It would be preferable if the tool conformed to the UNICODE standard.
- **Customer Support:** Customer support as defined here has several aspects. Not only does there need to be a good response (qualitatively and time-wise) to a technical question, but the technical representatives need to be accessible (preferably by phone and e-mail). The attitude of the technical representatives needs to also be friendly and helpful. In addition, the documentation must be thorough, well-written, and include many examples of complete example code. An established user-base at Los Alamos National Laboratory certainly would be an added bonus, as well.
- **Animation:** Animation for this paper is defined as the process of moving some UI component(s) along a specified path. Although statically creating animation is acceptable, the best way would be to have animation be dynamically createable (i.e. the path of the object specified at run time). It would also be helpful if an animation creation tool is included with the authoring tool.
- **Dynamic Graphics:** Dynamic Graphics are much like Static graphics (see above), but after they are created, can be manipulated and controlled via user interaction. Examples would include volume rendering and models.

Chapter 2

Authoring Tool Evaluations

2.1 Visix Galaxy

2.1.1 Overview

Galaxy is a C++ library that can be used to make GUI's and multiplatform support modules. It was created as a cross-platform interface and application development tool (and it does run on quite a few platforms). The idea is to make the functionality of your application with a combination of C++ and Galaxy library calls and to make the look-and-feel of your interface with VRE, the Visix Resource Editor. You then combine your C++ code and the resource file generated by VRE to get a working GUI. Galaxy has built-in functionality for custom widgets, static and dynamic graphics, and much more.

Galaxy uses an emulation paradigm to control the GUI look-and-feel. This allows for great flexibility in making interfaces. You can make an interface and then test the look-and-feel for all the target platforms on one computer. (e.g. a DOS machine can have an Motif look-and-feel).

Galaxy is a large, complex tool and has a lot of flexibility in the types of applications that it can create. There is very little that it cannot do, though some things may be very difficult and kludgy. The main problems with it are the lack of multimedia support, the steep learning curve, and licensing. The lack of multimedia support can probably be worked around and the steep learning curve will probably be helped by better documentation in 2.0 plus a new system of classes that Visix is thinking of implementing. Licensing is a problem, though, since Visix ties a license to a single person.

Galaxy is so complex that Visix requires evaluators to go to a week long class before they will give out evaluation copies. Without this class, Visix feels that Galaxy would quickly become overwhelming to the new user. Visix simply wants new customers to give their product a fair chance during evaluation. The learning curve for Galaxy is generally about 30 days with the class to become reasonably capable.

2.1.2 Evaluation Criteria

- **Multi-Platform Support:** This is one of Galaxy's major selling points, for the software does the same thing on any of the supported platforms seamlessly and requires no extra programming. In addition, it can handle doing one look-and-feel across all platforms. The only exception to this is emulation of the Macintosh look-and-feel on other platforms, as Apple has exclusive proprietary rights to it and will not allow it to be used on other platforms.
- **Widgets / Customizeable Widgets:** Galaxy has all the usual components, plus a few that are somewhat unexpected, such as an excellent help system; color, file, and font choosers; a very nice list manager; and more. Galaxy also allows custom widgets by subclassing their regular widgets and overriding and/or adding functionality to them.

