SOFTWARE—PRACTICE AND EXPERIENCE
Softw. Pract. Exper. 2000; **00**:1–7 *Prepared using* speauth.cls *[Version: 2002/09/23 v2.2]*

Comparing and Evaluating Computer Graphics and Visualization Software

Robert S. Laramee*,†

The Visual and Interactive Computing Group, Department of Computer Science, Swansea University, Swansea SA2 8PP, Wales, UK

SUMMARY

When starting a new computer graphics or visualization software project, students, researchers, and businesses alike must decide whether or not to start from scratch or with third party software. Since computer graphics and visualization applications are typically quite large, developers often build upon existing software libraries in order to take advantage of the tens of thousands of hours worth of development and testing already invested. Thus developers and managers must face the decision of which library to build on. We present a side-by-side comparison and evaluation of four popular, state of the art visualization and computer graphics libraries, namely the Visualization Toolkit, Open Inventor, Coin3D, and Hoops 3D. The evaluation is based on the feature set, ease of installation, development of a benchmark application, documentation, and technical support for each package. The results of our comparison and evaluation are described and recommendations are given as for whom the libraries are best suited. The Visualization Toolkit prevails on top in many of the aspects we compared and evaluated.

KEY WORDS: *Computer Graphics Software, Visualization Software, Graphics API, Visualization API*

1. Introduction

Students, researchers, and businesses often build on a pre-existing computer graphics or visualization software library when starting a new software project in these areas. The reasons may be three-fold: (1) no one wants to re-implement functionality that has already been implemented, *i.e.*, no one wants to reinvent the wheel, (2) building upon an existing software framework takes advantage of the already tens of thousands of hours worth of implementation and testing that have been invested into the library, and (3) building on a pre-existing,

*Correspondence to: The Visual and Interactive Computing Group, Department of Computer Science, Swansea University, Swansea SA2 8PP, Wales, U.K.
†E-mail: r.s.laramee@swansea.ac.uk
Contract/grant sponsor: ; contract/grant number:

Received 23 March 2007
Revised 6 July 2007

generic framework allows developers to concentrate their efforts on more specialized tasks, *e.g.*, their area of expertise. Plus, developers also have the benefit of inheriting common coding conventions rather than having to re-introduce a system themselves. However, there are several computer graphics and visualization libraries to choose from. Thus, everyone starting a new project is faced with the same difficult questions: Which software package should be chosen use to build upon and why?

In this project, we investigate, compare, and evaluate four popular, state-of-the-art computer graphics and visualization libraries:

1. The Visualization Toolkit (VTK) from Kitware Inc. [27]
2. Open Inventor (OI) from Mercury Computer Systems Inc. [35] This distribution of OI was formerly owned by Template Graphics Software (TGS) Inc. [52] TGS was acquired by Mercury Computer Systems circa May 2004.
3. Coin3D from Systems in Motion (SIM) [48]
4. Hoops 3D from Tech Soft America (TSA) [49]

We evaluate the libraries based on their set of features, ease of installation, the development of a small benchmark prototype, documentation, and support. We browsed thousands of pages of documentation, started dialogs with technical support and customer service, signed up on the mailing lists (where available), installed each library, and wrote a small benchmark application for each package that measures both performance and memory requirements. Here we present the findings of our evaluation based on this experience.

It is important to note that we made a breadth-for-depth trade-off. This is because we wanted to obtain some programming experience with each of the libraries –a task that becomes less and less feasible with the addition of each application. This research should help readers answer the question as to which software library is best suited for their particular application. To our knowledge, this is the first study of its kind that attempts to compare and evaluate four state of the art software libraries from the area of computer graphics and visualization. We note that this work can be seen as complementary to the work of Standard Performance Evaluation Corporation (`www.spec.org`).

2. Requirements

VRVis, a research center specializing in the fields of virtual reality and visualization, collaborates with, among others, a partner in the computational fluid dynamics (CFD) industry namely AVL [5]. AVL-AST, the software development department of AVL, requires platform independence. This is due to AVL's customers running AVL software on a variety of architectures. The evaluation was performed according to AVL's goal of writing CFD software applications. The one strict requirement of AVL is that whichever software library is chosen, it must be platform independent. Each of the four libraries we evaluated is platform independent. Table I lists the specific platforms that were explicitly mentioned as supported in each libraries documentation, or explicitly mentioned as supported by the respective company staff we corresponded with. It should be noted that because VTK and Coin3D are open source, they can all be compiled on a the target platform of choice. In the case of Hoops 3D, the obfuscated

	VTK	OIM	Coin 3D	Hoops 3D
Linux	+	+	+	+
BSD	+	-	+	-
Sun Sparc OS	+	-	+	+
Sun Solaris	-	+	+	+
IBM-AIX	+	+	+	+
HP-UX	+	+	+	+
Compaq/DEC Alpha	+	-	-	-
Compaq/DEC OSF1	+	+	-	-
Mac OSX	+	+	+	+
SGI-Irix	+	+	+	+
Windows	+	+	+	+

Table I. A side-by-side comparison of the platforms supported explicitly by the software libraries, as stated in their respective documentation. Note a plus (+) means supported, a minus (-) means not explicitly stated as supported according to the documentation.

source can also be compiled on a target platform. Thus it's likely that all four of these libraries can be adapted to compile on most, if not all, unix platforms.

3. Corporate Statistics

Since the lifetime of software development projects often span several years, a prospective user of third party software would theoretically like to know that their software library supplier will still be active for the foreseeable future. Thus information about a supplier's past and present development status might provide some insight. Some information comparing and summarizing such information for each corporation is given in Table II.

	VTK from Kitware	OI from Mercury	Coin3D from SIM	Hoops 3D from TSA
year founded	1998	1986	1994	1996
no. of employees	17	65	15	15
no. of developers	10	15	0-5	12
no. of apps	31	30	11	1
com. license cost	0	3,600 euro (one time)	1835 euro (annual)	$10,000 (annual)

Table II. A side-by-side comparison of general statistics about the corporations responsible for the software libraries.

Year company was founded: Mercury Systems Inc., the new owner of TGS, was founded in 1981. The year listed in Table II is the year TGS was founded. Note that we use OIM to

denote Open Inventor from Mercury throughout our work. SIM was founded in Trondheim, Norway, in cooperation with the Norwegian University of Science and Technology and Coin 1.0 was released in 2000. Hoops development started in the late 80's at Cornell University in Ithaca, New York. The technology was purchased by Autodesk in early 90's and subsequently spun-off to Tech Soft America in 1996. The core technology has been under development since 1989 with some of the initial Hoops developers still working for TSA. Kitware was founded in 1998.

Number of employees and developers: The four groups have approximately the same number of people involved in their respective projects with OIM having the largest group. VTK also has many contributers to the core implementation that do not work for Kitware. Eighteen outside developers contributed more than 1% of the code (per developer). This is one of the advantages of open source software.

Mercury has 15 Open Inventor developers plus a team for specific visualization projects. We report only on the TGS unit of Mercury and not the whole company which is much larger and responsible for many other products.

Coin3D does not have a fixed number of developers. Some periods are dedicated solely to bug fixing and technical support. Other periods may have up to five full-time developers adding new features. In other words, development on the core Coin technology is project-dependent.

Documented, derived applications: Before considering a third party library, prospective users and developers may be interested in prior successes. These are documented examples or case studies of clients that have used the library to build an application, a.k.a. '"success stories". By documented we mean featured on a web page or in a brochure. VTK has an impressive list of well documented, derived applications. OI from Mercury has both a "Customer spotlight" and a "testimonials" link on their web site however both links were broken at the time of this writing. The information we obtained comes from documentation sent to us directly from a sales representative [51]. For Hoops 3D we only found one documented success story.

Licensing costs: Our description of licensing costs is concerned with developer license costs. VTK has no annual licensing costs as long as VTK copyright notice remains in the source code.

Open Inventor from Mercury costs 3,600 euros for the development kit and is a one time cost only. Mercury also offers extension modules that can be added onto the basic Open Inventor library including: *DataViz*, *VolumeViz*, *HardCopy*, *TerrainViz*, *MultiPipe*, *FXViz*, and *SolidViz*. Each of these extensions modules is associated with additional license costs similar to the basic Open Inventor module. Mercury also has academic agreements with universities and educational institutions.

An annual license for Coin3D costs 1,835 euros and is free for non-commercial development. Coin3D also offers extensions to its basic library including: *Voleon*, *Scenery*, and *VRML View*. Each of these extensions requires additional licensing. For example, one annual SIM *Voleon* license, the volume rendering module for Coin3D, costs an additional 3,220 euros. Discounts on licenses are available of up to 20% for multiple licenses. SIM also offers a separate product, *Rational Reducer*. It is not an extension of Coin3D however.

The staff from TSA answered our inquiry about the number of employees and licensing costs with a telephone call. In addition to a licensing cost, TSA also charges between 1-4% royalty

on products sold that use Hoops 3D. TSA refers to this as a distribution cost. TSA also sells products that can be used in conjunction with Hoops 3D including: *Stream Toolkit*, *Net Server*, and *Parasolid*. However, unlike the OIM and Coin3D packages, these are not simply extensions to the basic graphics library, but separate products. Note that a Hoops 3D license does include a license to the Stream Toolkit while the Net Server component costs an additional $10,000 per year with no royalty charges. Parasolid is also a separate product with additional license fees.

