V. THREE-DIMENSIONAL REAL-TIME COMPUTER GRAPHICS

A. INTRODUCTION

This chapter describes the principal characteristics needed for the creation of object-oriented graphics viewers for visualizing a large-scale virtual world. Open standards, portability and versatility are emphasized over platform-specific performance considerations in order to support scaling up to very large numbers of users, platform types and information sources. The *Open Inventor* toolkit and scene description language has all of the functionality needed, and it is described briefly. The potential integration of network connections to logically extend graphics programs is examined in detail.

B. DESIRED CHARACTERISTICS OF GRAPHICS VIEWER PROGRAMS

A good graphics toolkit for building a virtual world viewer has many requirements to fill. Rendered scenes need to be realistic, rapidly rendered, permit user interaction, and capable of running on both low end and high end workstations. Graphics programmers must have a wide range of tools to permit interactive experimentation and scientific visualization of real world datasets (Nielsen 90) (Thalmann 90). The ability to read multiple data formats is also important when using scientific and oceanographic datasets. Scientific data format compatibility can be provided by a number of data function libraries which are open, portable, reasonably well standardized and usually independent of graphics tools (Fortner 92) (Rhyne 93b). Viewer programs need to be capable of examining high-bandwidth information streams and large archived scientific databases. Thus the ability to preprocess massive datasets into useful, storable, retrievable graphics objects will be particularly important as we attempt to scale up to meet the sophistication and detail of the real world. Adequate standardization of computer graphics and portability across other platforms is also desirable but has been historically elusive.

C. *Open Inventor*

Open Inventor is an object-oriented 3D graphics toolkit for graphics applications design (Strauss 92) (Wernecke 94a). Based on the *Open GL* graphics library, *Open Inventor* provides high-level extensions to the ***C***++ (or ***C***) programming language and a scene description language. It is designed to permit graphics programmers to focus on what to draw rather that how to draw it, creating scene objects that are collected in a scene database for viewpoint-independent rendering. User-triggered events are an integral part of the graphics rendering engine in order to permit rapid interactivity. A flexible design enables programmers to employ a variety of object representations and interaction modes. Object-oriented functionality allows users to customize and extend toolkit functionality through creation of new classes, subclassing and inheritance (Wernecke 94b).

The graphics capabilities of *Open Inventor* are extensive, including most (if not all) of the functionality described in canonical computer graphics reference (Foley, van Dam 90), as well as hooks to X-Windows and Motif-compliant window functions. *Open Inventor* is well suited to build graphics viewers for interactive real-time virtual worlds. It has been used to produce the graphics viewer for the NPS AUV underwater virtual world (Brutzman 94e). Particularly important and useful capabilities of *Open Inventor* are examined in the following paragraphs.

1. Scene Description Language

The ability to store graphics objects as readable, editable files is especially appealing for the creation of large-scale virtual worlds. Since the performance of computer graphics is highly dependent on the computational complexity of scenes to be rendered, it is inevitable that truly large-scale world scene databases will eventually overload viewing graphics workstations. Such overload will occur regardless of the efficiency of viewpoint culling algorithms and graphics pipeline optimizations, unless partitionable and networked scene databases are used. Furthermore, since populating a virtual world is a task that needs to be open and accessible to large numbers of people, an open graphics data standard is needed for virtual world construction. The ability to

selectively load graphics objects and scenes from files is also an important distribution mechanism which can take advantage of World-Wide Web connectivity. Thus use of *Open Inventor* scene description files permits individual workstations to act as graphics file servers, and also allows a large variety of viewers to examine individual graphics objects.

Elements in a scene graph can also be represented by icons pertaining to node type for easy reference. An abbreviated scene graph for the NPS AUV graphics object file appears in Figure 5.1.

