ARMY RESEARCH LABORATORY

BRL-CAD Tutorial Series: Volume IV – Converting Geometry Between BRL-CAD and Other Formats

by John R. Anderson and Eric W. Edwards

ARL-SR-121 **May 2004**

*NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5068

ARL-SR-121 **May 2004**

BRL-CAD Tutorial Series: Volume IV – Converting Geometry Between BRL-CAD and Other Formats

John R. Anderson
Survivability/Lethality Analysis Directorate, ARL

Eric W. Edwards
SURVICE Engineering Company

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
May 2004	Final	February 2003–December 2003

4. TITLE AND SUBTITLE
BRL-CAD Tutorial Series: Volume IV – Converting Geometry Between BRL-CAD and Other Formats

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
John R. Anderson and Eric W. Edwards

5d. PROJECT NUMBER
DAAD17-03-D-001, D.O.9

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: AMSRD-ARL-SL-BE
Aberdeen Proving Ground, MD 21005-5068

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-SR-121

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Since 1979, the U.S. Army Research Laboratory has been developing and distributing the BRL-CAD constructive solid geometry (CSG) modeling package for a wide range of military and industrial applications. The package includes a large collection of tools and utilities, including an interactive three-dimensional geometry editor, raytracing and generic framebuffer libraries, network-distributed image-processing and signal-processing capabilities, and an embedded scripting language.

As part of this effort, a multivolume tutorial series is being developed to assist users in the many features of the BRL-CAD package. "Converting Geometry Between BRL-CAD and Other Formats," which is the fourth volume in the series, discusses pertinent characteristics of the BRL-CAD file format and provides specific instruction on converting to/from various other modeling file formats. Also discussed are the steps involved in creating a new BRL-CAD converter and postconversion issues.

15. SUBJECT TERMS
BRL-CAD, geometry conversion, CAD, constructive solid geometry, boundary representation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			John R. Anderson
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL	54	19b. TELEPHONE NUMBER (*Include area code*) 410-278-7267

Contents

List of Figures

List of Tables

Acknowledgments

The authors would like to thank Messrs. Lee Butler, Edwin Davisson, William Landis, and David Davis for technically reviewing this document in a timely manner and for making numerous suggestions to improve its content and presentation. The authors also thank Dr. Paul Tanenbaum for the use of his graphics for figures 1 and 2 and Ms. TraNese Christy for her help in adapting them for presentation in this document. Finally, the authors would like to posthumously acknowledge Mr. Michael Muuss, the original architect of BRL-CAD. Without his vision, intellect, and diligence, this work would not have been possible. Thus, the BRL-CAD Tutorial Series is dedicated to his memory.

Supplementary Tutorial Boxes

Note that tutorial boxes have been used throughout this document to supplement the information presented in text. Each of these boxes is designated by a graduation hat icon.

INTENTIONALLY LEFT BLANK.

1. Introduction

1.1 Background, Purpose, and Scope

Since 1979, the U.S. Army Research Laboratory* has been developing and distributing the Ballistic Research Laboratory - Computer-Aided Design (BRL-CAD) three-dimensional (3-D) solid modeling package to support combat vehicle vulnerability studies and various other military and industrial applications. The software, which is now in its third generation, includes a large collection of tools and utilities, including an interactive geometry editor, raytracing and generic framebuffer libraries, network-distributed image-processing/signal-processing capabilities, and an embedded scripting language.

In support of the package, a multivolume tutorial series is being written to assist users with the many features and functionality of BRL-CAD. Three volumes have been published thus far. Volume I provides an overview of the package contents and installation (Butler and Edwards, 2002). Volume II addresses the basic functionality of the package's Multi-Device Geometry Editor (MGED) and offers a comprehensive list of the user commands available (Butler et al., 2001). Volume III discusses the modeling process as well as principles and techniques to help maximize BRL-CAD's effectiveness (Butler et al., 2003). All of these documents are available for download at <http://ftp.arl.army.mil/brlcad/> (U.S. Army Research Laboratory, 2003).

The purpose of Volume IV is to discuss issues of compatibility and conversion between the BRL-CAD geometry file format and the formats of various other computer-aided design (CAD), computer-aided manufacturing (CAM), and computer-aided engineering (CAE) packages. Conversion is increasingly important for BRL-CAD users who must interact with a growing number of Government and commercial organizations involved in the research, development, testing, and evaluation of today's combat systems.

Note that this document addresses BRL-CAD *geometry* converters, not BRL-CAD *image* converters (e.g., **pix-bw**, **pix-ps**, **pix-rle**, **bw-ps**, and **pl-ps**). For further information on image conversion, see the man page on the utility of interest.

Also, because of the many file formats in existence today and the rapidly changing nature of computer software and software companies, it would be impractical to try to address all of the conversion paths and methods that are currently or potentially possible. Many times, conversion from one file format to another is not a one-to-one process. Depending on the amount of time and effort the BRL-CAD user is willing to invest, seemingly incompatible formats can be

*On 30 September 1992, the U.S. Army Ballistic Research Laboratory (BRL) was deactivated and subsequently became part of the U.S. Army Research Laboratory (ARL) on 1 October 1992.

"forced" to convert via another CAD format or via a standardized CAD format (e.g., the Initial Graphics Exchange Specification [IGES] or the Standard for the Exchange of Product Model

Data [STEP]). In fact, when it comes to converting from BRL-CAD, the widely accepted Stereo Lithography Tessellation Language (STL) format offers a crude path to convert BRL-CAD objects to virtually any commercial CAD system. The user is reminded, however, that such forced conversions can sometimes produce geometry of such poor quality (e.g., low-resolution or lossy output formats) or poor performance characteristics (e.g., large or computationally intensive output files) that completely rebuilding a model from scratch might be a preferable alternative.

In any case, the focus of this document is on the *primary* formats that convert to and from BRL-CAD. Section 2 discusses pertinent characteristics specific to the BRL-CAD format. Section 3 addresses general conversion to/from the American Standard Code for Information Interchange (ASCII) format. Section 4 addresses the primary formats that convert to BRL-CAD. Section 5 addresses the primary formats that convert from BRL-CAD. Section 6 provides guidance for those users who desire to create their own customized converters. And section 7 addresses postconversion issues. In addition, the user is encouraged to consult the web sites and other resources cited at the end of each converter discussion to obtain the latest information on each format.

1.2 The Need for Conversion

Since its inception, BRL-CAD has proven itself to be a particularly effective tool for producing high-resolution and physically realistic geometry for ballistic penetration, radar signature, and other types of related analyses. However, several commercial CAD packages have also gained popularity, especially within organizations that design and manufacture military systems. Although these packages are not designed for vulnerability studies per se, their widespread use throughout military circles necessitates that BRL-CAD users be able to convert to and/or from them.

There are numerous benefits associated with the use of commercial packages in vulnerability studies. The Survivability/Vulnerability Information Analysis Center (SURVIAC) identified some of the most common benefits in its 2002 State-of-the-Art Report (SOAR) on geometric modeling. They include the following (SURVICE Engineering Company, 2002):

- **Reduced Modeling Time and Effort** - Manufacturers often spend hundreds of hours constructing detailed CAD models to streamline their design, production, and assembly processes (e.g., through computer numerical control equipment). It therefore makes economic sense—and is consistent with the military's ongoing commitment to leverage commercial off-the-shelf technology—to take advantage of existing geometric models and data where possible and avoid the significant cost and effort of building new models from scratch.

- **Increased Funding and Support** - Because commercial packages must maintain a larger user base to stay competitive in the open market, the most popular packages typically possess ample funding and personnel to continuously develop, support, and improve them.
- **Compatibility With Standardized Formats** - Most commercial packages possess the direct or indirect (through third-party vendors) capability to convert to standard or intermediary geometry formats, making the packages compatible (at least to some degree) with a wide range of other CAD formats.
- **Third-Party Add-On Support** - Large commercial packages typically offer a variety of plug-ins for other packages/utilities.

Of course, commercial CAD packages also have some common liabilities when used for vulnerability studies. They include the following (SURVICE Engineering Company, 2002):

- **Incompatible/Inaccessible File Formats** - As discussed in section 2, some CAD formats use the boundary representation (BREP) approach to solid modeling, which is largely incompatible with the constructive solid geometry (CSG) approach that BRL-CAD uses. In addition, although most commercial packages have some capability for data exchange conversion, data storage is often in a proprietary (and therefore inaccessible) native format. Moreover, when target descriptions are converted to a format designed for vulnerability assessment (i.e., BRL-CAD or FASTGEN), they often require manual checking, adjustment, and additional modeling (see section 7, Postconversion Issues). Typical problems that must be addressed include the translation of curved and irregular surfaces, the representation of solids of rotation, tolerancing, and interference handling.
- **Too Much Detail** - Commercial geometry files often contain too much of a good thing for vulnerability analysis. That is to say, commercial CAD packages often model geometry all the way to the "nuts and bolts" level, whereas vulnerability analyses are typically concerned only with details down to the level of shielding and critical components. Unfortunately, added detail produces unnecessarily large and complex input files and thus longer processing times for vulnerability assessments.
- **Too Little Detail** - In addition to providing too much detail, commercial packages sometimes provide too little detail for vulnerability studies. Vulnerability analysts and the applications designed to interrogate geometry rely on geometric measurements and material properties not always present in commercial CAD formats.
- **Package-Specific Naming Conventions** - Some organizations and CAD packages use unique object naming schemes that make geometry difficult to organize and work with when converted to or from BRL-CAD format.

- **Relatively Slow Raytracing Capability** - Commercial CAD packages typically have relatively slow raytracing speeds, and raytracing is the primary means of geometry interrogation in vulnerability assessment.
- **Limited Integrated Vulnerability Assessment Support** - Commercial CAD packages are designed for engineering analysis, not ballistic analysis and therefore offer few, if any, "shotlining" tools and limited integrated vulnerability assessment support.
- **High Cost** - In addition to facing the typically high cost of commercial CAD software, the user often faces the decision of whether or not to invest in non-PC hardware (e.g., UNIX workstations) to obtain maximum performance, especially with large, complex geometry. In addition, users may be required to pay continuing licensing and maintenance fees, often on a per-seat basis (although recent developments have offered PC-based implementations and short-term leasing "seats" to make these packages more affordable).

Not surprisingly, when considering the benefits and liabilities of using commercial packages in vulnerability assessment, one can see that the conversion of these formats to/from BRL-CAD is one of the most needed as well as one of the most challenging tasks the BRL-CAD user faces today.

2. Pertinent Characteristics of the BRL-CAD Format

2.1 CSG vs. BREP

As mentioned previously, there are two basic approaches to solid modeling: CSG and BREP. In CSG modeling, an object is represented as a Boolean combination of simple primitive shapes (e.g., spheres, cylinders, cones, etc.). In BREP modeling, an object is represented by a set of surfaces (e.g., facets, triangles, and splines) that are "stitched" together to completely enclose the object.

For example, as shown in figure 1, to geometrically represent a simple extrusion die used in material manufacturing, a CSG model might subtract a cone from a cylinder to achieve the desired shape. Alternatively, a BREP model might achieve the same shape by joining several surfaces (Tanenbaum, 2001).

Most of the major commercial solid modeling packages currently in distribution (e.g., Pro/ENGINEER [commonly known as Pro/E], Unigraphics, etc.) use the BREP approach, although they do have some CSG capability. BRL-CAD, on the other hand, is primarily a CSG modeling system with some BREP capability.

Conversions that occur between CSG and BREP typically involve their "lowest common denominator."