Number of clients: We also inquired about the number of clients each company product has. By number of clients we mean the number of institutions, both academic (non-commercial) or research related and commercial institutions currently using the library to develop applications. For VTK and OIM, it's more difficult to provide this information because of their cost structure. However, they may be able to make some estimates based on the number of clients paying for support. Coin3D and Hoops 3D may make some estimates based on their annual licensing activity. Although we did obtain some quantitative information on the matter, some of the companies requested that this information not be distributed. We can estimate that each product has hundreds, if not thousands (in some cases), of clients.

4. Software Installation

In this section we discuss the details of the compilation and installation process for each software library, with a short description of the demo applications themselves. We start with a general description of the source files.

4.1. Software Source

Here we present some details about the software source files. The goal of this information is to get some idea of how the software packages compare with one another in terms of characteristics such as size and structure. Table III summarizes information about the software source. In all cases, the information in Table III refers to the basic software libraries for the Linux platform without extension modules or windowing toolkits.

Required disk space: This is only meant to give a rough idea of the relative sizes of the packages. The disk spare required is for the Linux installation in each case. For VTK, 312MB represents the complete unix source code files, object files, examples, sample data, documentation. For OIM, 442MB includes source code files, data files, object files, examples, and documentation. For Coin3D, 241MB includes source code files, object files, and documentation. We did not find any data sources included. For Hoops 3D, 697MB includes source code files, object files, documentation, and sample data.

Number of files: Again, this data is here in order to get an overview of the relative sizes of the packages. For each package this includes unix source files, object files, example applications, and data files. A prospective user may be inclined to believe that the more files a library contains, the more functionality they are getting. This is a dangerous assumption however.

Open source: Open source software is important for many developers to ensure quality. The general belief is that open source software is higher quality because it can be inspected by any number of other developers. Developers are allowed to review the source code and make modifications. Others may notice bugs or implement improvements. VTK and Coin3D

	VTK	OIM	Coin3D	Hoops 3D
version of software	4.2.3	4.0	2.3	11.0
required disk space (MB)	312	442	241	697
number of files	8,947	10,095	6,811	8,688
programming language	C++	C++	C++	C,C++
open source	+	-	+	-
source modification	+	+	+	-
source API documentation	+	+	+	+
structural overview of source	+	-	-	+
available as download	+	+	+	+
time required to compile (minutes)	≈ 10	≈ 15	≈ 20	≈ 20
time required to install (days)	< 1	> 2	1-2	> 3

Table III. A side-by-side comparison and evaluation of the software source itself.

are open source, plus outside developers may inspect the code quality directly for themselves. There is more than one distribution of Open Inventor. Open Inventor 2.1 from SGI is open source. Open Inventor from Mercury and its extensions are not open source. A more thorough description of the differences between the SGI and Mercury distributions of Open Inventor can be found in the documentation provided by Mercury [36]. Coin3D is open source and the *Voleon* module is also open source however the *Scenery*, *Rational Reducer*, and *VRML View* modules are not. Although source is available under special licenses. Hoops 3D is not open source.

Source modification: Closely related to the open source characteristic is the opportunity to modify the source code. Prospective users may or may not want the option of modifying the software they purchase. Modification to the core library source code is possible with VTK and the basic Open Inventor module from Coin3D. In general, source code modification is not possible with the extension modules offered by Mercury and Coin3D however with the exception of *Voleon* from Coin3D. Source modification to the core OIM and Hoops 3D libraries is possible on Linux, since the source must be compiled on the target Linux platform in the first place.

Source API documentation: For VTK, Open Inventor from Mercury, and Coin3D each class documented with Doxygen (www.doxygen.org) including inheritance and collaboration diagrams. Doxygen is an open source documentation system for C++, C, Java, Objective-C, and IDL [55]. The Hoops 3D API is also documented with Doxygen, however collaboration or inheritance diagrams are not generated.

Structural overview of source: We are also interested in seeing if some higher-level overview diagram of each library was illustrated. This can help the developer gain a better understanding of how the library classes are organized. VTK and Hoops 3D provide some higher level of abstraction diagrams. For OIM and Coin3D, we found no such higher-level overview diagrams other than what is already contained in the Open Inventor book [61].

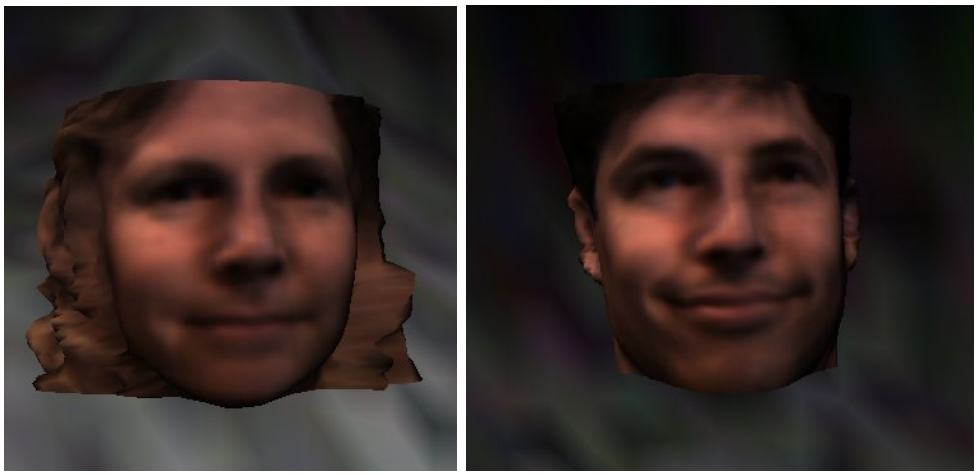

Figure 1. A sample morphing application supplied with OIM. (left) the female face is morphed into the (right) male face and vice-versa.

Available as download: All four libraries are available online for general purpose evaluation. Open Inventor from Mercury requires an additional license string which expires after 30 days.

Time required to find software: For each library, finding the software to download was easy. However, the Mercury web pages are very difficult to read because of the blue font color on darker blue background.

4.2. Compilation, Installation, and Demo Applications

In this section we discuss our hands-on experience with installing the libraries and running the demo programs provided by each distributor.

Time required to compile software: This is the amount of time taken to compile the base component(s) of each software library assuming the user already has already performed the required preparation. For all four packages the time required to compile the software is relatively short. VTK took only 10 minutes to compile on our Red Hat Linux 7.2 machine with a 1 GHz dual-processor Intel CPU and 1GB of RAM. Coin3D and Hoops 3D requiring a little longer in order to build in multiple directories.

Time to install software: This is the amount of time required we spent installing the software from the time it is downloaded. We had no prior experience installing these particular libraries, however we have had experience installing other libraries including Java 1.2/2.0 [3, 43], Java 3D [39] and VisAD [24] for an isosurfacing application [28]. Thus, this time also includes learning time and time to collect missing information. Other developers and administrators may be able to install the software faster, but for those installing the software for the very first time, we believe these times to be representative. We also believe that the relative amount of time taken to install the libraries is independent of the order in which

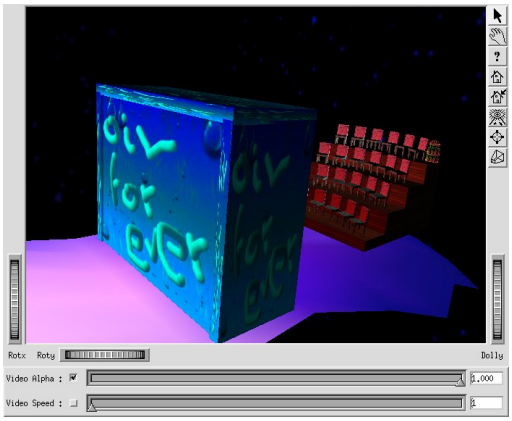

Figure 2. A well known cinema demo program supplied with OI from Mercury.

we worked on them. This is because the majority of the time was spent collecting missing information –an order-independent time investment.

The installation that was most straightforward in our experience was VTK. VTK required only a few hours including compilation, without language bindings. The installation procedure is fully documented in The VTK User's Guide [4]. The most time was spent learning how to use the CMake [33, 34] build utility from Kitware. CMake is a free tool whose goal is to provide a cross-platform build process.

OIM required a few days to install for us because the instructions are incomplete, thus we had to rely on an email exchange with technical support. The instructions are missing for the environment variable settings that are required in order for a successful build. Also, the installation directions are scattered rather than sorted in linear order. Response time from technical support varied from a few hours to more than 24 hours. The installation process is also complicated by the need for a unique license string. Our initial email requests for an extension on the trial license were not answered. However, we were able to obtain an extension on the trial license after phoning Mercury's customer service.

For Coin3D the installation directions are difficult to find, however once the instructions were found, the installation was fairly straightforward. The installation documentation for Coin3D is very terse and also incomplete, thus we had to rely on the Coin3D mailing list for a few missing details. Some important configure script options and environment variable settings are unclear from the installation instructions. However installation time was still not long because the responses from the mailing list came quickly, some after only five minutes.

