

Virtual Reality Toolbox

For Use with MATLAB[®] and Simulink[®]

Modeling

Simulation

Implementation

User's Guide

Version 3



How to Contact The MathWorks:



www.mathworks.com Web
comp.soft-sys.matlab Newsgroup



support@mathworks.com Technical support
suggest@mathworks.com Product enhancement suggestions
bugs@mathworks.com Bug reports
doc@mathworks.com Documentation error reports
service@mathworks.com Order status, license renewals, passcodes
info@mathworks.com Sales, pricing, and general information



508-647-7000 Phone



508-647-7001 Fax



The MathWorks, Inc. Mail
3 Apple Hill Drive
Natick, MA 01760-2098

For contact information about worldwide offices, see the MathWorks Web site.

Virtual Reality Toolbox User's Guide

© COPYRIGHT 2001-2002 by HUMUSOFT s.r.o. and The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by or for the federal government of the United States. By accepting delivery of the Program, the government hereby agrees that this software qualifies as "commercial" computer software within the meaning of FAR Part 12.212, DFARS Part 227.7202-1, DFARS Part 227.7202-3, DFARS Part 252.227-7013, and DFARS Part 252.227-7014. The terms and conditions of The MathWorks, Inc. Software License Agreement shall pertain to the government's use and disclosure of the Program and Documentation, and shall supersede any conflicting contractual terms or conditions. If this license fails to meet the government's minimum needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to MathWorks.

MATLAB, Simulink, Stateflow, Handle Graphics, and Real-Time Workshop are registered trademarks, and TargetBox is a trademark of The MathWorks, Inc.

Other product or brand names are trademarks or registered trademarks of their respective holders.

Printing History: August 2001 First printing New for Version 2.0 (Release 12.1)
July 2002 Second printing Revised for Version 3.0 (Release 13)

Preface

Required Products	viii
MATLAB	viii
Simulink	viii
VRML Viewer	ix
VRML Editor	x
Related Products	xi
Documentation and Help	xiii
Installing Online Documentation	xiii
Viewing Online Documentation	xiv
Printing the Documentation	xv
Product News Pages	xvi
Using This Guide	xvii
Expected Background	xvii
Organization	xviii
Conventions	xix
Terminology	xix
Typographical Conventions	xxi

What Is the Virtual Reality Toolbox?	1-2
Features of the Virtual Reality Toolbox	1-3
VRML Support	1-3
MATLAB Interface	1-4
Simulink Interface	1-4
VRML Viewers	1-5
VRML Editor	1-5
Real-Time Workshop Support	1-6
SimMechanics Support	1-6
Hardware Support	1-6
Client-Server Architecture	1-6
VRML Overview	1-8
VRML History	1-8
VRML Coordinate System	1-9
VRML File Format	1-10
Examples Using the Virtual Reality Toolbox	1-14
Simulink Interface Examples	1-14
MATLAB Interface Examples	1-21
Implementation Notes	1-23
Virtual Reality Toolbox Server	1-23
VRML Compatibility	1-24

System Requirements	2-2
Supported Computer Platforms	2-2
Host Computer	2-4
Client Computer	2-7
Installing the Virtual Reality Toolbox	
on the Host Computer	2-9
Getting or Updating Your License	2-9
Components on a Host Computer	2-10
Installing from CD (Windows)	2-11
Installing from CD (UNIX/Linux)	2-12
Downloading from the Web	2-14
Installing the VRML Viewer on the Host Computer	2-15
Virtual Reality Toolbox Viewer	2-15
Installing a VRML Plug-In (Windows)	2-16
Installing a VRML Plug-in (UNIX/Linux)	2-19
Setting the Default Viewer of Virtual Scenes	2-20
Installing the VRML Editor on the Host Computer	2-25
Installing VRML Editor (Windows)	2-25
VRML Editor (UNIX/Linux)	2-26
Setting the Default Editor of Virtual Scenes	2-26
Removing Components	2-31
Removing the Virtual Reality Toolbox and V-Realm Builder .	2-31
Removing the blaxxun Contact Plug-In	2-31
Installation on the Client Computer	2-33
Installing a VRML Plug-In (Windows)	2-33
VRML Plug-In (UNIX/Linux)	2-34
Testing the Installation	2-35
Running a Simulink Interface Example	2-35
Running a MATLAB Interface Example	2-40

Simulink Interface

3

Associating a Virtual World with Simulink	3-2
Adding a Virtual Reality Toolbox Block	3-2
Changing the Virtual World Associated with a Simulink Block	3-10
Using the Simulink Interface	3-13
Displaying a Virtual World and Starting Simulation	3-13
View a Virtual World with a Web Browser on the Host Computer	3-16
View a Virtual World with a Web Browser on the Client Computer	3-19

MATLAB Interface

4

Creating Virtual Reality Toolbox Objects	4-2
Creating a vrworld Object	4-2
Using the MATLAB Interface	4-4
Opening a Virtual World	4-4
Interacting with a Virtual World	4-5
Closing and Deleting a vrworld Object	4-8

Virtual Worlds

5

VRML Editing Tools	5-2
Editors for Virtual Worlds	5-2
V-Realm Builder	5-4

Deformation of a Sphere Example	5-5
Defining the Problem	5-5
Adding a Virtual Reality Toolbox Block	5-6
Creating a Sphere in a Virtual World	5-8
Creating a Box in a Virtual World	5-13
Connecting a Simulink Model to a Virtual World	5-17
Viewing a Virtual World	5-21
Virtual Reality Toolbox Viewer	5-21
blaxxun Contact VRML Plug-in	5-33
blaxxun Contact Settings	5-36
VRML Data Types	5-37
VRML Field Data Types	5-37
VRML Data Class Types	5-39

Block Reference

6 |

Function Reference

7 |

vrworld Object Reference

8 |

vrworld Object Properties	8-2
vrworld Object Methods	8-4

vrnode Object Reference

9

vrnode Object Properties	9-2
vrnode Object Methods	9-3

vrfigure Object Reference

10

vrfigure Object Properties	10-2
vrfigure Object Methods	10-5

Index

Preface

The Virtual Reality Toolbox is part of a family of software products that you use to create and visualize dynamic systems. Some of these products are required, while other products you use for special applications.

