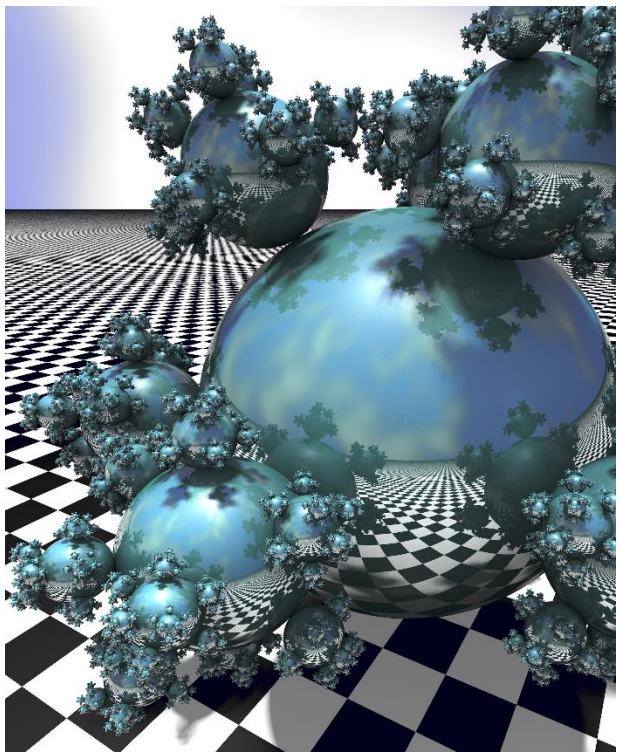
Discovering



A Contributor's Guide

BRLCAD

BRL CAD

Contents

Getting Started 1

A Call to Arms (and Contributors)

"The future exists first in the imagination, then in the will, then in reality." - Mike Muuss

Welcome to BRL-CAD! Whether you are a developer, documenter, graphic artist, academic, or someone who just wants to be involved in a unique open source project, BRL-CAD has a place for you. Our contributors come from all over the world and use their diverse backgrounds and talents to help maintain and enhance one of the oldest computer-aided design (CAD) packages used in government and industry today.

What is BRL-CAD?

BRL-CAD **(pronounced be-are-el-cad)** is a powerful, cross-platform, open source solid modelling system that includes interactive three-dimensional (3D) solid geometry editing, high-performance ray tracing support for rendering and geometric analysis, network-distributed framebuffer support, image and signal-processing tools, path tracing and photon mapping support for realistic image synthesis, a system performance analysis benchmark suite, an embedded scripting interface, and libraries for robust high-performance geometric representation and analysis

For more than two decades, BRL-CAD has been the primary solid modelling CAD package used by the U.S. government to help model military systems. The package has also been used in a wide range of military, academic, and industrial applications, including the design and analysis of vehicles, mechanical parts, and architecture. Other uses have included radiation dose planning, medical visualization, terrain

modelling, constructive solid geometry (CSG), modelling concepts, computer graphics education and system performance benchmark testing.

BRL-CAD supports a wide variety of geometric representations, including an extensive set of traditional implicit "primitive shapes" (such as boxes, ellipsoids, cones, and tori) as well as explicit primitives made from collections of uniform B-spline surfaces, non-uniform rational B-spline (NURBS) surfaces, n-manifold geometry (NMG), and purely faceted polygonal mesh geometry. All geometric objects may be combined using Boolean set-theoretic CSG operations such as union, intersection and difference.

Overall, BRL-CAD contains more than 400 tools, utilities, and applications and has been designed to operate on many common operating system environments, including BSD, Linux, Solaris, Mac OS X, and Windows. The package is distributed in binary and source code form as Free Open Source Software (FOSS), provided under Open Source Initiative (OSI) approved license terms.

History and Vision

BRL-CAD was originally conceived and written by the late Michael Muuss, the inventor of the popular PING network program. In 1979, the U.S. Army Ballistic Research Laboratory (BRL) (the agency responsible for creating ENIAC, the world's first general-purpose electronic computer in the 1940s) identified a need for tools that could assist with the computer simulations and analysis of combat vehicle systems and environments. When no existing CAD package was found to be adequate for this specialized purpose, Mike and fellow software developers began developing and assembling a unique suite of utilities capable of interactively displaying, editing, and interrogating geometric models. Those early efforts subsequently became the foundation on which BRL-CAD was built.

Development of BRL-CAD as a unified software package began in 1983, and its first public release came in 1984. Then, in 2004, BRL-CAD was converted from a limited-distribution U.S. government-controlled code to an open source project, with portions licensed under the LGPL and BSD licenses.

Today, the package's source code repository is credited as being the world's oldest, continuously developed open source repository. As a project, pride is taken in preserving all history and contributions.

The ongoing vision for BRL-CAD development is to provide a robust, powerful, flexible, and comprehensive solid modelling system that includes:

1. Faithful high-performance geometric representation.
2. Efficient and intuitive geometry editing.
3. Comprehensive conversion support for all solid geometry formats.
4. Effective geometric analysis tools for 3D CAD.