We were unable to find any documentation describing Hoops 3D installation, no README, INSTALL, nor configuration scripts or files. We did find a makefile however. We tried compilation using the supplied makefile and ran into some errors. We worked with TSA technical support via email and telephone in order to to resolve problems with the installation. The installation process for Hoops 3D requires the most amount of time due the combination

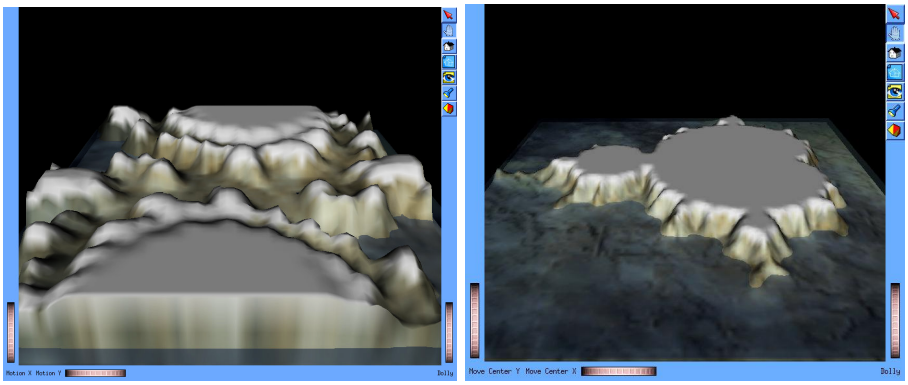

Figure 3. Two demo applications from Coin3D: (left) the landscape geometry is animated in time, (right) a fractal landscape, the complexity of the geometry adapts in complexity as the user zooms the viewpoint towards and away from the landscape.

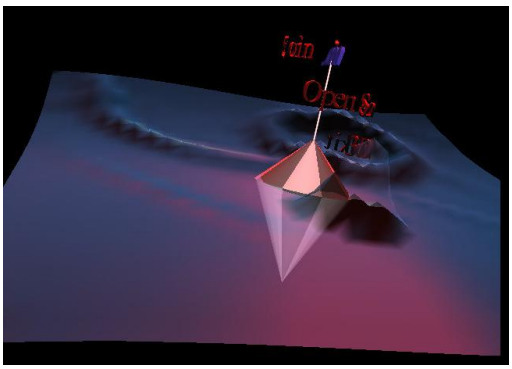

Figure 4. A sample demo program supplied with Coin3D. Buoyancy is simulated and animated in real time.

of lack of documentation and slow response time from technical support. It should be noted that for users in the U.S., the response time from technical support may be faster, since their technical support is located on the U.S. west coast and we were situated in Europe.

We also note that for each of these libraries, installation can be performed within a matter of an hour or two when the developer or administrator already has the required knowledge and experience. In fact we installed each library on two machines because we wanted to test our sample benchmark applications on more modern hardware, and the second time installing the libraries was much faster.

4.3. Discussion of Sample Programs and Data Sets

Here we make a distinction between the time taken to install the software libraries and the time required to run the demo programs. That's because in some cases, getting the demo programs running requires additional work like setting environment variables, acquiring data, additional downloads, additional windowing libraries, or additional compilation.

For VTK running the first sample program was relatively straightforward and required about two hours. The two hours was spent largely learning and configuring the CMake utility. After fumbling with the CMake utility for a few more hours, we were able to compile and run all the sample C++ programs. There are a wide variety of sample programs and sample data sets supplied with VTK. See Figures 5 and 6 for sample medical and information visualizations. Many of the sample programs require the Tcl/Tk [21, 60] language binding. However, in general, VTK Tcl/Tk programs can be converted to C++ without too much effort.

Some sample applications for OIM are shown in Figures 1 and 2. It took us some days of correspondence with technical support in order to find out that the sample demo programs using Motif require version 1.3 of the GLX library [2]. GLX is an OpenGL extension for the X windows system. The first machine we tried had GLX 1.2. Afterward, we ran into further time delay trying to obtain a working trial license string.

Coin3D also supplies an ample variety of sample programs. Obtaining and compiling the demo programs required CVS (Concurrent Versions System [53, 56]) access to a Coin3D server. Since we are behind a firewall that prevents CVS access, the demo programs took some extra time to acquire. The sample programs emphasize the computer graphics and interaction features of the library (see Figures 3 and 4). The drawback is that not very much visualization data is supplied (that we found).

Hoops 3D does not come with as many demo applications as the other libraries, however, the reference application they supply has many good features. Getting the reference application to compile and run took a few days. First, we had to install the Qt GUI library [9, 59] (`www.trolltech.com`). Then we had to make some small modifications to the source code in order to get the application compiling and running. The Hoops 3D demo application was the last one we had up and running, consistent with our experience up to this point.

	VTK	OIM	Coin3D	Hoops 3D
unstructured, 3D grids	+	+	-	-
data streaming	+	+	-	+
memory management	+	+	+	+
data caching	-	+	+	-
network interface	+	+	+	+
size of testing data (MB)	29	37	0	13

Table IV. A side-by-side summary of a subset of the data processing features in the respective libraries.

5. Data Processing Features

This section outlines features that are closely related to the data and not always tied directly to the computer graphics capabilities of the libraries. Nonetheless, the functionality described here may be very helpful to the goal of building a computer graphics or visualization application. Table IV provides a side-by-side summary and comparison of the data processing features.

Input and output file formats supported: VTK uses its own VTK data file format and in addition supports the following file formats: XML, STL-stereolithography, a file format commonly used by CAD applications, bitmap, DEM (Digital Elevation Model), DICOM (Digital Imaging and Communications in Medicine), GE Signal, JPEG, Binary UNC meta image data, PNG (Portable Network Graphics Specification), PNM (Portable aNY Map graphics), Postscript, SLC (Compiled SALT script, Telix), TIFF, RAW files, 3D Studio [31], AVS "UCD" format, Movie BYU (Brigham Young University), Renderman [47, 54], Open Inventor 2.0, Fluent GAMBIT ASCII, Unigraphics Facet Files, VRML 2.0, PLOT3D [26, 58], PLY (Autodesk Animator Polygon File Format), Ensight from CEI [7], and XYZ (RM2k XYZ Graphics Format).

Open Inventor from Mercury uses the proprietary Open Inventor format, VRML, MutltiGen Open Flight 3D model files, and X3D. The OIM *VolViz* module also supports reading the Amira Mesh [50], AVS field [38, 46], Segy, and Vol file formats for loading of volume data. The OIM *TerrainViz* module supports input and output in XML format, although the default file format for this module is proprietary. The OIM *TerrainViz* package also reads the GeoTiff and Tiff-tfw file formats. Others imports such as STL, CATIA [11], IGES, and STEP are available with the purchase of a separate licensed extension.

Coin3D supports the Open Inventor format, the DXF, MultiGen Open Flight 3D model files, and VRML. With the purchase of a *Rational Reducer* license, an extension software module offered by SIM, the 3D Studio [31] file format is also supported.

The base version of the Hoops 3D Part Viewer reads tessellated formats, including HSF (a proprietary Hoops Stream File), VRML, STL, OBJ (Alias Wavefront), and HMF (a proprietary Hoops Metafile format) files.

Unstructured, 3D grid support: Since we are evaluating the software for potential use in a CFD application, support for 3D, unstructured grids is important. VTK supports 3D, unstructured grids. In particular, VTK has support for tetrahedron, hexahedron, wedge, pyramid, quadric tetrahedron, and quadratic hexahedron volume cell types [44]. Open Inventor from Mercury supports 3D, unstructured grids and polar grids with a *DataViz* license, a separate licensed extension, in addition to an Open Inventor license. The DataViz module provides support for tetrahedron and hexahedron volume cell types [22]. Coin3D supports quad meshes, but not true 3D meshes, that is, meshes consisting of 3D cells. In Hoops 3D, all geometric entities are represented as flat, infinitely thin surfaces *i.e.*, there are no solid objects. In order to create an object that appears to be solid, one can define the collection of its surfaces.

Data streaming: CFD simulation data sets are usually very large. Therefore data streaming is also a desirable feature. VTK provides methods to stream data in pieces if an entire dataset cannot be held in memory. The *VolumeViz* extension model of OIM supports data streaming for extremely large data volumes. Coin3D does not support data streaming. Hoops

3D supports data streaming with the Hoops *Stream Toolkit* module included with the purchase of a Hoops 3D license. The Stream Toolkit is a set of platform and GUI-independent C++ classes that provide support for creating and reading Hoops Stream Files (HSF's) or data streams containing 2D & 3D scene-graph objects, attributes, and any kind of specialized application-specific data.

Memory management: We also looked into what memory management features each library had to offer since data sets resulting from CFD simulations are often very large. VTK, OIM, Coin3D, and Hoops 3D all support reference counting to avoid copying data (pg. 94 of Schroeder *et al.* [44].) Hoops 3D also supports what they call memory conservation. This instructs Hoops to save memory at the cost of rendering performance.

	VTK	OIM	Coin3D	Hoops 3D
multi-threading	+	+	+	+
parallel processing	+	-	-	
supported GUIs	X, W, T	X, Q, D	X, Q, W, M	X, Q, W
graphics libraries	OpenGL	OpenGL, Inventor	OpenGL, Inventor	OpenGL, Direct3D
language bindings	J, P, T	J	P, S	J, P, F
CAD kernel interface	-	-	-	+

Table V. A side-by-side comparison and summary of a subset of the application programming features contained within each package. supported GUI legend: X = Xt, Q = Qt, W = Win32, T = Tcl/Tk, M = Mac OSX. D = DialogViz language binding legend: J = Java, P = Python, T = Tcl/Tk, F = Fortran, S = Scheme.