Required Products (p. viii)	MATLAB®, Simulink®, Virtual Reality Toolbox Viewer, Web browser with VRML plug-in (optional if you use the viewer), VRML editor
Related Products (p. xi)	Stateflow®, Stateflow Coder, SimMechanics, Real-Time Workshop®, Real-Time Windows Target, and xPC Target
Documentation and Help (p. xiii)	Location and installation of online HTML and PDF files
Using This Guide (p. xvii)	Suggestions for learning the Virtual Reality Toolbox and a description of the chapters
Conventions (p. xix)	Terms that can have various meanings, and text formats used in this guide
Typographical Conventions (p. xxi)	Conventions used throughout this guide.

Required Products

The Virtual Reality Toolbox is part of a family of products from The MathWorks. You need to install some of these products and other third-party products to use the Virtual Reality Toolbox.

This section includes the following topics:

- **MATLAB** — Create objects in the MATLAB workspace, connect these objects to a virtual world, and then use a command-line interface to control and make changes to the virtual world.
- **Simulink** — Create a model of your physical system and controller using a block diagram, connect your block diagram to a virtual world, and then use the block diagram to make changes to your model and view those changes in the virtual world.
- **VRML Viewer** — View virtual worlds described with VRML.
- **VRML Editor** — Create virtual worlds described with VRML.

MATLAB

MATLAB provides the tools you use to write scripts and functions in M-code. You can use your M-code scripts to set positions and properties of VRML objects, create callbacks from GUIs, and map data to virtual objects.

Note Version 3.0 of Virtual Reality Toolbox requires MATLAB Version 6.5 on the Release 13 CD. The product is also available for Web download.

MATLAB documentation — For information on using MATLAB, see the MATLAB documentation. It explains how to work with data and how to use the functions supplied with MATLAB. For a reference describing the functions specific to the Virtual Reality Toolbox, see Chapter 7, “Function Reference” of this guide.

Simulink

Simulink provides an environment where you model your physical system and controller as a block diagram. You create the block diagram by using a mouse to connect blocks and a keyboard to edit block parameters.

With the Virtual Reality Toolbox, you can view the model you created with Simulink blocks. You can also view the simulation of your dynamic system over time.

Note Version 3.0 of the Virtual Reality Toolbox requires Simulink Version 5.0, which is available on the Release 13 CD.

Simulink documentation — For information on using Simulink, see the Simulink documentation. It explains how to connect blocks, build models, and change block parameters. For a reference describing the Virtual Reality Toolbox blocks, see Chapter 6, “Block Reference” in this guide.

VRML Viewer

You use a VRML viewer to visualize and explore virtual worlds described with VRML. The following are descriptions of VRML viewers:

- **Virtual Reality Toolbox viewer** — This viewer is installed with the Virtual Reality Toolbox and is the default viewer for virtual worlds. You can access this viewer from either a Virtual Reality Toolbox block in your Simulink model, or by using the `vrview` and `vrfigure` functions with MATLAB.
The Virtual Reality Toolbox viewer is a client to the Virtual Reality Toolbox server. It does not require a Web browser and it is available on more platforms than any other VRML97 viewer. It is supported on PC, UNIX, and Linux platforms. The viewer is the recommended method for viewing virtual worlds on a host computer.
- **blaxxun Contact Version 4.4** — VRML plug-in shipped with the PC version of the Virtual Reality Toolbox. This VRML plug-in allows you to view virtual worlds in your Web browser. The blaxxun Contact plug-in is the only supported VRML plug-in.

You can view a virtual world in the Virtual Reality Toolbox viewer as soon as you install the Virtual Reality Toolbox. If you want to view the virtual world in your Web browser, you need to use the `vrinstall` command to install the blaxxun Contact plug-in. See “Installing a VRML Plug-In (Windows)” on page 2-16.

For information on using a Web browser to view virtual worlds, see “Testing the Installation” on page 2-35. To download the blaxxun Contact plug-in, see <http://www.mathworks.com/support/product/VR>.

Note Every VRML plug-in installs Java classes into the Web browser. It is best to limit the number of plug-ins you use in order to avoid Java errors and conflicts. For this reason, use only the Virtual Reality Toolbox viewer or the blaxxun Contact VRML plug-in on PC platforms. On UNIX and Linux platforms, use only the Virtual Reality Toolbox viewer.

VRML Editor

You use a VRML editor to create the virtual worlds you connect to Simulink block diagrams:

- **PC platforms** — V-Realm Builder Version 2.0 is included with the Virtual Reality Toolbox. If you do not want to use V-Realm Builder, you can use your favorite VRML editor.

Use the command `vrinstall` to install the editor before editing a virtual world. See “Installing VRML Editor (Windows)” on page 2-25.

