The History of Computer Graphics Standards Development

Steve Carson
GSC Associates, Inc.

In keeping with the retrospective theme of this issue of *Computer Graphics*, Standards Pipeline takes a long look back at the history of the development of formal standards for computer graphics. As is fitting for a retrospective, there is a focus on where we have succeeded *and* where we have failed.

Three guest authors contributed to this column. First, Andy van Dam takes us back to the early beginnings of standards for computer graphics and the forces that led to the first attempts at standards. Andy ends by describing the beginnings of the PHIGS and PHIGS+ standards efforts. Our second contributor, Richard F. Puk, who was the editor of the ISO PHIGS standard, describes its development in more detail, ending with its successes and its failures and a vision for the future. Our final contributor, Lofton Henderson brings us up to date on the development of the Computer Graphics Metafile and its amendments.

Some Personal Recollections on Graphics Standards[1]

Andries van Dam
Brown University

Standards

Standards are a necessary evil. They are necessary because they allow portability of programs (and programmers) and interoperability, thereby reducing time-to-market of applications. They are evil because they are the result of a consensus-building process that takes many years and therefore produces specifications and implementations that lag the state of the art. Experience has shown that the best designs in any field are typically the product of a very small, elite group of system architects, while standards are the product of the art of techno-political compromise by larger committees. My experience with several decades of standards-creation efforts has been frustrating, but rewarding, and helped shape my professional development.

GINO

My interest in graphics standards came from a visit to Cambridge University and the Computer-Aided Design Centre (in the UK) in the late sixties where I was shown the device- (and platform) independent graphics package GINO (Graphical Input/Output System [WOOD71]). It stood in sharp contrast to the device-dependent graphics libraries that most of us were stuck with in these early days of plotters, various kinds of vector refresh tubes and the occasional vector storage tubes, although occasionally someone ported the Calcomp plotter or Tektronix tube libraries to other devices (typically by emulating the device!). When I took a crew of my Brown University students with me in 1971 for my sabbatical to Nijmegen in my home country of the Netherlands to set up a graphics lab and start computer science teaching and research, we decided to create an even richer package than GINO for the brand new graphics system the Math-Science faculty procured for us. This was a recently announced, hot DEC PDP 11/45 minicomputer with a 3D Vector General display that had built-in hardware for real-time traversal of hierarchical display lists (a simple form of what today is often called a scene graph), matrix transformations and clipping.

To divide the labor, I formed a collaboration with Peter Veenman and Edwin Herman in Delft Technical University in the Netherlands and with Peter Woodsford, GINO's boss, in Cambridge. Nijmegen was largely responsible for design of the new package (GPGS/the General Purpose Graphics System) [VAND77] and for the implementation for the /360 IBM mainframe, the host driving the intelligent graphics satellite of the PDP 11/45 + VG. Delft did the stand-alone PDP11 version, and Cambridge provided advice. Two of my students who did much of the work were my then-Ph.D. student, Dan Bergeron, now Chair of the Computer Science Department at the University of New Hampshire, and master's student Larry Carruthers, who has remained in the Netherlands.

One key contribution was generalizing GINO's concept of logical output devices to the idea of logical input devices. Logical output surfaces, regardless of size or coordinate-addressing technique, were uniformly addressed with floating-point normalized device coordinates, typically in the range of 0-1, rather than integer device coordinates. Logical input devices abstracted our standard set of hardware interaction devices: the keyboard, the locator (e.g., lightpen, joystick or cursor thumbwheels), the pick/selection device (typically a lightpen), valuator devices (e.g., a box of dials) and the button device (e.g., a set of physical function keys). These logical input devices were later adapted and extended by Core, GKS and PHIGS. They provided the first good abstraction of the many input devices then available and were discussed a few years later by Wallace and Foley in the first important papers on interaction for graphics.

GPGS

GPGS supported full hierarchy with transforms, guaranteeing simulation of that machinery and of all logical input devices on any interactive display device lacking that functionality. Thus, even a cheap direct view storage tube terminal without display lists, hardware transforms or anything more than keyboard and cursor thumbwheels, could run, albeit slowly, a program intended for the far more expensive high-end displays. We did not succumb to a lowest common denominator design. By 1972, GPGS was usable, if not yet robust. The later FORTRAN version, GPGS-F, done at Delft, was distributed to some 60 sites worldwide and was used as the Scandinavian standard.