- **Interactive Builder:** The Visual Resource Editor makes the programmer's job much easier. It allows a developer to make windows, put buttons, etc. in them, and more all by dragging and dropping the appropriate widgets to the interface window. It also has a large number of programmer conveniences that make it extremely easy and efficient to use. An example is the spring/strut system of the geometry manager that allows flexible placement of widgets, etc.
- **C++/dynamic-link/fork-exec/etc:** Since Galaxy is a C/C++ library, it can do everything that C/C++ can do. The forking, etc. is limited by OS/Platform, and there is nothing that Galaxy (or any other tool, for that matter) can do about that. Galaxy also has a transparent networking library that allows seamless communication between different networking methods.
- **Multimedia:** Galaxy does not support multimedia. Since Galaxy is a C++ library, though, a third party multimedia library could be used (keeping in mind that the multimedia library needs a counterpart library when porting to different platforms).
- **Static Graphics:** Galaxy has a good set of graphics primitives which will work the same on any platform (no modifications necessary). It can also make images from data, since it is done on top of C/C++ code.
- **Hypertext:** Galaxy has no built-in hypertext support. However, a hypertext widget can be created somewhat easily using other Galaxy widgets.
- **International Character Sets:** Not only will Galaxy handle all sorts of international character sets, it automatically looks at the country code set on the machine that the application is running on and sets the character set accordingly. This is all done dynamically and with no work by the programmer. In addition, Galaxy will be fully UNICODE compatible (as of version 2.1).
- **Customer Support:** Galaxy has an excellent support staff. They are generally very knowledgeable and try to help you with problems in a timely manner. Visix has e-mail, fax, and regular phone help support. During this evaluation process, they have been extremely helpful and willing to answer questions. The documentation is the only part of this category that Galaxy falls down on. The documentation, though fairly complete and thorough, is not organized very well and it is difficult to get a good overall picture of how to put various elements of Galaxy together. It does have complete code samples that are fairly relevant in many places, though. Release 2.0 of galaxy is supposed to have MUCH better documentation according to Visix.
- **Animation:** Animation in Galaxy is done with graphics primitives and timer tools. This works reasonably well, though it is a bit tedious to program. However, because Galaxy is such a flexible and powerful tool, it would

be possible to create our own animation-editor tool that could be used to make dynamic animation.

- **Dynamic Graphics:** One of the main advantages of being a C/C++ library is that complicated applications using dynamic graphics can be created. One application that has been made uses real-time data to make a pictographic representation which can then have operations done to it by the user (such as zooming, rotating, filtering, etc).
- **Cost:** For the power of Galaxy, the cost of approx. \$10,000 to \$12,000 per license is not unreasonable, but it is on the high end of the spectrum. Galaxy also has a personal license scheme that ties a copy of Galaxy to a single developer. The license can be transferred to another developer after purchase as many times as necessary.

2.1.3 References

- 800-832-8668
703-758-0233 (fax)
703-758-8230 (regular)
galaxy@visix.com

11440 Commerce Park Drive
Reston, VA 22091
- contacts:
Lee Tanner/Sales
Don Warner/West Coast Federal Sales
Larry Spector/Engineering
- [2]
- [3]

2.2 NeuronData Open Interface Elements

2.2.1 Overview

Neuron Data provides a multi-layered product for authoring. Their product is divided into three versions. The basic version is Open Interface and includes just the GUI design and application generation. All of the features listed below are part of the Open Interface. The next version is C/S Elements. This version provides client/server hooks into databases as well as all of Open Interface. Their coverage of databases is extensive. Finally the last version is Smart Elements. This version includes all of the previous two versions plus the ability to create/use rules and objects. Neuron Data's history began with work in expert systems thus explaining the inclusion of this technology in Smart Elements. This paper will concentrate on Open Interface.

To use Neuron Data's OI, first use the interactive builder to create your interface. Attach script code to any widgets and/or objects (most of which are capable of utilizing it with some exceptions). OI then generates C code to match your GUI and a resource file specific to the platform. The GUI widgets and their supporting script code is stored in the resource file while the C code file generated loads all the widgets and does the grunt work of interfacing your C code and your resource file.

Resource files can be changed and the builder will "regenerate" the C code so that you don't lose any other code you've put in the old code skeleton. Also, multiple resource files can be loaded in one program.

Open Interface does have a few shortcomings, however. Although the Neuron Data products do work with C++ compilers, they accomplish this by the use of C wrappers in included header files. In addition, the GUI builder only generates C code. This will change with future versions, however, as Neuron Data is aware of the shortcoming. They expect that OI will be able to generate C++ code by early 1995.

2.2.2 Evaluation Criteria

- **Multi-Platform Support:** Neuron Data currently supports over 40 platforms. Moving from one platform to another involves simply moving source files and recompiling. It also has the look-and-feel feature across almost all platforms (the exception being, of course, the Macintosh).
- **Widgets / Customizeable Widgets:** Open Interface has all the appropriate widgets. It even provides several power widgets, such as a business chart widget and hypertext widget. OI also allows customization of widgets through subclassing or creation of custom widgets from the group up since widgets are just C++ classes/C structures.
- **Interactive Builder:** The visual editor is very user-friendly and allows for quick construction of applications. The builder provides several features

that make it stand out including a geometry manager, widget drag-and-drop placement, and a test mode that allows easy testing of the application look-and-feel.