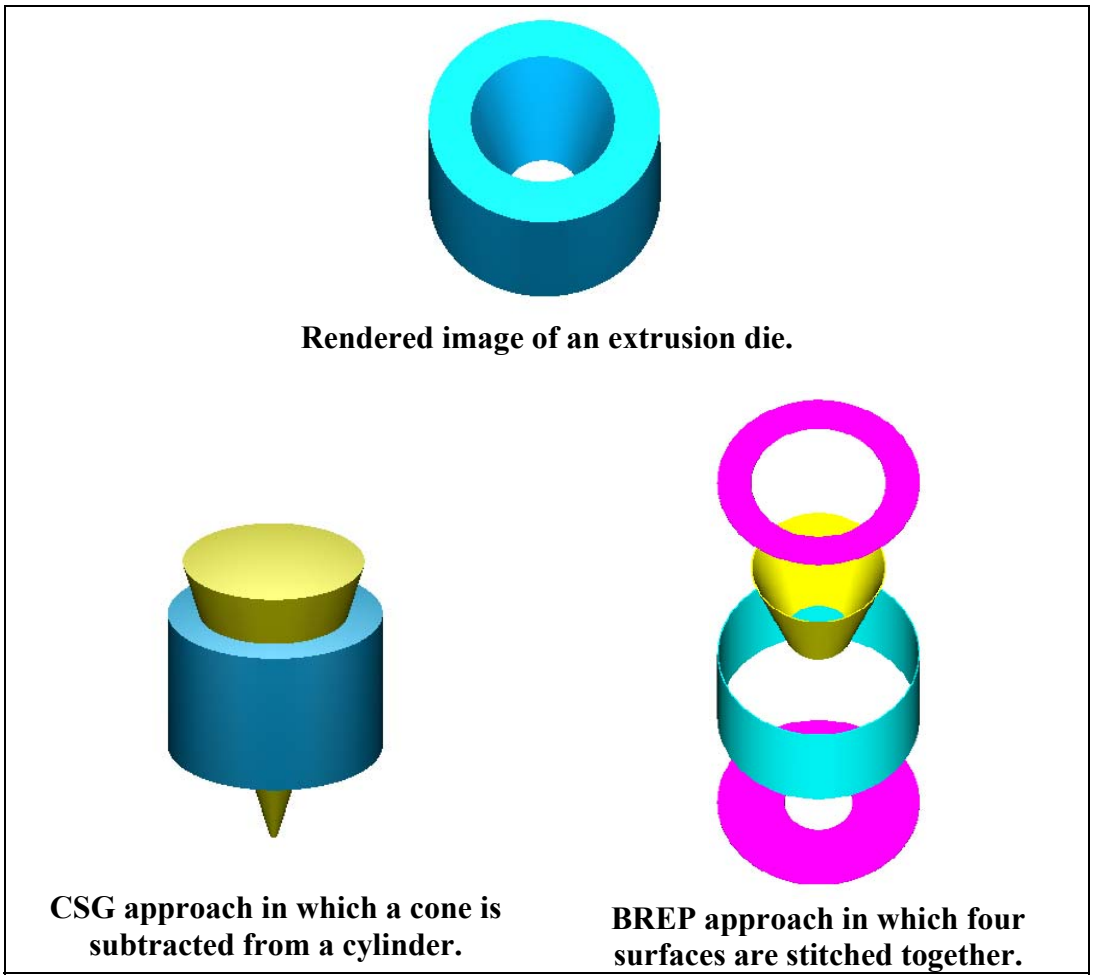

Figure 1. CSG and BREP approaches to representing an extrusion die (Tanenbaum, 2001).

For BRL-CAD, a simple type of BREP is available in the form of triangles; it is called the Bag of Triangles (BOT) primitive. If the BREP object consists of all triangles, the BRL-CAD representation can be an exact duplicate. But more often, a BREP object consists of smooth, curved surfaces, meaning the BRL-CAD triangle representation will be an approximation of the object. Note that this approximation may be made as close to the true surface as desired, but at the cost of more, smaller triangles (see figure 2) (Tanenbaum, 2001). Thus, the higher degree of smoothness means the greater the impact on file size and performance of any application trying to employ all the triangles.

Because most commercial CAD systems have the capability to produce tessellated approximations of their BREP objects, some converters to BRL-CAD (e.g., those with Pro/E and Unigraphics) take advantage of these capabilities. In addition, the previously mentioned STL format, which represents solid objects entirely with triangles, offers the community a universal (albeit crude) way to convert BRL-CAD geometry (via the **g-stl** converter) to nearly any commercial CAD system.

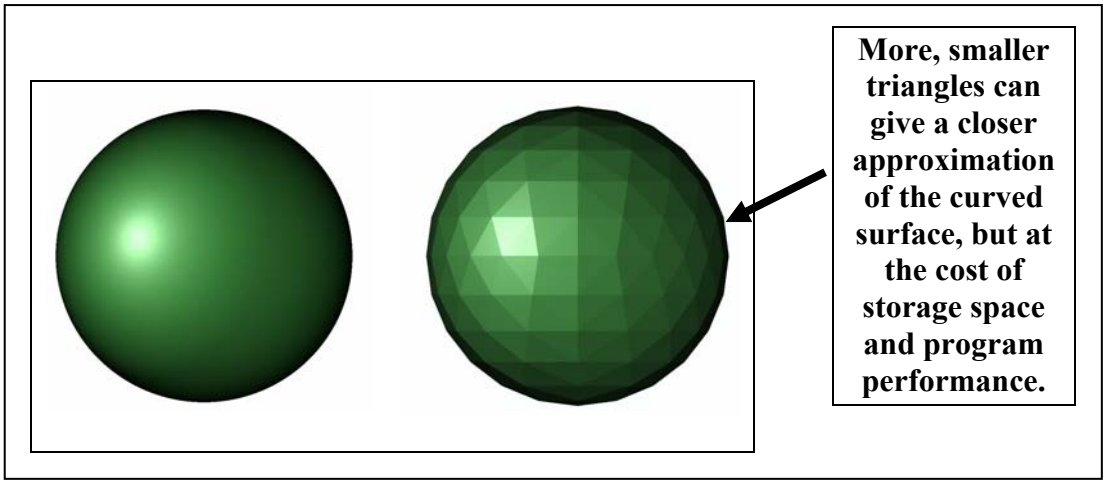

Figure 2. Approximating a smooth, curved surface (left) with a BOT (right) (Tanenbaum, 2001).

BREP Terminology

Boundary representations can take several forms. Although the terms for these forms are sometimes used interchangeably, the following list provides some key distinctions:

- **Polygonal (or Facetized) Geometry** - A type of BREP geometry that represents objects with a collect of the plane bounded by a closed path of line s (polygons).
- **Triangle** - A special type of polygon that has sides.
- **Non-Uniform Rational B-Spline (NURB) Surface** - A piecewise polynomial representation of a smooth, curved surface (e.g., an airplane wing).
- **BOT (Bag of Triangles)** - A BRL-CAD primitive that consists of a collection of triangles and that gives the package limited BREP capability by approximating solid geometry.

In the end, although there are benefits to using the BREP methodology (e.g., ease of use, software availability/compatibility, etc.), BRL-CAD developers and vulnerability analysts have found CSG modeling to be the best approach in terms of model accuracy, storage efficiency, precision, and analysis speed.

CSG or BREP?

Depending on one's point of view, there are advantages and disadvantages to using CSG vs. BREP. Although "the right" approach for a given project ultimately depends on many specifics (e.g., model purpose, hardware/software availability, and compatibility with other users), the following areas should be considered when choosing between CSG and BREP:

- **Realism** - Does the model need to approximate real material densities, thicknesses, and volumes, or is the realistic appearance of object surfaces more important?
- **Resolution** - How geometrically precise must the model be? For example, radar cross-section studies are sensitive to surface detail; ballistic penetration analyses usually are not.
- **Resources** - How much disk space and memory will be needed to store the geometry and data structures?
- **Time** - How much CPU time is needed to read, write, and process the model, and how much user time will be needed to understand, modify, and conduct analyses with it?

2.2 Prerelease 6.0 Database vs. Current Format

With the release of BRL-CAD 6.0, an entirely new BRL-CAD database format was introduced. Additional capabilities provided by the new format include unlimited name lengths, object attributes, machine independence (for IEEE floating point architectures), reduced disk space requirements, opaque binary objects, and the capability to hide objects. (For more information on the new database format, see Butler et al. [2003]).

Older BRL-CAD databases may be upgraded to the current database (db) format using the **dbupgrade** utility. To execute this command, the user types the following command at the command-line prompt:

```
dbupgrade oldformat.g currentformat.g
```

This command reads an existing **input.g** database (in the older format) and writes a new **output.g** in the latest format. Remember that all BRL-CAD geometry files are designated by a ".g" suffix. Thus, the conversion commands discussed in the following sections use a "g" to represent the BRL-CAD file format (e.g., asc2g, dxf-g, g-acad, etc.).

What About Databases in the Old Format?

Although databases created with BRL-CAD release 6.0 or later are in the new format by default, BRL-CAD retains the ability to work in the old format. Thus, new-database users need not be concerned about losing databases in the old format. Also, whenever a new database format is implemented in the future (typically a rare event), the **dbupgrade** utility will be modified so that it will always be able to upgrade databases to the current format.

2.3 Converters Currently Available in BRL-CAD

Table 1 shows the CAD formats for which BRL-CAD currently converts from (imports) or to (exports).

Table 1. BRL-CAD conversion capabilities.

CAD Format	Import Capability?	Export Capability?
ACAD	—	✓
ASCII	✓	✓
AutoCAD DXF	✓	✓
Elysium Neutral Facetted	✓	—
EUCLID[a]	✓	✓
FASTGEN	✓	—
IGES	✓	✓
Jack	✓	✓
NASTRAN	✓	—
Pro/E	✓	—
STL	✓	✓
TANKILL	✓	✓
Unigraphics	✓	—
Viewpoint	✓	—
VRML	—	✓
Wavefront (.obj)	—	✓
X3D	—	✓

[a] Conversion is to/from the ASCII EUCLID decoded format, not native format.

3. Converting From/to ASCII

With the implementation of BRL-CAD's machine-independent database format, the need for converting to or from ASCII format has rapidly diminished. Nonetheless, BRL-CAD still supports ASCII conversions, and release 6.0 introduced a new ASCII format. This format is

simply a Tcl* script that (when executed by a properly empowered Tcl interpreter) rebuilds the database from which it was derived. In contrast, the old ASCII format consisted of a sequence of one-line records that corresponded to the old database format records.

3.1 Converting From ASCII

The **asc2g** utility is used to convert ASCII format databases into the new binary format. This utility will always create the new binary format, but it will accept either of the two ASCII formats.

The usage for this utility is as follows:

```
asc2g input.asc output.g
```

In this command, **input.asc** is a file previously created using **g2asc** and **output.g** is the name where the resulting binary database should be stored.

3.2 Converting to ASCII

The converse of the **asc2g** utility is the **g2asc** utility, which is used to convert from either the current database format or the prerelease 6.0 database format to an ASCII format. Databases in the new format will be converted to the new (Tcl script) ASCII format. Databases in the old format will be converted to the old ASCII format.

The command for this utility is as follows:

```
g2asc input.g output.asc
```

In this command, the **input.g** is a BRL-CAD database (old or new format) and **output.asc** is the name where the ASCII output is placed. As mentioned previously, the form of the ASCII output depends on which database format is used by the input database. If it is known that the input database is in the older BRL-CAD format, then the **g2asc** may be executed using redirected input and output, as follows:

```
g2asc < input.g > output.g
```

If this is attempted using the newer database format, however, an error message will be displayed and no conversion will be performed.