[1]As with anyone's recollections spanning nearly three decades, I have undoubtedly exhibited "selective memory" and bias, not to mention having had to omit important people, places and things due to space limitations. I refreshed my memory by reviewing the Special Issue on Graphics Standards of the ACM Computing Surveys, December, 1978, and checked my "facts" with Dan Bergeron, Jim Foley and Jose Encarnação. The errors that remain are purely my responsibility.

The SIGGRAPH Graphics Standards Planning Committee (GSPC)

In 1974, Jim Foley and my former student Ira Cotton organized a workshop on the potential of graphics standards, held at Ira's workplace, the National Bureau of Standards in the United States, (now the National Institute of Standards and Technology - NIST). I recall little of the content, but vividly remember a late night gathering at a whiteboard with half a dozen folks, including Dan Bergeron, outlining the set of GPGS features we felt comfortable proposing for a standard, based on broad experience with that package. The SIGGRAPH Graphics Standards Planning Committee (GSPC) was formed at the workshop, but there was little progress subsequently on actually specifying a standard.

In fact, nothing significant happened until Richard Guedj, under IFIP WG5.2 sponsorship, convened most of the top several dozen graphics leaders at a workshop in Seillac, France to consider the methodology of graphics standards. Among those I remember were William Newman (of Newman and Sproull, authors of the first graphics book), Tom Sancha and Martin Newell of the CAD Centre, Jose Encarnação, Jim Foley, Bert Herzog and guest speakers Alan Kay and Nick Negroponte. There was fierce debate on whether the time was ripe to try to produce such a standard, and if so, what features, indeed what model of graphics, it should encode. We agreed that the graphics standard should represent what was generally accepted in the community as a "code of practice," a criterion used to separate *modeling*, in particular various forms of representing component hierarchy, à la Sketchpad, from *viewing*, i.e., simulating a "synthetic camera." The latter might be ready for standardization while the former was judged still to be in the research domain. The modeling/viewing dichotomy persists to this day as designers still heatedly argue the pros and cons of retained mode graphics packages that support some form of scene graph versus more RISC-like immediate mode packages.

I remember Nick Negroponte warning that if people like me succeeded in promulgating a graphics standard, "it would set the field back at least a decade." With the benefit of hindsight, I can't say he was completely wrong, but I do believe that what happened next was, in fact, mostly beneficial, if of less impact than we proponents had hoped. In any case, Jim Foley, Bert Herzog, Jose Encarnação and I left determined to design a proto-standard despite all the concerns that the opposition had expressed, and we convened a small group that met regularly for a year, mostly at Brown, while Jose continued working in parallel with his German colleagues.

CORE

Our committee was chaired by Jim Foley and Dan Bergeron, under the aegis of SIGGRAPH's GSPC, which was chaired by Bert Herzog and Bob Dunn and included two future graphics stalwarts, my former Ph.D. student Jim Michener and Peter Bono, as well as Ph.D. student Ingrid Carlbom and Elaine Sonderegger. We produced a specification for the 3D Core Graphics System, published in the proceedings of SIGGRAPH 77 [GSPC77]. I lost the fight to include the simple modeling that GPGS and high-end hardware supported, i.e., hierarchical display lists, but saw many other features of GPGS incorporated in whole or in part, such as logical input devices.

We did stick pretty much to accepted codes of practice except in the area of viewing where we came up with an ambitious (and admittedly somewhat complex) unified viewing model that made 2D a subset of 3D viewing and allowed arbitrary perspective and parallel view volumes to be constructed with nearly identical parameters. Ingrid did much of the research on viewing systems, including Dick Puk's library. Our idea was to provide a device-independent viewing package that would allow other, more application-specific packages including modeling packages (hence the term "Core") to be built "on top."