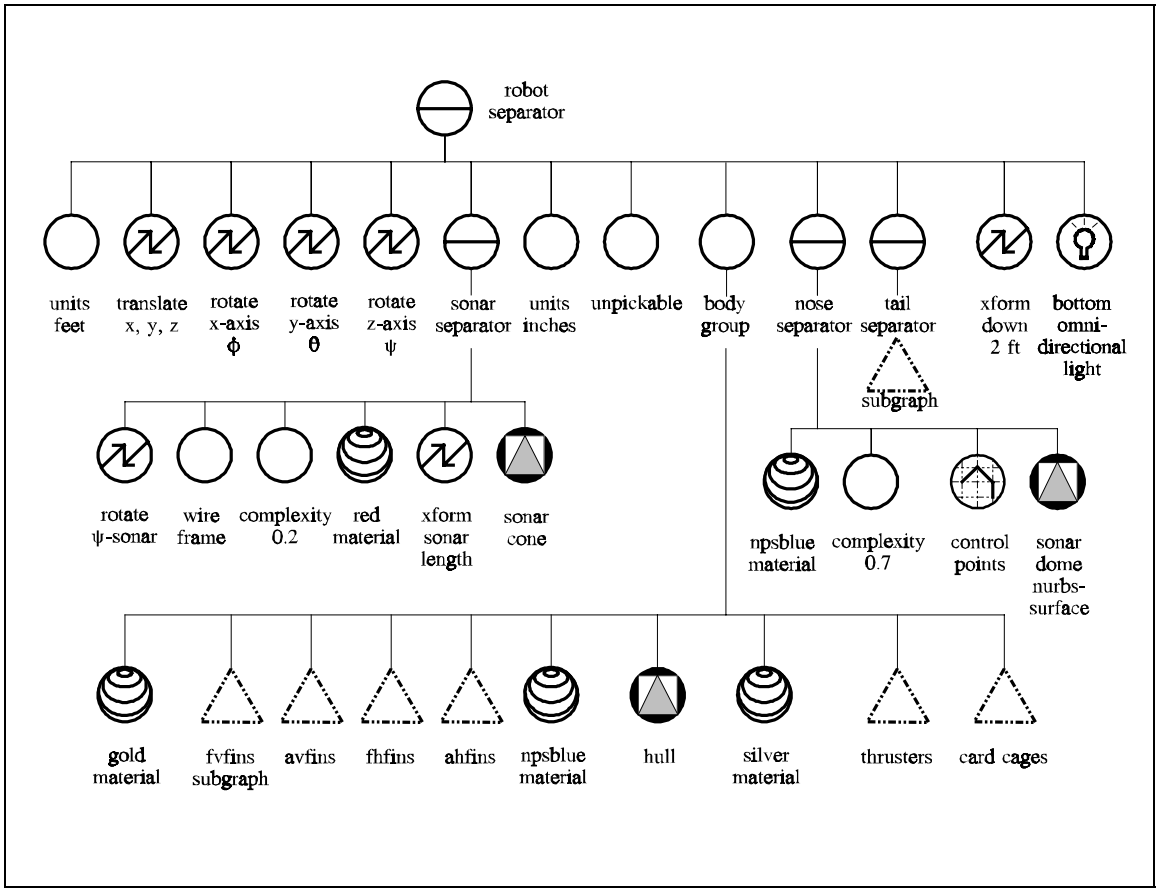

Figure 5.1. *Open Inventor* scene graph for the NPS AUV graphics model (*auv.iv*).

Open Inventor is release 2 of the Inventor toolkit and includes numerous performance improvements. A performance optimization guide and an offline scene

graph optimization tool have also been included. These are both useful for tuning Open Inventor applications to achieve near-optimal graphics pipeline performance.

2. Open Standards and Portability

Silicon Graphics Inc. (SGI) is the preeminent company producing 3D computer graphics workstations and software, including *Open Inventor*. SGI has made a corporate commitment to maintain the *Open Inventor* scene description language as their preferred open standard graphics file format. Although *Open Inventor* syntax is not in the public domain like *Open GL*, SGI has further committed to maintaining backwards compatibility through future versions of *Open Inventor*. (Wernecke 94c) provides a methodology for writing translators from other scene description languages to *Open Inventor*, further encouraging nonproprietary portability among graphics models. Numerous third-party vendors are porting the *Open GL* and *Open Inventor* programming environments to other operating systems and architectures (Macintosh, Windows, Sun, HPUX etc.), further extending the expected portability of *Open Inventor* models and viewers. Ubiquitous portability for analytic, hypermedia, network, multicast and graphics tools is an extremely desirable feature for virtual world model builders. Suitability of *Open Inventor* for this role was recently underscored by an open working group examination and ballot which chose *Open Inventor* over a dozen competitors as the baseline for the draft Virtual Reality Modeling Language (VRML) specification (Pesce 94) (Bell 94).