For information on using V-Realm Builder with the Virtual Reality Toolbox, see Chapter 5, “Virtual Worlds.”

- **UNIX/Linux platforms** — The default VRML editor for UNIX/Linux platforms is the MATLAB editor. If you do not want to use the MATLAB editor, you can set the Editor preference to your favorite text editor.

Note V-Realm Builder is the only supported VRML editor. It is provided with the PC version of the Virtual Reality Toolbox.

Related Products

The MathWorks provides several products that are especially relevant to the kinds of tasks you can perform with the Virtual Reality Toolbox.

For more information about any of these products, see either

- The online documentation for that product if it is installed or if you are reading the documentation from the CD
- The MathWorks Web site, at <http://www.mathworks.com>; see the “products” section

Note The toolboxes listed below all include functions that extend the capabilities of MATLAB. The blocksets all include blocks that extend the capabilities of Simulink.

Product	Description
Aerospace Blockset	Model, analyze, integrate, and simulate aircraft, spacecraft, missile, weapon, and propulsion systems
Dials & Gauges Blockset	Monitor signals and control simulation parameters with graphical instruments
Real-Time Windows Target	Run Simulink and Stateflow models on a PC in real time
Real-Time Workshop	Generate C code from Simulink models
SimMechanics	Model and simulate mechanical systems
Simulink	Design and simulate continuous- and discrete-time systems
Stateflow	Design and simulate event-driven systems

Product	Description
Stateflow Coder	Generate C code from Stateflow charts
xPC Target	Perform real-time rapid prototyping using PC hardware
xPC Target Embedded Option	Deploy real-time applications on PC hardware

Documentation and Help

The Virtual Reality Toolbox software ships with printed documentation. This documentation is available online through the MATLAB Help browser, or as a PDF file that you can print or view online.

Documentation from The MathWorks does not include the VRML97 standard reference. You can view this reference online at http://www.web3d.org/fs_specifications.htm.

This section includes the following topics:

- **Installing Online Documentation** — Install HTML files from the Documentation CD or from a Web download.
- **Viewing Online Documentation** — View HTML files from your hard drive, the documentation CD, or the MathWorks Web site.
- **Printing the Documentation** — Locate and print PDF files on the Documentation CD or the MathWorks Web site.
- **Product News Pages**— View product information for the Virtual Reality Toolbox.

Installing Online Documentation

Installing the online documentation is part of the normal MathWorks installation process:

- **Documentation from a CD** — Start the MathWorks installer, and when prompted, select the **Product** and **Documentation** check boxes. During the installation process you are asked to insert the documentation CD.
- **Documentation from a Web download** — If you update the Virtual Reality Toolbox using a Web download, and you want to view the documentation with the MathWorks Help browser, you must install the documentation on your hard drive. Start the Web installer, and as before, select the **Product** and **Documentation** check boxes.

Note During normal installation of the Virtual Reality Toolbox from a CD or a Web download, the PDF files for the documentation are not copied to your hard drive. When you select to install the documentation, only the HTML files are copied.

Viewing Online Documentation

You can access the online documentation from HTML files you install on your hard drive, from the documentation CD, or through the MathWorks technical support Web pages.

To Access HTML Documentation on Your Hard Drive or the Documentation CD

- 1** In the MATLAB window, from the **Help** menu, click **Full Product Family Help**.

The Help browser window opens.

- 2** In the left pane, click **Virtual Reality Toolbox**.

In the right pane, the Help browser displays the Virtual Reality Toolbox Roadmap page. If you did not install the HTML help files on your hard drive, a message box opens asking you to insert the documentation CD.

- 3** Under the section titled Required and Related Products, select Related Products.

The Help browser displays the Virtual Reality Toolbox Release 13 Related Products information.

Note If you installed the Virtual Reality Toolbox from a Web download, and you chose not to install the HTML help files, the current documentation is neither on your hard drive, nor on the documentation CD. You need to use the MathWorks technical support Web site.

To Access HTML Documentation from MathWorks Technical Support

Alternatively, you can view the documentation from the MathWorks technical support Web site. The Web pages are identical to the latest release whether it was distributed from a CD or a Web download:

- 1 Open a Web browser.
- 2 In the address box, enter
`http://www.mathworks.com/access/helpdesk/help/toolbox/vr/vr.shtml`
- 3 Under the section titled Required and Related Products, select Related Products.
- 4 The Web browser displays the Virtual Reality Toolbox Related Products information.

Printing the Documentation

The documentation for the Virtual Reality Toolbox is available as PDF files. You need to install Adobe Acrobat Reader 4.0 or later to open and read these files. To download a free copy of Acrobat Reader, see <http://www.adobe.com/products/acrobat/main.html>.

To Access PDF Documentation on the Documentation CD

- 1 Insert the documentation CD into your CD drive.
- 2 In the MATLAB window, from the **Help** menu, click **Full Product Family Help**.

The MathWorks Help browser window opens.

- 3 In the left pane, click **Virtual Reality Toolbox**.

In the right pane, the Help browser displays the Virtual Reality Toolbox Roadmap page.

- 4 Under the section titled Printing the Documentation, select the PDF file you want to print.

Note If you installed the Virtual Reality Toolbox from a Web download, and you chose not to install the HTML help files, the current documentation is neither on your hard drive, nor on the documentation CD. You need to use the MathWorks technical support Web site.

Product News Pages

The developers for the Virtual Reality Toolbox maintain a Product News page. Information such as bug fixes, enhancements, and tips on how to use the product can be found on this page. To view the Product News page, navigate to

<http://www.mathworks.com/support/product/VR/>

Then, click on the Product News link.