GKS

Jose and his committee at the same time used various European packages as well as an early version of our Core specification as inspiration for their GKS (Graphical Kernel System) specification [GKS85]. Viewed from the 50,000 foot level, GKS, which eventually became an ISO standard, is a 2D version of Core, servicing the 80 percent or so of then-popular applications that were 2D and non-hierarchical, e.g., GIS, CAD and electrical circuit layout. The design center for Core was 3D applications on high-end hardware while GKS was more oriented towards the far more common and cheaper storage tubes and plotters. GKS was later extended to 3D. Both Core and GKS were designed in an era of vector graphics, though each had elementary raster graphics extensions that were rarely used. [FOLE79]

PHIGS

Soon enough, 3D hierarchical display lists and transformation hardware became more prevalent and a mostly new set of people started rethinking various design decisions for Core and GKS. These efforts eventually led to the ISO standardization of PHIGS (Programmer's Hierarchical Interactive Graphics System). Unfortunately, it took at least six years of hard work by an international committee to converge on this spec [PHIG89], which still was primarily a line-drawing package, in an era that saw the rapid conversion from vector graphics to color raster graphics. As it became clearer in the mid-80s that hardware would soon be fast enough to do real-time rendering using Gouraud and Phong lighting and shading models, there was less enthusiasm for a large, complex package that did not support such primitives as triangle meshes and lighting/shading models. Device-dependent packages, particularly industry leader SGI's GL and later OpenGL [OGL95], became more popular than the platform-independent PHIGS.

PHIGS+

In 1986, I again convened a small, ad hoc committee, this time with some 20 representatives mostly from the workstation industry and some from academe, to try to extend PHIGS for the modern era of real-time graphics. We started with a proprietary specification donated by Raster Technology and in the space of less than two years produced a specification for PHIGS+ [PHIG88]. PHIGS+ was the base document for the ISO standard PHIGS PLUS , finalized in 1992 and in turn served as the basis for an extension to the X Consortium's X protocols, PEX (PHIGS Extensions to X.) For a while, there was a big fight primarily among the workstation vendors about which of OGL or PHIGS+/PEX should be adopted, but clearly formal (ANSI/ISO) standards in graphics lost the battle to leaner, faster-moving de facto industry standards.

Lessons Learned

What lessons can be drawn from this very compressed history? Those of us who fought for standards believe that particularly in the `70s, when graphics was still a nascent field, having a generally agreed-upon set of concepts and terminology helped form the field by educating the developer and instructor community. Furthermore, standards (including the Computer Graphics Metafile) [CGM92] provided a baseline for the development of more state-of-the-art packages. Finally, applications were, in fact, written to these early standards, making possible platform – and therefore vendor – independence, one of our key goals.

And those who warned that premature standardization would inhibit progress were fortunately wrong: However good or bad the standards were (often a matter of point of view), they clearly did not inhibit the wonderful progress of our field over the last several decades. By now, the wheel of reincarnation has turned again and the world waits to see which of the many proposed 3D graphics (and a bit of other media) APIs will survive: SGI's OGL/OGL++, Microsoft's Direct3D, and their impact on the future Fahrenheit set of APIs to be designated jointly by SGI and Microsoft, the VRML Consortium's VRML (the only ISO standard covering graphics on the Web) or Sun's Java3D. It is disheartening to me that after all these years, no clear winner has emerged that is platform independent, widely implemented and neither too lean nor too rich. What is clear is that the time-scale compression induced by the phenomenon of Web time, with its emphasis on time-to-market rather than on quality and robustness, militates against standards design processes that take more than a few years; this in turn argues against design-by-committee and the kind of consensus building required by an open standards process.

PHIGS: The Programmer's Hierarchical Interactive Graphics System

Dick Puk
Intelligraphics

Background

In 1979, the Graphics Standards Planning Committee published the specifications of a prototype for a standard for three-dimensional computer graphics called "Core" [GSPC77]. The intent was that this specification be introduced as the initial working document for an American National Standard for computer graphics. At the same time, a specification for a two-dimensional standard (GKS) was submitted to the International Standards Organization as a working draft. In order to avoid competing standards, it was decided by international agreement that the GKS standard introduced by Germany would form the basis for a two-dimensional standard after which work would commence on a three-dimensional standard based on the United States proposal. This three-dimensional standard would be called The Programmer's Hierarchical Interactive Graphics System (PHIGS). Work on GKS commenced immediately. It was not until 1984 that work on PHIGS began.