- **C++/dynamic-link/fork-exec/etc:** Open interface is a C/C++ library so it can do all that C/C++ can do. Since OI can only generate C code - not C++ - developers may have some problems interfacing generated modules with C++ code.
- **Multimedia:** Does not support multimedia capabilities at all except to provide QuickTime support for the Mac.
- **Static Graphics:** Open Interface has a set of graphics primitives that allow graphics to be dealt with transparently across all platforms.
- **Hypertext:** Open Interface has a hypertext widget that handles dynamic page setup, but only works with Windows Help Format hypertext. Neuron Data also has a visual hypertext editor in alpha version currently - a definite plus. However it is important to note that it only directly supports hypertext as opposed to hypermedia.
- **International Character Sets:** Open Interface doesn't explicitly support UNICODE, however it does deal with international character sets in a decent manner. It is able to support large numbers of languages by placing the strings in the resource files. Since Open Interface is capable of loading resource files dynamically this also allows for dynamic changing of languages. Neuron Data has said that they hope to have their UNICODE version out by October or November of 1994.
- **Customer Support:** Neuron Data's customer support appears to be extremely good. They provide hotline service as well as a bulletin board. Their documentation is well-written and contains numerous code examples. They even provide onsite training if desired. They have been extremely helpful during this evaluation as well, even going so far as to set up a conference call with one of their engineers to answer our technical questions.
- **Animation:** Open Interface does not provide an Animation tool, but like Galaxy and XVT, it can be accomplished through graphics primitives and timers.
- **Dynamic Graphics:** Open Interface can do dynamic graphics because of its C/C++ library.
- **Cost:**
Cost can be a disadvantage depending on the version chosen. Open Interface is \$5,000 for the Mac or PC platform and \$10,000 for the Unix platform. C/S Elements is \$7,000 for either Mac or PC, and \$14,000 for Unix. Smart Elements is \$10,000 for Mac or PC, and \$20,000 for Unix.

2.2.3 References

- 800-876-4900
415-321-3728 (fax)
214-789-5340 (Dallas)

156 University Avenue
Palo Alto, California 94301
- contacts:
Frank Peel/Sales (frankpeel@neurondata.com)
Michael Harper/Sales
- [4]

2.3 Sybase Gain Momentum

2.3.1 Overview

Gain Momentum is an authoring tool designed for standalone applications or ones working with databases. The tool works by designing the interface with an interface builder and then adding functionality to the widgets with the built-in GEL scripting language.

The GUI builder is fully functional, though it could use a slightly better interface. It uses a canvas and icon-placement of widgets. Since GM uses a scripting language, these widgets are fully functional immediately and there is no compiling to slow things down. The main problems with the GUI builder interface is the lack of a widget "palette" (it uses pull-down menus only) and the frustrating debugging process (scripts attached to objects can easily get "lost" in large applications).

Gain Momentum does have a few major flaws however. One of these flaws is the lack of an object-oriented paradigm. The scripting language does not support any data types more complicated than basic arrays, making usage of CORBA objects very difficult (the object structures have to be "flattened"). This also means that programmers will have to break their own learning of higher-level programming practices in order to use this product effectively.

The other major flaw is the inability to work well with outside C or C++ code. While it can do so and even pass data back and forth dynamically, it seems difficult and the shared data is limited in size. GM uses dynamic linking to pass data back and forth between GM and the other program, but this is limited to 1000 bytes (for now - this could be fixed soon).

Overall, Gain Momentum is an excellent tool and has some features that really make it stand out. It falls down, however, in a couple of major areas, as noted above, which will probably make it less than desirable until fixed. If Sybase takes this tool seriously and DOES fix the problems, though, this tool could come out way ahead of the pack.

2.3.2 Evaluation Criteria

- **Multi-Platform Support:** Gain works well with Unix and PC (Windows NT only) platforms, but has problems with full functionality on Macintoshes (at this point, Gain Momentum will only run on a PowerMac).
- **Widgets / Customizeable Widgets:** The interface builder has a list of standard widgets which can be used to make front-end GUI's. The list doesn't include complex widgets like file selectors, but is complete enough that most applications can be built using them. Custom widgets can be created using inheritance from widgets above the new one in the container class hierarchy.