Key Strengths

All CAD packages are not alike. Among the many strengths of the BRL-CAD package are the following:

1. BRL-CAD is **open source**! Don't like something? You can make it better.
2. You can leverage **decades of invested development**. BRL-CAD is the most feature-filled open source CAD system available, with hundreds of years time invested.
3. **Your work will get used**. BRL-CAD is in production use and downloaded thousands of times every month by people all around the world.
4. You have the ability to create extensively **detailed realistic models.**
5. You can model objects on scales ranging from (potentially) the subatomic through the galactic, while essentially providing **all the details, all the time**.
6. You can leverage **one of the fastest raytracers** in existence (for many types of geometry).
7. You can convert to and from a wide range of **geometry file formats**
8. BRL-CAD has a powerful, **customizable scripting interface** with many advanced editing and processing capabilities.

Want to Be a Contributor?

With BRL-CAD being a part of the open source community since 2004, contributors from all over the world are able to enhance the features and functions of the package in many different ways. In return, these contributors have had a unique opportunity to:

1. Join a team of passionate and talented contributors who share the common values of open source development. Open source emphasizes free redistribution; openly available source code; full, open participation; and non-discrimination against individuals, groups, technologies, or fields of interest. (To learn more, see http://opensource.org.)
2. Drive needed improvements in the open source software community's support for solid modelling and CAD software capabilities.
3. Experiment with new and state-of-the-art algorithms and ideas within the context of a fully open CAD system that is in production use and has an established user community
4. Become a better developer. Whether you're a newbie or a seasoned developer with decades of experience, you can always work on a BRL-CAD project that is catered toward improving your abilities.
5. Become part of a legacy. Participate in a robust and historically significant open source project that goes all the way back to the days of the DEC PDP-11/70 and VAX-11/780./Gain practical experience working on a real-world, large-scale software project.

If you would like to be a BRL-CAD contributor, the primary areas currently identified for future development and enhancement include the following:

1. **Improved graphical user interface and usability** to accommodate increasingly varied user needs and participation levels. This includes improving the look-and-feel and features of:
    a. the primary editing graphical interface (MGED)
    b. the geometric visualization and interaction management system (libdm).
2. **Improved hybrid boundary representation geometry support** to support all 3D CAD models regardless of whether they use implicit or explicit geometric representation. Geometry formats we are particularly focusing on include:
    a. volumetric models (VOL)
    b. spline-surface (for example, NURBS) and polygonal (for example, triangle mesh) boundary representations (BREP)
    c. implicit primitives.
3. **Improved geometry services and functionality**, including the ability to provide:
    a. multiuser access controls
    b. comprehensive revision history
    c. collaborative enhanced multiuser modelling
    d. more flexible application development.

In addition, BRL-CAD's existing geometry kernel functions are continuously being refactored. Help turn them into a comprehensive, scriptable command framework, create an object-oriented geometry kernel application programming interface (API), or build a lightweight network daemon protocol for language agnostic client application development.

Improved open source awareness and increased development participation via:

- establishing strong open source community ties
- improving software documentation
- openly welcoming new contributors and users.