Using This Guide

To help you effectively read and use this guide, here is a brief description of the chapters and a suggested reading path.

This section includes the following topics:

- **Expected Background** — Working knowledge of MATLAB and Simulink
- **Organization** — Chapters, sections, and topics with procedures and reference information

Expected Background

This manual assumes that you are already familiar with

- MATLAB, to write scripts and functions with M-code, and to use functions with the command-line interface
- Simulink and Stateflow, to create models as block diagrams and simulate those models

If You Are a New User — You might want to review

- Chapter 1, “Introduction” — This chapter gives you an overview of the Virtual Reality Toolbox features.
- Chapter 3, “Simulink Interface” — Interact with a virtual world from Simulink.
- Chapter 4, “MATLAB Interface” — Interact with a virtual world from MATLAB.

If You Are an Experienced Virtual Reality Toolbox User — You might want to review

- Chapter 6, “Block Reference” — Additional functionality has been added to the Virtual Reality Toolbox library.
- Chapter 8, “vrworld Object Reference” — Description of vrworld object properties and methods.
- Chapter 9, “vrnode Object Reference” — Description of vrnode object properties and methods.

- Chapter 10, “vrfigure Object Reference” — Description of vrfigure object properties and methods. Version 3.0 of the Virtual Reality Toolbox includes the new object vrfigure.

Organization

The following table lists the chapters of this guide.

Chapter or Appendix	Description
Chapter 1, “Introduction”	Overview of the functions and features of the Virtual Reality Toolbox
Chapter 2, “Installation”	Procedures to install the Virtual Reality Toolbox, VRML plug-in, and VRML editor
Chapter 3, “Simulink Interface”	Procedures to add Virtual Reality Toolbox blocks to your Simulink model and associate those blocks with virtual worlds
Chapter 4, “MATLAB Interface”	Procedures for creating scripts and functions to control and change virtual worlds using MATLAB objects
Chapter 5, “Virtual Worlds”	Procedures for creating a simple virtual world using V-Realm Builder
Chapter 6, “Block Reference”	Reference for blocks in the Virtual Reality Toolbox
Chapter 7, “Function Reference”	Functions for the Virtual Reality Toolbox environment
Chapter 8, “vrworld Object Reference”	Reference for vrworld objects, including methods and properties
Chapter 9, “vrnode Object Reference”	Reference for vrnode objects, including methods and properties
Chapter 10, “vrfigure Object Reference”	Reference for vrfigure objects, including methods and properties

Conventions

To help you effectively use this guide, there are some conventions. Conventions consist of ways of consistently formatting the text and graphics, and the meanings for common terms.

The topics in this section are

- **Terminology**— Terms specific to the Virtual Reality Toolbox and terms that can have multiple meanings
- **Typographical Conventions** — Formatting conventions to indicate user-selected objects, system messages, and filenames

Terminology

The following table lists some of the terms used in this guide.

Term	Definition
application	See <i>real-time application</i> .
build process	The process of generating C code from your Simulink model, compiling and inlining the generated code to create a <i>real-time executable</i> .
external mode	A Simulink mode that uses a Simulink block diagram as a graphical user interface to a <i>real-time executable</i> . This interface provides parameter downloading and signal uploading for display using Scope blocks.
real-time application	Code that is ready to run in real time with the <i>kernel</i> .
simulation	The process of running a dynamic system in nonreal time to observe its behavior.
Virtual Reality Modeling Language	The specification for displaying three-dimensional objects using a VRML viewer.

Term	Definition (Continued)
virtual figure object	A handle to a Virtual Reality Toolbox viewer window.
virtual node object	A handle to a node in a virtual world that allows access to the node's properties.
virtual world	An imaginary world where you can navigate around objects in three dimensions.
virtual world object	A handle to a virtual world that allows you to interact with and control the world.
VRML	See "VRML Overview" on page 1-8 of this guide.

Typographical Conventions

This manual uses some or all of these conventions.

Item	Convention	Example
Example code	Monospace font	To assign the value 5 to A, enter <code>A = 5</code>
Function names, syntax, filenames, directory/folder names, user input, items in drop-down lists	Monospace font	The <code>cos</code> function finds the cosine of each array element. Syntax line example is <code>MLGetVar ML_var_name</code>
Buttons and keys	Boldface with book title caps	Press the Enter key.
Literal strings (in syntax descriptions in reference chapters)	Monospace bold for literals	<code>f = freqspace(n, 'whole')</code>
Mathematical expressions	<i>Italics</i> for variables Standard text font for functions, operators, and constants	This vector represents the polynomial $p = x^2 + 2x + 3$.
MATLAB output	Monospace font	MATLAB responds with <code>A =</code> <code>5</code>
Menu and dialog box titles	Boldface with book title caps	Choose the File Options menu.
New terms and for emphasis	<i>Italics</i>	An <i>array</i> is an ordered collection of information.
Omitted input arguments	(...) ellipsis denotes all of the input/output arguments from preceding syntaxes.	<code>[c, ia, ib] = union(...)</code>
String variables (from a finite list)	<i>Monospace italics</i>	<code>sysc = d2c(sysd, 'method')</code>

Introduction

The Virtual Reality Toolbox allows you to connect a virtual world, defined with VRML, to Simulink and MATLAB. Understanding the features of the Virtual Reality Toolbox and some basic VRML concepts will help you to use this product more effectively.