Data caching: By data caching, we mean any software data structures or classes that support caching in addition to already existing hardware caching. All four libraries support OpenGL display lists which are one form of caching. Open Inventor from Mercury and Coin3D also support bounding box caching [61]. Bounding box caching is used to speed up picking.

Network interface features: Being able to share data over a network can prevent copying and storing huge data sets locally thus wasting resources. VTK supports VRML and Java. In addition, an application called Paraview (`http://www.paraview.org`) has been developed on top of VTK whose goal is the implementation of a scalable parallel processing tool with an emphasis on distributed memory architecture. Open Inventor from Mercury and Coin3D support VRML. In the case of Hoops 3D, the additional *Hoops/Net* package consists of a multi-platform server and client-side library that allows developers to implement message passing scheme between networked clients. The server provides session management control including late join support and session and user password control. The toolkit supports HTTP and TCP/IP protocols for internet or intranet use and can be customized to handle any kind of application-specific data as well as binary stream-capable data from the Hoops/Stream or BaseStream toolkits. The Hoops/Net library requires the purchase of an additional annual license ($10,000).

Size of data supplied for testing: We wanted to get an idea of the size of the data sets that each vendor provides for testing and whether or not any data was supplied for testing. In the case of OIM, the emphasis on their demo programs is rather on computer graphics and interaction widgets, thus we didn't find much in the way of data sets. Overall, the size of the

data sets we found were small in terms of the number of polygons contained in a sample mesh. That is why we introduced two larger data sets more typical of a mechanical engineer's work in CFD. Section 12 describes the data sets we used for testing in more detail.

When considering the data processing features of the four libraries, we note that VTK provides support for many different file formats and types. We interpret this as a sign that VTK is being used for many different applications in a wide variety of ways. We also note that their support for a wide variety of volume cell types for unstructured meshes is very useful in the case of CFD applications.

6. Application Programming Features

This section describes some features that can potentially impact or influence the whole application project including the design of the application and how it appears to the user. Some application programming features are summarized in Table V.

Multi-threading: Multi-threading can be very important for an application's performance. Often the user-GUI interaction is handled by a separate thread as well as file IO since file IO is relatively slow. Multi-threading for VTK is supported by a class called `vtkMultiThreader`. In Hoops 3D, various levels of multi-threading support may be used. Different levels enable Hoops 3D to be partially or fully reentrant (thread-safe). Specifically, two different threads could be traversing, querying, or modifying the Hoops geometry database.

Parallel processing: Parallel processing is very important for applications that required long processing times. For example, CFD simulations can require days or even weeks. VTK is the only library we saw with explicit support for parallel processing (via VTK distributed data filter).

Tested GUI libraries (*e.g.*, Motif, Qt, Mac OSX): Here we note which graphical user interface libraries are discussed in association with each graphics toolkit. In addition to Motif (Xt) [16, 17], Trolltech's Qt [18, 59] (`www.trolltech.com`), and Win32 [20, 23], OIM provides DialogViz–a C++ class library that extends Open Inventor by adding cross-platform GUI support [22]. Interestingly, we did not find any mention of the open source GUI library we currently use, namely, FOX (`www.fox-toolkit.org`). We use FOX for its platform independence and because it is open source.

Compatible graphics libraries, *e.g.*, OpenGL, DirectX: Each package uses OpenGL. Hoops 3D also supports Direct3D. Unlike OIM, which is build on the Open Inventor library originally from SGI [19, 61], Coin3D was written from scratch, guided by an older version of Open Inventor. Thus the code bases for Coin3D and Open Inventor from SGI are different.

Language bindings: Compiled C++ code wrapped with interpreted languages allows the developer to build efficient algorithms and retain the rapid code development features of interpreted languages by avoiding the compile/link cycle. VTK supports Tcl/Tk, Python, and Java bindings. Coin3D maintains a binding called *pivy*, for Python and another language binding called *Ivy* for Scheme. There is also a Java language binding for Coin3D v1.0. The SIM staff informed us that they plan on bringing this up-to-date in the coming months. Hoops 3D supports Fortran, Java, and Python.

CAD kernel interface, *e.g.*, to Parasolid, Acis, Granite: AVL-AST starts with CAD modeling data from partners, and use this to generate a true volumetric mesh before carrying

	VTK	OIM	Coin3D	Hoops 3D
transparency and texturing	+	+	+	+
shading	+	+	+	+
off-screen rendering	+	+	+	+
image processing	+	-	-	-
LoD	+	+	+	+
animation	-	+	-	+
picking	+	+	+	+
hidden surface removal algorithms	-	(+)	-	(+)

Table VI. A side-by-side summary and comparison of some of the rendering features within the respective packages. A feature in parenthesis () indicates the feature claims to be supported, but is not documented.

out a CFD simulation. It might be of interest to use a third party software library that can interface with CAD software, *i.e.*, a CAD kernel like ACIS [8].

VTK, OIM, and Coin3D do not support interfacing with a CAD kernel. This is one of the strengths of the Hoops 3D library. However, the staff at Kitware informed us that they have a work-in-progress project to be able to link the VTK to any third party CAD library. The Hoops 3D Part Viewer has three types which feature the Hoops 3D application framework's integration with the three leading modeling kernels: Parasolid, ACIS and Granite One [41]. Parasolid, also sold by TSA, also requires an annual update maintenance fee of $9,600.

7. Rendering Features

This section compares some basic features closely related to computer graphics and rendering. We discuss features such as texturing, off-screen rendering, level-of-detail (LoD) and picking. The feature set contained by the four libraries in our evaluation is summarized in Table VI.

Transparency and texture mapping: All four packages support transparency and texture mapping. OIM and Coin3D provide explicit support for 3D textures.

Shading options: In addition to the standard flat and Gouraud shading offered by OpenGL, Open Inventor from Mercury offers higher quality real-time shading supported by the *FxViz* module and supports programmable shaders such as those supported by the OpenGL ARB and Cg [32, 42]. Programmable shaders for Coin3D have already been developed and are planned for the next release according to the SIM staff we corresponded with.

Off-screen rendering: Off-screen rendering is useful for creating high-quality printouts that can be used for posters or presentations. Off-screen rendering with VTK requires downloading an additional Mesa library.

Open Inventor from Mercury supports SGI's RGB format, encapsulated postscript, TIFF (Tagged Image File Format), JPEG and PNG (Portable Network Graphics). Also, the *HardCopy* module can output the CGM (Computer Graphics Metafile), HPGL (Hewlett-Packard Graphics Language), and Postscript vector file formats.

Coin3D provides support for Postscript and RGB. The *simage* library from Coin3D also supports off-screen rendering for PNH, JPEG, TIFF, and GIF.

Hoops 3D supports Postscript, TIFF, CGM, WMF/EMF (Windows Metafile/Enhanced Windows Metafile), and HPGL off-screen formats. Hoops 3D can also interface with the ImageMagick library (`http://www.imagemagick.org`). ImageMagick is a collection of open-source tools and libraries to read, write, and manipulate an image in over 90 image formats.

Image processing: Only VTK supports image processing directly. In fact, The Visualization Toolkit [44] dedicates an entire chapter solely to this topic.

Support for level of detail (LoD): A level of detail (LoD) approach allows for faster frame rates by trading off rendering quality for speed. Each package supports some sort of LoD approach. All four libraries include explicit support for rendering at a target frame rate. With Coin3D, more advanced LoD features are included with the purchase of a *Rational Reducer* license. Rational Reducer costs 4,850 euros with 10-20% discounts for group purchases.

Animation features: Animation is useful for presentations, visual analysis as well as many computer graphics applications. VTK does not explicitly support animation, however, the related Paraview product does. Paraview is an application developed jointly by Kitware, Inc., Los Alamos National Laboratory, and Sandia Corporation. Open Inventor from Mercury supports MPEG generation. Many of the Coin3D demo programs use animation, however, we didn't find explicit documentation of these features. The Hoops *MVO* module provides a collection of classes that manage object behaviors. Most commonly those are defined as keyframe-based animations that manipulate an object's position or orientation over time.

Interaction and Picking: Picking is an important interaction technique in the case of CAD modeling and CFD analysis.

VTK supports multiple picking options. Depending on the type of picking class, the information returned from the pick may be as simple as an x,y,z global coordinate, or may include cell IDs, point IDs, cell parametric coordinates, the object that was picked, and/or assembly paths. Open Inventor from Mercury and Coin3D support basic picking. Picking capabilities are also demonstrated in the Coin3D demo programs.

Hoops 3D supports the most selection methods including picking based on:

- aperture–a point location. Aperture is defined by a radial distance from the selection point.
- area–a rectangular region
- polygon (lasso)–a polygonal region
- polyline (fence)–a series of points connected by line segments
- volume–a 3D object space volume
- object–a Hoops 3D shell primitive enables clash-detection/interference-checking between objects

Hoops 3D also supports a range of picking granularity within selected objects including: (1) vertex, (2) edge, (3) face, and (4) object selection. Hoops 3D will also compute the analytical point of intersection with the selected geometry in object, world and camera spaces.