- **Interactive Builder:** Objects can be interactively placed on the screen and groups of widgets aligned easily enough, but there doesn't seem to be extensive support for things like how to handle window resizes. Other GUI builders such as Galaxy and XDesigner (a Motif user-interface builder) appear to do a better job of this.
- **C++/dynamic-link/fork-exec/etc:** External C and C++ code can be linked in and called from within a GEL script. There are some definite problems with data passing and data structures, though. For example, a maximum of 1000 bytes can be passed between the GM application and an external routine. Also, the most complicated data structure that Gain Momentum has is arrays and so external data structures must be "flattened" to be compatible with the GM application.
- **Multimedia:** Gain Momentum can handle video, sync'd sound etc. in a very efficient manner (by a timeline, similar to Macromind Director). It can support video input from analog (NTSC) and digital devices. It has no built-in device handlers for laserdiscs, etc. though, and so C/C++ code would have to be written to handle these.
- **Static Graphics:** Gain provides interactive utilities and script commands for dealing with bitmaps. It also has fairly extensive utilities and script commands for creating and manipulating low-level graphic objects, such as lines, arcs, and rectangles. Images can be imported in several formats, such as TIFF and XBM. Including complex pictures would probably best be done by creating them with a paint program and then importing the images into Gain.
- **Hypertext:** This is accomplished in Gain/Momentum by the use of "hot spots." It is certainly not the best way to define a hyperlink in text, but it is workable. Pages must be statically created with these "hot spots" and cannot be dynamically changed.
- **International Character Sets:** There doesn't appear to be any direct support for different languages. Supplying text for text objects with something like an X resource file, so that displayed text can quickly and easily be changed, doesn't appear to be an option, either. The only option appears to be changing the fonts manually.
- **Customer Support:** The on-line documentation is excellent (when it doesn't crash the program; that happened several times during this evaluation). There are on-line tutorials, user guides, and reference manuals. Each script command has a help page that includes an example of its use. Their phone and email support is not very good, however, and they have consistently ignored my phone calls without calling me back.
- **Animation:** Any object can be animated along a user-described path.

- **Dynamic Graphics:** There doesn't seem to be direct support for this kind of thing, though it can be done with external code. A demo package already exists that displays 3D wire-frame images in a window and allows the user to rotate them with the mouse. GM does at least have very good drag & drop support for icons, etc.

- **Cost:**

The cost of this package is a pretty flat fee of approximately \$10,000. This includes the base package plus one of the option packages listed below. If more option packages are desired, then the additional price listed must be paid. Sybase does give volume discounts for buying many copies of Gain Momentum.

- Sybase SQL Option \$2,500
- Oracle SQL Option \$2,500
- Gain Exposure Option \$2,500
- CAI Option \$2,500
- Document Conversion Option \$2,500

Unfortunately, GM has some hefty run-time fees running from \$100 (for 10,000+ users) to \$795 (for 1 user, "advanced" runtime license). Technical Support is not cheap either, at about \$7,000 per year per copy.

The good news is that their licensing options are VERY flexible. GM is run off a central server and users can access a number of copies at once equal to the number of licenses that have been bought. Who the user is or what machine they're on is irrelevant.

2.3.3 References

- 1-800-879-2273
- contacts:
Deborah Christoffers/Sales (415-813-8341)

2.4 XVT Power++

2.4.1 Overview

XVT Power++ is a C++ authoring tool library that can be used to create applications on many different platforms. XVT has been around for quite a while, and although it doesn't directly have a lot of the support we need, it does have many independent vendors who have made add-on libraries and tools for more functionality. For example, Power++ does not come with network or file abstractions, but independent vendors have created packages that do. As a matter of fact, this is a deliberate philosophy, for they have tried to unbundle everything that is not essential to the package so that they can lower the cost and let the developers buy only the extra tools that they need from secondary vendors. This can be a good and bad thing, for if you want a LOT of extra functionality, you can find yourself paying a lot more than you would by just buying a more expensive package like Galaxy or Neuron Data.

Power++ uses a layered approach for GUI creation. Users can be guaranteed the same look-and-feel as they are used to on each platform (and XVT supports a lot of platforms). Because Power++ is C++ code, it runs fairly quickly. Also, Rogue Wave components (the pre-ANSI "standard" for C++ classes) are integrated into Power++.