What Is the Virtual Reality Toolbox? (p. 1-2)	Solution for virtual interaction with models of dynamic systems over time
Features of the Virtual Reality Toolbox (p. 1-3)	Description of the many features available to create and view dynamic systems
VRML Overview (p. 1-8)	Brief history of VRML, differences between the VRML and MATLAB coordinate systems, and the format of VRML files
Examples Using the Virtual Reality Toolbox (p. 1-14)	VRML worlds with an interface to Simulink block diagrams and an interface to MATLAB objects and functions
Implementation Notes (p. 1-23)	Outlines the Virtual Reality Toolbox Server and VRML compatibility

What Is the Virtual Reality Toolbox?

The Virtual Reality Toolbox is a solution for viewing and interacting with dynamic systems in a three-dimensional virtual reality environment. It extends the capabilities of MATLAB and Simulink into the world of virtual reality graphics.

- **Virtual worlds** — Create virtual worlds or three-dimensional scenes using standard Virtual Reality Modeling Language (VRML) technology.
- **Dynamic systems** — Create and define dynamic systems with MATLAB and Simulink.
- **Animation** — View moving three-dimensional scenes driven by signals from the Simulink environment.
- **Manipulation** — Change the position and properties of objects in a virtual world, or change parameters in your Simulink model while running a simulation.

To provide a complete working environment, the Virtual Reality Toolbox includes additional components:

- **VRML viewer** — Use either the Virtual Reality Toolbox viewer or, for PC platforms, the blaxxun Contact plug-in for Web browsers to display your virtual worlds.
- **VRML editor** — For PC platforms, use V-Realm Builder to create and edit VRML code. For UNIX or Linux platforms, use the MATLAB text editor to write VRML code to create virtual worlds.

Features of the Virtual Reality Toolbox

The Virtual Reality Toolbox includes many features for you to create and visualize dynamic systems. It also provides real-time virtual interaction with dynamic models.

This section includes the following topics that describe these features:

- **VRML Support** — Use VRML to define a virtual world
- **MATLAB Interface** — Control the virtual world from the MATLAB interface
- **Simulink Interface** — Use Virtual Reality Toolbox blocks to connect your Simulink model to a virtual world
- **VRML Viewers** — View your virtual world with the Virtual Reality Toolbox viewer or your Web browser
- **VRML Editor** — Create virtual worlds using a VRML authoring tool or text editor
- **Real-Time Workshop Support** — Support for simulations that use code generated by Real-Time Workshop
- **SimMechanics Support** — View the behavior of your SimMechanics model in a virtual world
- **Hardware Support** — Functions for using special hardware devices
- **Client-Server Architecture** — Provide client-server architecture for a single computer or network operation

VRML Support

The Virtual Reality Modeling Language (VRML) is an ISO standard that is open, text-based, and uses a WWW-oriented format. You use VRML to define a virtual world that you can display with a VRML viewer and connect to a Simulink model.

The Virtual Reality Toolbox uses many of the advanced features defined in the current VRML97 specification. The term VRML, in this guide, always refers to VRML as defined in the VRML97 standard ISO/IEC 14772-1:1997. This format includes a description of 3-D scenes, sounds, internal actions, and WWW anchors.

The Virtual Reality Toolbox analyzes the structure of the virtual world, determines what signals are available, and makes the signals available from MATLAB and Simulink.

The Virtual Reality Toolbox viewer supports the majority of VRML97 standard nodes, allowing you almost complete control over associated virtual worlds. The blaxxun Contact plug-in supports all of VRML97 standard nodes.

The Virtual Reality Toolbox makes sure that the changes made to a virtual world are reflected in MATLAB and Simulink. If you change the viewpoint in your virtual world, this change occurs in the vrworld object properties in MATLAB and Simulink.

The Virtual Reality Toolbox includes functions for retrieving and changing virtual world properties.

MATLAB Interface

The Virtual Reality Toolbox provides a flexible MATLAB interface to virtual reality worlds. After creating MATLAB objects and associating them with a virtual world, you can control the virtual world by using functions and methods.

From MATLAB, you can set positions and properties of VRML objects, create callbacks from graphical user interfaces (GUIs), and map data to virtual objects. You can also view the world with a VRML viewer, determine its structure, and assign new values to all available nodes and their fields.

The Virtual Reality Toolbox includes functions for retrieving and changing the virtual world properties and for saving the VRML files corresponding to the actual structure of a virtual world.

MATLAB provides communication for control and manipulation of virtual reality objects using MATLAB objects.

Simulink Interface

With a Simulink model, you can observe a simulation of your dynamic system over time in a visually realistic 3-D model.

The Virtual Reality Toolbox provides blocks to directly connect Simulink signals with virtual worlds. This connection lets you visualize your model as a three-dimensional animation.

You can implement most of the Virtual Reality Toolbox features with Simulink blocks. Once you include these blocks in a Simulink diagram, you can select a virtual world and connect Simulink signals to the virtual world. The Virtual Reality Toolbox automatically scans a virtual world for available VRML nodes that Simulink can drive.

All the VRML node properties are listed in a hierarchical tree-style viewer. You select the degrees of freedom to control from within Simulink. After you close a **Block Parameters** dialog box, Simulink updates the block with the inputs and outputs corresponding to selected nodes in the virtual world. After connecting these inputs to appropriate Simulink signals, you can view the simulation with a VRML viewer.

Simulink provides communication for control and manipulation of virtual reality objects, using Virtual Reality Toolbox blocks.