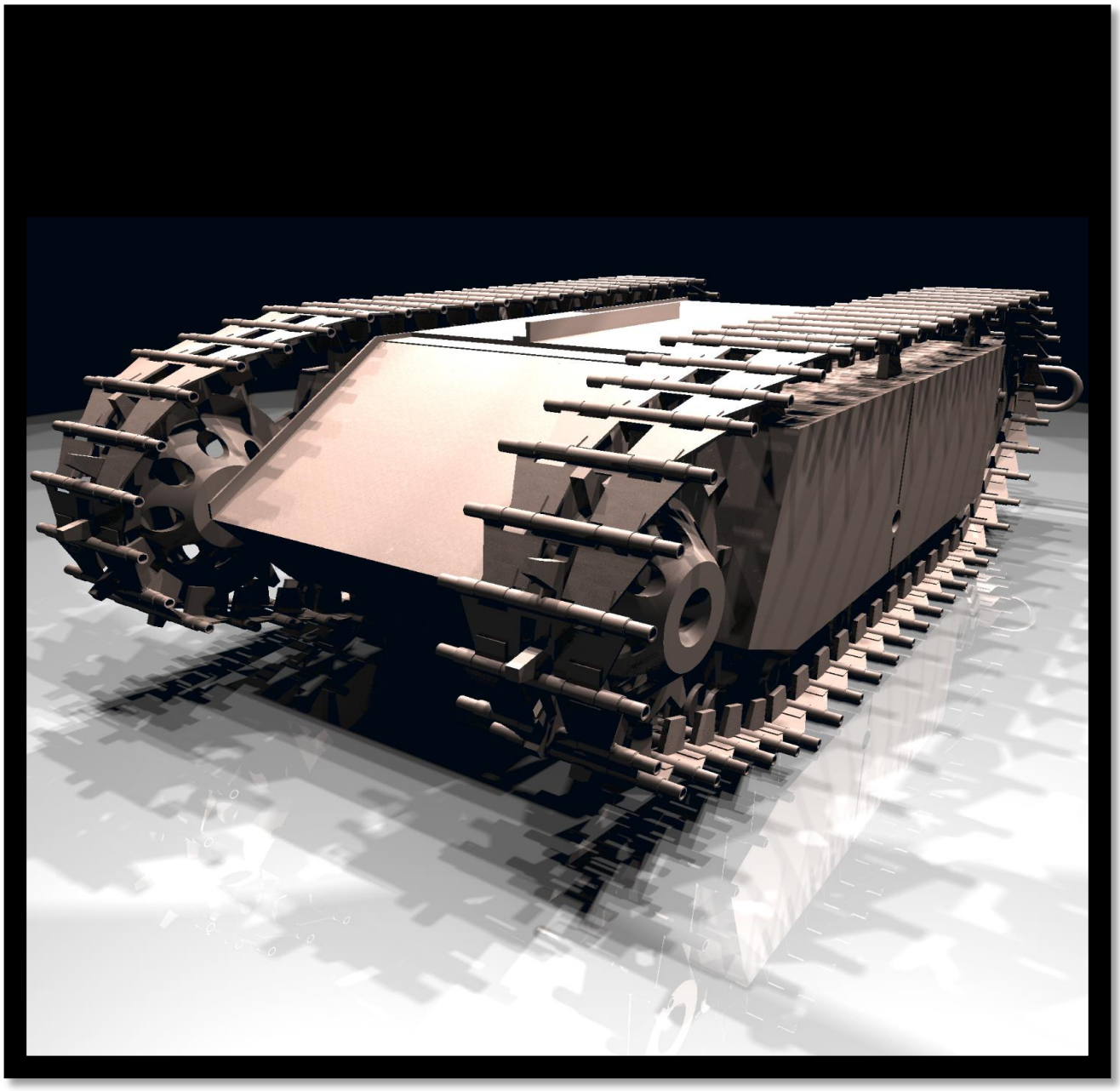
Let the contributions begin!

Feature Overview

BRL-CAD has thousands of distinct features that have been developed over a number of decades. One strength of a solid modelling system with integrated high-performance rendering is the ability to showcase some of those features graphically.

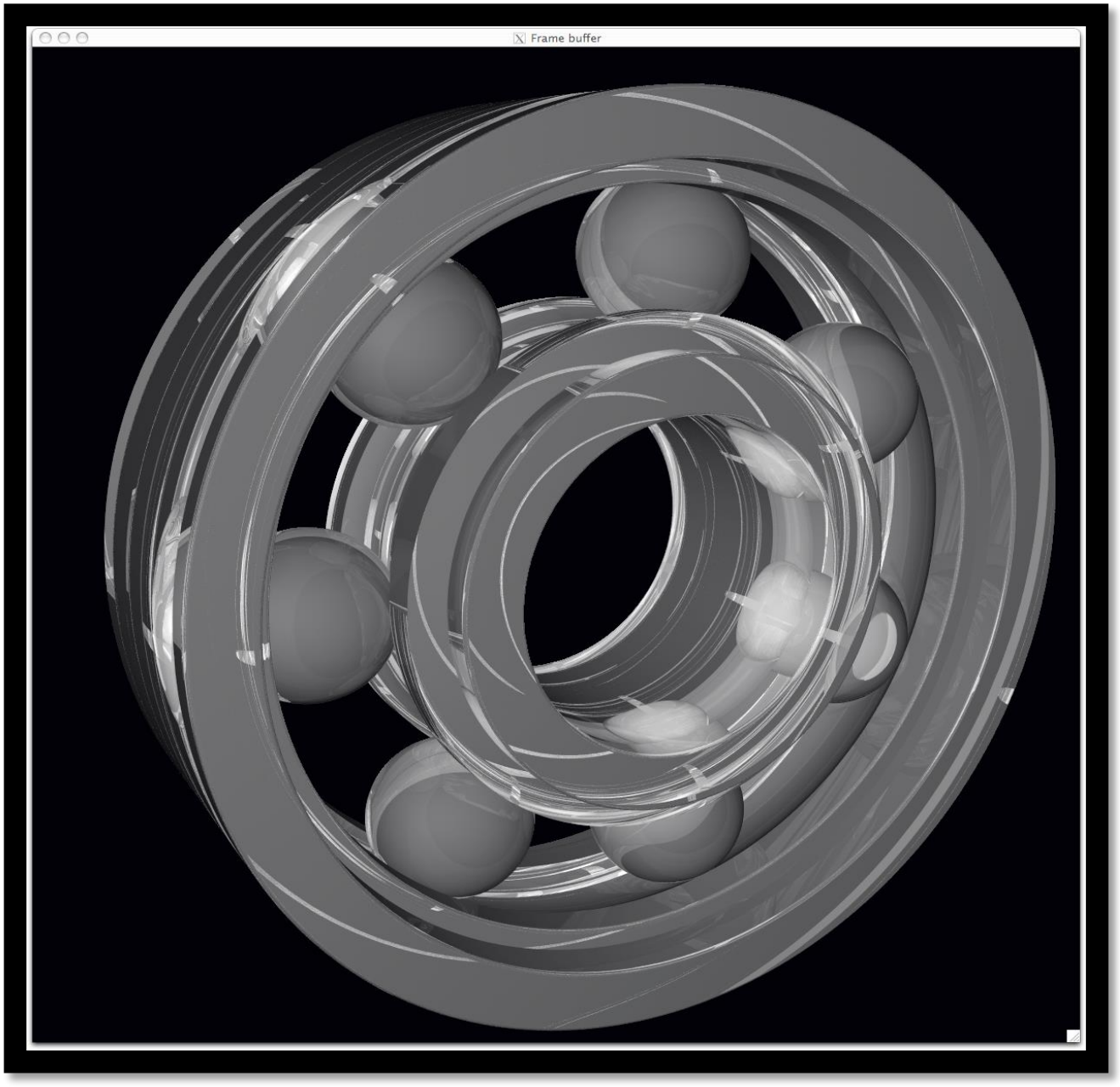
Let's take a quick look at just some of the high-level features provided by BRL-CAD.