*Tcl/Tk is an interpreted programming language that can be used to extend/customize BRL-CAD functionality. Users are encouraged to consult one of the many texts on Tcl/Tk syntax and use, including Practical Programming in Tcl and Tk (Welch, 1999) and TCL/TK in a Nutshell (Raines and Tranter, 1999).

4. Converting to BRL-CAD

4.1 General Information About Converting to BRL-CAD

Table 2 contains the primary file formats (other than ASCII) that currently convert to BRL-CAD. Note that, as discussed previously, the "-g" designation at the end of each converter name indicates that the converted file will be in BRL-CAD's standard geometry (.g) format.

Table 2. BRL-CAD import converters.

CAD Format	Import Converter
AutoCAD DXF	dxf-g
Elysium Neutral Facetted	enf-g
EUCLID[a]	euclid-g
FASTGEN (FASTGEN4 and PATCH)	fast4-g/patch-g
IGES	iges-g
Jack	jack-g
NASTRAN	nastran-g
Pro/E	—[b]
STL	stl-g
TANKILL	tankill-g
Unigraphics	ug-g
Viewpoint datalabs	viewpoint-g

[a] Conversion is from the ASCII EUCLID decoded format, not native format.
[b] Converter is part of the Pro/E GUI.

4.2 Converting From AutoCAD DXF Format

DXF is Autodesk Inc.'s proprietary data specification that has been developed to support links with AutoCAD, the company's popular CAD software. The format, which has been implemented in many CAD systems (particularly those that work on PCs), is simple and relatively limited; however, it has become a well-established means of exchanging engineering drawings.

The DXF format allows for the specification of the units used in the DXF file; however, not all DXF files include this information. Thus, if units information is not found in the DXF file, millimeters (which is also the default and underlying unit of measure in BRL-CAD) is the assumed unit of measure. Of course, if millimeters is not the right choice for a particular user or application, the **s** (scale) option can be used. This option and the other available options for the **dxf-g** converter are discussed in the text that follows.

When using this converter, polygons and meshes are converted to BOT primitives. Each layer becomes a region. Lines, circles, and arcs become n-manifold geometry (NMG) objects, and points become spheres.

	The NMG primitive represents geometry that is defined by points, lines, polygonal facets, and collections thereof. It is the primary BRL-CAD primitive for encoding one- or two-dimensional (2-D) data. For more information on the NMG, see Muuss and Butler (1991) and Weiler (1987).

The command for the **dxf-g** converter is as follows:

```
dxf-g [options] input.dxf output.g
```

The options for the command are as follows:

- **c** - specifies that only the first color encountered for a layer in the DXF file is used; additional colors for the same layer are ignored.
- **d** - debug; produces core dump on failure.
- **s** - applies the specified scale factor to the DXF data (takes an argument).
- **t** - sets tolerance distance (in millimeters) (default is 0.005) (takes an argument).
- **v** - verbose; prints out progress information.

For more information on the DXF file format and import possibilities, see the on-line documentation on the Autodesk web site at http://adeskftp.autodesk.com/prodsupp/downloads/dxf.pdf (Autodesk, 2003).

4.3 Converting From Elysium Neutral Facetted Format

Elysium Inc. was founded in 1984 to provide product structure synchronization and compatibility between different CAD systems. The company claims to hold the industry's highest translation success rate and relies on three of its products (CADporter, CADdoctor, and CADserver) to convert geometry between a large number of formats, including Pro/E, Unigraphics, ABAQUS/CAE, ACIS, CATIA, Inventor, I-deas, Metrix Build!IT, One Space Designer, Parasolid, and SolidWorks (Elysium, 2003).

The **enf-g** converter was originally written when BRL-CAD developers needed an intermediate format—the Elysium neutral facetted file format—to allow the conversion of Unigraphics geometry to BRL-CAD. The converter converts each part to a BRL-CAD region consisting of one BOT primitive. Ident numbers are incremented for each region. If a part name mapping file is provided, part names in the input file will be output using the corresponding names from the mapping file. The part name mapping file may look similar to the following:

01a3-011-03 washer, flat

01a2-011-02 nut, flex

.
.
.

The syntax for the **enf-g** command is as follows:

```
enf.g input_file output.g
```

The options for the command are as follows:

- **`i starting_ident`** - sets the starting ident number (default is 1000).
- **`n part_name_mapping`** - sets the mapping from input names to output names.
- **`t tolerance_distance`** - sets distance calculation tolerance (in millimeters) (default is 0.005).

For more information on the Elysium file format and import possibilities, see the Elysium web site at http://www.elysiuminc.com (Elysium, 2003).

4.4 Converting From EUCLID Format

EUCLID is one of Europe's primary product design, manufacturing, and engineering CAD packages (though it is quickly being superceded by CATIA). Formerly distributed by the European Aeronautic Defence and Space Company's Matra Datavision subsidiary and now a product of the IBM subsidiary MDTVISION, the package was developed for the design and manufacture of complex models and has been used on major systems such as the Airbus, Eurocopter, Ariane, Eurofighter, Astrium, and Euromissile.

Package features include specialized applications for design, styling, drafting, analysis, machining, and product data management. Recent enhancements have focused on automatic creation of 2-D drawings from 3-D models, milling cycles, two- to four-axis wire cutting, sheet metal design, mold design, and standard data exchange format interfaces.

EUCLID offers several data translation interfaces, including those for DXF, IGES, VDA-FS, STL, and SET formats. In addition, other modules are available to help refine and customize data transfer, with direct connectivity at two levels available for CATIA (V4/V5) data transfer using BREPs to handle surface data. EUCLID geometry can be output to standard graphic (e.g., PostScript, Encapsulated PostScript, and Interleaf) and plotting (e.g., CalComp, HP, HP/GL2, OCE, and Versatec) formats as well as to an STL file.

The **euclid-g** converter converts an ASCII EUCLID "decoded" format file to BRL-CAD. Each part is converted to a BRL-CAD region consisting of a single BOT primitive.

The syntax for the converter is as follows:

```
euclid-g [options]
```

The options for the command are as follows:

- **`v`** - verbose; prints out progress information.
- **`i input_euclid_db`** - sets input file name (default is stdin).
- **`o output_brlcad_db`** - sets output file name (default is stdout).
- **`d tolerance_distance`** - sets distance calculation tolerance (in millimeters) (default is 0.005).
- **`n`** - produces NMG primitives (default is BOT primitives).
- **`x lvl`** - sets librt debug flag (see **raytrace.h** in the binary distribution for definitions of **DEBUG_xxx**).
- **`X lvl`** - sets NMG library debug flag (see **nmg.h** in the binary distribution for definitions of **DEBUG_xxx**).

For more information about the EUCLID file format and import possibilities, see the MDTVISION web site at http://support.mdtvision.com (MDTVISION, 2003).

Note that all header (h) files—including **raytrace.h**, **nmg.h**, and **bu.h**—are accessible in both the source and binary distributions of BRL-CAD. In the binary distribution, these files are installed in the **/user/brlcad/include/brlcad** directory.

4.5 Converting From FASTGEN Format

Developed by the Falcon Research and Development Company over 30 years ago, FASTGEN has been widely used in the Department of Defense air combat system assessment community (e.g., the Air Force Research Laboratory, the Joint Technical Coordinating Group for Munitions Effectiveness, and the Joint Technical Coordinating Group on Aircraft Survivability. Now distributed by the Survivability/Vulnerability Information Analysis Center (SURVIAC) as part of the Vulnerability Modeling Tool Set, the FASTGEN format uses geometry BREP based on NASTRAN, with data presented in a generic, open text-based file format.

Components of a target are represented in FASTGEN by triangles, quadrilaterals, cones, cylinders, spheres, and hexahedrons. These basic elements are designated as either plate (hollow) or volume (solid) mode and combined using a hierarchical structure for the formation of components and groups (SURVICE Engineering Company, 2002).

Notable aerospace firms and support organizations that use FASTGEN include Lockheed Martin, Boeing, Bell Helicopter, Northrop Grumman, Pratt & Whitney, General Electric, KETRON, ITT, BAH, SURVICE Engineering, and ASI.

There are several versions of FASTGEN currently in use. FASTGEN converters include FASTGEN4 and preprocessed FASTGEN Version 3 (also referred to as PATCH).

4.5.1 FASTGEN4

The **fast4-g** converter converts FASTGEN4 to BRL-CAD. FASTGEN4 entities are converted to corresponding equivalent BRL-CAD objects.

The syntax for the **fast4-g** command is as follows:

```
fast4-g [options] fastgen4_input output.g
```

The options for the command are as follows:

- **`d`** - prints verbose debugging output.
- **`q`** - prints nothing except errors.
- **`w`** - prints warnings about creating default names.
- **`plot_file_name`** - creates a UNIX plot file of all CTRI and CQUAD elements processed.
- **`c component_list`** - processes only the listed region ids; may be a list (e.g., 3001,4082,5347) or a range (e.g., 2314–3527).
- **`m muves_file_name`** - creates a MUVES warnings file containing CHGCOMP and CBACKING elements.
- **`b lvl`** - sets libbu debug flag (see **bu.h** in the binary distribution for definitions of **BU_DEBUG_xxx**).
- **`x lvl`** - sets librt debug flag (see **raytrace.h** in the binary distribution for definitions of **DEBUG_xxx**).

4.5.2 Preprocessed FASTGEN Version 3/PATCH

The **patch-g** converter converts preprocessed FASTGEN version 3 files to BRL-CAD format. The syntax for the command is as follows:

```
patch-g [options] output.g
```

The options for the command are as follows:

- **`f fastgen.rp`** - specifies preprocessed FASTGEN file (default is stdin).
- **`a`** - processes phantom armor.
- **`n`** - processes volume mode as plate mode.
- **`u #`** - specifies the number of union operations per region (default is five).
- **`c "x y z"`** - specifies the center of the object (in inches) (for some surface normal calculations).
- **`t title`** - specifies optional title (default is "Untitled MGED database").
- **`o object_name`** - specifies optional top-level name (default is "all").
- **`p`** - writes volume and plate mode components as BOTs.
- **`6`** - processes plate mode triangles as ARB6 solids (overrides **p** option for triangles).
- **`i group.file`** - specifies group labels source file.
- **`m mat.file`** - specifies materials information source file.

- **r** - reverses normals for plate mode triangles.
- **d lvl** - sets debug level.
- **x lvl** - sets librt debug flag (see **raytrace.h** in the binary distribution for definitions of **DEBUG_xxx**).
- **X lvl** - sets librt NMG debug flag (see **nmg.h** in the binary distribution for definitions of **DEBUG_xxx**).
- **T tolerance_distance** - sets distance tolerance (in inches).
- **A parallel_tolerance** - sets parallel tolerance (sine).

For more information about the FASTGEN file formats and import possibilities, see the SURVIAC web site at http://www.bahdayton.com/surviac/fastgen.htm (SURVIAC, 2003).

4.6 Converting From IGES Format

IGES was developed in 1979 by a consortium of government, industry, and academia representatives. Originally intended to provide a means of exchanging graphics and engineering drawings, IGES was extended to include solid models as well. This specification includes so many different implementations within itself that conversion between IGES flavors has become a small industry. As a result, conversion to/from IGES is a "hit-or-miss" proposition (NIST, 2003).

The Product Data Exchange using STEP specification is intended to replace IGES and correct the aforementioned deficiency by explicitly identifying different types of conversion and requiring converters to conform to those types. BRL-CAD supports conversion of two implementations of IGES, entirely facetted BREP and CSG with facetted BREP primitives.