VRML Viewers

The Virtual Reality Toolbox contains a viewer that is the default viewing method for virtual worlds. This Virtual Reality Toolbox viewer is supported on PC, UNIX, and Linux platforms.

If you are on a PC or SGI platform, you can install a VRML plug-in and view a virtual world in your preferred Web browser. For PC platforms, the Virtual Reality Toolbox includes the popular VRML plug-in blaxxun Contact. This is the only supported VRML plug-in. For SGI platforms, use the Cosmo Player VRML plug-in.

The Virtual Reality Toolbox connects MATLAB and Simulink with a VRML-enabled browser to display a simulated process using the TCP/IP protocol. This allows you to watch a simulated virtual world not only on the computer where MATLAB and Simulink are running, but also on other computers connected through the Internet.

VRML Editor

For PC platforms, the Virtual Reality Toolbox includes one of the classic VRML authoring tools, V-Realm Builder by Ligos Corp. With the addition of this VRML authoring tool, the Virtual Reality Toolbox provides a complete authoring, development, and working environment for carrying out 3-D visual simulations.

For UNIX and Linux platforms, you can use the MATLAB text editor to write VRML code to create virtual worlds. You can also use your favorite text editor.

Real-Time Workshop Support

The Virtual Reality Toolbox seamlessly integrates with Real-Time Workshop targets. It supports simulations that use code generated by Real-Time Workshop and a third-party compiler on your desktop computer. The Virtual Reality Toolbox also supports code executed in real time on external target computers. It enables interaction with real-time code generated by Real-Time Workshop and compiled with a third-party C/C++ compiler.

SimMechanics Support

You can use the Virtual Reality Toolbox to view the behavior of a model created with SimMechanics. First, you build a model of a machine in Simulink using SimMechanics blocks. SimMechanics creates a rough visual schematic of the model. You can then view the animation of this model using a VRML viewer.

Alternatively, you can create a more detailed picture of your machine in a virtual world, connect this world to the SimMechanics body sensor output, and then view the behavior of the body in a VRML viewer.

Hardware Support

The Virtual Reality Toolbox contains functions for using special hardware devices, including Joystick and SpaceMouse. It can also connect to common hardware devices, including joysticks and Magellan SpaceMouse, using Simulink blocks.

Client-Server Architecture

- Multiple clients connected to one server
- Adjustable parameters for tuning network performance
- Provides client-server architecture for a single computer or network operation

The Virtual Reality Toolbox connects MATLAB and Simulink to a VRML-enabled Web browser using the TCP/IP protocol. The toolbox can be used in two configurations:

- **Single computer** — MATLAB, Simulink, and the virtual reality representations run on the same host computer.
- **Network computer** — You can view an animated virtual world on a computer separate from the computer with the Virtual Reality Toolbox server.

VRML Overview

The Virtual Reality Modeling Language (VRML) is the language you use to display three-dimensional objects with a VRML viewer.

This section includes the following topics:

- **VRML History** — Events leading up to the creation of the VRML97 standard.
- **VRML Coordinate System** — The VRML coordinate system is different from the MATLAB coordinate system.
- **VRML File Format** — VRML files use a hierarchical structure to describe three-dimensional objects and their movements.

VRML History

Since people started to publish their documents on the World Wide Web (WWW), there has been an effort to enhance the content of Web pages with advanced three-dimensional graphics and interaction with those graphics.

The term Virtual Reality Markup Language (VRML) was first used by Tim Berners-Lee at a European Web conference in 1994 when he talked about a need for a 3-D Web standard. Soon afterward, an active group of artists and engineers formed around a mailing list called `www-vrml`. They changed the name of the standard to Virtual Reality Modeling Language to emphasize the role of graphics. The result of their effort was to produce the VRML 1 specification. As a basis for this specification they used a subset of the Inventor file format from Silicon Graphics.

The VRML 1 standard was implemented in several VRML browsers, but it only allowed you to create static virtual worlds. This limitation reduced the possibility of its widespread use. Quickly it became clear that the language needed a robust extension to add animation and interactivity, and bring life to a virtual world. The VRML 2 standard was developed, and in the year 1997 it was adopted as International Standard ISO/IEC 14772-1:1997. Since then it is referred to as VRML97.

VRML97 represents an open and flexible platform for creating interactive three-dimensional scenes (virtual worlds). As computers improve in computational power and graphic capability, and communication lines become faster, the use of 3-D graphics becomes more popular outside the traditional

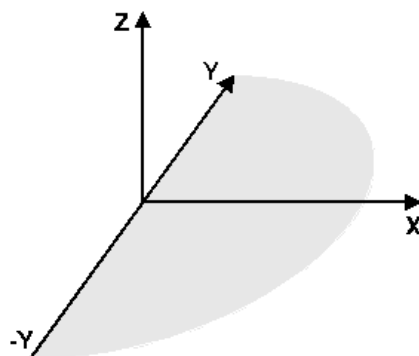
domain of art and games. There are now a number of VRML97-enabled browsers available on several platforms. Also, there are an increasing number of VRML authoring tools from which to choose. In addition, many traditional graphical software packages (CAD, visual art, and so on) offer VRML97 import/export features.

The Virtual Reality Toolbox uses VRML97 technology to deliver a unique, open 3-D visualization solution for MATLAB users. It is a useful contribution to a wide use of VRML97 in the field of technical and scientific computation and interactive 3-D animation.