Basic PHIGS

It took until 1989 to specify PHIGS and win its approval as International Standard 9592 [PHIG89]. This basic version of PHIGS included the following major features:

- A editable hierarchical organization of graphics data called the structure store
- A powerful logical input model in support of interaction devices
- A workstation mechanism to allow support of multiple simultaneous input and output devices
- 3D graphics primitives which could be specified using either 2D or 3D functions
- Separate attributes based on primitive type

Implementations of basic PHIGS were provided by all of the major workstation vendors. However, it soon became apparent that a 3D graphics system had to support much more than just the features listed above. Therefore, as the work on basic PHIGS neared completion, a new project was initiated in support of changes in the state of the art. This new project was entitled PHIGS Plus Lumière Und Surfaces or PHIGS PLUS.

PHIGS PLUS

The PHIGS PLUS project was initiated by a working group formed outside of the standards organization [PHIG88]. The goal of this group was to define a standard, fully upward compatible extension to PHIGS to provide support for the following new capabilities:

- 3D surface primitives with material properties
- Non-rational B-Spline line and surface primitives
- Support for direct color (non-indexed color)
- Enhanced rendering with support for lighting and shading
- Hidden line and hidden surface removal

This effort was successfully completed with the approval of IS 9592-4 in 1992. Once again, implementations were provided by the major workstation vendors.

As commercial applications began to use PHIGS and PHIGS PLUS, some additional areas of improvement in overall PHIGS functionality became recognized. Many extensions to PHIGS were incorporated in the ISO Registry of Graphical Items. However, it soon became apparent that this was an inadequate mechanism of standardizing the proposed enhancements.

Full PHIGS

In order to more clearly specify the desired enhancements, a series of amendments were approved each with a narrowly defined goal. The amendments themselves would not be published. Instead, after approval of all amendments, PHIGS would be republished incorporating not only all of the amendments but also the PHIGS PLUS functionality into a single document. PHIGS republication was completed in 1997 and includes the following new features:

- Support for direct interpretation structures (so-called "immediate mode") including picking of direct interpretation primitives
- Programmer-definable logical input devices
- Association of logical input reports with workstation state table entries to provide automatic updating of values
- Programmer-definable highlighting methods, linetypes, and marker styles
- Support for texture mapping and transparency
- Rendering to targets other than the display surface (including double buffering)
- Incorporation of most registered items including line caps and line joins

Most of the capabilities defined in these enhancements had already been incorporated in commercial PHIGS implementations at the request of users of those implementations.

PHIGS Support

PHIGS has defined three profiles to encourage interoperability of programs. These three profiles are:

- Basic PHIGS
- PHIGS PLUS
- Full PHIGS

which match the functional categories defined above. In addition, definition of these profiles was necessary also to ensure that existing PHIGS language bindings did not become obsolete.

In order to use the abstract functionality defined by the PHIGS standards, bindings of the abstract functionality to four specific programming languages occurred throughout the PHIGS development period [PHLB90]. Bindings to support Fortran-77, Pascal, Ada, and C are available for the Basic PHIGS and PHIGS PLUS profiles. An amendment to the C binding will soon be published to support the Full PHIGS profile.

PHIGS Successes

The Basic PHIGS and PHIGS PLUS profiles are supported on major workstation platforms. Many applications have been implemented using these products. Full PHIGS implementations have not yet been provided by most workstation vendors. However, an implementation of much of Full PHIGS is currently used in a major Japanese CAD program which is used by the major Japanese automotive companies.

Where PHIGS Has Failed

PHIGS is no longer used extensively for the implementation of graphics applications. It has been superseded by OpenGL [OGL95] which presently is enjoying wide success as a de facto standard. One primary reason has been the inability of standards organizations to react to the needs of the industry in a timely fashion. In addition, PHIGS never had the large marketing budget needed to promote it. Finally, Full PHIGS, while providing the requisite capabilities desired by the industry, became available too late. Interestingly, OpenGL is now incorporating some of the capabilities of PHIGS to shore up its own functional weaknesses.