Solid Geometry



BRL-CAD focuses on solid modeling CAD. Solid modeling is distinguished from other forms of geometric modeling by an emphasis on being physically accurate, fully describing 3D space. Shown is a 3D model of a Goliath tracked mine, a German-engineered remote controlled vehicle used during World War II. This model was created by students new to BRL-CAD in the span of about 2 weeks, starting from actual measurements in a museum.

Raytracing



Raytracing is central to BRL-CAD as a means for performing geometric analysis (e.g., calculating weights and moments of inertia) and for rendering images for visualization purposes. The image shown is a BRL-CAD 2D framebuffer screenshot displaying the rendering of a ball bearing. The bearing is modeled with a material appearance resembling acrylic glass, and this raytracing result shows reflection, refraction, shadowing, and some caustic effects.

Working With Our Code

BRL-CAD consists of more than 1 million lines of source code spanning more than 20 foundation libraries and 400 application modules.

The majority of BRL-CAD is written in highly portable C and C++, with some GUI and scripting components written in Tcl/Tk1. There is also some support for, and bindings to, other languages available. POSIX2 shell scripts are used for deployment integration testing. BRL-CAD uses the CMake3 build system for compilation and unit testing.

The Big Picture

The source code and most project data are stored in a Subversion4 version control system for change tracking and collaborative development. Trunk development is generally stable, but cross-platform compilation is not guaranteed. A separate branch (named STABLE) provides a higher level of quality assurance. Every released version of BRL-CAD is tested and tagged.

The project aims for an It Just Works approach to compilation whereby a functional build of BRL-CAD is possible without needing to install more than a compiler, CMake, and a build environment--for example, GNU Make or Microsoft Visual Studio. BRL-CAD provides all of the necessary third-party dependencies for download and compilation convenience within source distributions but by default will build using system versions of those dependencies if available.

As with any large system that has been under development for a number of years, there are vast sections of code that may be unfamiliar, uninteresting, or even daunting. Don't panic. BRL-CAD has been intentionally designed with layering and modularity in mind.

You can generally focus in on the enhancement or change that interests you without being too concerned with other portions of the code. You should, however, do some basic research to make sure what you plan to contribute isn't already in the BRL-CAD code base.

History of the Code

As mentioned previously, the initial architecture and design of BRL-CAD began in 1979. Development as a unified package began in 1983. The first public release was in 1984. And on December 21, 2004, BRL-CAD became an open source project5.

BRL-CAD is a mature code base that has remained active over decades due to continual attention on design and maintainability. Since the project's inception, more than 200 people have directly contributed to BRL-CAD. The project has historically received support from numerous organizations within academia, commercial industry, various government agencies, and from various independent contributors. We credit all contributors in BRL-CAD's authorship documentation6.

The following diagram illustrates how the number of lines of code in BRL-CAD has changed over time:

System Architecture

BRL-CAD is designed based on a UNIX7 methodology of the command-line services, providing many tools that work in harmony to complete a specific task. These tools include geometry and image converters, signal and image processing tools, various raytrace applications, geometry manipulators, and much more.

To support what has grown into a relatively large software system, BRL-CAD takes advantage of a variety of support libraries that encapsulate and simplify application development. At the heart of BRL-CAD is a multi-representation ray tracing library named LIBRT. BRL-CAD specifies its own file format (files with the extension .g or .asc) for storing information on disk. The ray tracing library uses a suite of other libraries for other basic application functionality.

