



www.hoops3d.com

The Added Value of HOOPS® Relative to OpenGL

1 Introduction	1
2 What is OpenGL?	1
3 What is HOOPS?	3
4 HOOPS' Added Value over OpenGL	5
4.1 PRIMITIVE SET	5
4.2 3D SPLINED OUTLINE FONTS.....	5
4.3 HIDDEN LINE REMOVAL ALGORITHMS	6
4.4 HARDCOPY OUTPUT	6
4.5 SELECTION ALGORITHMS.....	7
4.6 RENDERING TEXTURED OBJECTS.....	7
4.7 DATABASE TRAVERSAL OPTIMIZATIONS	8
4.8 MIXED MODE STRUCTURE: HOOPS INTERMEDIATE MODE	8
4.9 OBJECT MODELING FLEXIBILITY	9
4.10 TECHNICAL SUPPORT AND CONSULTING SERVICES.....	9
5 Summary	10

1 Introduction

The HOOPS graphics system is a high-level graphics system which, among other things, stores 2D & 3D geometric information and renders it to screens and printers. Workstations and printers supply interface languages for rendering and HOOPS has the ability to communicate to output devices via several distinctly different interface languages, including OpenGL.

HOOPS and OpenGL are complementary technologies which when combined give developers a powerful cross-platform graphics development environment. In fact, HOOPS is listed as one of the higher level OpenGL API's on the OpenGL website:

www.opengl.org/Developers/developers.html

This paper starts with high-level descriptions of both OpenGL and HOOPS then discusses some of the specific functionality already in HOOPS that a developer would need to implement if they choose to write directly to an immediate mode interface like OpenGL.

2 What is OpenGL?



OpenGL (www.opengl.org) is a specification for an immediate mode 3D graphics library based on streaming, vertex-oriented z-buffering and has its historical roots in the GL library from Silicon Graphics, Inc. The OpenGL Architecture Review Board governs the specification and as of April 1998, its members consist of: 5 UNIX hardware vendors (DEC, Evans & Sutherland, HP, IBM, Silicon Graphics), one PC manufacturer (Intergraph), a chip maker (Intel), and an operating system developer (Microsoft). The ARB meets every 3 months and the most current version of the OpenGL specification is 1.2.

In order for an interface specification to be of use to an application programmer it must be connected to a hardware device; i.e., it must be implemented. Most of the UNIX vendors have OpenGL implementations available for their workstations and most IBM PC accelerator board manufacturers provide OpenGL drivers for their new products. Not all vendors support the same version of OpenGL.

See: www.opengl.org/Developers/overviews.html

As is common with committee based APIs, the OpenGL specification does not provide everything a given hardware or software company may need and there is a formal method for defining vendor specific extensions to OpenGL. Thus, the various implementations available are slightly different and out of phase, requiring the application developer to monitor these differences and to create and maintain multiple versions of their interface code.

See:

www.opengl.org/Developers/Documentation/aboutextensions.html

As an immediate mode library, OpenGL cannot directly provide higher level geometric optimizations that API's like SGI's Optimizer, HP's Direct Model and Microsoft's Farenheit are attempting to address. Unfortunately, if a development group has written directly to OpenGL and they wish to leverage such emerging higher level technology, this will require a complete rewrite of their graphics sub-system.

While it is difficult to imagine that OpenGL will vanish, precedent in the computer hardware industry suggests that eventually it will. Specifically, who knows if the joint Microsoft/SGI/HP Farenheit project will actually succeed in melding OpenGL, Direct3D, Direct Model, & Optimizer? But if it does, one of the stated goals of the Farenheit project is to maintain backward compliance with the OpenGL specification. The implication being OpenGL may not be its main immediate mode API and performance may suffer.

See:

www.sgi.com/Headlines/1997/December/fahrenheit_release.html



3 What is HOOPS?

HOOPS (www.hoops3d.com) is a software component providing for the creation, storage, manipulation, querying and rendering of 3D graphical information. HOOPS is developed, distributed and supported by Tech Soft America. It is provided as a set of dynamically linked libraries, is available on all UNIX workstation and the IBM PC and the libraries are 100% source code compatible. Support for the Dec Ultrix, HP-UX for HP300/400, Intergraph Clipper Unix, IRIX 4.0x for SGI, Linux, OpenVMS VAX, XDOS MetaWare / Phar Lap, Windows NT for Intel using WATCOM, Windows NT for MIPS and SunOS platforms are available via obfuscated source code ports. As of March 1999, the current release of HOOPS is 4.41.

Fundamentally HOOPS consists of two main parts: the object database and the structured device interface layer.