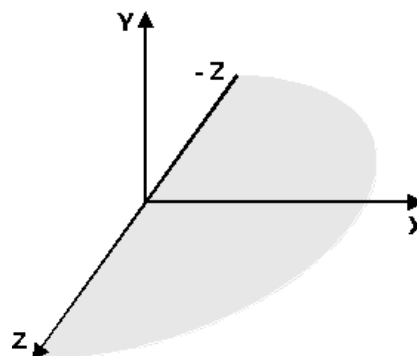
The VRML97 standard continues to be improved by the Web 3D Consortium. The newly released X3D (eXtensible 3D) standard is the successor to VRML97. X3D is an extensible standard that provides compatibility with existing VRML content and browsers. For more information, see <http://www.web3d.org>.

VRML Coordinate System

VRML uses the right-handed *Cartesian coordinate system*. If your thumb, index finger, and middle finger of the right hand are held so that they form three right angles, then your thumb symbolizes the x -axis, your index finger the y -axis (pointing up), and your middle finger the z -axis.



MATLAB graphics coordinate system

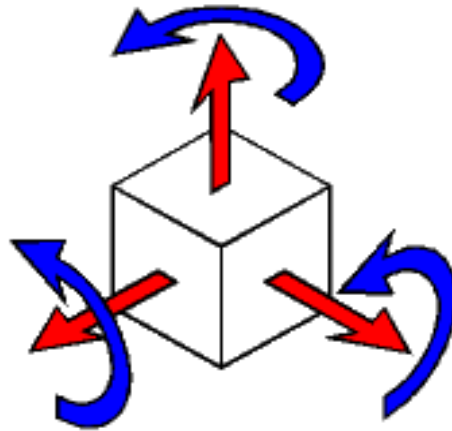


VRML coordinate system

The VRML coordinate system is different from the MATLAB, Aerospace Blockset, and SimMechanics coordinate systems. VRML uses the *world coordinate system* in which the y -axis points upward and the z -axis places objects nearer or farther from the front of the screen. It is important to realize

this fact in situations involving the interaction of these different coordinate systems.

Rotation angles — In VRML, rotation angles are defined using the *right-hand rule*. Imagine your right hand holding an axis while your thumb points in the direction of the axis towards its positive end. Your four remaining fingers point in a counter-clockwise direction. This counter-clockwise direction is the positive rotation angle of an object moving around that axis.



Child objects — In the hierarchical structure of a VRML file, the position and orientation of child objects are specified relative to the parent object. The parent object has its local coordinate space defined by its own position and orientation. Moving the parent object also moves the child objects relative to the parent object.

Measurement units — All lengths and distances are measured in *meters*, and all angles are measured in *radians*.

VRML File Format

You need not have any substantial knowledge of the VRML format to use the VRML authoring tools to create virtual worlds. However, it is useful to have a basic knowledge of VRML scene description. This helps you to create virtual worlds more effectively, and gives you a good understanding of how the virtual world elements can be controlled using the Virtual Reality Toolbox.

This section is an introduction to VRML. For more information, refer to the VRML97 Reference. This reference is available online at <http://www.vrml.org/Specifications/VRML97>. There are many specialized VRML books that can help you understand VRML concepts and create your own virtual worlds. We recommend the book *Teach Yourself VRML2 in 21 days* by Chris Marrin and Bruce Campbell (1997).

In VRML, a 3-D scene is described by a hierarchical tree structure of objects (nodes). Every node in the tree represents some functionality of the scene. There are 54 different types of nodes. Some of them are *shape nodes* (representing real 3-D objects), and some of them are *grouping nodes* used for holding child nodes. Here are some examples:

- **Box node** — Represents a box in a scene.
- **Transform node** — Defines position, scale, scale orientation, rotation, translation, and children of its subtree (grouping node).
- **Material node** — Corresponds to material in a scene.
- **DirectionalLight node** — Represents lighting in a scene.
- **Fog node** — Allows you to modify the environment optical properties.
- **ProximitySensor node** — Brings interactivity to VRML97. This node generates events when the user enters, exits, and moves within the defined region in space.

Each node contains a list of fields that hold values defining parameters for its function.

Nodes can be placed in the top level of a tree or as children of other nodes in the tree hierarchy. When you change a value in the field of a certain node, all nodes in its subtree are affected. This feature allows you to define relative positions inside complicated compound objects.

You can mark every node with a specific name by using the keyword DEF in the VRML scene syntax. For example, the statement `DEF MyNodeName Box` sets the name for this box node to MyNodeName. You can only access the fields of those nodes that you name in a virtual world.

In the following example of a simple VRML file, two graphical objects are modeled in a 3-D scene: A floor is represented by a flat box with a red ball above it. Note that VRML file is a readable text file that you can write in any text editor.

