Progress? What Progress?

Alan Bridges

This paper briefly reviews some of the history of computer graphics standardisation and then presents two specific case studies: one comparing HTML with SGML and Troff and the other comparing VRML with the Tektronix® Interactive Graphics Language implementation of the ACM Core Standard. In each case, it will be shown how the essential intellectual work carried out twenty years ago still lies at the foundations of the newer applications.

Keywords*: SGML, HTML, VRML.*

A Brief History of Graphics Standardisation

The standardisation of computer graphics has a history that goes back to the late 1960's. With the growth of interactive computer graphics following the introduction of the first Tektronix storage-tube displays, the portability of computer graphics programs became a real issue. A number of packages, including Tektronix' own Plot-10 (Plot-10, 1971) and Cambridge University's GINO-F (CAD Centre, 1975), acquired the status of "de facto" standards.

The formal development of standards began in the 1970's with:

- the establishment (in 1972) of the Graphics Standards Planning Committee (GSPC) by ACM SIGGRAPH (GSPC,1977), and
- the Seillac I workshop (held in May 1976) called by Richard Guedj under the auspices of IFIP WG 5.2 (Guedj, 1976)
- a meeting of international experts, following the proposal that GINO-F should be adopted as an international standard, deciding that no existing computer graphics system could he considered suitable for a standard

Typical configurations of the "best practice" systems at that time consisted of a device independent "front end" driving one or more device specific 'back ends" variously called "code generator" or "device driver" (see Figure 1). The front end was driven by the applications program via a set of (normally Fortran) subroutine calls, whilst the back end would typically be a single entry point routine, which for portability, would supply any of the features not directly supported on a device, within the limits of portability supported by the system. The back end would typically have portions of code specific to the requirements of interfacing to a particular operating system and would thus be both device and machine dependent.

This overall architecture was recognized as giving rise to two principal contenders for standardization. The first is the application/graphics system front end interface and the second the Device Independent/ Device Dependent interface between the front and back ends. The decision taken at Seillac was that the first step should be to standardise a kernel graphics system on top of which applications-orientated graphics packages would sit. It was generally agreed that any standard would specify a set of virtual input and output facilities, which would be realized in terms of the functions of particular graphics devices. These

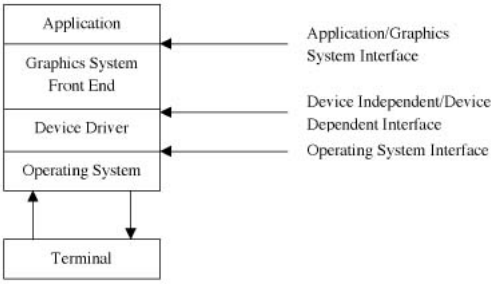

Figure 1 (left). Structure of a device independent graphics system

basic principles underlie much of todays computer applications, in particular, the structure of HTML and VRML.

Computer Graphics Standardisation

The most important work in these early standards was in the field of portability. In computer graphics the following types of portability may be defined:

- Applications Portability - the ability to move a suite of application programs from one system configuration to another with minimum (preferably no) work in adapting the code, whilst keeping as much of the "look and feel" of the application as possible.
- Graphics Package Portability - the ability to use the same range of graphics facilities in a variety of situations. This is typically a combination of the host machine independence and device independence listed below, but also needs to reflect how thorough the implementation of the system is in simulating any features which the underlying hardware and devices do not supply directly.
- Host Machine Independence. A graphics package, which is host independent, is able to run successfully on many underlying computer systems, from a variety of suppliers. In practice, this may well mean that a supplier has implemented the system on a variety of hardware rather than supplying a system, which can genuinely be moved between machines.
- Device Independence. The ability to use the same code to control a wide range of graphics devices implies a careful structuring of the graphics package implementation to isolate the parts of the code which are specific to a particular display or input device in a small proportion of the overall system. Substitution of this small portion of code then allows the addition of a different device to the repertoire of devices supported by a graphics package.
- Programming Language Independence implies that the same facilities should be available from all common programming languages.

The World Wide Web and HTML

Whilst undertaking consultancy for CERN in 1980, Tim Berners-Lee, wrote a notebook program "Enquire-Within-Upon-Everything". Back at CERN in 1989, he produced two internal papers, based on his earlier work, "Information Management: A Proposal" and "HyperText and CERN". The proposals were approved and in 1990, Berners-Lee began work on a hypertext browser, which he called "WorldWideWeb". This ongoing work was publicised on the net at alt.hypertext, comp.sys.next, comp.text.sgml, and comp.mail.multi-media. Marc Andreessen at NCSA released the first alpha version of Mosaic in February 1993 and, on April 30, 1993 CERN's directors declared that WWW technology would be made freely available with no royalty payable to CERN. In March 1994 Andreessen left NCSA to form "Mosaic Communications Corp" (later Netscape). The rest of the history is relatively well known.