Hidden surface removal algorithms: Also known as occlusion culling, we looked for *software* based algorithms that prevented occluded polygons from being sent to the graphics card, not just the standard back face culling supported by OpenGL. Open Inventor from Mercury claims to support an "extended" version of the painter's algorithm [10], object space hidden line removal, and image space hidden line and surface removal. However we were not

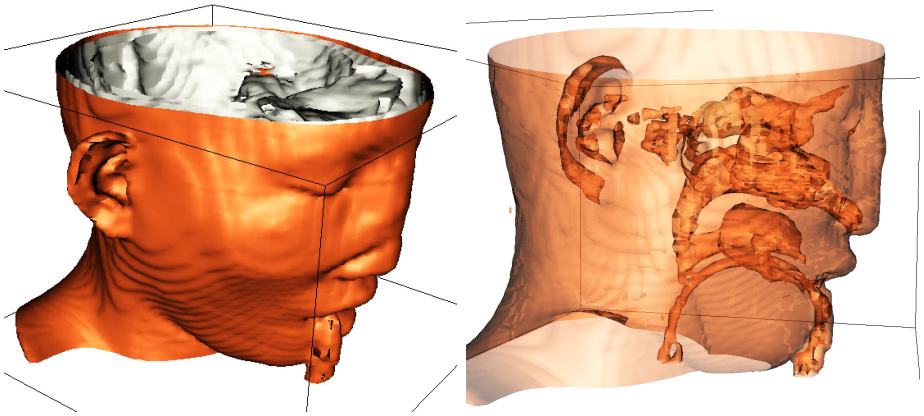

Figure 5. Two sample medical application visualizations from VTK (top) illustrating multiple isosurfaces (bottom) illustrating multiple isosurfaces and transparency.

able to find documentation to support this. Hoops 3D supports a software z-buffer algorithm. They also claim to support an extended painter's algorithm, a quick painter's algorithm, object-space hidden line removal, and image-based hidden line removal, however, we did not find descriptions of these in the documentation.

Looking back at the side-by-side summary and comparison of rendering features shown in Table VI, we can see that the four packages offer similar features in this category. That suggests that rendering features may not be an important deciding factor when trying to choose which library to employ.

8. Visualization Features

This section outlines a side-by-side comparison and evaluation of data visualization features contained within the libraries. The distinction between computer graphics and scientific visualization is not always 100% clear, thus some of the features here may fall into both categories. A selection of visualization features is shown in Table VII.

Color mapping: By color mapping we mean specifically the mapping between a range of scalar values and colors using a color bar-like object. All four packages support color mapping, however Coin3D supports color mapping (only) through the SIM Voleon volume rendering library.

Slicing: Each package supports some sort of slicing operation through a geometry. The slicing mechanism in Hoops 3D provides a very nice interaction widget for user interaction (Figure 8).

Interpolation: By interpolation we mean software interpolation between two arbitrary points in 3D space, a feature supported by all four libraries in some form. Open Inventor from Mercury supports interpolation along a pre-defined path.

Interpolation of orientation: We also looked to see if any of the packages supported interpolation of orientation via the use of quaternions [37, 45]. We did not find this advanced

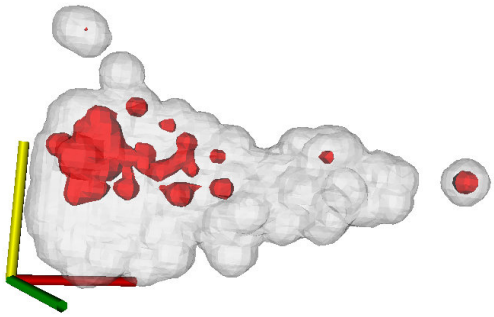

Figure 6. A sample application of information visualizations from the VTK. Some financial data is plotted in 3D. The gray, wireframe surface represents the total data population. The dark surface represents data points delinquent on loan payment.

	VTK	OIM	Coin3D	Hoops 3D
color mapping	+	+	+	+
slicing	+	+	+	+
interpolation	+	+	+	+
intersection detection	+	+	+	+
isosurfacing	+	+	-	-
flow visualization	+	+	-	-
volume rendering	+	+	+	-
time dependent data	-	-	-	-

Table VII. A side-by-side comparison and summary of selected visualization features contained in each library.

feature in VTK or Coin3D. Open Inventor from Mercury does support interpolation of position and orientation along a pre-defined path (via the use of a class `SoTrackFollower`). Hoops 3D explicitly supports interpolation orientation using quaternions. This functionality is used to offer a smooth transition between different 3D viewpoints.

Line and cell intersection: All four packages feature some collision intersection methodology. Coin3D includes a sample application demonstrating collision detection features. Hoops 3D supports intersection of surfaces.

Isosurfacing: Isosurfacing is a very valuable feature for the visual analysis of CFD simulation data. For OIM, isosurfacing is supported by the *DataViz* module, an additional package with a separate license fee. VTK also supports isosurfacing with the Marching Cubes algorithm [30]. Coin3D and Hoops 3D do not support isosurfacing but Hoops 3D does feature isolines.

Flow visualization features: CFD simulations generate vector field data, thus we are interested in visualization features that convey magnitude *and* direction. VTK features glyphs,

hedgehog plots, streamlines, and streampolygon generation. Open Inventor from Mercury supports streamlines, animated streamlines, streamsurfaces, all of which are part of the *DataViz* module. We saw no flow visualization features in Coin3D or Hoops 3D.

Volume rendering: Three of the libraries include volume rendering functionality. Open Inventor from Mercury requires the purchase of a *VolViz* license in addition to the normal Open Inventor license. Coin3D requires with the purchase of a SIM *Voleon* license: 3,220 euros (or $3,880 at the time of this writing) for a single license. VTK also supports volume rendering.

Time-dependent data: CFD simulations usually result in data sets that contain several time steps. Thus we also looked for any explicit support for time-dependent data. However, we did not see any features especially targeted at time-dependent data.

In the category of visualization features, VTK and OIM are the strongest. However, the case of OIM requires the purchase of an additional DataViz license. We consider this a disadvantage due the extra overhead resulting from multiple license strings, multiple releases, patches, and complexity.

9. Other Features

Here we outline some features that might be of interest but do not fit cleanly into the previous categories. Topics include stereo viewing and support for NURBS curves and surfaces.

Stereo viewing: Stereo viewing is standard in OpenGL 1.2 via the use of separate left and right frame buffers. In addition, Open Inventor from Mercury supports interleaved stereo, Anaglyph Stereo [57] and Half-Screen Stereo. With Anaglyph stereo a stereoscopic motion or still picture in which the right component of a composite image usually red, is superposed on the left component in a contrasting color like green. The goal is to produce a 3D effect when viewed through correspondingly colored filters in the form of spectacles. Coin3D supports interleaved and Anaglyph stereo.

Remote rendering: Virtual Network Computing (VNC, `www.uk.research.tt.com/vnc`), allows a server machine to perform all rendering while client machine displays resulting images. VTK also uses MPI [12, 15] (the Message Passing Library/Interface) and SNL's (Sandia National Labs, `http://www.sandia.gov`) ICE-T [40] for parallel rendering.

Support for NURBS (Non-Uniform Rational B-Spline) curves and surfaces: Support for NURBS curves and surfaces is part of the standard GLUT library (the OpenGL Utility Library) [63]. Open Inventor from Mercury also supports solid modeling operations such as merging, intersection, and subtraction with the purchase of a *SolidViz* license. Hoops 3D can be combined with Parasolid, a CAD modeler supplied by TSA. Parasolid is a separate product that requires an additional license.

10. Support

Different projects require different types of support. Typically students and researchers use mailing list support while businesses may be interested in more advanced support options. The support options offered by Kitware, Mercury, SIM, and TSA are compared and summarized side-by-side in Table VIII. We describe some of our experiences with support along with options.

	VTK	OIM	Coin3D	Hoops 3D
email	+	+	+	+
mailing list	+	-	+	-
telephone	+	+	-	+
training	+	+	-	+
web site	4	1	3	2

Table VIII. A side-by-side summary of the support options offered by Kitware, Mercury, SIM, and TSA. The subjective web site rating is on a scale from 1-5.

VTK offers three levels of support. The first level includes *The VTK User's Guide* [4], release notes, and software updates. Support is via e-mail and phone and includes installation assistance, bug fixes, limited development support, and one work day response time. Limited development support means that the staff will help a customer find the right VTK classes to use, and suggest approaches for an application. The cost is $1,500 annually. The second level of VTK support provides 24 hours of consulting labor to use as needed over the course of one year. This contract may be used for example to have Kitware extend existing VTK classes, develop new customized or higher-level classes, or to improve and extend proprietary VTK projects. The cost is $3,000 annually. The third level of support provides custom consulting services and training to help customers create products based on VTK. The price is negotiated.

Open Inventor from Mercury offers custom development, consulting, and training. The quote we received was for 800 euros annually and includes includes telephone support, software updates and bug fixes, new library versions, and documentation updates.

The Coin3D support model is email based and support requests are handled by the developers of Coin3D. It should be noted that both VTK and Coin3D mailing lists offer speedy, convenient support at no charge.