Tenets of Good Software

BRL-CAD's architecture is designed to be as cross-platform and portable as is realistically and reasonably possible. As such, BRL-CAD maintains support for many legacy systems and devices provided that maintaining such support is not a significant burden on new development.

The code adheres to a published change deprecation and obsolescence policy8 whereby features that have been made publicly available are not removed without appropriate notification. Generally there should be a compelling motivation to remove any existing functionality, but improvements are encouraged.

BRL-CAD has a longstanding heritage of maintaining verifiable, validated, and repeatable results in critical portions of the package, particularly in the ray tracing

library. BRL-CAD includes regression tests that will compare runtime behavior against known results and report any deviations from previous results as failures. Considerable attention is put into verification and validation throughout BRL-CAD. Incorrect behavior does not need to be preserved simply to maintain consistency, but it is rare to find genuine errors in the baseline testing results. So, anyone proposing such a behavior change will have to conclusively demonstrate that the previous result is incorrect.

Code Layout

The basic layout of BRL-CAD's source code places public API headers in the top-level include directory and source code for both applications and libraries in the src directory. The following is a partial listing of how the code is organized in a checkout or source distribution. Note that some subdirectories contain a README file with more details on the content in that directory.

Code Conventions

BRL-CAD has a STABLE branch in SVN that should always compile and run on all supported platforms. The primary development branch trunk, unlike STABLE, is generally expected to compile but may occasionally fail to do so during active development.

Languages

The majority of BRL-CAD is written in ANSI/POSIX C with the intent of strictly conforming with the C standard. The core libraries are all C API, though several--such as the LIBBU and LIBRT libraries--use C++ for implementation details. Our C libraries can use C++ for implementation detail, but they cannot expose C++ in the public API.

Major components of the system are written in the following languages:

Application & Resources

db/

- Example Geometry

doc/

- Project documentation

doc/docbook

- User documentation in XML format
- See doc/docbook/README for more details

include/

- Public API headers

regress/

- Scripts and resources for regression testing

src/

- Application and library source code
- See src/README for more details

src/conv/

- Geometry converters

src/fb/

- Tools for displaying data in windows

src/mged/

- Main GUI application: Multi-device Geometry EDitor

Working With Our Documentation

BRL-CAD provides documentation in the following formats:

- UNIX man pages.
- HyperText Markup Language (HTML) for the web.
- PDF for documents needing a well-defined, consistent appearance.

Our challenge is to maintain BRL-CAD's documentation in multiple formats. It is difficult enough to keep software documentation up to date without needing to update multiple documents using different formats that contain the same information. As well, it is not possible to supply documentation in a single format that works optimally on all platforms. For example, while UNIX man pages are standard across all UNIX and UNIX-like systems, most Windows systems will not understand that format and will require HTML versions of those documents.

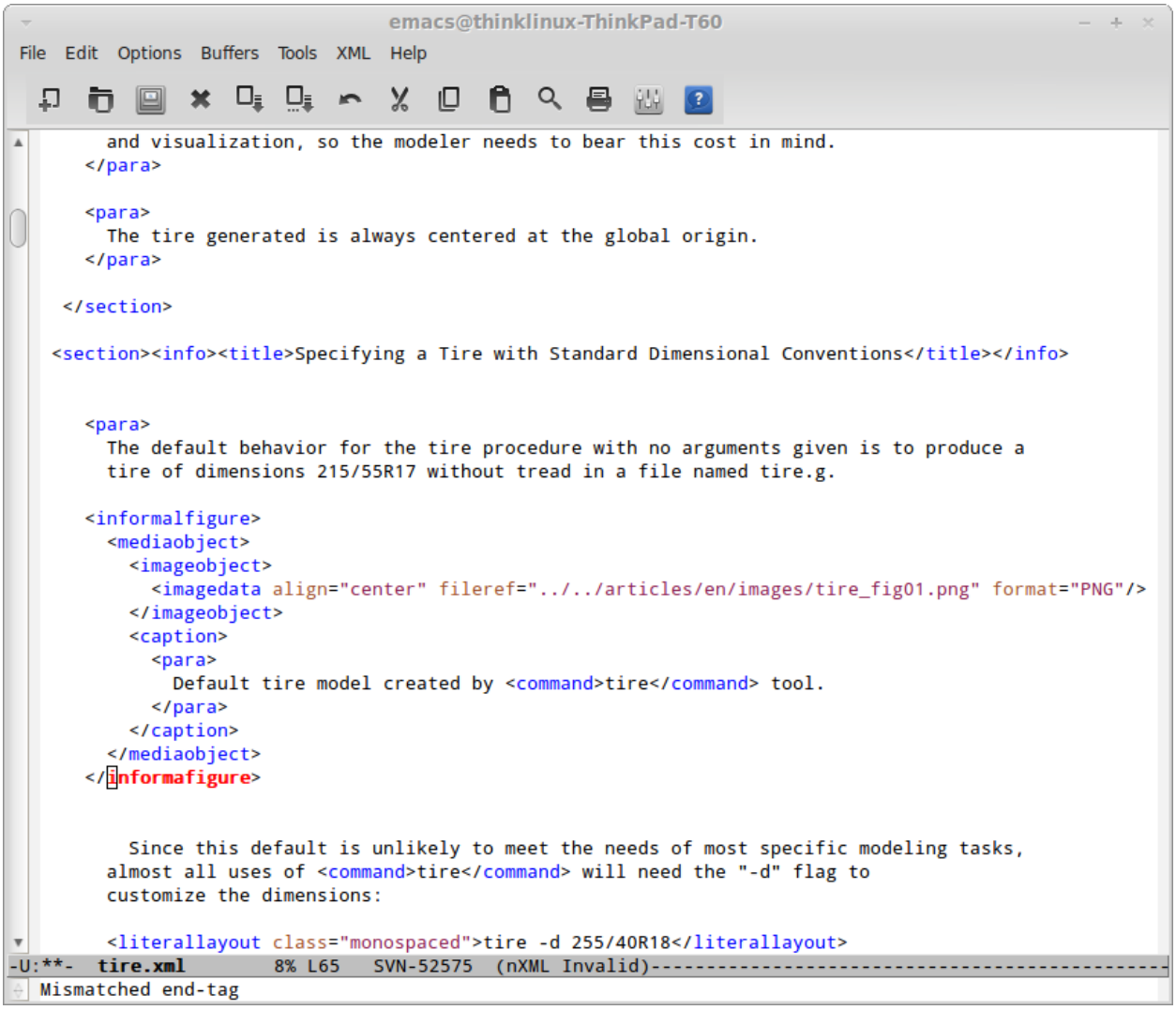
Instead of using a mix of formats and tools, BRL-CAD uses the DocBook documentation format and toolchain to produce documentation in the range of required formats.