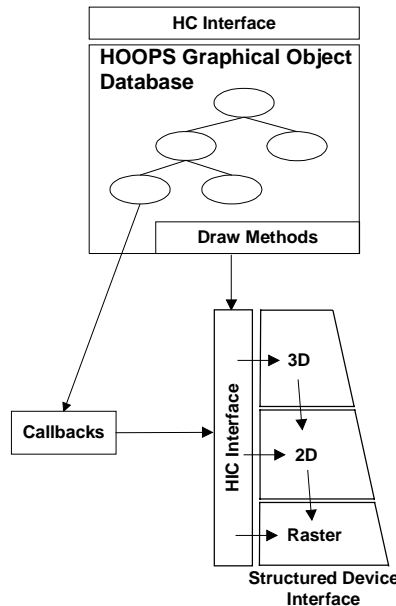


Figure 1. The HOOPS Graphics System: Object Database & Structured Device Interface

The database stores graphical objects and provides methods for traversing and sending the objects to the structured device interface. This interface accepts information found in the object database, reformats it for the hardware output device interface and then sends it to the device for drawing. A successive decomposition technique for reformatting is used where the information in the database passes through software mapping layers until it is in the format the output device can handle.



If the hardware device interface understands 3D information, as with OpenGL, HOOPS can pass along information without much change, but if the device interface only understands 2D pixels, like PostScript, GDI or HPGL, several layers of decomposition must be used. This approach ensures both optimal rendering throughput on a given device and consistent functionality of the HOOPS API across multiple platforms and devices.

Given that 70% of all UNIX machines and a majority of PCs in use today lack hardware acceleration for 3D graphics, special emphasis has gone into optimizing the performance of the HOOPS software rendering pipeline. In some cases on systems with graphics hardware, HOOPS software can be faster than the hardware due to highly optimized rendering algorithms.

For more detailed information on HOOPS' capabilities, see the document, [HOOPS: A Technical Overview](#), available on the HOOPS website.



www.hoops3d.com

4 HOOPS' Added Value over OpenGL

Like all immediate mode graphics libraries, OpenGL is unable to remember what has been drawn. Therefore to interact with 3D information, the application needs to store this information itself, compute the effects of user interaction (e.g., mouse movement) on the data and then communicate the new set of information representing the current state of the 3D information.

Use of OpenGL alone requires the application developer to create data-structures for storage and implement algorithms for traversing, rendering & querying of all graphics objects. Below are listed some of the specific functionality in HOOPS a developer would need to re-create if they were writing directly to OpenGL.

4.1 Primitive Set

OpenGL has a limited set of low-level primitives consisting primarily of polylines, polygons and triangle strips. Consequently, developers must provide the logic to tessellate their geometric data into these primitives. This requires a significant amount of effort when a developer is working with non-trivial data sets, particularly if performance is an issue. Here are two examples:

- 1) If your application needs to represent N-faceted, irregular polyhedra you will need to tessellate them into a set of optimal length triangle strips.
- 2) OpenGL does not directly support concave polygons and you will need to break these up into a convex set.

HOOPS provides a full set of geometric primitives for the needs of CAD/CAM/CAE, Scientific Visualization and GIS applications. Highly optimized routines ensure the most efficient primitives are passed to OpenGL. HOOPS directly supports:

Circles	Circular Arcs
Circular Wedge	Circular Chord
Ellipses	Elliptical Arc
Grids	Images
Lines	Markers
Meshes	Polygons
Polylines	Shells
Text	

HOOPS also provides distant and spot lights, cutting planes and cap planes (cutting planes that cap their regions of intersection).

4.2 3D Splined Outline Fonts

OpenGL has very limited font support and the ones that are available cannot be transformed in 3D or added to a scene where hidden surface



calculations are being considered. HOOPS supports True Type, Adobe Type 1, and all system supplied fonts and they are treated like any other piece of geometry in the scene. Thus, your HOOPS text is fully transformable in 3 dimensions and comes with all appropriate database methods like selection, editing and attribute settings.

HOOPS also includes support for the Japanese Kanji language and ISO-Latin double byte character sets as well as a portable font-naming scheme.

4.3 Hidden Line Removal Algorithms

As OpenGL is based on z-buffering and rasterization of primitives, it cannot supply any true analytical hidden line algorithms such as those found in HOOPS. In addition to providing access to hardware z-buffering when graphics accelerators are present, HOOPS also supplies several different software hidden line/hidden surface removal techniques including true analytical hidden line, z-sorting, and full painters algorithm.

End-user requirements in the manufacturing industry often call for a true analytical hidden line drawing on both the computer screen and paper hardcopy. OpenGL cannot provide this capability. HOOPS' true analytical hidden line algorithm provides for dynamic calculation of lines obscured from the viewer at any given vantage point. These "hidden lines" can either be not drawn or drawn with user selected attributes, e.g. a slightly dimmer or different color, a different line pattern, a different line thickness, etc.

4.4 Hardcopy Output

OpenGL does not support hardcopy output. In order to have hardcopy support the developer would have to write a completely separate module to output to a hardcopy-rendering pipeline. Furthermore, this work would have to be replicated for each additional hardcopy format. HOOPS can output in over 8 different hardcopy types including Postscript, CGM, Image and HPGL.



4.5 Selection Algorithms

OpenGL does not perform picking operations other than a square box pick. HOOPS can pick by point, window space 2D area, 3D model space volume, polygonal boundary and polyline fence. OpenGL, like all low level APIs, performs picking by re-drawing the world and watching which of the pixels would have hit the pick rectangle (i.e., the selection comparison takes place in device coordinates).