Tech Soft America, the sellers of Hoops 3D, offers three different levels of support: The first level provides support to commercial development efforts. First level support customers may provide feedback and influence TSA's product development process and future software releases. In certain cases, first level customers may receive custom maintenance releases with special features and modifications. The cost is $10,000 annually. The second level of support provides the base-level of support recommended for incorporating Hoops 3D technology into a new or existing application. Developer support includes direct email and telephone access to the Hoops 3D Consulting Engineering group, training services and when possible, special consideration during bug fixing periods. The cost is $30,000 per year. The third level of support offers custom software development for $175,000 annually. Note that there is a distinction between the information presented here, which was provided to us by customer service and the information that may be presented on the Hoops 3D web site (`http://www.hoops3D.com`). According to the representative we spoke to, the web site is being updated.

Training: Kitware offers on-site training for 1-3 day courses. Mercury offers similar three day courses in Houston Texas and Bordeaux, France. Hoops 3D offers 3-5 day, on-site courses in California.

Telephone: Systems in Motion is the only company that does not offer telephone support.

Mailing list: Mailing lists have the advantage that they bring together the knowledge and experience of a potentially large group of people together in a common forum. And since mailing lists usually have many subscribers, developers may avoid some temporal bottlenecks associated with time-zone differences or limited numbers technical support staff. VTK has 1,900 subscribers to its mailing list and the messages are stored in a fully searchable archive. Coin3D also has an active mailing list however, archives are not searchable. Open Inventor from Mercury and Hoops 3D do not have mailing lists however, there is a general Open Inventor newsgroup. We do note that Hoops 3D does provide a developer web site (`http://developer.hoops3d.com`) where developers may post specific questions. This web site is not nearly as active as VTK or Coin3D mailing lists however.

Email responsiveness: VTK has a very active electronic mailing list. We received speedy, helpful responses to all of our questions. Since Mercury has no newsgroup, we sent email to technical support. Responses came anywhere between 1-48 hours later. Coin3D also has a helpful mailing list with very fast response time. Sometimes responses took only five minutes. Hoops 3D has no mailing list. In our experience, responses from tech support always came within 24 hours. It should be noted that TSA also offered very helpful complementary telephone support during our evaluation.

Website, subjective rating: For completeness, we also give each companies respective web site an arbitrary rating from 1-5, 5 being the best rating. VTK has a good web site however finding and downloading the source code is not obvious. Open Inventor from Mercury has the worst web site of the four because (1) it features blue text on dark blue background, (2) is difficult to navigate, (3) information is not easy to find, *e.g.*, the year TGS was founded, and (4) it contains broken links on the front page. The Coin3D web site is good overall, but documentation is scattered. The Hoops 3D web site is not well organized and the API documentation is presented in a very small size font.

11. Documentation

Documentation is important to help programmers understand the source code. Relying solely on the source code to understand a library without accompanying documentation is unrealistic in our experience.

In terms of documentation, VTK is a clear winner. They offer the most complete and thorough set of documentation. Kitware offers a mature VTK textbook [44] in its third edition and a user's guide [4]. The user's guide gives a thorough description of the installation process as well as writing a first application.

For the general Open Inventor library, there are The Inventor Mentor [61], The Inventor Toolmaker [62], and The Open Inventor C++ Reference Manual [19]. For Open Inventor from Mercury the user manual is supplied as a free PDF file, 558 pages, available for download [22].

Coin3D and Hoops 3D documentation is mainly limited to the online API. No manuals or books have been written especially for Coin3D that we found and overall, the quality of the documentation is poor. Information is scattered throughout the SIM web site. Only an occasional brochure is available for download.

Hoops 3D is even missing README and INSTALL files for their software. We did order a Hoops 3D book [29] however it was out of print at the time of this writing. It may be possible to still get used copies of this book. TSA sent us a complimentary copy including their software

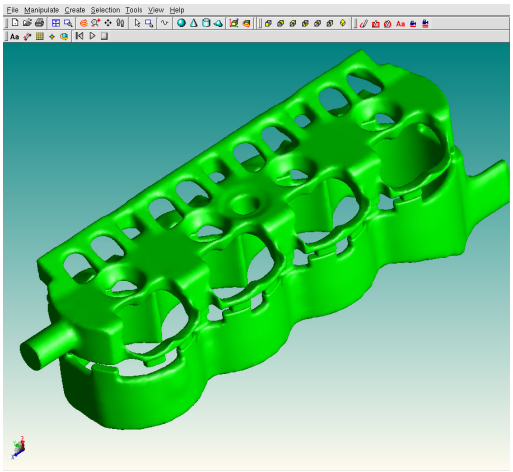

Figure 7. One of our benchmark data sets shown in an a Hoops 3D application. The cooling jacket shown here is a complex, unstructured, adaptive resolution mesh composed of 228,000 polygons.

on CD. However, this documentation is rather outdated. We note that Hoops 3D does offer technical articles from their developer web site (`http://developer.hoops3d.com`).

12. Writing a Benchmark Application

As part of our evaluation we wanted to compare writing a small sample application with each package. Thus for each library we undertook writing an application to do the following:

1. load an unstructured, triangulated mesh
2. rotate the mesh automatically
3. measure the frames per second for rotating the input mesh
4. output the result using off-screen rendering.

Furthermore, we wanted to use the same data sets across each sample application. For our benchmark data sets, we chose two larger unstructured, adaptive resolution, triangulated meshes not typically found with sample programs, but which are typical to the daily work of mechanical engineers using CFD software, a cooling jacket and a terrain data set. The cooling jacket data set is composed of 228,000 unstructured, adaptive resolution polygons and is shown in Figure 7 within a Hoops 3D application. The terrain data set is composed of 372,000 unstructured polygons (see Figure 9).

To our surprise, writing the benchmark application was generally easier than installing the software and getting the demo programs running. For each library we tried to find a demo program that we could start with and modify in order to suit our goal outlined above.

For VTK, we started with a sample Tcl/Tk script (`CADPart.tcl`), converted it to C++, and added the frames-per-second performance measure. We already had our sample data in STL format, which VTK supports, so loading the data was trivial. VTK rotated the cooling jacket

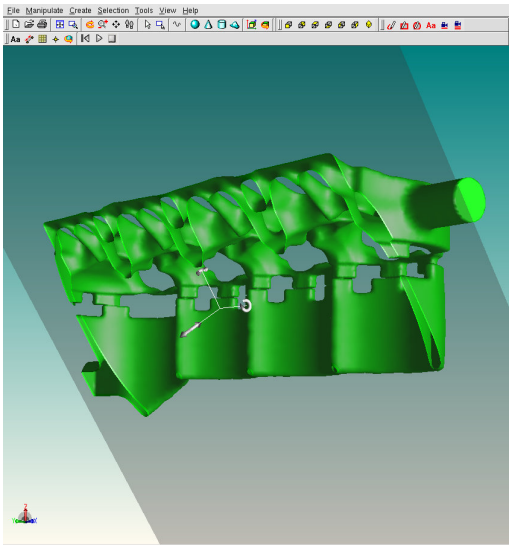

Figure 8. The Hoops 3D cutting plane and associated widget slicing through the cooling jacket geometry.

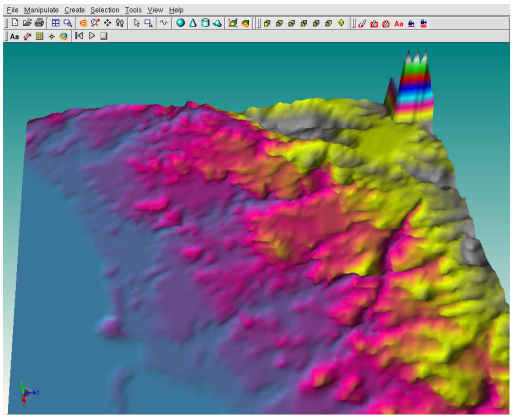

Figure 9. A terrain data set similar to the one we used in our benchmark application.

data set at 35 fps, and VTK has built-in support for LoD, thus an arbitrary frame rate can be set by the user. We also tested this feature with success. One problem however, is the off-screen rendering which requires downloading and installing an additional Mesa library.

Our OIM benchmark application is a modified version of the Large Model Viewer (`LargeModelViewer.cxx`) demo program provided by TGS. This demo application will load meshes in Open Inventor format and provide the rendering performance in frames per second.

	cooling jacket 228,000 polygons	terrain 372,000 polygons
VTK	35	19
OIM	33	20
Coin3D	30	19
Hoops 3D	35	3.2

Table IX. The frame-rate of each benchmark application rotating the cooling jacket and terrain data sets, in frames per second (fps).

We converted our data to Open Inventor format using a tool
called `ivconvert` [6]. All we had to do was add the off-screen rendering by using the `SoOffScreenRenderer` class. For the implementation of the off-screen rendering, we looked at the source code of Example 9–1 (Chapter 9) of The Inventor Mentor [61]. One very nice feature of the large model viewer demo application was the ability to generate triangle strips. After generating triangle strips, the rendering performance increased to 80 and 50 fps for the cooling jacket and terrain data sets respectively. We note that triangle strip generation should also be possible in Coin3D since it is supported by the basic Open Inventor library via the `SoTriangleStripSet` class.

For Coin3D we also started with a demo C++ application (`multifileviewer.cpp`) and adapted that accordingly. Coin3D does not support STL format. We
used `ivconvert` [6] to automatically convert STL files to Open Inventor format. Adding the performance benchmark is trivial and adding off-screen rendering was also rather straightforward. The online API documentation was very helpful for this task.