On the Horizon

Both PHIGS and OpenGL are legacy standards (formal and de facto, respectively). Three-dimensional graphics functionality is now being integrated with other media as part of a greater multimedia whole. In addition, applications are appearing which incorporate behaviors as an integral part of virtual worlds along with audio and video. Nowhere is this more evident than on the World Wide Web where the Virtual Reality Modeling Language is gaining acceptance and, indeed, has recently been adopted as ISO/IEC 14772. [VRML97] While both PHIGS and especially, OpenGL, will continue to be used, applications are now being designed with an object-oriented paradigm which relegates such legacy systems to a support role.

CGM: The Computer Graphics Metafile

Lofton R. Henderson
Inso Corporation

A Standard Whose Time Has Come

In 1996, CGM was registered as a MIME type for Web usage. Currently, a Web profile is being developed, and W3C gives positive support for vector graphics usage of CGM in appropriate Web applications. Purveyors of CGM Web browser plug-ins note rapidly rising inquiries and sales in the past year alone. Commercial activity around CGM technology in general continues to accelerate in the U.S., Europe and Asia. There is little doubt that the time has come for CGM, as one of the most successful of the formal ISO graphics standards.

Definition

CGM is formally designated: ISO/IEC 8632/1-4:1992. It is a four-part standard [CGM92] defining a file format for the device-dependent and application-independent capture, storage and transfer of graphical pictures (targeted principally at vector graphics, but also containing advanced features for raster graphics).

The Beginning

CGM's roots are in the late 1970s, when SIGGRAPH's Graphics Standards Planning Committee (GSPC) proposed the graphics standard "Core" [GSPC77], and appended a proposal for an associated graphical file storage format — a "metafile" (for one explanation of the oft-questioned term "metafile," see [HEND93]).

Formative Years

In the early 1980s, the accredited graphics standards committee in the U.S. known as X3H3 took over the Core proposal and began to develop it as an American National Standard. This included continuing the development of Core's metafile proposal as an annex (called at that time the VDM — Virtual Device Metafile) to the Virtual Device Interface (VDI) standardization project.

Concurrently in Europe, the use of the graphical standard GKS (Graphical Kernel System, an alternative to Core) [GKS85] was rapidly progressing and GKS eventually became an ISO standard. The standard contained a non-normative annex defining a GKS Metafile (GKSM).

In 1982, representatives of ISO and X3H3 began to discuss the separation of VDM (then a 70-page draft document) from VDI, and its rapid progression as a separate ISO metafile standard. The assertion that "VDM is already mostly what is needed in an ISO metafile standard" was appealing. At the same time this statement concealed the deep architectural and philosophical differences between the metafile concepts of two significant application groups — the static picture capture metafile of VDM and the audit trail metafile of GKSM [HEND93].

Finally, A Standard

Understanding and eventually overcoming this philosophical divide accounted for much of the effort of the metafile standard development project during the next four to five years. Nevertheless, the metafile standardization project became a truly international effort, and the contribution of unique talent and ideas from both Europe and North America ultimately resulted in the definition of the first international metafile standard. The nature of the metafile was resolved (virtually on the eve of publication) as "picture capture," rather than "audit trail" and in 1987 ISO published the metafile standard, Computer Graphics Metafile (CGM), Version 1 [CGM86].

Development and Growth

As CGM Version 1 progressed toward completion, numerous proposals for advanced graphics capabilities were considered. There was no dispute about the utility or desirability of features such as Bezier curve primitives, but there was concern that "creeping functionality" could lead to endless delay in the completion and publication of the standard.

The CGM group decided that an initial, Version 1, standard defining a "basic picture drawing format" would be completed as rapidly as possible. Even before publication, work commenced to define and satisfy the requirements for additional, upward compatible functional levels.

In 1989 CGM Amendment 1 was published, defining Version 2 metafiles, and in 1991 Amendment 3 was finished, defining CGM Version 3 (see [HEND93] for an explanation of the never-completed Amendment 2). Specific application areas — technical illustration, cartography, presentation graphics, and graphic arts — drove the functionality of these addenda, culminating in the advanced, state of the art, integrated vector and raster picture format of Version 3.

Completion

The CGM International Standard was republished in 1992, integrating the two amendments and a number of defect corrections. At the time of publication, work was already well along on Amendment 1 to CGM:1992, Rules for Profiles (the "conformance amendment"). This amendment addressed the issues of interoperability, conformance, testing and product certification — issues critical to the success of *any* open data interchange format.