The way that Power++ works is that you can create layouts with a GUI creation tool (a simple one is available now, but their next version, Viper, is extremely powerful. It should be out in November of '94 or so). This will then generate a C++ code shell and a resource file to load the widgets from. This is very similar to Neuron Data's approach except that Power++ does not have a scripting language and is built from the ground up in C++. Also similar to Neuron Data, the code generated can be "regenerated."

XVT's main goal is to increase developer productivity, and they seem to succeed at this fairly well. Their next goal is to provide graphical tools to use this increased productivity even better.

The main drawbacks of Power++ are the lack of multimedia support and the somewhat limited widget functionality.

2.4.2 Evaluation Criteria

- **Multi-Platform Support:** XVT works on most Unix, PC (Windows, OS/2), and Mac platforms. To switch between them is "simply" a recompile. The recompiling may have problems at times due to differing setups and resource locations. Some tweaking may be involved, but overall, it should be pretty easy.
- **Widgets / Customizeable Widgets:** The interface builder (described below) does have the standard widget components. However, in order to do better, the tool would have to have more higher level widgets, which the basic package does not seem to have. Power++ can do widget customization

very easily, simply by subclassing, and can use third party widget libraries easily as well.

- **Interactive Builder:** The current GUI builder for C++ is primitive, but workable. It allows the developer to graphically “paint” the interface and then it generates the code shell and resources for it. Independent comments [5] state that the builder actually puts out buggy code, doesn’t give you many options, and is not very intuitive. The new version, Viper, provides much greater functionality such as an Object hierarchy browser, a GUI layout tool, and a graphical class attribute editor. Viper also has a very nice interface.
- **C++/dynamic-link/fork-exec/etc:** XVT itself does not have inherent extra capabilities for dynamic linking, etc. Since it is a C++ library, however, all those capabilities are covered by the C++ language itself and should therefore not be a problem.
- **Multimedia:** XVT does not directly support multimedia devices. Like other libraries of this nature, third party products would have to be used.
- **Static Graphics:** XVT does have a set of graphics primitives that allow dynamic drawing of images from data. In addition to these primitives, there are several vendors that provide more advanced graphics library packages for XVT.
- **Hypertext:** The new version of Power++ does support hypertext in the Windows help format.
- **International Character Sets:** Power++ has support for all 8-bit European languages. It does not currently have UNICODE support, but it is planned within the next year.
- **Customer Support:** Their documentation is “award-winning” and they have plenty of example code. They also have knowledgeable customer support for each of the platforms that they support. In making this evaluation, they have been very good at answering my questions and when they couldn’t do so right away, they did their research and called back within a day or so. In addition, XVT offers C++ classes about once a month for Power++ developers. Source code for all their classes (except Rogue Wave) is provided.
- **Animation:** XVT does not have any animation support beyond its graphics primitives and the fact that it is embedded in C++.
- **Dynamic Graphics:** These graphics primitives could be used to make dynamic images because Power++ is embedded in C++, which can handle user input and graphics manipulation. There is at least one vendor that provides good functionality in this area. Power++ also has drag & drop support.

- **Cost:**

- \$1,700 PC, \$6,800 Unix
- Site license available (normally done per user, but they're willing to work with us)
- no run-time fees, etc.
- This cost is for the basic Power++ package. While it seems much lower than many of the other tools look at in this report, it does not take into account the independent vendor packages that may have to be bought to get all required functionality. After all that, it could very well end up comparable in price to other more expensive packages in this report.

2.4.3 References

- 800-678-7988
303-443-4223
303-443-0969 (fax)

info@xvt.com

4900 Pearl East Cir.
Box 18750
Boulder, CO 80308

- contacts:
Robert Tedesco/Sales (303-545-3244, 303-447-2109 (fax)), robert@xvt.com
Dave Locke/Product Mgr.

- [6]

2.5 Aimtech IconAuthor

2.5.1 Overview

IconAuthor is an authoring tool used to create interactive multimedia applications that combine text, graphics, animation, full-motion video and audio. This product concentrates on making standalone multimedia presentations and training programs, but at the expense of interoperability with languages such as C++.