The syntax for the **iges-g** converter is as follows:

```
iges-g [options] –o  output.g input.igs
```

The options available for this command are as follows:

- **n** - combines all the rational B-spline surfaces in the file into one BRL-CAD spline primitive.
- **d** - converts drawings in the IGES file to 2-D BRL-CAD NMG primitives.
- **3** - same as the **d** option, but the final projection to 2-D is not performed. This can produce 3-D drawings in some cases.
- **t** - converts all trimmed surfaces in the IGES file to a single BRL-CAD NMG primitive.
- **N primitive_name** - specifies a name for single primitive created using the **t** or **n** options.
- **p** - converts all BREP entities to BRL-CAD NMG primitives rather than the default BOT primitives.
- **x lvl** - sets librt debug flag (see **raytrace.h** in the binary distribution for definitions of **DEBUG_xxx**).
- **X lvl** - sets librt NMG debug flag (see **nmg.h** in the binary distribution for definitions of **DEBUG_xxx**).

The **n**, **d**, **t**, and **3** options are mutually exclusive. If none of these four options is provided, the default action is to convert only IGES solid model entities (CSG and planar face BREP) to BRL-CAD.

For more information about the IGES file format and import possibilities, see the National Institute of Standards and Technology (NIST) web site at http://www.nist.gov/iges (NIST, 2003).

4.7 Converting From Jack

Jack is a 3-D interactive ergonomics and human factors CAD package developed by the University of Pennsylvania's Center for Human Modeling and Simulation. Now maintained and distributed by Electronic Data Systems (EDS) (the company that now also distributes Unigraphics and NASTRAN), the package enables users to study and improve the ergonomics of product design and workplace tasks through the positioning of biomechanically accurate digital humans of various sizes in virtual environments. Jack and Jill digital "humans" can tell engineers what they can see and reach, how comfortable they are, when and why they're getting hurt, when they're getting tired, and other important ergonomics information. The package's principal features include a detailed human model, realistic behavioral controls, anthropometric scaling, task animation and evaluation systems, view analysis, automatic reach and grasp, and collision detection and avoidance (The University of Pennsylvania, 2001; EDS, 2003a).

The **jack-g** converter creates a single region consisting of a single BOT primitive. The syntax for the converter is as follows:

```
jack-g [options] input.jack output.g
```

The options for this command are:

- **`r region_name`** - specifies a name for the BRL-CAD region created. If this option is not specified, the input file name will be used to construct a region name.
- **`g group_name`** - specifies the name of a combination to create to hold the BRL-CAD region created. If this option is not specified, the combination will not be created.

For more information on the Jack file format and import possibilities, see the EDS web page at http://www.eds.com/products/plm/efactory/jack/ (EDS, 2003a).

4.8 Converting From NASTRAN Format

Originally developed under National Aeronautics and Space Administration (NASA) sponsorship in the mid-1960s, the NASA Structural Analysis (NASTRAN) program was one of the first efforts to consolidate structural mechanics into a single computer program. It has since been used as a general-purpose software tool in numerous industries, including aerospace, automotive, medical, heavy machinery, electronic devices, and consumer products. The program is developed and distributed by the MSC.Software Corporation (MSC) and (as of June 2003) the EDS Corporation. It employs advanced finite element analysis computational techniques to

analyze material strength/performance and evaluate static structures and the dynamic motion of structures (SURVICE Engineering Company, 2002; MSC.Software Corporation, 2003).

NASTRAN's nonlinear analysis capabilities can address a wide range of static and dynamic problems exhibiting both material and geometric nonlinear behavior. Heat transfer problems can also be solved using conduction, convection, and radiation methods under a variety of applied loads and boundary conditions.

The NASTRAN finite element modeling program is one of the general-purpose structural analysis programs used worldwide. Even though it was originally intended for structural analysis problems, its current applications include aeroelasticity, heat transfer, fluid structure interaction, acoustics, electromagnetics, and many other applications.

NASTRAN includes a file specification for describing geometric data. NASTRAN's wide use and adoption by CAD vendors make it well suited as a file standard.

The **nastran-g** converter currently only converts CBAR, CROD, CTRIA3, and CQUAD4 elements of NASTRAN files to BRL-CAD format. CBAR and CROD elements become cylinders in BRL-CAD. CTRIA3 and CQUAD4 elements become BOT facets.

The syntax for the converter is as follows:

```
nastran-g [options]
```

The options for the command are as follows:

- **`i NASTRAN_input_file`** - sets input NASTRAN file (default is stdin).
- **`o output.g`** - sets output file name (default is "nastran.g").
- **`n`** - produces NMG primitives (default is BOT primitives).
- **`x lvl`** - sets librt debug flag (see **raytrace.h** in the binary distribution for definitions of **DEBUG_xxx**).
- **`X lvl`** - sets NMG library debug flag (see **nmg.h** in the binary distribution for definitions of **DEBUG_xxx**).
- **`t tolerance_distance`** - sets distance calculation tolerance (in millimeters) (default is 0.005).
- **`m`** - sets input units to millimeters (default is inches).

For more on the NASTRAN file format and import possibilities, see the MSC web site at http://www.mscsoftware.com (MSC.Software Corporation, 2003) and the EDS web site at http://www.eds.com/products/plm/nastran/ (EDS, 2003b).

4.9 Converting From Pro/E Format

Distributed by the Parametric Technology Corporation (PTC), Pro/E is one of the most widely used commercial CAD packages for designing, engineering, and manufacturing products. The long list of major corporations that use Pro/E software for Product Lifecycle Management includes Boeing, Rolex, Audi, Dell, Nike, Maytag, Braun, and Hewlett-Packard (PTC, 2003).

Because of Pro/E's popularity in the Defense community, the Pro/E-to-BRL-CAD converter is one of the most important conversion utilities that BRL-CAD offers. Note that unlike the converters for other formats, the Pro/E converter is no longer command-line activated. This converter was written using the Pro/Toolkit module of Pro/E and therefore runs as part of the Pro/E program and GUI.

Accordingly, in order to use the converter, the user must have a seat of Pro/E as well as the BRL-CAD distribution. Currently, the converter is only supported on Silicon Graphics (SGI) machines with MIPS processors running the Irix operating system. The source code for this converter is included in the binary distribution, so users can compile it for different platforms if they have the Pro/Toolkit module for that platform.

Pro/E models are made up of two elements: parts and assemblies. Part files (which are designated by a **.prt** extension) are the basic building blocks of Pro/E geometry. Assembly files (which are designated by a **.asm** extension) are composed of parts and/or other assemblies. The converter produces a BRL-CAD *region* for each Pro/E part that is converted and a BRL-CAD *combination* for each Pro/E assembly that is converted. Each of these regions will consist of a single BOT primitive.

The conversion of geometry from Pro/E to BRL-CAD is a two-stage process. This converter first produces the ASCII form of BRL-CAD databases. The user then converts these databases to binary form using the **asc2g** utility.

In addition, because Pro/E files for most combat vehicles are so large (often requiring a stack of CDs to store them), entire geometries typically cannot be loaded all at once. Thus, the BRL-CAD user often has to convert geometry system by system (e.g., engine, transmission, and suspension) and then concatenate (i.e., join) them together in a single BRL-CAD geometry file. For more detailed information about this process, see the discussion on **dbconcat** in Volume III of this tutorial series.

Pro/E makes extensive use of referenced geometry. As discussed in the previous volume of this tutorial series (see section 5 of Butler et al. [2003]), referencing is the method by which multiple occurrences of objects are created by referring to a single object numerous times with different orientations and locations for each reference. These references are duplicated in BRL-CAD using combinations and transformation matrices. In some cases—such as when geometry is used with vulnerability codes that require each region to have a unique ident number—users may need to use the **xpush** command in MGED after the conversion is complete to replace the references with real geometry. For more information about this procedure, users should consult MGED's on-line help or the **xpush** entry in volume II of this tutorial series (see appendix A of Butler et al. [2001]).

Keys to Converting Pro/E Geometry

To maximize the efficiency and effectiveness of converting Pro/E geometry, the BRL-CAD user should first acquire the following information from the Pro/E designer:

- **A list of top-level assemblies** - This information is vital in helping the BRL-CAD user gain an understanding of the overall model structure and know where to begin the conversion process.
- **A mapping of part numbers to part names** - Although part numbers can be an important aspect of design and manufacturing, they have little meaning for vulnerability analysts, who are more concerned with the descriptive names of the components than the numbers represent.

The command to start the Pro/E program is specified by each installer. When Pro/E is started, the program looks for a file named **protk.dat** in a few specific places, one of which is in the current directory. This file informs Pro/E about Pro/TOOLKIT modules it should load. There is a **protk.dat** file for the Pro/E-to-BRL-CAD converter, and it is included in the distribution under the **pro-engineer** directory. Users should copy this file to the directory where they will be starting Pro/E. After that file is in place, Pro/E will load the converter at startup. If the loading succeeds, users will see a message saying "Installation of Proe-BRL converter succeeded." With Pro/E started and the converter module loaded, the user can open any Pro/E model he wants to convert.

The conversion process is started by selecting the **ProE-BRL** item in the Pro/E drop-down File menu. The converter dialog box, shown in figure 3, will then appear.

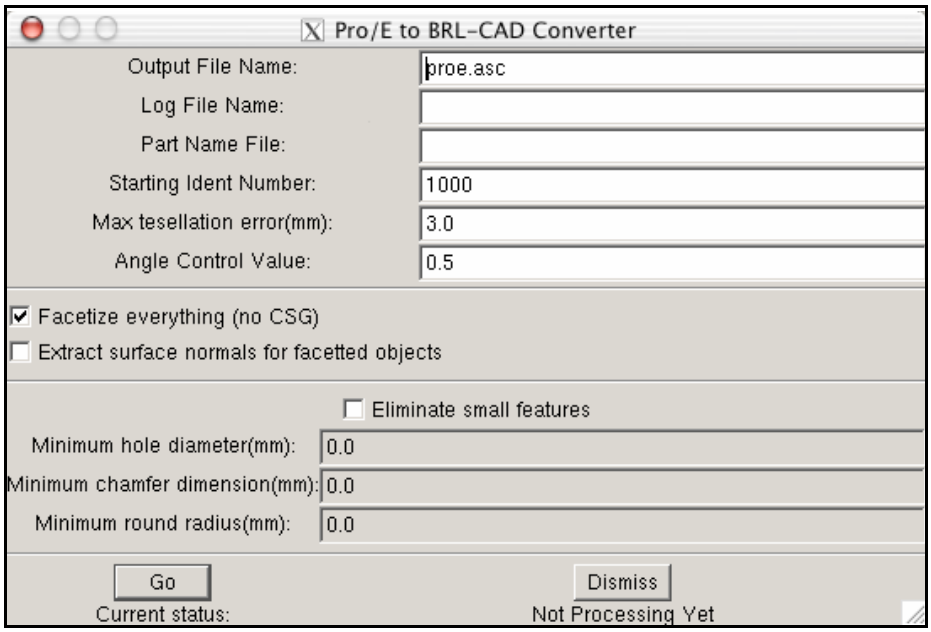

Figure 3. Pro/E to BRL-CAD converter dialog box.

The following list provides a description of the use and functionality of the primary elements in the dialog box. Note that the box is preloaded with reasonable defaults for many of the inputs. In addition, if the user has a question about any of the entry windows, check boxes, or buttons, he can move the mouse over them to see a brief explanation of their use.