Open Inventor file formats can be specified as ASCII (plain text) or binary format files. *Open Inventor* specifications require that the first line of any file (ASCII or binary) contain a plain-text declaration of Inventor version number for forward compatibility with current and future file readers. Binary file formats are not openly published by SGI but binary file readers are openly available, a design decision made to ensure efficient backward format compatibility in future versions. As might be predicted from an information-theoretic perspective, compressed ASCII *Open Inventor* files are about the same size as binary files. Thus compressed ASCII (i.e. human readable) *Open Inventor* files are suitable for network distribution with minimal

bandwidth load. *Open Inventor* scene description files in text format (or forthcoming VRML file format extensions) are therefore excellent candidates for object definitions in a large-scale virtual world.

3. Behavior Animation through Data Sensors, Timer Sensors and Engines

Once graphics objects are specified, most graphics programmers expend a great deal of effort connecting devices, data or algorithms to animate the scene. These animation techniques are typically the heart of any graphics program and the specific reason that most graphics programs are needed, because the only way to explicitly specify behaviors is through the programming language itself. This also means that most graphics scenes are not portable as scene description files, only as programs. *Open Inventor* significantly extends the capabilities of scene description files by providing data sensors, timer sensors and behavioral "engines" which can be connected to automatically animate elements of the scene graph. Sensor and engine functionality and connections can still be written out to file, preserving behavioral connections.

Behavioral extensions to a scene description language are very useful. A simple example of engine functionality from (Wernecke 94a) is used to animate the static *JASON* ROV graphics model (which was originally donated via electronic mail). A graphics rendering of *JASON* appears in Figure 5.2. This figure moves about the base of an oil platform in the underwater virtual world. The corresponding animation scene graph is shown in Figure 5.3. Further work on extending behavioral definitions to include detailed physically-based dynamics is desirable and has been demonstrated independently (Zyda 92a).