IconAuthor provides the user with an icon-based programming language. This language is object-oriented and each "command" has its own set of functions, properties, etc. which can be utilized. This command language can be easily integrated with the output of other (included) editors such as a graphics editor, a user-interface editor, and an animation editor. The output of these editors is treated as an object that can be manipulated by functions within the IconAuthor Programming language. In addition, third party image, audio, video, etc. files can be imported and used (the supported types include a fairly broad range). Also, other programs can be run from the IconAuthor language so that you are not tied to just what IconAuthor can do. Finally, IconAuthor has its own database functionality (which I haven't really tested or looked into).

2.5.2 Evaluation Criteria

- **Multi-Platform Support:** IconAuthor is made to support the Unix and Windows platforms. A Macintosh playback program is available but its capabilities are limited. Also, a major limitation on the PC side is that it will only run in Windows, which can be some major overhead.
- **Widgets / Customizeable Widgets:** Iconauthor has most of the usual widgets, etc. Widgets are treated as objects with distinct properties and (global) variables can be used to store data and exchange it between objects. Customizeable widgets consist of building a widget structure and then collapsing it into one icon to use later (essentially making it a subroutine).
- **Interactive Builder:** The SmartObject Editor is very handy for making interfaces. Used in conjunction with the flow control of IconAuthor, it can also be very powerful.
- **C++/dynamic-link/fork-exec/etc:** In order to run anything not part of the IconAuthor domain, you must run a separate pre-compiled executable.
- **Multimedia:** This is one of IconAuthor's main selling points. It has the ability to interface with almost any multimedia tool or player around. It does this seamlessly and with very little effort to the programmer. It also comes with a whole slew of drivers for various multimedia products.
- **Static Graphics:** IconAuthor includes a set of drawing primitives that allow users to create images dynamically.

- **Hypertext:** IconAuthor Support Hypertext links and can read the links from a windows-help format text file.
- **International Character Sets:** Able to handle Spanish, at least.
- **Customer Support:** The Salesman stated that their customer support was rated "best of the industry." Documentation is pretty good, although it could use better organizing. There are very few, if any, code samples, due to the nature of the visual scripting language.
- **Animation:** IconAuthor includes an animation editor that allows you to move pre-created images around on the screen. It is easy to program, and is reasonably sophisticated in what it can do.
- **Dynamic Graphics:** Once images are created (using the drawing primitives), simple things can be done to them through user interactivity. This would be difficult to program, however, and would only be able to do simple things.
- **Cost:**
 - approximately \$5,000 windows, \$10,000 Unix.
 - Run time fees are based upon who they sell the product to. There are no run-time fees for government. For commercial developers, run-time fees depend on their particular licensing agreement and are usually done for developers that want to put out commercial products using IconAuthor.
 - Site licenses available

2.5.3 References

- 1-800-289-2884
- Mark Robertson/Sales (303-671-4444)
- [7]

2.6 Liant C++/Views

2.6.1 Overview

This is a C++ library based on the Smalltalk model (single inheritance, all classes derived from one superclass). It includes a class browser and a WYSIWYG GUI builder.

Like most other C++ libraries looked at in this report, C++/Views uses a GUI builder to make the interface (it has geometry management, a palette of objects to place, etc. All the common amenities). It then creates resource files with the objects in them and a skeleton on C++ source code that uses the objects in the resource files. This is pretty standard fare, but the price is much lower most other tools in this report.

One very nice feature of their code generation is being able to make the methods for objects in the GUI builder, attach it to the object, and then generate the code with the method already integrated. This is the only tool I've seen that does this with true C++ methods in the GUI builder.

C++/Views is mainly for cross-platform business applications. It is really designed to build things conforming to the look-and-feel of the various supported platforms (dialog boxes, scroll bars, etc.). The product is also very "Microsoft Windows-centered," having many functions that are only supported in the Windows environment. The company seems to concentrate on this, to the detriment of functionality on other platforms.

2.6.2 Evaluation Criteria

- **Multi-Platform Support:** C++/Views supports Microsoft Windows, DOS, OS/2 Presentation Manager, OSF/Motif, and Macintosh. Code is recompiled across platforms.
- **Widgets / Customizeable Widgets:** C++ Classes. Visual environment to create customized widgets, etc. System is not designed with multiple inheritance in mind. Also, customized widgets cannot be added to the GUI builder widget palette.
- **Interactive Builder:** The GUI builder, as noted above, has all the normal GUI builder amenities, including geometry management, widget palettes, etc. However, it also has some very nice features, such as being able to attach straight C++ code to objects right in the GUI builder, and a class browser.
- **C++ / Dynamic Link / fork-exec / etc.:** Since C++/Views is basically a C++ wrapper library around native GUI toolkits, you have access to everything you need. It has no network or file abstraction support.
- **Multimedia:** The only multimedia support this package has is the ability to load images (.bmp format only) as resources.