In 1994, work was completed on the final significant addition to the CGM standard — CGM:1992 Amendment 2, Application Structures (APS). Again, no new graphical elements were involved. Rather APS provide the means to define graphical objects within CGM pictures and associate non-graphical application information, such as hotspot and hyperlink information.

Acceptance

As early as the late 1980s, the potential of the CGM as a neutral interchange format was demonstrated in such expositions as NCGA and Eurographics UK. However, commercial adoption lagged. The principal reason was the prevalence of bad implementations — incomplete, incorrect and unable to interoperate. Amendment 1, Rules for Profiles, enabled the establishment of certification testing, and this was a key development in resolving these interoperability barriers. The first implementations were certified in 1995, and since then the general improvement in quality has been key to enabling CGM's uptake. The APS of Amendment 2 finally made CGM an ideal SGML peer and an ideal Web format. Together with good certified implementations, it has enabled the real explosion in CGM usage.

It Wasn't All Grim

What about the process of developing an international standard such as CGM? Some observed that it looked grim, and sometimes it was. The hours of debate about silly and unimportant details tend to confirm the adage: the time spent on a particular detail is inversely proportional to its importance and to the likelihood that it will ever matter.

It wasn't all grim, however. The people, places and anecdotes make for a rich history: the comic and slapstick speed with which these allegedly slow and sluggish standards writers clear the room when a small earthquake interrupted a meeting; the enjoyment of invention of the "pointless element" (an accurately named but essential feature of Version 2); the invention of the Graphics Interface to Minimal Programmers (GIMP) as a vehicle to prove the necessity of formal specifications as part of the standard (PMIG — Programmers Minimal Interface to Graphics — was a real attempt to preserve the Core standard as a subset of the PHIGS standard).

The people are the true stars of any retrospective. Over the years, a unique collection of bright individuals from North America, Europe and Asia struggled with numerous conflicting ideas (and personalities!). The final result is a fusion of the best aspects of competing viewpoints, and a technically strong and sound consensus standard.

Was There a Better Way?

When VDM separated from VDI and became CGM, VDI proceeded as CGI (Computer Graphics Interface) [CGI91]. Unlike CGM, CGI undertook to finish all parts of a much-bigger-than-CGM standard as a monolithic whole. The result was an excellent and impressive piece of work, but during the 11 years it took to complete, its market disappeared and CGI was moot by the time it was finished.

As the CGM group received requests for increased functionality, the approach taken was: finish the basic level, and let the needs of user constituencies drive a series of addenda. The approach was effective, but it was a lot of work. Another approach has been suggested. Define the a basic and lean standard, but with easy and powerful extensibility mechanisms built in. This is effectively the approach of VRML. In retrospect, it might have worked for CGM, might have been less work and would better fit with modern object-oriented paradigms. Nevertheless, CGM has ultimately met its users' requirements in a reasonably timely manner.

What Next

CGM formal standardization is finished. The standard is being republished again in 1998 to roll in the two amendments (Profiles and APS) and to fix some 95 identified defects. This is not to say that development is finished. Indeed, the exciting work is just beginning, with the adoption of CGM as an Internet and intranet format, and with its integrated use in advanced, interactive electronic document architectures. This work will not, however, be initiated in standards committees, but rather in industry user groups, in the W3C and in the new CGM Open consortium. The resulting industry specifications will then likely be transposed into International Standards following the model recently pioneered by VRML [VRML97].

The ISO/IEC JTC 1/ SC 24 WG6 Metafiles Working Group is largely focused on other technologies now — the collaboration with the VRML Consortium that has made VRML an ISO standard and the collaboration with W3C to make the PNG (Portable Network Graphics) specification an International Standard.

Acknowledgments
I want to acknowledge the helpful suggestions for this CGM retrospective of the following people, each of whom has also been instrumental in the work of the ISO CGM committee — Chris Osland, Alan Francis, Eckhard Moeller and Anne Mumford — all key contributors without whom CGM would not have turned out as it has.