Given this timetable, Berners-Lee did not start from scratch. His initial specifications of URLs, HTTP and HTML were based on existing technology, although they have been refined as the Web technology spread. HTML is a form of SGML. The

Standard Generalised Mark-up Language (SGML) is an international standard for the definition of device-independent, system-independent methods of representing texts in electronic form.

Historically, the word markup has been used to describe annotation or other marks within a text intended to instruct a compositor or typist how a particular passage should be printed or laid out. Examples include wavy underlining to indicate boldface, special symbols for passages to be omitted or printed in a particular font and so forth. As the formatting and printing of texts was automated, the term was extended to cover all sorts of special markup codes inserted into electronic texts to govern formatting, printing, or other processing. A markup language specifies what markup is allowed, what markup is required and how markup is to be distinguished from text.

There are three characteristics of SGML which distinguish it from other markup languages: its emphasis on descriptive rather than procedural markup; its document type concept; and its independence of any one system for representing the script in which a text is written

A descriptive markup system uses markup codes which simply provide names to categorize parts of a document. Markup codes such as **<**para**>** simply identify a portion of a document and assert of it that "the following item is a paragraph". By contrast, a procedural markup system defines what processing is to be carried out at particular points in a document e.g. "call procedure PARA with parameters 1, b and x here". In SGML, the instructions needed to process a document for some particular purpose (for example, to format it) are sharply distinguished from the descriptive markup which occurs within the document. Usually, they are collected outside the document in separate procedures or programs. With descriptive instead of procedural markup the same document can readily be processed by many different pieces of software, each of which can apply different processing instructions to those parts of it which are considered relevant.

SGML introduces the notion of a document type, and hence a document type definition (DTD). Documents are regarded as having types, just as other objects processed by computers do. The type of a document is formally defined by its constituent parts and their structure. The definition of a report, for example, might be that it consisted of a title and possibly an author, followed by an abstract and a sequence of one or more paragraphs. Anything lacking a title, according to this formal definition, would not formally be a report, and neither would a sequence of paragraphs followed by an abstract, whatever other report-like characteristics these might have for the human reader. If documents are of known types, a special purpose program (called a parser) can be used to process a document claiming to be of a particular type and check that all the elements required for that document type are indeed present and correctly ordered. More significantly, different documents of the same type can be processed in a uniform way. Programs can be written which take advantage of the knowledge encapsulated in the document structure information, and which can thus behave in a more intelligent fashion.

A basic design goal of SGML was to ensure that documents encoded according to its provisions should be transportable from one hardware and software environment to another without loss of information. The two features discussed so far both address this requirement at an abstract level; the third feature addresses it at the level of the strings of bytes (characters) of which documents are composed. SGML provides a general purpose mechanism for string substitution, that is, a simple machine-independent way of stating that a particular string of characters in the document should be replaced by some other string when the document is processed. One obvious application for this mechanism is to ensure consistency of nomenclature; another, more significant one, is to counter the notorious inability of different computer systems to understand each other's character sets, or of any one system to provide all the graphic characters needed for a particular

application, by providing descriptive mappings for non-portable characters. The strings defined by this string-substitution mechanism are called entities.

SGML and HTML

SGML is used to define HTML. There are many common features between HTML and SGML including the descriptive markup nature, notion of a document type and DTD, platform-independence, tags like "DOCTYPE" (used to indicate document type), and comment structure. Formally, HTML is a SGML DTD. Various implementations of document mark-up languages have existed. One of the oldest, and perhaps best known is troff (Ossanna, 1976), a formatting and phototypesetting program, written originally in 1973 in PDP-11 assembler and then in barely-structured early C in 1975 by Joseph Ossanna (who was killed in an accident in 1977). It was modelled on the earlier roff which was in turn based on Multics' RUNOFF by Jerome Saltzer. In 1979, Brian Kernighan modified troff so that it could drive phototypesetters other than the Graphic Systems CAT. His paper describing that work (Kernighan, 1979) explains troff's durability. This software is part of the Unix operating system which was the environment used by Berners-Lee (in the form of a Next cube) in developing the WWW.

VRML

VRML was conceived in the spring of 1994 at the first annual World Wide Web Conference in Geneva, Switzerland. Tim Berners-Lee and Dave Raggett organized a Birds-of-a-Feather (BOF) session to discuss Virtual Reality interfaces to the World Wide Web. Several attendees described projects already underway to build three-dimensional graphical visualization tools, which inter-operate with the Web. Attendees agreed on the need for these tools to have a common language for specifying 3D-world description and WWW hyper-links. The term Virtual Reality Markup Language (VRML) was coined, and the group resolved to begin specification work after the conference. The word "Markup" was later changed to "Modelling" to reflect the graphical nature of VRML.