- **Static Graphics:** The system has a good variety of graphics primitives, including some non- standard ones like pie charts and tables.
- **Hypertext:** C++/Views will not handle hypertext as is (there is no widget included). If hypertext is desired, the widget must be created separately.
- **International Character Sets:** Supports double-byte (Asian) characters, but only in the MS Windows environment.
- **Customer Support:** Liant does not seem to have very good customer support. They were not very responsive in returning phone calls. The technical support representative did appear to know the material he was speaking about, however, and was able to answer most questions that were asked.
- **Animation:** This can be done using graphics primitives and C++ timers, but there is no support built in.
- **Dynamic Graphics:** User interaction is not supported, not even drag & drop.
- **Cost:**
 - Unix software: \$1999 (support = \$1000/yr including upgrades)
 - Windows software: \$759 (support = \$500/yr including upgrades)
 - licensing is non-machine specific and based on maximum number of users at once
 - onsite training classes available

2.6.3 References

- (800) 237-1873, ext. 247
(508) 872-8700, ext. 247
(508) 820-0035 FAX

LPI Division
Object Technology Division
Liant Software Corporation
959 Concord Street
Framingham, MA 01701-4613

- Sabina Chadha/Account Manager (sabina@lpi.liant.com)
Monzurul Eshan/Tech Support

2.7 Inmark Technology zApp

2.7.1 Overview

zApp is a C++ class library that encapsulates all the functionality necessary to write a GUI application. zApp is a collection of multiple class hierarchies which allows you to add new classes to the hierarchy. It provides object persistence so that objects can be saved and retrieved from disk.

Originally developed for the financial community, zApp has become a high-level application framework for developers who wish to create Windows programs and then port them to Windows NT, OS-2 and Unix X/Motif platforms. Because of its roots, it lends itself well to business application programming.

Inmark's zApp offers a well designed set of C++ classes for a very low cost. Many users give it high marks on how simple and intuitive it is for developing GUIs in an object oriented framework. The major complaint is that zApp is MS-Windows oriented and many of their classes do not translate well to other platforms.

zApp, like Liant, appeared to be another "MS Windows-centered" company, only more so. The GUI builder itself only works on Windows.

2.7.2 Evaluation Criteria

- **Multi-Platform Support:** zApp supports Windows, Windows NT, and the Unix X/Motif platforms. You can take a zApp program and move it from one platform to another by recompiling and linking the proper library, although users complain that many of zApp's classes do not translate well to other platforms. A Macintosh library is due 4th quarter 94.
- **Widgets / Customizeable Widgets:** There are a large number of classes provided for GUI development: windows, scroll bars, tool bars, check boxes etc.; graphics support, dialogs, forms, clipboard support, fonts, bitmaps, size and positioning services, memory management, data collection and print services. In addition, you can customize widgets by subclassing.
- **Interactive Builder:** zApp Factory for Windows is a visual application designer/code generator, and zApp Interface Pack is a collection high-level controls (widgets) for the zApp environment. The visual interface builder is currently only available on Windows. An interactive builder for Unix is due the first quarter of '95.
- **C++/dynamic-link/fork-exec/etc:** zApp is a C++ library, it can do everything that C++ can do.
- **Multimedia:** zApp does not support multimedia. Since zApp is a C++ library, though, a third party multimedia library could be used (keeping in mind that the multimedia library needs a counterpart library when porting to different platforms).