What is DocBook?

DocBook is a schema (a structured approach to organization of information) that uses the eXtensible Markup Language standard (XML) as its fundamental framework and builds atop that framework a vocabulary for describing the content and structure of technical documentation. BRL-CAD uses the DocBook 5.0 documentation format to describe its documentation. For detailed documentation for DocBook 5.0, see http://www.docbook.org/tdg5/en/html/docbook.html.

Tools for Working with DocBook

While you can write documentation in DocBook using WYSIWYG (What You See Is What You Get) editors, we require that a document saved to DocBook from an editing tool should be inspected for human readability and, if necessary, reformatted for simplicity.



If you are comfortable with working with DocBook XML directly, we recommend that you use the Emacs editor and its nXML module. nXML can automatically recognize and highlight mistakes in the structure of a document while you are editing. The following image illustrates nXML identifying an incorrect closing tag for an informal figure object:Aside from error checking tools like nXML, the ability to pinpoint errors in a document's formatting is built into the BRL-CAD compilation process. That process uses a tool called xmllint to report incorrect formatting. When, for example, the error illustrated in the image above is encountered during BRL-CAD's build, xmllint produces the following error:

[40%] Validating DocBook source with xmllint:

/home/user/brlcad/doc/docbook/articles/en/tire.xml:65: parser error : Opening and ending tag mismatch: informalfigure line 54 and informafigure

</informafigure>

^

CMake Error at tire_validate.cmake:39 (message):

xmllint failure: 1

In this case, the error is reasonably informative. However, xmllint is not the only tool available for this sort of error checking. You can specify the following validation tools when you configure your environment:

1. Oracle Multi-Schema XML Validator (https://msv.java.net) - specified as msv on the command line.
2. oNVDL (http://sourceforge.net/projects/onvdl) - specified as nvld on the command line.
3. Relax NG Validator (rnv) (http://sourceforge.net/projects/rnv) - specified as rsv on the command line.

Note that these alternative validation tools must be installed on the system on which you are working; they are not provided with BRL-CAD. To specify an alternative tool, use the VALIDATE_EXECUTABLE option. For example, run the following command to use the Oracle Multi-Schema XML Validator:

```
CMAKE -DVALIDATE_EXECUTABLE=MSV ...
```

While BRL-CAD provides enough DocBook support to guarantee that HTML files and UNIX man pages are generated, you can only generate PDF documents if the Apache Formatting Objects Processor (FOP) (http://xmlgraphics.apache.org/fop) is installed on your system. When FOP is available, BRL-CAD can automatically produce PDF outputs.

For more information, including how to use alternative tools for other DocBook processing steps besides validation, see the file **doc/docbook/README** in the BRL-CAD source code archive.