The sample application supplied by Hoops 3D already fulfills our needs, so no further programming was necessary after the installation was complete. The reference application loads STL files, measures performance times automatically, and outputs off-screen prints. We did encounter some bugs with the off-screen rendering of Postscript files on Linux however, and the level of detail features in the application did not work with our data sets, only the data sets supplied by TSA.

Table IX compares the performance times of our benchmark applications written using each library. The benchmark applications were run on an *HP Workstation xm6000* with 2.8 GHz dual-processor running Red Hat Linux and an NVIDIA Quadro4 980 graphics card. In general, each package performaned similarly. We urge caution when interpreting the results of Table IX however. Any number of measures can be taken to improve rendering performance like using triangle strip generation, display lists, occlusion culling and LoD techniques. What we attempted was likely a worst-case scenario with the goal of being objective and as unbiased as possible. Each library supports LoD with target rendering frame rates, so performance alone may not be the most important aspect when considering which library to choose.

We also note that we ran into a problem when running our benchmark Hoops 3D application using the terrain data set on our performance evaluation workstation. Hoops 3D offers two graphics modes of operation: an OpenGL mode and an X11 mode. We had to perform the evaluation of the terrain data set in the slower X11 mode because the application crashed in

	stand alone	cooling jacket data set	terrain data set
VTK	0.4%	2.5%	3.8%
OIM			
Coin3D	0.8%	4.5%	7.4%
Hoops 3D	1.5%	4.6%	6.6%

Table X. The memory usage of each benchmark application stand-alone and with our two test data sets. Units are expressed in terms of percentage of main memory occupied by the application. The workstation we measured this on has 1GB of memory.

OpenGL mode. This was not a problem with our initial older workstation we used for the initial installation however. We supplied the terrain data set to TSA and they were able to re-create the problem. They offered to fix the crash via a patch or include a fix in the next release (version 12) depending on the urgency of the problem.

As part of evaluation, we not only looked at rendering performance, but also memory occupancy. We did this by noting the amount of memory taken by each respective benchmark application after loading the cooling jacket and terrain data sets. The results are shown in Table X and are expressed in terms of the percentage of main memory (with one gigabyte total) occupied by the application when rendering the benchmark data sets.

Unfortunately, we were not able to run this test for the OIM benchmark application. This is because our Open Inventor license string expired. Mercury expects that the trial period lasts from two to four weeks. Mercury refused our request for another license string. In essence, our trial period had passed and was over by this time from the viewpoint of Mercury Systems. This is likely related to the fact that we did a longer, more thorough evaluation than most customers, thus was required more time than average. An evaluation of this nature is unusual.

In our tests, VTK occupies the smallest amount of memory at run time. Again, we advise the use of caution when interpreting the results. This is because any application can be tailored to be more memory efficient. Each benchmark application has a different feature set, influencing the results. That is why we have listed the percentage of memory occupied by each benchmark application without data separately. Plus, data storage formats could have an influence on these results. For VTK and Hoops 3D, the cooling jacket and terrain data sets were stored in STL format, for OIM and Coin3D in Inventor format.

13. Conclusions

First, we note that VTK and Coin3D are more educational and research oriented than Open Inventor from Mercury and Hoops 3D. Open Inventor from Mercury and Hoops 3D are more targeted at businesses rather than researchers, universities, and individuals. For visualization purposes, we recommend VTK since it has the strongest list of visualization features (Table VII). Open Inventor from Mercury has an equally full set of features but requires the purchase of an additional *DataViz* license. We also recommend the VTK for those users that place a high importance on documentation since the VTK has arguably the best of the group.

For businesses interested in writing computer graphics applications, Open Inventor from Mercury may be a good choice. Mercury supplies the greatest selection of demo applications related to computer graphics. However, customer service and support is the weakest for Mercury in our experience. Obtaining information from Mercury customer support was the most difficult of the four companies in our experience, although obtaining information from Mercury technical support was not a problem. We speculate that this could be related to the transition process TGS was going through after being acquired by Mercury.

For students and researchers interested in building pure computer graphics applications, we recommend Coin3D. Coin3D is open source and free for educational and research purposes. Open source software is also valuable for developers interested in furthering their software engineering skills. Plus we found the mailing list to be a great source of helpful information.

For businesses wishing to write software closely related to CAD modeling, we recommend Hoops 3D. The Hoops 3D library contains many useful features related to computer aided design and is the only library that can interface with CAD kernels (Section 6). On the other hand, the Hoops 3D library is missing the true 3D geometric cell types and data structures to carry on a CFD simulation after the modeling phase is completed. Support for unstructured 3D meshes is featured in the VTK (Section 5). Hoops 3D has very strong customer and technical support. The Hoops 3D support team was by far the most proactive. The drawbacks with Hoops 3D are the documentation and the cost. The pricing is the highest of the libraries we evaluated.

14. Future Prospects

One interesting future prospect is the possibility of combining the libraries such that prospective developers may have the chance of inheriting the benefits of more than one library. During the course of our work, we were informed that Kitware and Tech Soft America were in talks over the possibility of building an interface between their two systems. The ultimate outcome of this project is still uncertain. Such a task would be easier if the computer graphics and visualization communities tried to develop standard APIs for their software, in the same spirit as OpenGL.

Future work here could take on many different directions. For example, there are several other computer graphics and visualization libraries that could be added to the evaluation including VisAD [24], OpenVIZ from Advanced Visual Systems [1], Iris Explorer [13, 14], the Insight Segmentation and Registration Toolkit (ITK) [25, 64], and/or Amira [50]. All the libraries here could be evaluated in more depth or with a more specific focus.

15. Acknowledgments

We would like to thank all those who have contributed to financing this research, including AVL (`www.avl.com`) and the Austrian national research program Kplus (`www.kplus.at`). We thank Pete Sampl and Jürgen Krasser of AVL for valuable discussions and feedback. We thank Mathieu Malaterre of Kitware for providing very valuable feedback on this work and the rest of the VTK mailing list for their support both technical and non-technical. We thank Harold Stöckl and Jean-Luc Garnier of TGS, Mercury for their support and for reading this paper and providing feedback. We thank Leopold Palomo-Avellaneda and Volker Enderlein of

the Coin3D mailing list for their help and Lars J. Aas of SIM for his very valuable feedback and proofreading. We thank Yanick Fluhmann, Guido Hoffmann, and Gavin Bridgeman of TSA for their valuable help and for looking at the paper. Gavin Bridgemen gave us very helpful feedback on this research as well.

REFERENCES

1. Advanced Visual Systems Inc., 300 Fifth Avenue, Waltham, MA 02451. *OpenVIZ*. http://www.avs.com [September 2007].
2. OpenGL Architecture Review Board. *OpenGL Reference Manual: The Official Reference Document to OpenGL, Version 1.2*. Addison Wesley, 2000. D. Schreiner, editor.
3. K. Arnold, J. Gosling, and D. Holmes. *The Java Programming Language Third Edition*. Addison-Wesley, Reading, MA, 3 edition, 2000.
4. L. S. Avila, S. Barre, B. Geveci, A. Henderson, W. A. Hoffman, B. King, C. C. Law, K. M. Martin, and W. J. Schroeder. *The VTK User's Guide (VTK 4.2)*. Kitware, Inc., 2003.
5. AVL LIST GMBH, Hans-List-Platz 1, 8020 Graz, Austria. *AVL-AST (Advanced Simulation Technologies)*. http://www.avl.com [September 2007].
6. J. Burkardt. *IVCON–3D Graphics File Conversion*. Pittsburgh Supercomputing Center (PSC), Carnegie Mellon University, University of Pittsburgh, Pennsylvania, July 2000. http://www.psc.edu/ burkardt/src/ivcon/ivcon.html [September 2004].
7. Computational Engineering International (CEI), 2166 N. Salem St., Suite 101 Apex, North Carolina 27523-6456. *Ensight*. http://www.ceintl.com/ [September 2007].
8. J. Corney and T. Lim. *3D Modeling with ACIS*. Saxe-Coburg Publications, UK, 2 edition, 2001.
9. M. K. Dalheimer. *Programming with Qt: Writing Portable GUI Applications on UNIX and Win32*. O'Reilly & Associates, Inc., 981 Chestnut Street, Newton, MA 02164, 1999.
10. J. D. Foley, A. Van Dam, and S. K. Feiner. *Introduction to Computer Graphics*. Addison-Wesley, August 1993.
11. P. Forestier. The CATIA Integrated Geometry Modeler. In *Proceedings of the International Conference on Computer Graphics '85*, pages 381–391. Online Publications, Pinner, Middx., England, 1985.
12. I. Foster, D. R. Kohr, R. Krishnaiyer, and A. Choudhary. A Library-based Approach to Task Parallelism in a Data-Parallel Language. *Journal of Parallel and Distributed Computing*, 45(2):148–158, September 1997.
13. D. Foulser. IRIS Explorer: A Framework for Investigation. *Computer Graphics*, 29(2):13–16, May 1995.
14. C. Goodyer and M. Berzins. Eclipse and Ellipse: PSEs for EHL Solutions Using IRIS Explorer and SCIRun. In P. M. A. Sloot, C. J. K. Tan, J. Dongarra, and A. G. Hoekstra, editors, *International Conference on Computational Science (ICCS), Part I*, volume 2329 of *Lecture Notes in Computer Science*, pages 523–532. Springer-Verlag, April 21–24 2002.
15. S. Gorlatch and H. Bischof. A Generic MPI Implementation for a Data-Parallel Skeleton: Formal Derivation and Application to FFT. *Parallel Processing Letters*, 8(4):447–458, December 1998.
16. K. Gottheil, H. J. Kaufmann, T. Kern, and Z. Zhao. *X und Motif*. Springer, Berlin, 1992.
17. K. D. Gregory. *Programming with Motif*. Springer-Verlag, Berlin, Heidelberg, London, 1992.
18. A. Griffith. *KDE/QT Programming Bible*. IDG Books Worldwide, Indianapolis, IN, 2001.
19. Open Inventor Architecture Group. *Open Inventor C++ Reference Manual: The Official Reference Document for Open Systems*. Addison-Wesley, Reading, MA, 1994.
20. P. Handsman. Porting to the Win32 API. *Dr. Dobb's Journal of Software Tools*, 18(1):74, 76–78, January 1993.
21. M. Harrison and M. McLennan. *Effective Tcl/Tk programming: writing better programs with Tcl and Tk*. Addison-Wesley, Reading, MA, 1998.
22. M. Heck, T. Dufour, R. Albou, M. Alcala, F. Arnaud, P. Barthélemy, R. Berthon, N. Daguise, P. Estrade, O. Fedkiw, S. Gateuil, J.M. Gadinaud, J. Hummel, F. Jautard, C. Ornier, and K. Tinoco. *Open Inventor from TGS, Users Guide*. TGS unit of Mercury Inc., Chelmsford, Massachusetts, 2003. http://www.tgs.com [September 2007].
23. M. Heller. *Advanced Win32 Programming*. Wiley, New York, NY, 1993.
24. W. Hibbard. Connecting People to Computations and People to People. *Computer Graphics*, 32(3):10–12, 1998.