- **Output File Name** - This is the name of the file to receive the ASCII output.
- **Log File Name** - If provided, verbose status logging will be written to that file.
- **Part Name File** - If provided, the converter will use that information to map Pro/E part and assembly names into the specified BRL-CAD object names. This file is simply a text file with each line specifying a Pro/E part or assembly name and a BRL-CAD object name separated by white space. Note that the BRL-CAD names should not include any special characters such as "/", "[", "]", or white space.
- **Starting Ident Number** - Ident numbers will be assigned to the resulting BRL-CAD regions sequentially starting with the number that appears in the window.
- **Max Tessellation Error** - This value (expressed in millimeters) is used to control the coarseness of the tessellation. It is the maximum distance between the actual surface and its tessellated approximation. Smaller values here will result in finer tessellations and more triangles. The default value is reasonable for ballistic vulnerability analysis purposes.
- **Angle Control Value** - This is a number between 0 and 1 that provides additional control over the coarseness of the tessellation. The exact relationship between the tessellation and this value is not specified in the Pro/E documentation, but the default value of 0.5 seems to work well.

For more information about the Pro/E file format and import possibilities, see the Parametric Technology Corporation (PTC) web site at http://www.ptc.com (PTC, 2003).

4.10 Converting From STL Format

The STL format was developed by 3D Systems, Inc., in the 1980s for use with its StereoLithography Apparatus (SLA). The SLA device produces a physical 3-D model based on an STL format file. Because of its simplicity, the STL format has become an industry standard for exchanging 3-D models. Unfortunately, this simplicity also presents some limitations.

The format consists only of triangles, and each triangle is represented by three vertices and a surface normal vector. Because the vertices for each triangle are explicitly listed, rather than indexed from a list, the topology must be inferred by the receiving system, which can sometimes lead to incorrect geometry.

STL files may be either ASCII or binary. The ASCII format includes the capability of including more than one solid part and an optional name for each part, while the binary format can only support a single solid part with no naming.

The **stl-g** converter converts STL format to BRL-CAD. The STL format is entirely triangles. The resulting BRL-CAD database will consist of one or more regions and a top-level combination named "all," which contains all the regions produced. Each region will consist of a single BOT primitive.

Note that the ASCII format STL file includes a capability to contain more than one solid part. The regions created will be named according to the name specified in the STL file unless a name is provided on the command line. If the STL file does not specify any name, and the user does not provide a name, then the regions produced in the BRL-CAD database will be constructed from the name of the STL file.

The syntax for the **stl-g** converter is as follows:

```
stl-g [options] input.stl output.g
```

where **input.stl** is the STL file to be converted and **output.g** is the name of the BRL-CAD database to receive the converted output.

The options for this command are as follows:

- **`b`** - designates that the input STL file is in binary format (the default is ASCII).
- **`c units`** - specifies the units used in the STL file. Choices include "cm," "m," "in," "ft," and many others (the default is millimeters).
- **`N name`** - specifies a name for the resulting BRL-CAD region. If more than one region is produced, unique region names will be constructed by adding a suffix consisting of an underscore and an integer.
- **`d`** - designates that additional debugging information be printed during the conversion.
- **`i ident`** - specifies the ident number assigned to the first region created during the conversion. Additional regions will be assigned sequential ident numbers.
- **`I ident`** - specifies the ident number to assign to all the BRL-CAD regions created during this conversion. (This option and the **i ident** option are mutually exclusive.)

- **`m material_code`** - specifies the integer material code to be assigned to each BRL-CAD region created during this conversion.
- **`t tolerance_distance`** - specifies the minimum distance (in millimeters) allowed between distinct vertices. Vertices closer than this minimum will be considered to be the same vertex (the default value is 0.005 mm).
- **`x librt_debug_flag`** - specifies a flag for the raytracing library that will result in additional debug log messages (see **librt/debug.h** in the binary distribution for details).

For more information on the STL file format and import possibilities, see the 3D Systems web site at http://www.3dsystems.com (3D Systems, 2003).

4.11 Converting From TANKILL Format

Distributed by the Advantage Business Group, a contractor to the British Ministry of Defence, the TANKILL format is used with the TANKILL vulnerability and lethality assessment code. This format is another purely triangulated representation of solid objects.

The syntax for the **tankill-g** converter is as follows:

```
tankill-g [options]
```

The options for the command are as follows:

- **`v`** - verbose; prints out progress information.
- **`n`** - produces NMG primitives (default is BOT primitives).
- **`i input.tkl`** - specifies the input TANKILL file (default is stdin).
- **`o output.g`** - specifies the output BRL-CAD file (default is tankill.g).
- **`k`** - keeps components with id = 1001 (normally skipped).
- **`x lvl`** - sets librt debug flag (see **raytrace.h** in the binary distribution for definitions of **DEBUG_xxx**).
- **`X lvl`** - sets NMG library debug flag (see **nmg.h** in the binary distribution for definitions of **DEBUG_xxx**).

For more information on the TANKILL file format and import possibilities, contact the Advantage Business Group at the Barbician, East Street, Farnham, Surrey GU9 7TB or visit the web site at http://www.advantage-business.co.uk (Advantage Business Group, 2003).

4.12 Converting From Unigraphics Format

Like Pro/E, Unigraphics is a widely popular CAD format used by thousands of companies in the United States and abroad, including General Motors, Ford, Kodak, General Electric, Pratt & Whitney, Boeing, and Samsung. Now distributed by EDS (the company that also distributes Jack and a version of NASTRAN), the Unigraphics toolset addresses traditional CAD/CAM/CAE, conceptual and industrial design, knowledge-based engineering, real-time design collaboration, and process automation (EDS, 2003c).

There are three modeling methodologies offered with Unigraphics. First, explicit (or traditional) modeling uses points, curves, and surfaces with no associativity or history. Second, history-based modeling uses associative geometric entities. Third, direct modeling uses a combination of both explicit and history-based modeling and also allows the global application of geometric rules and constraints across geometry of all origins (SURVICE Engineering Company, 2002).

Unigraphics bases its component geometric modeling capabilities on the Parasolid geometry engine (developed by EDS in Cambridge, England) and related XT file format. This enables Parasolid-based systems (e.g., Unigraphics, Solid Edge, and systems by Parametric Technology, SolidWorks, Bentley Systems, CADKEY, ANSYS, Mechanical Dynamics, and MSC.Software) to share and exchange geometric data without translation via an interoperable data pipeline (SURVICE Engineering Company, 2002).

Because the Unigraphics-to-BRL-CAD converter, **ug-g**, was written using the Unigraphics UG/Open API library, users must have a Unigraphics UG/Open execute or development license in order to run it. This converter is compiled for SGI Irix machines running on MIPS processors; however, users can compile it for other platforms by obtaining the BRL-CAD source distribution and a UG/Open development license from Unigraphics.

This converter creates a BRL-CAD region consisting of a single BOT primitive for each Unigraphics part and a combination for each Unigraphics assembly. Each instance of a Unigraphics part is converted independently, so there are no transformation matrices created in the resulting BRL-CAD model. The BRL-CAD regions are given the same name as the parts are assigned in the Unigraphics model, unless a part-name mapping file is provided. Region names are made unique, if necessary, by adding a suffix consisting of a dot and an integer number.

The syntax for this converter is as follows:

```
ug-g [options] o output.g UG_part_file [subpart1 subpart2 …]
```

where the **UG_part_file** is a Unigraphics part file. If subparts are listed on the command line, only those named parts in the specified part file will be converted.

The available options are as follows:

- **`d level`** - specifies a debug level for additional log messages. Currently, any nonzero value here provides additional logging.
- **`i initial_ident`** - specifies the ident number for the first region created by the conversion. Subsequent regions are assigned sequential ident numbers.
- **`n part_name_file`** - specifies a file containing a mapping of Unigraphics part and/or assembly names to BRL-CAD object names. If this file is provided, it will be used to create object names in the BRL-CAD model. The format of this file is simply a line per part, with the Unigraphics part/assembly name followed by the desired BRL-CAD object name, separated by white space. Note that the usual restrictions for BRL-CAD object names apply (e.g., no spaces, no special symbols, etc.) For a detailed discussion of

recommended naming schemes and restrictions, see section 4 of Volume III (Butler et al., 2003).

- **`R refset_name`** - specifies a desired reference set (which Unigraphics uses to provide additional control over assembly components), overriding the reference set specified in the Unigraphics model.
- **`f`** - facetizes all the geometry. If this option is not specified, the converter will attempt to create CSG equivalent geometry wherever possible. (*Note that this option is experimental.*)
- **`s`** - lists all features that were attempted to convert to CSG. (This option and the **f** option are mutually exclusive.)
- **`u`** - extracts surface normals from the Unigraphics model. Using this option will result in a BRL-CAD model that is significantly larger than when not using it, but raytraced objects will appear much smoother.
- **`o output.g`** - specifies the name of the file to receive the BRL-CAD model.
- **`t tolerance_distance`** - specifies the minimum distance (in millimeters) allowed between distinct vertices. Vertices closer than this minimum will be considered to be the same vertex (the default value is 0.005 mm).
- **`a surface_normal_tolerance`** - specifies a surface normal error tolerance (in degrees) for the facetization process. By default, the surface normal is not considered during facetization.
- **`c min_chamfer`** - specifies that any chamfer with dimensions less than the provided minimum (in millimeters) will be ignored. By default, no features are ignored.
- **`r min_round`** - specifies that any round (or fillet) with a radius less than the specified minimum (in millimeters) will be ignored. By default, no features are ignored.

For more information on the Unigraphics file format and conversion potential, see the EDS web site at http://www.eds.com/products/plm/unigraphics_nx/ (EDS, 2003c).

4.13 Converting From Viewpoint Datalabs Format

Viewpoint Datalabs started out as a commercial supplier of 3-D models, maintaining a large repository of facetted models of many objects. The company has since grown to provide more services than models, and its model repository is now maintained by Digimation, Inc.

The **viewpoint-g** converter converts the Viewpoint Datalabs coor/elem format to BRL-CAD format. Objects in the input files are converted to regions, each consisting of a single BOT primitive. The converter will assign vertex normals if they are present in the input files. Two files are expected, one that contains vertex coordinates (and optional normals) and one that lists the vertex numbers for each polygonal face. This format was used by the original Viewpoint Datalabs model repository. The current repository uses more common formats such as DXF and VRML.

The syntax for the converter is as follows:

```
viewpoint-g [options]
```

The options for the command are as follows:

- **`c coord_file_name`** - sets the input vertex coordinates file name (required).
- **`e elements_file_name`** - sets the input faces file name (required).
- **`o output_file_name`** - sets the output BRL-CAD database name (default is viewpoint.g).
- **`t tolerance_distance`** - sets distance calculation tolerance (in millimeters) (default is 0.005).

For more information about the Datalabs format and conversion potential, see the Digimation web site at http://www.digimation.com (Digimation, 2003).

5. Converting From BRL-CAD

5.1 General Information About Converting From BRL-CAD

Table 3 contains the primary file formats (other than ASCII) to which BRL-CAD currently converts. Once again, the "g-" designation at the beginning of each converter name indicates that the conversion process begins with BRL-CAD's standard geometry (.g) format.

Note that most of the BRL-CAD export converters operate by tessellating all the primitive shapes in the BRL-CAD model, then evaluating the Boolean formula for each region. The result of this is a facetted representation of each region. The tessellation tolerances are used to determine the coarseness of the tessellation of curved surfaces. Smaller tolerances result in more, smaller facets and a larger output file. The calculation tolerance is the minimum distance between any two vertices in a region. Any vertices closer than this are fused into a single vertex.

Table 3. BRL-CAD export converters.