Adding a New Document to BRL-CAD

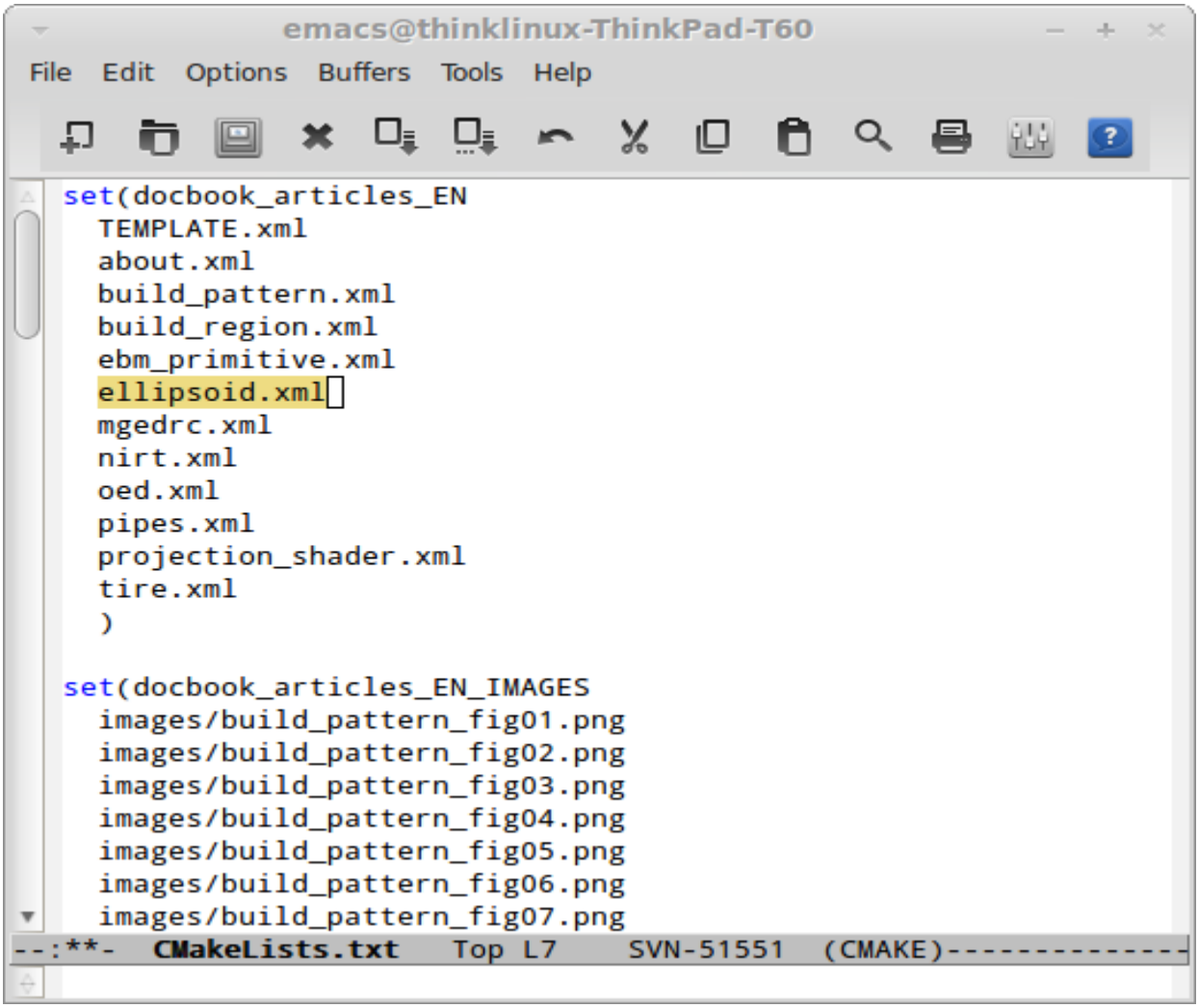
Because creating and editing DocBook documentation is greatly simplified by BRL-CAD's management of the conversion process, it is usually a good idea to add a new document to the build system at the beginning of the document creation and editing

process. To do this, copy a template file from the source directories to the file name to be used for the new document.

For example, if you are writing a new DocBook article in English about the ellipsoid, use the following command to copy the article template to the filename **ellipsoid.xml** in the English articles directory:

```
~/brlcad$cpdoc/docbook/articles/en/TEMPLATE.xmldoc/docbook/articles/en/
ellipsoid.xml
```

Next, open the file doc/docbook/articles/en/CMakeLists.txt in a text editor. Then, add the name of the new document to the file to alert the build system of its existence:



BRL-CAD now knows about the new file and can generate output for it.