- **Static Graphics:** zApp has graphic primitives and they are well suited to creating business graphics such as 3D charts with dynamic drawing of graphs from data. The appropriate platform-specific graphics libraries handle the graphic primitives.
- **Hypertext:** Not supported. Inmark is currently looking into it, but there are no projection dates.
- **International Character Sets:** Not supported.
- **Customer Support:** zApp comes with on-line and printed form documentation that includes a Programmers Guide and a Programmer's Reference. PC Techniques calls it "thorough and easy-to-follow." Inmark offers free technical support through voice, fax, BBS, CompuServe and on the Internet. zApp comes with full source code. Training for developers is offered every other month in Santa Clara CA. (\$1495 3 day course).
- **Animation:** Again, graphics primitives and timers would have to be used.
- **Dynamic Graphics:** Any support for user interaction would have to be handled by the developer, as zApp does not.
- **Cost:**
 - zApp Developer's Suite for Windows (zApp for Windows, zApp Interface Pack for Windows, zApp Factory for Windows) \$695
 - zApp for Win 32 (Windows NT) \$295
 - zApp for Unix X/Motif (Sun SunOS 4.1x, Solaris 1.x) \$1,495
 - (SGI) \$1,495
 - (HP HPUX 9.x) \$1,495
 - zApp requires no runtime royalties.

2.7.3 References

- 415-691-9000 (regular)
415-691-9099 (fax)
- support@inmark.com
- 2065 Landings Drive
Mountain View, CA 94043
- contact: John Hudson / Sales

Chapter 3

Conclusions

3.1 The Closest Fit

After evaluating these products, the tough decision came in which one fit the LIST project the best. This product turned out to be Galaxy for several reasons, which will be discussed later. First, this paper shall examine why the other products do not fit the LIST project as well.

Neuron Data Open Interface Elements has excellent functionality and an easy-to-use GUI builder. In addition, it seems to be a fairly complete development environment. The main problem with it is the fact that it was not built from the ground up in C++. It uses wrappers to make C++ convert to C, and this can cause strange problems in development sometimes.

Sybase Gain Momentum is also an excellent tool, but was not really designed for what the LIST project wants to do. Gain Momentum is a fairly complete development environment and has excellent multimedia capabilities, but since it is a scripting language, it has problems interfacing with outside programs.

XVT Power++ will have an excellent GUI builder (Viper) and is a complete development environment, however, it has limited functionality in comparison with the other tools we looked at. This can of course be corrected by third party software, but would end up being more expensive for this project in the long run.

Aimtech's IconAuthor is a wonderful tool for doing presentations and computer based training and has excellent multimedia capabilities as well, but is lacking when it comes to trying to hook other applications (such as independently written C++ programs) into it.

Liant C++/Views and InMark Technology zApp both had the same problems for this project: too "MS Windows-centered." While they both had good functionality, their lack of support for other platforms ruled them out.

The LIST project also looked at a couple more products during this evaluation, Kaleida Labs ScriptX and Blythe Software Omnis 7.0. These were eliminated from consideration early on, however, for the reasons below. The LIST project does intend to keep watch on these products and perhaps do another evaluation when they are mature.

ScriptX is a scripting language, somewhat like Postscript, that would allow users to utilize multimedia files and devices on any platform supported. It would come with a large library of predefined objects, multimedia capability, an "authoring player," a graphics library, and the ability to create custom components. ScriptX has been plagued by development problems, however, and does not expect a Unix version any time soon.

Blythe Software Omnis 7.0 is another authoring tool that has good functionality and even some multimedia capability, but does not support the Unix Platform (only Mac and PC). This is expected to change by sometime in early 1995, when they come out with a version for SunOS.

The reasons that Galaxy did fit the LIST project best were that it was a complete development environment, had extremely good functionality, a well-thought out and powerful architecture, it was written from the ground up in C++, had a good GUI builder, had good customer support and a knowledgeable technical staff, was a C++ library for easy integration with outside applications,

and supported all platforms necessary for the LIST project.

Bibliography

- [1] Phillip Robinson. Authoring software. *New Media*, (11), 1993.
- [2] Inc. Visix Software. Galaxy technical description. 11440 Commerce Park Dr., Reston, VA 22091, 1993.
- [3] Stephen K. Kavanagh. Galaxy: Cross-platform development star. *Open Systems Today*, pages 50,56,58, July 1994.
- [4] Inc. Neuron Data. Neuron data elements. White Paper.
- [5] Wade Guthrie. Platform independent graphical user interface kits faq. FAQ available at <ftp://rtfm.mit.edu/pub/usenet/comp.windows.misc>.
- [6] Catherine Connor Michael Sher, Dave Locke. Xvt technical overview for c++. Technical Report by XVT.
- [7] Luisa Simone. Iconauthor for windows. *PC Magazine*, pages 225-231, February 1994.