CAD Format	Export Converter
ACAD	g-acad
AutoCAD DXF	g-dxf
EUCLID [a]	g-euclid
IGES	g-iges
Jack	g-jack
STL	g-stl
TANKILL	g-tankill
VRML	g-vrml
Wavefront (.obj)	g-wave
X3D	g-x3d

[a] Conversion is to the ASCII EUCLID decoded format, not native format.

Occasionally, the Boolean evaluation process will fail. When this happens, an error message specifying the region that failed to convert will be generated. The converter will then proceed to work on the next region to be converted.

5.2 Converting to ACAD Format

The Advanced Computer Aided Design (ACAD) format is the format for the in-house CAD package developed in the early 1980s by Lockheed Martin Tactical Aircraft Systems (then General Dynamics - Fort Worth) and distributed since 1995 by the Electromagnetic Code Consortium. The package was developed to improve the product design process by eliminating manual drawing boards and reducing design iteration cycle times. Notable aircraft programs that have used ACAD include the Joint Strike Fighter (F-35), the Raptor (F-22), the Advanced Technology Fighter, the National Aerospace Plane (X-30), and the Attack/Fighter-Experimental (A/F-X) (SURVICE Engineering Company, 2002).

A notable characteristic of ACAD is the layers by which modeled parts are separated and managed (e.g., by subsystem). The package stores geometric data in a relational database that uses parent-child dependencies to enable automatic and rapid geometric modification. For example, changing a control spline of a fuselage will automatically regenerate any surface(s) built with that spline. Any geometry that is associated to the fuselage surface (i.e., plane/curve and surface intersections, fillets, etc.) will then regenerate.

One of ACAD's strengths is its bidirectional IGES translator, which provides compatibility with other IGES-supported formats, including CADAM, CATIA, COMPUTERVISION, AutoCAD, and Pro/E.

Is it ACAD or AutoCAD?

Although these packages are often confused, ACAD was developed by Lockheed Martin and is not connected with AutoCAD, which was developed by Autodesk Inc.

As its name implies, the **g-acad** utility converts a file in BRL-CAD format to a file in facetted ACAD format. The form of the command is as follows:

```
g-acad [options] –o output_file input.g object(s)
```

The options for the **g-acad** command are as follows:

- **`i`** - designates that the output be in inches (the default is millimeters).
- **`x lvl`** - sets **librt** debug flag (see **raytrace.h** in the binary distribution for definitions of **DEBUG_xxx**).
- **`X lvl`** - sets the NMG library debug flag (see **nmg.h** in the binary distribution for definitions of **DEBUG_xxx**).
- **`e error_file`** - sends error messages to specified file (default is standard error [stderr]).
- **`v`** - verbose; prints out progress information.

- **`D tolerance_distance`** - sets distance calculation tolerance (in millimeters) (default is 0.005).
- **`r rel_tess_tol`** - sets relative tessellation tolerance (normally 0.0 to 1.0) (default is 0.1).
- **`a abs_tess_tol`** - sets absolute tessellation tolerance to specified value (in millimeters) (overrides **`r`** option).
- **`n norm_tess_tol`** - sets surface normal tessellation tolerance (angle in radians) (overrides **`r`** option).

For more information on the ACAD file format and conversion potential, contact Lockheed Martin Tactical Aircraft Systems at http://www.lockheedmartin.com (Lockheed Martin, 2003).

5.3 Converting to AutoCAD DXF Format

The **g-dxf** command converts BRL-CAD objects to the previously mentioned AutoCAD DXF format. The syntax for this command is as follows:

```
g-dxf [options] input.g object(s)
```

The options for the **g-dxf** command are as follows:

- **`i`** - requests the output DXF file to be in inches (default is millimeters).
- **`o output.dxf`** - specifies the file to receive the DXF output (default is stdout).
- **`p`** - requests that the output DXF file consist of POLYFACE MESH entities (the default is 3DFACE entities).

The command also accepts the **v**, **r**, **a**, **n**, **x**, and **X** options, which have been discussed in the import converters portion of this document (section 4).

For more information on the DXF file format and export possibilities, see the on-line documentation on the Autodesk web site at http://www.autodesk.com (Autodesk, Inc., 2003).

5.4 Converting to EUCLID Format

The **g-euclid** command converts BRL-CAD objects to a EUCLID "decoded" facetted format. Note that, as previously mentioned, this format is not native EUCLID format. The form of the command is as follows:

```
g-euclid [options] input.g objects(s)
```

The options for the **g-euclid** command are as follows:

- **`u units`** - sets the output units (the default is millimeters).
- **`o output_file_name`** - sets the output file name (the default is stdout).

For more information about the EUCLID file format and export possibilities, see the MDTVISION web site at http://support.mdtvision.com (MDTVISION, 2003).

5.5 Converting to IGES Format

The syntax for the BRL-CAD to IGES converter is as follows:

```
g-iges [options] input.g object(s)
```

The options for the **g-iges** command are as follows:

- **o output_name** - indicates the name of the resulting IGES file. If the **m** option is specified, then this is the name of a directory where resulting IGES files will be placed (the directory must already exist). (The default is stdout.)
- **f** - designates that the resulting IGES file will be entirely facetted BREP entities. The **f** and **t** options (the next option discussed) are mutually exclusive.
- **t** - designates that the resulting IGES file will be entirely trimmed NURB entities similar to the **f** option, but each facet is represented as a trimmed NURB. The **f** and **t** options are mutually exclusive.
- **m** - Each region in the specified **objects** is output in an individual IGES file and placed in the directory specified with the **o** option. This option implies the **t** option.
- **s** - Used in conjunction with the **f** or **t** options to indicate that the facets should all be expressed as planar NURB surfaces rather than the default simple planar surface.

With the **t** and **f** options, the converter will employ Boolean evaluation of each region. If neither option is specified, then a CSG IGES file will be produced. The command also accepts the previously mentioned **v**, **r**, **a**, **n**, **x**, and **X** options.

Package-Preferred IGES Converter Options

Note that different CAD packages seem to "prefer" certain options for the IGES converter. In particular, Pro/E works well with IGES files that use the **m** option, and Unigraphics works well with IGES files that use the **s** option.

For more information about the IGES file format and export possibilities, see the NIST web site at http://www.nist.gov/iges (NIST, 2003).

5.6 Converting to Jack

The syntax for the **jack-g** converter is as follows:

jack-g [options] input.jack output.g

The options for the **jack-g** command are as follows:

- **r region_name** - specifies the name of the region to create in the BRL-CAD output (by default, this name is constructed from the input file name).
- **g combination_name** - specifies the name of a combination to be created to hold the region produced. If not specified, no combination will be created.

For more information about the Jack file format and export possibilities, see the EDS web page at http://www.eds.com/products/plm/efactory/jack/ (EDS, 2003a).

5.7 Converting to STL Format

The syntax for the BRL-CAD to STL converter is as follows:

```
g-stl [options] input.g object(s)
```

The options for the **g-stl** command are as follows:

- **`o output_name`** - specifies the name of the file to receive the STL format output. This option is mutually exclusive with the **m** option. (The default is stdout.)
- **`m directory_name`** - specifies the name of an existing directory where output STL files will be placed. This option is mutually exclusive with the **o** option. When this option is exercised, each region in the specified **object(s)** is output in a separate file written to the specified directory. The file names will be based on the BRL-CAD database *path* to the region, with "/" characters replaced by "@" and "." characters replaced by "_".
- **`b`** - produces binary format STL files. The combination of this option and the **o** option produces a binary STL file containing one solid object representing all the regions in the specified **object(s)**. (The default output format is ASCII.)
- **`D calculation_tolerance`** - sets the distance calculation tolerance (in millimeters) (the default is 0.005).
- **`i`** - produces an STL file in units of inches (the default is millimeters).

The command also accepts the previously mentioned **v**, **r**, **a**, **n**, **x**, and **X** options.

For more information on the STL file format and export possibilities, see the 3D Systems web site at http://www.3dsystems.com (3D Systems, 2003).

5.8 Converting to TANKILL Format

As discussed previously, the TANKILL format is another purely triangulated representation of solid objects.

The syntax for the **g-tankill** command is as follows:

```
g-tankill [options] input.g object(s)
```

The options for the **g-tankill** command are as follows:

- **`i idents_file`** - specifies that the ident numbers in the output file should be assigned sequentially as BRL-CAD regions are encountered (rather than using the ident numbers assigned in the BRL-CAD file). A mapping of the assigned ident numbers and the BRL-CAD regions is written to the specified **idents_file**.
- **`s surroundings_code`** - specifies the surroundings code, which is a code TANKILL uses to indicate the type of volume that surrounds an object (the default is 1000).
- **`o output_name`** - specifies the name of a file to receive the output (default it stdout).

The command also accepts the previously mentioned **v**, **r**, **a**, **n**, **x**, and **X** options.

For more information on the TANKILL file format and export possibilities, contact the Advantage Business Group at the Barbician, East Street, Farnham, Surrey GU9 7TB or visit the web site at http://www.advantage-business.co.uk (Advantage Business Group, 2003).

5.9 Converting to VRML Format

The Virtual Reality Modeling Language (VRML) began in 1994 at the first World Wide Web Conference. The format—which is maintained by the nonprofit, vendor-neutral Web3D Consortium—was designed to be a Web-interoperable standard for specifying multiparticipant 3-D graphical visualization tools. When a user selects a hyperlink to a VRML document from within a correctly configured WWW browser, a VRML viewer is launched to navigate through a given geometry (e.g., a model of a building). Future versions of VRML are planned to include more advanced features, including animations, motion physics, and real-time multiuser interaction (Web3D Consortium, 2003b).

The **g-vrml** command converts BRL-CAD objects to the VRML 2.0 format. It takes the following form:

```
g-vrml [options] input.g object(s)
```

The options for the command are as follows:

- **`d tolerance_distance`** - sets distance calculation tolerance in millimeters (the default is 0.005).
- **`u units`** - sets the desired output units (the default is millimeters).
- **`o output_file_name`** - sets output file name (the default is stdout).

The command also accepts the previously mentioned **v**, **r**, **a**, **n**, **x**, and **X** options.

For more information on VRML, see the Web3D Consortium web site at http://www.web3d.org /fs_technicalinfo.htm (Web3D Consortium, 2003b).

The Web3D Consortium

The Web3D Consortium is a nonprofit organization promoting open standards for 3-D Web and broadcast applications. Members include leading corporate and educational institutions, including 3Dlabs, ATI Technologies, blaxxun, Nexternet, OpenWorlds, Sony, SGDL Systems, ParallelGraphics, the Naval Postgraduate School, and the Army Simulation, Training, and Instrumentation Command (STRICOM). Notable efforts include developing VRML specifications and an improved, componentized new-generation specification for Web3D, including X3D (see subsection 5.11) (SURVICE Engineering Company, 2002).

5.10 Converting to Wavefront Format

The Wavefront **OBJ** format was developed in 1984 by Wavefront Technologies (now Alias, a company that merged with Wavefront under SGI). The format, which was intended for use with the Wavefront Advanced Visualizer, consists of lines, polygons, and smooth surfaces.

This **g-wave** converter produces an **OBJ** file that consists entirely of polygons (triangles). The command takes the following form:

```
g-wave [options] input.g object(s)
```

The options for the command are as follows:

- **`m`** - includes "usemtl" statements in the output (encoded aircode, los, and material number).
- **`i`** - sets the output units to inches (the default is millimeters).
- **`d tolerance_distance`** - sets distance calculation tolerance in millimeters (the default is 0.005).
- **`u`** - includes vertexuse normals in the output.
- **`o output.obj`** - specifies the output file name (default is stdout).