You will generally only want to rebuild a specific output (say, HTML) to confirm that output renders properly. To set up the specific targets for the new file, run the command below to refresh the build targets (in this example, the build output directory is called build):

```
~/brlcad/build $ cmake ..
```

This creates a new build target, ellipsoid_article_html, which will build only the HTML output of the document and its dependencies:

```
~/brlcad/build $ make ellipsoid_article_html

[  0%] Built target printtimestamp
[  0%] Built target buildtimestart

Build Time: Tue Oct 15 19:14:42 2013
[  0%] Built target timestamp
[  0%] Built target zlib
[100%] Built target xml
[100%] Built target xslt
[100%] Built target exslt
[100%] Built target xmllint
[100%] Built target xsltproc
[100%] Built target schema-expand
[100%] Built target fonts-dejavu-expand
[100%] Built target fonts-stix-expand
[100%] Built target offo-2-expand
[100%] Built target svg-dtd-expand
[100%] Built target xsl-expand
[100%] Built target docbook_articles_EN_IMAGES_cp
Scanning dependencies of target ellipsoid_article_html
[100%] Validating DocBook source with xmllint:
/home/cyapp/brlcad/doc/docbook/articles/en/ellipsoid.xml validates
[100%] Generating ../../../../share/doc/html/articles/en/ellipsoid.html
[100%] Built target ellipsoid_article_html
~/brlcad/build $
```

Selecting Output Formats

Although you can produce HTML, UNIX man pages, and PDF files from the DocBook sources, you don't have to produce all of them. By default, PDF output is not produced because it takes longer to generate than other formats. UNIX man pages are not generated by default for Windows (where they generally are of little use) to avoid wasting configuration and compilation time.

You can use the following configuration options to turn the compilation of various formats on and off:

Option	Description	Setting
BRLCAD_EXTRADOCS	Enable DocBook documentation	ON
BRLCAD_EXTRADOCS_HTML	Enable HTML output	ON
BRLCAD_EXTRADOCS_MAN	Enable UNIX man page output	ON (OFF on Windows)
BRLCAD_EXTRADOCS_PDF	Enable PDF output (needs FOP)	OFF
BRLCAD_EXTRADOCS_PDF_MAN	Enable PDF man page output	Defaults to setting of BRLCAD_EXTRADOCS_PDF

The option to disable the PDF man page output exists to support situations where someone wants the article and tutorial PDFs, without the overhead of generating hundreds of PDFs for the various manual pages. If you do not specifically want PDF versions of the individual manual pages, set the BRLCAD_EXTRADOCS_PDF_MAN option to OFF.

Introduction to MGED 4

In this Lesson You Will….

Launch the MGED program.

- Enter commands at the MGED prompt in the Command Window.
- Use the MGED Graphical User Interface (GUI).
- Open or create a new database when launching MGED.
- Use the GUI to open or create a new database.
- Title a database.
- Select a unit of length for your design.
- Select a primitive shape.
- Create a primitive shape using the make command.
- Use the Z command to clear the Graphics Window.
- Draw a previously created shape using the draw command.
- Use the erase command to delete an item in the Graphics Window display.
- Create a sphere using the GUI menu.
- Use the l command to list a shape's attributes or parameters.
- Use the ls command to list the contents of the database.
- Eliminate a shape or object from the database using the kill command.
- Edit a command.
- Use the q or exit commands to quit the program

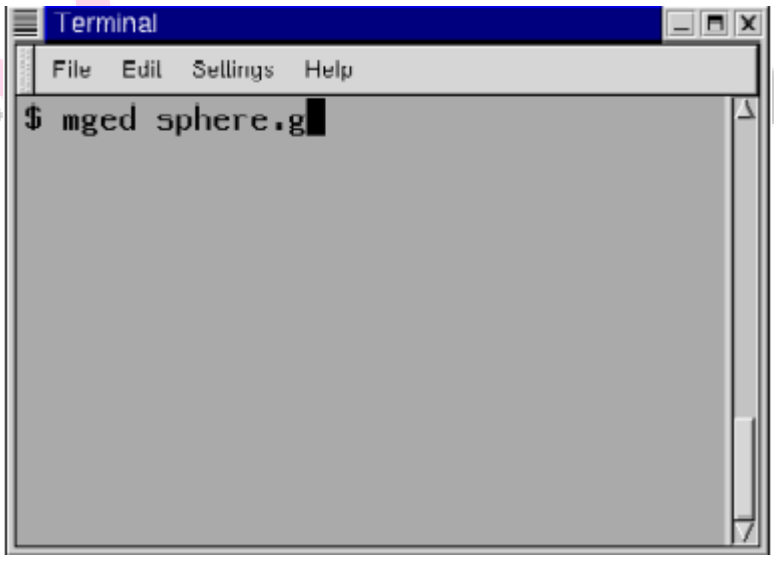
Launching the MGED Program

To launch the MGED program, type mged at the Terminal (tty) prompt and then press the ENTER key. This brings up two main windows: the MGED Command Window and the MGED Graphics Window (sometimes called the Geometry Window). Both windows will initially be blank, awaiting input from you. To leave the program at any time, at the Command Line type either the letter q or the word quit and then press the ENTER key.

Opening or Creating a New Database when Launching MGED

When launching MGED, you can open or create a database at the same time. At the shell prompt (usually a $ or %), in the Terminal Window, type mged followed by a new or existing database name with a .g extension. For example:

```
mged sphere.g<ENTER>
```



Terminal Window

If you are creating a new database, a small dialog box asking if you want to create a new database named sphere.g will appear. Click on Yes, and a new database will be created.