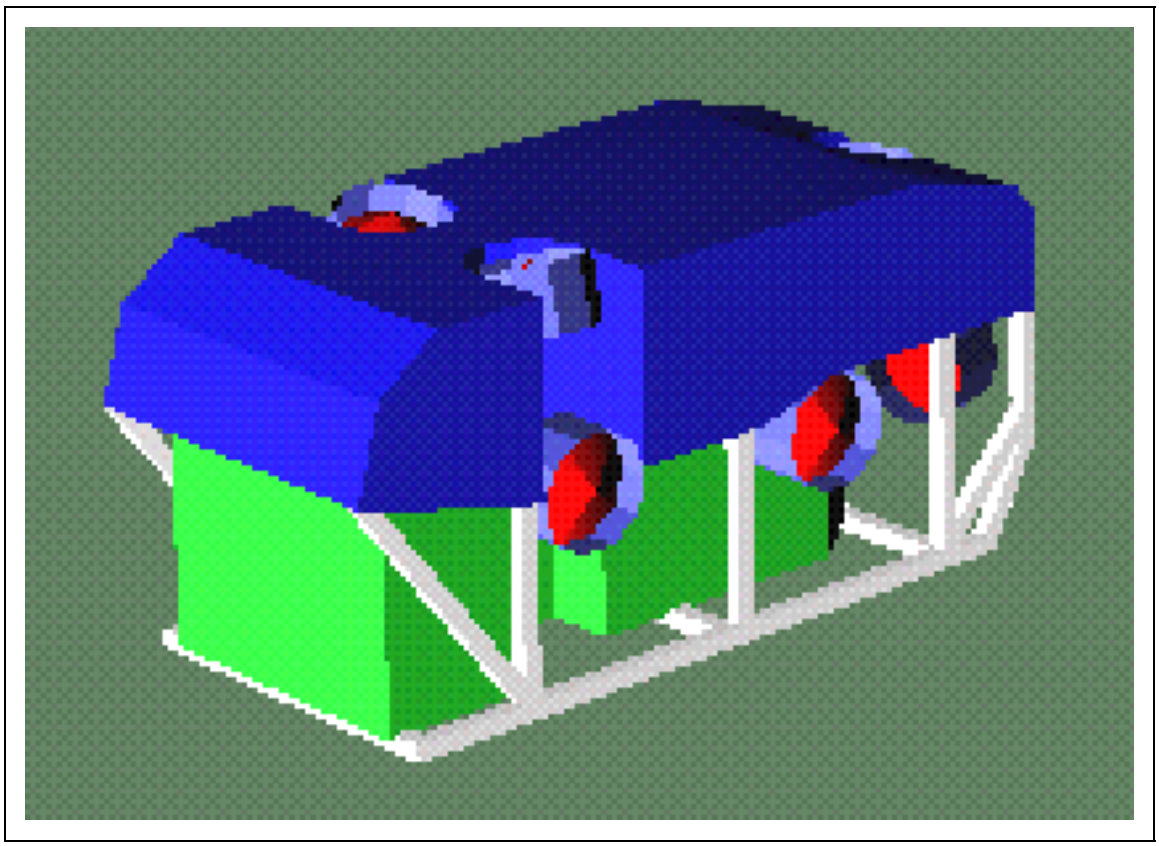

Figure 5.2. *Open Inventor* rendering of *JASON* ROV graphics model.

D. NETWORK LINKS TO GRAPHICS OBJECTS

As the use of the DIS standard becomes widespread, implementation of DIS library functionality will be more frequent and a good candidate for tool automation. Currently, vehicle graphics model connections to a DIS interface can be manually programmed or specified through initialization files within virtual world viewers such as NPSNET (Pratt 93). Increased user familiarity and availability of DIS libraries will increase the population of DIS-compatible graphics-based entities. Creation of DIS-compliant physical and graphical models is becoming progressively easier.

A recommended area for future implementation is the use of the DIS Message PDU to augment the announced arrival of new entities. The Message PDU might specify Internet Universal Resource Locator (URL) addresses which contain the