The command also accepts the previously mentioned **v**, **r**, **a**, **n**, **x**, and **X** options.

For more information on the Wavefront file format and export possibilities, see the Alias web site at http://www.alias.com/eng/index.shtml (Alias, 2003).

5.11 Converting to X3D Format

The successor to the Web3D Consortium's VRML format, the Extensible 3D (X3D) format was designed to be a standard for interactive web- and broadcast-based 3-D content. Its intended uses include engineering and scientific visualization, multimedia presentations, entertainment and educational applications, web pages, and shared virtual worlds. It is also intended to serve as a universal interchange format for integrated 3-D graphics and multimedia. X3D possesses

several improvements over VRML, including advanced APIs, additional data encoding formats, stricter conformance, and a componentized architecture that allows for a modular approach (Web3D Consortium, 2003a).

The **g-x3d** command converts BRL-CAD objects to the X3D format. It takes the following form:

```
g-x3d [options] input.g object(s)
```

The options for the command are as follows:

- **`d tolerance_distance`** - sets distance calculation tolerance in millimeters (the default is 0.005).
- **`u units`** - sets the desired output units (the default is millimeters).
- **`o output_file_name`** - sets output file name (the default is stdout).

The command also accepts the previously mentioned **v**, **r**, **a**, **n**, **x**, and **X** options.

For more information on the X3D file format and export possibilities, see the Web3D Consortium web site at http://www.web3d.org/technicalinfo/specifications/ISO_IEC_19775 (Web3D Consortium, 2003a).

---

6. Building a New Converter

6.1 General Information About Building a New Converter

If the previously mentioned import/export converters do not meet the user's needs, a custom converter can be built. The effort required to accomplish this, of course, depends on the user's coding ability, the intended model use, and the available resources. If one needs to convert from a simple ASCII format consisting entirely of triangles to BRL-CAD, a few days of effort will probably suffice. On the other hand, if one wants to develop a converter from a proprietary commercial CAD system to BRL-CAD, several months of effort and training may be required.

For example, Pro/E, Unigraphics, and similar commercial CAD systems provide libraries and APIs that a developer can use to access their models. Typically, training is required in order to understand what these libraries can do and how they work, and training in the use of the CAD system itself may be required as a prerequisite to understand the package environment and terminology. The purchase of expensive licenses is also a typical requirement, and, in some cases, the use of the code developed will require the purchase of another license.

But before one begins developing a new converter, one should first consider the possibility of using combinations of existing converters. For example, if one needs to convert from a commercial CAD system to BRL-CAD and no converter exists for that system, he might consider importing his model into Pro/E or Unigraphics and converting to BRL-CAD from there.

Understandably, this may result in the loss of some information (and repeated conversions usually mean greater losses), but restoring that information manually may be more cost effective than building a new converter from scratch. Another possibility might be to convert the geometry to STL format and then convert the STL to BRL-CAD using **stl-g** (see section 4.10).

6.2 Converting From BRL-CAD

Converters from BRL-CAD to another format generally come in one of two varieties. The first type converts BRL-CAD objects to their CSG counterparts in the receiving system. This type of conversion rarely happens as most commercial CAD systems have extremely limited CSG capabilities. If this is the case, however, the converter merely needs to visit every object in the BRL-CAD model and output it in the form required by the receiving system. A sample code (named **g-xxx.c**) to accomplish this is included in the **conv** directory in the BRL-CAD source distribution. The only missing piece of this code is the actual output of the receiving format.

The more common type of conversion is when the desired output format is not CSG. Currently, the best choice in this situation is to produce a triangular facetted approximation of the BRL-CAD regions and output that approximation in the receiving format. A sample code (named **g-xxx_facets.c**) for this approach is also included in the **conv** directory in the BRL-CAD source distribution. Again, only the part where the facets are output in the receiving format is missing.

The sample code performs Boolean evaluation of the regions using tessellated versions of the CSG primitives. This technique is not always successful, so the sample code catches the failures, notifies the user, and continues when a failure occurs.

6.3 Converting to BRL-CAD

A similar situation exists for converting to BRL-CAD. If one is fortunate enough to be converting from a CSG system, then just converting everything to its corresponding BRL-CAD representation is all that is required. The **libwdb** library distributed with BRL-CAD contains all the routines needed to build primitives, regions, and combinations in BRL-CAD.

To create an empty BRL-CAD model, use the following code snippet:

```
struct rt_wdb *wdbp;
if( (wdbp=wdb_fopen( new_file_name ) ) == RT_WDB_NULL ) {
bu_log( “Failed to open output file (%s)\n”, new_file_name );
exit( 1 )
}
```

Then use the **wdbp** pointer to pass to all the routines in **libwdb**.

For example, to build a sphere centered at coordinates (1.0, 2.0, and 3.0) with a radius of 5.0 mm in BRL-CAD, the following call to **mk_sphere** will suffice:

```
char *sphere_name=”s1”;
point_t center;
fastf_t radius;
```

```
/* set the center and radius values */
VSET( center, 1, 2, 3 );
radius = 5.0;

/* actually build the sphere */
if( mk_sph( wdbp, sphere_name, center, radius ) ) {
bu_log( "Failed to make a sphere (%s)\n", sphere_name );
exit( 1 );
}
```

To build a region or combination in BRL-CAD, a call to **mk_comb** will do the job. This is a bit more complicated as it requires building a list of members prior to the call to **mk_comb**. In the example that follows, a region consisting of primitive "s2" subtracted from primitive "s1" is constructed:

```
struct bu_list head;
struct wmember *wmp;

/* initialize the list of members for this combination */
BU_LIST_INIT( &head );

/* add the first member (use the UNION operator here) */
if( (wmp=mk_addmember( "s1", &headp, NULL, WMOP_UNION) ) == WMEMBER_NULL
) {
bu_log( "Failed to add member(s1)\n" );
exit( 1 );
}

/* subtract the second member */
if( (wmp=mk_addmember( "s2", &headp, NULL, WMOP_SUBTRACT) ) ==
WMEMBER_NULL ) {
bu_log( "Failed to add member(s2)\n" );
exit( 1 );
}

/* build the actual region */q
if( mk_comb( wdbp, "region_name", &head, 1, NULL, NULL, NULL, 1000, 0,
1, 100, 0, 0, 0 ) ) {
bu_log( "Error building region\n" );
exit( 1 );
}
```

If the system you are converting from is not a CSG system (which is the most likely case), then you will need to do some processing on the originating system to get the data in a form that BRL-CAD can use. If you have the API module for the originating system, you may be able to use one of those routines to create a triangulated representation of the model. This triangle data can then be used to build BOT primitives in the BRL-CAD model using the **mk_bot** routine. Each BOT primitive can be placed in a region of its own (as shown previously).

A hybrid approach is also possible where simple shapes that can be identified as BRL-CAD primitive shapes are converted as BRL-CAD primitives, and everything else is converted as BOT primitives.

If the converter requires building BOT primitives, you might want to consider a converter that produces a BRL-CAD ASCII format database. The "C" **mk_bot** routine is a very thin wrapper for the actual internal format of the BOT primitive and requires an intimate knowledge of its

structure. The ASCII format is considerably simpler and easier to work with in this situation. As mentioned previously, the current ASCII format is simply a Tcl script. For example, the following is the ASCII form of a BRL-CAD database containing a single BOT:

```
title {example of a simple BOT primitive}
units mm
put {tetra} bot mode volume orient rh flags {} V { { 0 0 1000 } { -500 0
-1000 } { -500 -500 -1000 } { 500 -500 -1000 }} F { { 0 1 3 } { 0 1 2 }
{ 0 2 3 } { 1 2 3 }}
```

The aforementioned example creates a BRL-CAD database with the title of "example of a simple BOT primitive," with preferred units of millimeters and a single tetrahedron-shaped BOT primitive. The line that begins with "put" creates the BOT primitive (note that there are actually only three lines in this example; the last line is wrapped in this report for clarity). This line is a command to the Tcl interpreter. The "put" is the command, "tetra" is the name of the object to be created, and "bot" is the type of object to create. The remainder of the line consists of key/value pairs. This line creates a BOT primitive named "tetra," using volume mode (meaning this BOT encloses space), with right-hand (rh) rule triangle orientation, and no flags.

The **V** key introduces the list of vertices for this BOT primitive, and the **F** key introduces the list of triangle faces (each integer is an index into the list of vertices). Additional **put** commands build more objects in the database. For example, to create a region using the **tetra** object, the user would append the following line to the aforementioned file:

```
put {aregion} comb region yes tree {l tetra}
```

This line creates a BRL-CAD combination named **aregion**, which is a region and consists only of the *tetra* object created earlier. The resulting file is converted to BRL-CAD binary format using **asc2g**.

For an example of code that produces this type of output, see the **g2asc.c** file in the **conv** directory of the source distribution.

7. Postconversion Issues

Before converted geometry can be used in a given application or analysis (e.g., a vulnerability study), there are several tasks that often need to be performed. They include the following:

- **Modeling Objects That Failed to Convert** – Even in the best conversion processes, there can be individual objects that, for some reason, do not successfully convert to or from the BRL-CAD format. When this happens, the converter reports an error message, and the modeler/analyst must either manually rebuild the objects or—if they are not critical to the model—omit them (see the following bulleted item).

- **Remodeling or Eliminating Excessively Detailed, Complex, or Unnecessary Objects** - In some cases, the problem with converted geometry is not about objects that failed to convert but rather about objects that did convert. For example, highly detailed objects such as screens, splined grids, and geometrically complex exterior components—which are often not needed for vulnerability studies—may unnecessarily enlarge a model's file size and slow its performance speed. Thus, the modeler/analyst has to check for these objects and either remodel them in a simpler manner or eliminate them altogether.
- **Modeling Air** - Some applications that evaluate ballistic penetration, fire, vaporific effects, etc., require that the interior space within a given geometry contain continuous regions of air. Therefore, it is sometimes necessary to take a converted model and model various air compartments within them (e.g., the crew, passenger, and engine compartments). In addition, in cases where one air region contacts another (e.g., in an open turret hatch, a gun turret, or an exhaust grill), a thin layer of "phantom armor" is sometimes needed to separate the two. For more information about air components and codes, see Winner et al. (2002).
- **Assigning Material Codes and Line-of-Sight (LOS) Densities** - To more realistically approximate actual material weights and properties, modelers often have to assign material codes to converted geometry regions. In addition, if a region is not solid material, a density percentage is assigned. For a list of standardized material codes and densities, see Winner et al. (2002).
- **Assigning Optical Shaders and Colors** - Although most engineering analysis applications focus on what geometry is made of, rather than what it looks like, it is often necessary for viewing, publication, or presentation purposes to assign visual properties to converted objects in order to better simulate material characteristics (e.g., mirrored or transparent effects) or to match standard system colors (e.g., electrical systems are forest green [50 145 20]). For a list of standardized system color values, see Volume III of this tutorial series (Butler et al., 2003) or Winner et al. (2002). For more information on coloring and shading in MGED, see Volume II of this tutorial series (Butler et al., 2001).
- **Assigning Component Identification Numbers** - Some vulnerability interrogation applications rely on components being grouped into common regions. For example, the tank track ident number needs to be assigned to each track link region.
- **Assigning Meaningful Names to the Converted Objects** - As mentioned previously, some packages (e.g., Pro/E) typically label their geometry with part numbers instead of part names. Although this practice can be effective for manufacturing and other production purposes, when the objects are converted, the modeler/analyst often has to assign meaningful part names so that the model can be more easily organized, analyzed, and manipulated. Accordingly, it is important that complete mappings of part numbers to part names are supplied during the conversion process.
- **Combining Multiple Converted Files Into a Single BRL-CAD Model** - Files from packages such as Pro/E and Unigraphics are often so large that entire models cannot be loaded all at once, and thus the BRL-CAD import converters for these formats must convert the geometry file by file and system by system. As a result, the BRL-CAD modeler/analyst often has to combine all the files of converted geometry so that they can be displayed and evaluated as a single, integrated model (see the discussion on **dbconcat** in Volume III of this tutorial series [Butler et al., 2003]).