References (All Three Contributions)

[CGI91] ISO/IEC 9636-1:1991 Information technology -- Computer graphics -- Interfacing techniques for dialogues with graphical devices (CGI) -- Functional specification -- Part 1: Overview, profiles, and conformance, Part 2: Control , Part 3: Output, Part 4: Segments, Part 5: Input and echoing and Part 6: Raster.

[CGM86] ISO/IEC 8632:1986 Information technology -- Computer graphics -- Metafile for the storage and transfer of picture description information -- Part 1: Functional specification, Part 2: Character encoding, Part 3: Binary encoding, and Part 4: Clear text encoding.

[CGM92] ISO/IEC 8632:1992 Information technology -- Computer graphics -- Metafile for the storage and transfer of picture description information -- Part 1: Functional specification, Part 2: Character encoding, Part 3: Binary encoding, and Part 4: Clear text encoding.

[FOLE79] Foley, J. C. "Some Raster Graphics Extensions to the Core System," Proceedings of SIGGRAPH 79, published as *Computer Graphics* 13(2) August 1979, pp 15-24.

[GKS85] ISO/IEC 7942:1985 Information technology -- Computer graphics -- Graphical Kernel System (GKS) -- Part 1: Functional description, Part 2: NDC metafile, Part 3: Audit, and Part 4: Archive.

[GKS94] ISO/IEC 7942:1994 Information technology -- Computer graphics and image processing -- Graphical Kernel System (GKS) -- Part 1: Functional description, Part 2: NDC metafile (updated in 1997), Part 3: Audit, and Part 4: Archive.

[GSPC77] Graphics Standards Planning Committee. "Status Report of the Graphics Standards Planning Committee of ACM SIGGRAPH," *Computer Graphics*, 11(3), Fall 1977.

[HEND93] Henderson, Lofton R. and Anne M. Mumford, *The CGM Handbook*, Academic Press Inc., 1993.

[OGL95] Segal, Mark and Kurt Akeley, The OpenGL Graphics System: A Specification (Version 1.1), Silicon Graphics, Inc., 1995.

[PHIG88] PHIGS+ Committee, Andries van Dam, chair, "PHIGS+ Functional Description, Revision 3.0," *Computer Graphics*, 22(3), July 1988, pp. 125-218.

[PHIG89] ISO/IEC 9592:1989 Information technology -- Computer graphics and image processing -- Programmer's Hierarchical Interactive Graphics System (PHIGS) -- Part 1: Functional description, Part 2: Archive file format and Part 3: Specification for clear-text encoding of archive file.

[PHIG97] ISO/IEC 9592:1997 Information technology -- Computer graphics and image processing -- Programmer's Hierarchical Interactive Graphics System (PHIGS) -- Part 1: Functional description, Part 2: Archive file format and Part 3: Specification for clear-text encoding of archive file.

[PHLB90] ISO/IEC 9593:1990 Information processing systems -- Computer graphics -- Programmer's Hierarchical Interactive Graphics System (PHIGS) language bindings -- Part 1: FORTRAN, Part 3: ADA and Part 4: C.

[VAND77] Van Den Bos, J., et al "GPGS-a device-independent general-purpose graphics system for stand-alone and satellite graphics," *Computer Graphics*, proceedings of SIGGRAPH 77, 11(2) Summer 1977, ACM SIGGRAPH, New York, NY, pp. 112-119.

[VRML97] ISO/IEC 14772-1:1997 Information technology -- Computer graphics and image processing -- The Virtual Reality Modeling Language (VRML) -- Part 1: Functional specification and UTF-8 encoding.

[WOOD71] Woodsford, P.A. "The design and implementation of the GINO 3D graphics software package," *Softw. Pract. Exper.* 1, (Oct. 1971), pp. 335.

George S. (Steve) Carson is President of GSC Associates of Las Cruces, NM, a systems engineering consulting firm specializing in real-time signal and information processing systems. He is the Chairman of ISO/IEC JTC 1/SC 24 (Computer Graphics and Image Processing) and has been involved in ANSI and ISO standards development for 20 years.

George S. Carson
GSC Associates, Inc.
5272 Redman Road
Las Cruces, NM 88011
Tel: +1-505-521-7399
Email: *carson@siggraph.org*