HOOPS performs the pick operations by computing the analytical intersection of the geometry and the selection region and returns precise information. Additionally, HOOPS can leverage higher level object information such as visibility, selectability and bounding box attributes to optimally traverse the object database in searching for primitives to hit-test against the pointing device position. The performance of picking is often several orders of magnitude faster in HOOPS.

For the developer, the various selection methods supplied by HOOPS enable the fine grain interaction techniques required for applications interested in creating and editing 3D geometry.

4.6 Rendering Textured Objects

OpenGL pre-computes lighting effects in texture maps; thus, dynamically moving textured objects or textured objects lit by moving lights will have incorrect lighting.

HOOPS provides for per pixel lighting calculations on textures. Thus if a scene has a local light and a flat surface with a texture, the reflections will be correctly computed at all times. OpenGL does not provide this capability and lit textured objects will not have correctly computed specular highlights.

OpenGL also requires the size of texture maps to be in powers of 2. HOOPS imposes no such limit.



4.7 Database Traversal Optimizations

Due to its immediate mode nature, OpenGL cannot perform large model drawing optimizations. HOOPS maintains bounding hierarchies and change flags in its database and uses this information when deciding what information actually needs to be sent to the low level immediate mode API for drawing. For example, if you have just finished drawing a Boeing 747 that has hundreds of thousands of polygons, it would be extremely inefficient to have to redraw the entire scene if a dialog box covers only the landing gear.

The logic enabling these intelligent traversal-time decisions about what must be redrawn, typically gets implemented in the late stages of a software components development cycle, requires effort from very highly skilled programmers and is often never completed within an in-house development effort.

4.8 Mixed Mode Structure: HOOPS Intermediate Mode

Often it is useful for application developers to program with a procedural, non-retained mode interface language like OpenGL. For example, if one wished to represent railroad tracks in a map, it would be very inefficient to store all of the cross ties in the graphics database. One could simply store the start and end points of the tracks and supply a procedure for computing where the cross-ties should be. HOOPS provides an additional library, HOOPS I.M., which provides such a procedural interface.

With HOOPS I.M. it is possible to interrupt the normal HOOPS update cycle at various points during traversal and rendering and have process control returned to the application via use of callback functions. The callback routines are registered with HOOPS as attributes associated with segments and/or geometry and when invoked, the callback routines are passed all the information for the associated geometry and attributes. When traversal is trapped at a callback point, decisions can then be made about what and how something should be drawn, or the traversal process itself can be aborted.

From within the callbacks, the HOOPS HIC immediate mode interface is available for query the graphics database and device characteristics, set attributes and draw all of the HOOPS primitives. The traversal process may be trapped at either the 3D or 2D layers in the rendering pipeline and drawing functions are available at both of these levels.



4.9 Object Modeling Flexibility

OpenGL only supports right-handed coordinate systems where in HOOPS right and left handed coordinate systems may be mixed in the same scene-graph. Polygonal facets must have a handedness defined for lighting and clipping calculations and again HOOPS provides for both right and left handed polygon handedness attribute values.

With HOOPS the application programmer is free to choose the method most appropriate for their task, even if it means intermingling right and left handed handedness attributes.

4.10 Technical Support and Consulting Services

HOOPS is more than just a set of libraries: Tech Soft America provides a complete set of technical support and consulting services and has a proven track record in supporting developers from design through delivery.

HOOPS has been in use by industry leading software development organizations for over 10 years and the technical support and consulting group has gained valuable insight and experience with each implementation effort. In many instances, the product direction of HOOPS, from functionality in the library to specialized demonstration or integration code, stems directly from this on-going dialogue with the customer base.



5 Summary

HOOPS and OpenGL are complementary rather than competing technologies.

OpenGL has established itself as the current dominant 3D immediate mode API and commercial implementations of the specification are available on most workstations today. The implementations may not be based on the same version of the OpenGL specification and may include vendor specific extensions. Thus an OpenGL application written for one platform may not run on another.

HOOPS contains higher level data structures and algorithms needed for commercial-grade CAD/CAM/CAE applications. With over 110+ man-years of effort, HOOPS delivers extensive functionality, is stable, mature, and well documented. Developers writing applications to the OpenGL immediate mode interface are choosing to re-implement this work.

OpenGL support spans most of the workstations available today, but in many cases is not the optimal programming interface for existing hardware installed in the manufacturing sector. And if history is any indicator, nothing is certain about the long-term availability of any immediate mode APIs. HOOPS offers developers the ability to deliver applications today that can be configured in the field by the end-user to use the optimal hardware device interface whether that is OpenGL, Starbase, XGL, PEX, GDI, XLib or any as yet to emerge API. Thus, the HOOPS architecture continues to deliver long-term flexibility in supporting immediate mode APIs resulting in significant reduction of risk to its user base over time.

If the functionality implemented in HOOPS matches the needs of your application, using HOOPS rather than building your own retained mode graphics system will save significant time and reduce their risk when developing 3D interactive graphics applications.