- **Combining or Breaking Down Regions to Match Analyst Requirements** - In some cases, converted geometry does not match the "grouping" requirements of an analyst or application and therefore must be regrouped. For example, the converted objects of an engine might need to be grouped into a single engine region. Conversely, a single engine region might need to be broken down into separate regions.
- **Reorganizing or Creating a Model Hierarchy** - Some geometry (e.g., an STL file) is converted with little to no tree structure or organization. In such cases, the modeler/analyst often needs to set up a basic tree hierarchy of parts and assemblies so that the model can be more easily handled. For more information about model structuring, see Volume III of this tutorial series (Butler et al., 2003).

8. References

3D Systems, Inc. http://www.3dsystems.com (accessed October 2003).

Advantage Business Group. http://www.advantage-business.co.uk/ (accessed October 2003).

Alias. http://www.alias.com/eng/index.shtml (accessed October 2003).

Autodesk, Inc. AutoCAD 2004 DXF. http://adeskftp.autodesk.com/prodsupp/downloads /dxf.pdf (accessed February 2003).

Butler, L. A.; Edwards, E. W. *BRL-CAD Tutorial Series: Volume I – Overview and Installation;* ARL-SR-113; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, February 2002.

Butler, L. A.; Edwards, E. W.; Schueler, B. J.; Parker, R. G.; Anderson, J. R. *BRL-CAD Tutorial Series: Volume II – Introduction to MGED*; ARL-SR-102; U.S Army Research Laboratory: Aberdeen Proving Ground, MD, April 2001.

Butler, L. A.; Edwards, E. W.; Kregel, D. L. *BRL-CAD Tutorial Series: Volume III – Principles of Effective Modeling*; ARL-SR-119; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, September 2003.

Butler, L. A.; Muuss, M. J.; Tanenbaum, P. J.; Anderson, J. R.; Parker, R. G.; Bowers, R. A.; Johnson, C. T.; Edwards, E. W. *New BRL-CAD Database Format, Version 5, Draft*, U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, December 2003.

Digimation, Inc. http://www.digimation.com (accessed October 2003).

Electronic Data Systems (EDS). Jack. http://www.eds.com/products/plm/efactory/jack/ (accessed October 2003a).

Electronic Data Systems (EDS). NX Nastran: Overview. http://www.eds.com/products /plm/nastran/ (accessed October 2003b).

Electronic Data Systems (EDS). Unigraphics NX: Overview. http://www.eds.com/products/plm/unigraphics_nx/ (accessed October 2003c).

Elysium Inc. http://www.elysiuminc.com (accessed October 2003).

Lockheed Martin. http://www.lockheedmartin.com (accessed November 2003).

Muus, M. J.; Butler, L. A. Combinatorial Solid Geometry, Boundary Representations, and Non-Manifold Geometry. Paper published in *State of the Art in Computer Graphics*: *Visualization and Modeling*, Rogers, D. F., Earnshaw, R. A., Eds.; Springer-Verlag: New York, 1991, pp 185–223.

MSC.Software Corporation. http://www.mscsoftware.com (accessed October 2003).

MDTVISION. http://support.mdtvision.com (accessed December 2003).

National Institute of Standards and Technology (NIST). http://www.nist.gov/iges (accessed October 2003a).

Parametric Technology Corporation (PTC). http://www.ptc.com (accessed October 2003).

Raines, P.; Tranter, J. *TCL/TK in a Nutshell.* Sebastopol, CA: O'Reilly & Associates, Inc., March 1999.

SURVICE Engineering Company. *Computerized Geometric Information to Support Vulnerability Assessments State-of-the-Art Report (SOAR)*; SURVICE-TR-02-009; prepared for the Survivability/Vulnerability Information Analysis Center, Wright-Patterson AFB, OH, November 2002.

Survivability/Vulnerability Information Analysis Center (SURVIAC). FASTGEN 4. http://www.bahdayton.com/surviac/fastgen.htm (accessed October 2003).

Tanenbaum, P. J. Geometric Modeling for SMART: One Size Fits All? *Presentation at the 2001 SMART Conference*, Salt Lake City, UT, 18 April 2001.

The University of Pennsylvania. http://www.cis.upenn.edu/~hms/jack.html (accessed 2003).

U.S. Army Research Laboratory. http://ftp.arl.army.mil/brlcad (accessed October 2003).

Web3D Consortium. Information Technology — Computer Graphics and Image Processing — Extensible 3D (X3D). ISO/IEC 19775: 200x, X3D. http://www.web3d.org/technicalinfo /specifications/ISO_IEC_19775 (accessed October 2003a).

Web3D Consortium. The Virtual Reality Modeling Language. Version 1.0 Specification. http://www.web3d.org/fs_technicalinfo.htm (accessed September 2003b).

Weiler, K. The Radial Edge Structure: A Topological Representation for NonManifold Geometric Modeling. Paper published in *Geometric Modeling for CAD Applications*, Springer-Verlag: New York, 1987.

Welch, B. *Practical Programming in Tcl and Tk*, Third Edition; Prentice Hall: Upper Saddle River, NJ, 1999.

Winner, W. A., et al. *Target Description Standards for Ballistic Survivability, Lethality, and Vulnerability Analyses of Ground Mobile Vehicles and Aircraft*; Special report (draft), U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, November 2002.

NO. OF COPIES ORGANIZATION

1 (PDF Only) DEFENSE TECHNICAL INFORMATION CENTER
DTIC OCA
8725 JOHN J KINGMAN RD
STE 0944
FT BELVOIR VA 22060-6218

1 COMMANDING GENERAL
US ARMY MATERIEL CMD
AMCRDA TF
5001 EISENHOWER AVE
ALEXANDRIA VA 22333-0001

1 INST FOR ADVNCD TCHNLGY
THE UNIV OF TEXAS
AT AUSTIN
3925 W BRAKER LN STE 400
AUSTIN TX 78759-5316

1 US MILITARY ACADEMY
MATH SCI CTR EXCELLENCE
MADN MATH
THAYER HALL
WEST POINT NY 10996-1786

1 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL CS IS R
2800 POWDER MILL RD
ADELPHI MD 20783-1197

3 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL CI OK TL
2800 POWDER MILL RD
ADELPHI MD 20783-1197

3 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL CS IS T
2800 POWDER MILL RD
ADELPHI MD 20783-1197

NO. OF COPIES ORGANIZATION

ABERDEEN PROVING GROUND

1 DIR USARL
AMSRD ARL CI OK TP (BLDG 4600)

NO. OF COPIES ORGANIZATION

1 OASD C3I
J BUCHHEISTER
RM 3D174
6000 DEFENSE PENTAGON
WASHINGTON DC 20301-6000

1 OUSD(AT)/S&T AIR WARFARE
R MUTZELBURG
RM 3E139
3090 DEFENSE PENTAGON
WASHINGTON DC 20301-3090

1 OUSD(AT)/S&T LAND WARFARE
A VIILU
RM 3B1060
3090 DEFENSE PENTAGON
WASHINGTON DC 20310-3090

1 UNDER SECY OF THE ARMY
DUSA OR
RM 2E660
102 ARMY PENTAGON
WASHINGTON DC 20310 0102

1 ASST SECY ARMY
ACQSTN LOGISTICS & TECH
SAAL ZP RM 2E661
103 ARMY PENTAGON
WASHINGTON DC 20310 0103

1 ASST SECY ARMY
ACQSTN LOGISTICS & TECH
SAAL ZS RM 3E448
103 ARMY PENTAGON
WASHINGTON DC 20310 0103

1 DIRECTOR FORCE DEV
DAPR FDZ
RM 3A522
460 ARMY PENTAGON
WASHINGTON DC 20310 0460

1 US ARMY TRADOC ANL CTR
ATRC W
A KEINTZ
WSMR NM 88002-5502

1 USARL
AMSRD ARL SL M
J PALOMO
WSMR NM 88002 5513

NO. OF COPIES ORGANIZATION

1 USARL
AMSRD ARL SL EA
R FLORES
WSMR NM 88002 5513

1 USARL
AMSRD ARL SL EI
J NOWAK
FT MONMOUTH NJ 07703-5601

ABERDEEN PROVING GROUND

1 US ARMY DEV TEST COM
CSTE DTC TT T
APG MD 21005 5055

1 US ARMY EVALUATION CTR
CSTE AEC SVE
R BOWEN
4120 SUSQUEHANNA AVE
APG MD 21005-3013

1 US ARMY EVALUATION CTR
CSTE AEC SVE S
R POLIMADEI
4120 SUSQUEHANNA AVE
APG MD 21005-3013

1 US ARMY EVALUATION CTR
CSTE AEC SV L
R LAUGHMAN
4120 SUSQUEHANNA AVE
APG MD 21005-3013

13 DIR USARL
AMSRD ARL SL
J BEILFUSS
P DEITZ
AMSRD ARL SL B
J FRANZ
M PERRY
P TANENBAUM
AMSRD ARL SL BB
D BELY
D FARENWALD
S JUARASCIO
M RITONDO
AMSRD ARL SL BD
R GROTE

NO. OF
COPIES ORGANIZATION

AMSRD ARL SL BE
L ROACH
AMSRD ARL SL E
M STARKS
AMSRD ARL SL EC
J FEENEY
E PANUSKA

NO. OF
COPIES ORGANIZATION

1 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL D
J MILLER
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL CI AP
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL SL
J PALOMO
WSMR NM 88002-5513

25 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL SL E (25 CPS)
WSMR NM 88002-5513

1 NAWC
WEAPONS DIVISION
CODE 418300D A WEARNER
BLDG 91073
1 ADMIN CIR
CHINA LAKE CA 93555-6100

1 AIR FORCE ARMAMENT CNTR
AAC/ENMA
D MCCOWN
101 W EGLIN BLVD
EGLIN AFB FL 32542-5549

1 USAF
46 OG OGMLV
B THORN
104 CHEROKEE AVE
EGLIN AFB FL 32542-5600

1 USAF WRIGHT LAB
46TH OG OGM AL AC
M LENTZ
2700 D STREET BLDG 22B
WRIGHT PAT AFB OH
45433-7605

NO. OF
COPIES ORGANIZATION

1 SURVIAC
ABERDEEN SATELLITE OFC
A LAGRANGE
4695 MILLENNIUM DR
BELCAMP MD 21017-1505

6 THE SURVICE ENGNRG CO
E EDWARDS
D KREGEL
C BOYER
M HARDIN
M BUTKIEWICZ
L MCKAY
4695 MILLENNIUM DR
BELCAMP MD 21017-1505

ABERDEEN PROVING GROUND

223 DIR USARL
AMSRD ARL SL
C HARDIN
AMSRD ARL SL E
D BAYLOR
AMSRD ARL SL EC
AMSRD ARL SL B (5 CPS)
AMSRD ARL SL BB (75 CPS)
AMSRD ARL SL BD (15 CPS)
AMSRD ARL SL BE (25 CPS)
J ANDERSON (100 CPS)