25. L. Ibanez, W. J. Schroeder, L. Ng, and J. Cates. *The ITK Softwre Guide: The Insight Segmentation and Registration Toolkit (version 1.4)*. Kitware, Inc., 3rd edition, 2003.
26. G. A. Keramidas, E. W. Miner, and W. Bauman. PLOT3D, An Interactive Graphics Code for Three Dimensional Plots. *Microsoftware Eng. (GB)*, 1(1):49–64, July 1985.
27. Kitware Inc., 28 Corporate Drive, Suite 204, Clifton Park, New York 12065. *The Visualization Toolkit*. http://www.kitware.com [September 2007].
28. R. S. Laramee and R. D. Bergeron. An Isosurface Continuity Algorithm for Super Adaptive Resolution Data. In *Advances in Modelling, Animation, and Rendering: Computer Graphics International (CGI 2002)*, pages 215–237. Computer Graphics Society, Springer, July 1-5 2002.
29. W. Leler and J. Merry. *3D with Hoops*. Autodesk Inc. Addison-Wesley, Reading, MA, 1996.
30. W. E. Lorensen and H. E. Cline. Marching Cubes: a High Resolution 3D Surface Construction Algorithm. In *Computer Graphics (Proceedings of ACM SIGGRAPH 87, Anaheim, CA)*, pages 163–170. ACM, July 27–31 1987.
31. G. Loveria. Low-Cost 3-D Animation Materializes for PC Users: 3D Studio brings affordable animation to the PC. *Byte Magazine*, 16(4):259–262, April 1991.
32. W. R. Mark, R. S. Glanville, K. Akeley, and M. J. Kilgard. Cg: A System for Programming Graphics Hardware in a C-like Language. In *Proceedings of ACM SIGGRAPH 2003*, volume 22(3) of *ACM Transactions on Graphics*, pages 896–907, 2003.
33. K. Martin. The CMake Build Manager. *Dr. Dobb's Journal of Software Tools*, 28(1):40, 42, 44, 46, January 2003.
34. K. Martin and B. Hoffman. *Mastering CMake, Version 2.0*. Kitware Inc., 2004.
35. Mercury Computer Systems Inc., Chelmsford, Massachusetts. *Open Inventor from Mercury*. http://www.mc.com [September 2007].
36. Mercury Inc., Chelmsford, Massachusetts. *Open Inventor from Mercury Release 5.0 Versus SGI Open Inventor 2.1 , Technical White Paper*, april 2003 (upgrade sept 2004) edition, September 2004. http://www.tgs.com/pro_div/oiv_vs_sgi_whitepaper.htm [September 2004].
37. T. Möller and E. Haines. *Real-Time Rendering*. A. K. Peters Limited, 2 edition, 2002.
38. H. G. Pagendarm and S. I. Choudhry. Visualization of Hypersonic Flows - Exploring the Opportunities to Extend AVS. In *The 4th EUROGRAPHICS Workshop on Visualization in Scientific Computing (Abingdon, UK)*, 1993.
39. I. Palmer. *Essential Java 3D Fast*. Essential. Springer_Verlag, April 2001.
40. Allen Partridge. *Internet Collaboration and Exchange for Theatre (ICE-T):–Developiong Interactive Network Conferencing Software for Theatrical Collaboration*. PhD thesis, Texas Tech University, 2000.
41. C. Potter. The Engines of Mechanical CAD — As modeling kernels such as ACIS, Parasolid, and DesignBase continue their battle for market share, the debate rages. *Computer Graphics World*, 20(5):31–38, May 1997.
42. J. Ratcliff. The Cg Programming Language. *Dr. Dobb's Journal of Software Tools*, 27(10):40, 42, 44, October 2002.
43. J. Sanchez and M. P. Canton. *Java Programming for Engineers*. CRC Press, 2000 N.W. Corporate Blvd., Boca Raton, FL 33431, 2002.
44. W. J. Schroeder, K. M. Martin, and W. E. Lorensen. *The Visualization Toolkit, An Object-Oriented Approach to 3D Graphics*. Kitware, Inc., 3rd edition, 2003.
45. K. Shoemake. Animating Rotations with Quaternion Curves. *Computer Graphics (SIGGRAPH '85 Proceedings, San Francisco, CA)*, 19(3):245–254, July 1985.
46. C. S. Siegel. Creating 3D Models from Medical Images Using AVS: A User's Perspective. *Computer Graphics*, 29(2):59–60, May 1995.
47. P. Slusallek, T. Pflaum, and H.-P. Seidel. Implementing RenderMan - Practice, Problems, and Enhancements. *Computer Graphics Forum (Eurographics '94, Oslo, Norway)*, 3:443–454, 1994.
48. Systems in Motion (SIM) AS, Bygdøy allé 5, N-0257 Oslo, Norway. *Coin3D*. http://www.sim.no [September 2007].
49. Tech Soft America (TSA), 1830 Embarcadero Suite 103, Oakland, CA 94606. *Hoops 3D*. http://www.hoops3D.com [September 2007].
50. TGS Inc. (Template Graphics Software), 5330 Carroll Canyon Road, Suite 201, San Diego, California 92121-3758. *Amira 3.1, User's Guide and Reference Manual*. http://www.amiravis.com [September 2007].
51. TGS Inc. (Template Graphics Software), 5330 Carroll Canyon Road, Suite 201, San Diego, California 92121-3758. *Customer Spotlights, Brochure*. http://www.tgs.com [September 2007].

52. TGS Inc. (Template Graphics Software), 5330 Carroll Canyon Road, Suite 201, San Diego, California 92121-3758. *Open Inventor from TGS*. http://www.tgs.com [September 2007].
53. D. Thomas and A. Hunt. *Pragmatic Version Control Using CVS*. The Pragmatic Programmers, Lewisville, TX, 1 edition, sep 2003.
54. S. Upstill. *The Renderman Companion*. Addison-Wesley, Reading, MA, 1989.
55. Dimitri van Heesch. *Doxygen, Manual for version 1.3.9.1*. The Netherlands, 197-2004.
56. J. Vesperman. *Essential CVS*. O'Reilly, 1 edition, jun 2003.
57. S. Volbracht, K. Shahrbabaki, G. Domik, and G. Fels. Perspective Viewing, Anaglyph Stereo or Shufter Glass Stereo? In *Proceedings of the IEEE Symposium on Visual Languages*, pages 192–195, Washington, September 3–6 1996. IEEE Computer Society Press.
58. P.P. Walatka, P.G. Buning, L. Pierce, and P.A. Elson. *PLOT3D User's Manual*. NASA, mar 1990. http://www.openchannelfoundation.org/ [September 2007].
59. P. Ward. *Qt Programming for Linux and Windows 2000*. Hewlett-Packard professional books. Prentice-Hall PTR, Upper Saddle River, NJ 07458, 2001.
60. B. B. Welch. *Practical Programming in Tcl and Tk*. Prentice-Hall, Upper Saddle River, NJ 07458, 1995.
61. J. Wernecke. *The Inventor Mentor, Programming Object-Oriented 3D Graphics with Open Inventor, Release 2*. Addison-Wesley, March 1994.
62. J. Wernecke. *The Inventor Toolmaker, Extending Open Invetor, Release 2*. Addison Wesley, April 1994.
63. M. Woo, J. Neider, T. Davis, and D. Shreiner. *OpenGL Programming Guide, The Official Guide to Learning OpenGL, Version 1.2*. Addison Wesley, 3 edition, 1999.
64. T. S. Yoo. *Insight into Images, Principles and Practice for Segmentation, Registration, and Image Analysis*. A. K. Peters, Wellesley, MA, May 2004.