Shortly after the Geneva BOF session, the www-vrml mailing list was created to discuss the development of a specification for the first version of VRML. The list quickly agreed upon a set of requirements for the first version, and began a search for technologies which could be adapted to fit the needs of VRML. The search for existing technologies turned up a several worthwhile candidates. After much debate the Open Inventor ASCII File Format from Silicon Graphics, Inc.was selected.

VRML 1.0 was designed to meet the following requirements:

- Platform independence
- Extensibility
- Ability to work well over low-bandwidth connections

Early on, the designers decided that VRML would not be an extension to HTML. This is as well because VRML is a file format rather than a mark-up language. The Virtual Reality Modelling Language (VRML) is a file format for describing 3D interactive worlds and objects and may be used in conjunction with the World Wide Web. VRML has been designed to fulfill the following requirements:

- Authorability. To make it possible to develop application generators and editors, as well as to import data from other industrial formats.
- Completeness. Provide all information necessary for implementation and address a complete feature set for wide industry acceptance.
- Composability. The ability to use elements of VRML in combination and thus allow re-usability.
- Extensibility. The ability to add new elements.
- Implementability. Capable of implementation on a wide range of systems.

- Orthogonality. The elements of VRML should be independent of each other, or any dependencies should be structured and well defined.
- Performance. The elements should be designed with the emphasis on interactive performance on a variety of computing platforms.
- Scalability. The elements of VRML should be designed for infinitely large compositions.
- Standard practice. Only those elements that reflect existing practice, that are necessary to support existing practice, or that are necessary to support proposed standards should be standardised.
- Well-structured. An element should have a well-defined interface and a simply stated unconditional purpose. Multipurpose elements and side effects should be avoided.

There is a mapping between VRML elements and commonly used 3D application programmer interface (API) features. The scope of the standard incorporates:

- a mechanism for storing and transporting two-dimensional and three-dimensional data elements.
- methods of representing two-dimensional and three-dimensional primitive information
- elements for defining characteristics of such primitives elements
- methods for viewing and modelling two-dimensional and three-dimensional information
- a container mechanism for incorporating data from other metafile formats
- mechanisms for defining new elements which extend the capabilities of the metafile to support additional types and forms of information

IGL

The Author, together with Graham Ellis, a software analyst from the Information Display Group of Tektronix®, made the first U.K. installation of Tektronix' Interactive Graphics Library (IGL) in 1978. This library, in effect, replaced the Plot-10 Storage Tube library with a new library to deal with the emerging raster-scan devices and was an early implementation of the ACM Core Standard (GSPC, 1977). The main features of the standard are remarkably similar to the requirement specifications of VRML 1.0, in that it was to be platform independent, extendable and was constrained, not so much by band-width considerations, as the (relatively) very limited computer power available at that time. The set of requirements for VRML listed above (extracted from the Standard) would serve just as well for IGL/Core as VRML.

One of the important features of the "Core System" was the distinction made between modelling and viewing functions. The standard was, essentially, a viewing rather than a modelling system. The expectation was that given a clear definition in the standard, modelling capabilities could be built on top of the other capabilities of the system: this is what IGL did. Because of the expectation that a variety of higher-level systems would be built on top of the proposed standard, the standard was viewed as a set of core capabilities, and thus became known as the "Core System". This approach reflects the situation where many VRML models are built in dedicated modelling systems, and, given the well-specified file-format of VRML, the data model is then exported to VRML.

Conclusion

The constraints of low-speed dial –up lines to remote time-sharing mainframe computers that were prevalent in the 1970's forced researchers to carefully consider the structure of software programs. The spread of relatively affordable minicomputers and

raster graphic terminals forced a consideration of device dependence. Both of these features have laid foundations that have been successfully built on by researchers working in the current distributed environment of personal networked computers.

References

GINO-F, the General Purpose Graphics Package Reference Manual. CAD Centre, Cambridge, 1975.

GSPC - Status Report of the Graphics Standards Planning Committee of ACM/SIGGRAPH. Published as Computer Graphics Quarterly, volume 11, number 3, Fall, 1977.

Guedj, R.A. et al, Report on the IFIP W.G. 5.2 Workshop, Methodology in Computer Graphics. IFIP, July 1976.

ISO 8879:1986 Information processing; Text and office systems; Standard Generalized Markup Language (SGML).

ISO/IEC 14772 Information Technology - Computer Graphics and Image Processing - Virtual Reality Modelling Language (VRML).

Brian W. Kernighan, A Typesetter-independent troff, AT&T Bell Laboratories Computing Science Technical Report no. 97, 1979.

Joseph F. Ossanna, nroff/troff User's Manual. AT&T Bell Laboratories Computing Science Technical Report no. 54, 1976.

Plot-10 Terminal Control System. Document 062-1438-00. Tektronix Inc. Beaverton, Oregon, 1971.

Alan Bridges
University of Strathclyde
Department of Architecture and Building Science
Glasgow G4 ONG
a.h.bridges@strath.ac.uk