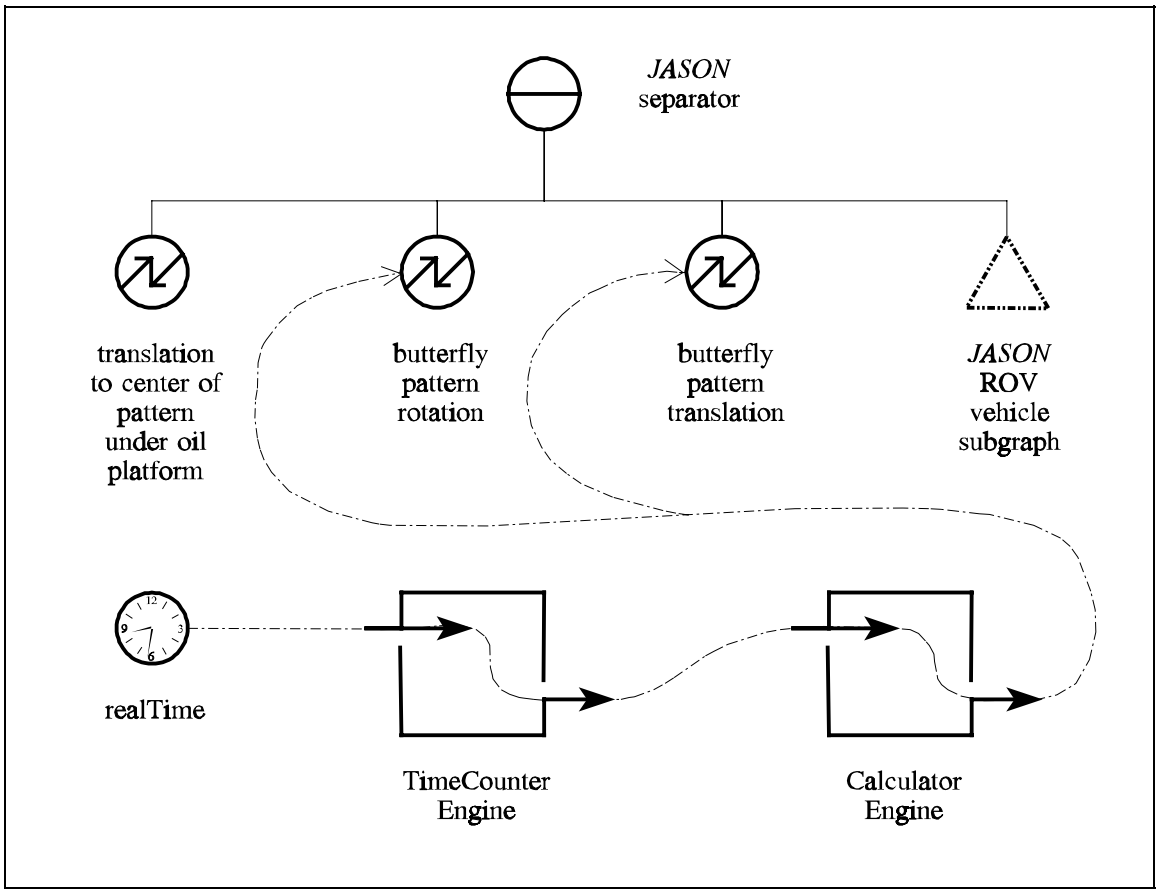

Figure 5.3. Engine animation scene graph for *JASON* ROV wandering behavior.

graphics model and operational characteristics for entity types that were previously unknown or unavailable. Such an extension permits the introduction of new DIS entities automatically without requiring pre-exercise coordination. Another interesting use of DIS Message PDUs in this underwater virtual world application might be to relay the robot commands which are being spoken to all viewers. Possible formats for this information include the original text, or a URL pointing to the synthesized audio file to minimize duplicate sound server queries.

As the use of the World-Wide Web becomes ubiquitous, the placement of graphics objects, images and datasets at well-defined network locations on public servers will become commonplace. Individual and institutional domain experts can maintain and update sophisticated world databases for open retrieval on demand.

Textures corresponding to specific locations in terrain datasets can be used to map available imagery to the real world. An example texture image manually collected from a MBARI ROV *Ventana* video stream via the MBone is shown in Figure 5.4.

Figure 5.4. Example Monterey Canyon bottom image recorded via MBone video from the MBARI ROV *Ventana*. This image is applied as a bottom texture in the underwater virtual world. Used with permission.

Widespread application of textures is particularly suited to the automatic collection of image data by robots. Automated collection and recording of video mosaics can be registered with terrain and stored on public file servers to build textured maps for large-scale virtual worlds. Extension and standardization of such approaches is also furthered by the combination of graphics and networking mechanisms proposed in the draft VRML specification (Bell 94).

E. SPECIAL METHODS

Much more work is possible to extend and augment the graphics viewer. User interface extensions will be focused exclusively on X-Windows Motif, Tk/Tcl (Osterhout 94) or hypertext markup language in order to maximize portability. Sound and data sonification can add an extra dimension to the display of scientific information. Automatically embedding hypermedia links inside scene graphs is expected to be possible using VRML extensions to *Open Inventor*, hopefully through use of embedded comments or future compatibility between the two scene graph languages.

F. SUMMARY AND FUTURE WORK

The characteristics of an open graphics viewer for underwater virtual world rendering are presented. Principal requirements include capable flexibility for scientific visualization and portability across multiple hardware and software platforms. *Open Inventor* is demonstrated as an effective programming toolkit in this regard. The desirability of scaling to very large numbers of users and information sources leads to a great deal of interesting future work which can extend graphics capabilities by embedding network capabilities. The use of multicast DIS message PDUs for distribution of World-Wide Web pointers, extending scene description languages to include dynamic behaviors and the proposed functionality of the Virtual Reality Modeling Language (VRML) are especially promising possibilities.