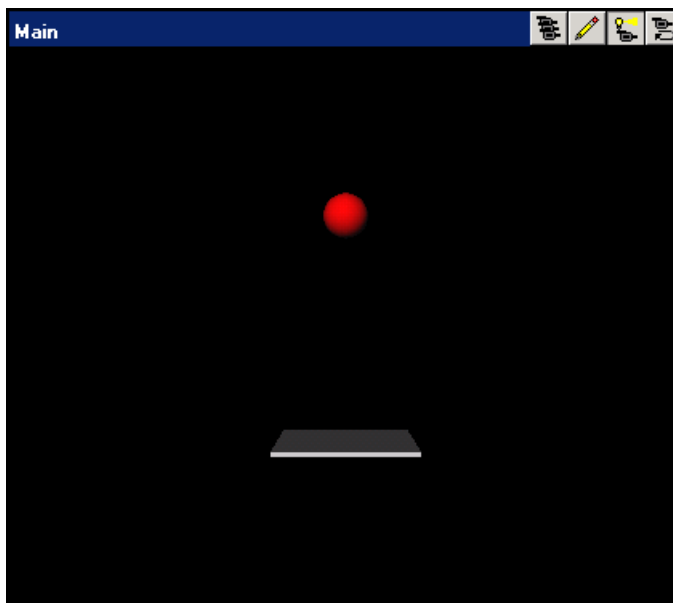
```
#VRML V2.0 utf8
# This is a comment line
WorldInfo {
  title "Bouncing Ball"
}
Viewpoint {
  position 0 5 30
  description"Side View"
}
DEF Floor Box {
  size 6 0.2 6
}
DEF Ball Transform {
  translation 0 10 0
  children Shape {
    appearance Appearance {
      material Material {
        diffuseColor 1 0 0
      }
    }
    geometry Sphere {
    }
  }
}
```

The first line is the VRML header line. Every VRML file must start with this header line. It indicates that this is a VRML 2 file and that the text objects in the file are encoded according to the UTF8 standard. You use the number sign (#) to comment VRML worlds. Everything on a line after the # sign is ignored by a VRML viewer, with the exception of the first header line.

Most of the box properties are left at their default values — distance from the center of coordinate system, material, color, and so on. Only the name `Floor` and the dimensions are assigned to the box. To be able to control the position and other properties of the ball, it is defined as a child node of a `Transform` type node. Here, the default unit sphere is assigned a red color and a position 10 m

above the floor. In addition, the virtual world title is used by VRML viewers to distinguish between virtual worlds. A suitable initial viewpoint is defined in the virtual world VRML file.

When displayed in V-Realm builder, the floor and red ball look like



Examples Using the Virtual Reality Toolbox

The Virtual Reality Toolbox includes examples using both the Simulink and MATLAB interfaces. You can use these examples to learn what you can do with the Virtual Reality Toolbox.

This section includes the following topics:

- **Simulink Interface Examples**— Examples that use the VR Sink and VR Source blocks in Simulink block diagrams
- **MATLAB Interface Examples** — Examples that use MATLAB objects to interact with a virtual world

Simulink Interface Examples

For all the examples that have a Simulink model, use the following procedure to view a virtual world.

- 1** In the MATLAB Command Window, enter the name of a Simulink model. For example, enter

```
vrbounce
```

A Simulink window opens with the block diagram for the model. If the **Open VRML viewer automatically** check box is selected, a virtual world opens in the Virtual Reality Toolbox viewer or in your VRML-enabled Web browser.

- 2** In the Simulink window, double-click the VR Sink block.

A **Block Parameters** dialog box opens. Notice that the **Open the VRML viewer automatically** check box is selected for all Virtual Reality Toolbox demos.

If you close the virtual world window, you can display it again by clicking the **View** button.

- 3** In the Simulink window, from the **Simulation** menu, click **Start**.

A simulation starts running and the virtual world is animated using signal data from the simulation.

The following table lists the Simulink examples provided with the Virtual Reality Toolbox. Descriptions of the examples follow the table.

Example	RTW Ready	VR Source	VR Sink	Joystick	SpaceMouse
vrbounce	X		X		
vrcrane		X	X		
vrlights			X		
vrmaglev	X	X	X		
vrmaglev_rtwin	X	X	X		
vrmanipul			X		X
vrmemb1			X		
vrpend	X	X	X		
vrplanets	X		X		
vrtkoff			X		

Bouncing Ball Example (vrbounce)

The vrbounce example represents a ball bouncing from a floor. The ball deforms as it hits the floor, keeping the volume of the ball constant. The deformation is achieved by modifying the scale field of the ball.

Tower Crane with Weight Example (vrcrane)

The vrcrane example illustrates how to control a Simulink model from a virtual world. In the associated VRML file, the **Joystick** PROTO is defined. The VR Source block receives the X output from this PROTO to actuate the motor that drives the crane. The crane dynamics are modeled according to the following equations, using the differential equations editor.

$$\begin{bmatrix} m + M & Ml \cos(\theta) \\ Ml \cos(\theta) & Ml^2 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} F + Ml \sin(\theta) \dot{\theta}^2 \\ -Mgl \sin(\theta) \end{bmatrix}$$

The VR Sink block is used to output the position of the motor and the angle of the rope to the virtual world.

Lighting Example (vrlights)

vrlights is an example with light sources. In the associated VRML file, several viewpoints are defined that allow you to observe the gradual changes in light from various perspectives.

Magnetic Levitation Model Example (vrmaglev)

vrmaglev is an example showing the interaction between dynamic models in Simulink and virtual worlds. The Simulink model represents the HUMUSOFT CE 152 Magnetic Levitation educational / presentation scale model. The plant model is controlled by a PID controller with feed-forward to cope with the nonlinearity of the magnetic levitation system.

The position of the ball responds to the changing value of the set point. You can observe this change not only in the Scope window, but also with a VRML viewer displaying the virtual world. To display the virtual world, double-click the VR Sink block, then click the **View** button in the dialog box. After switching the input from the signal generator to the VR Source block, you can use your mouse to drag the ball to a new position. Note that the position of the ball represents the real dynamics of the system. If you switch the VRML viewer to the Camera 3 viewpoint, you can observe and control the ball more easily.

You achieve the dragging effect by using the VRML PlaneSensor attached to the ball geometry with its output restricted to <0,1> in the vertical coordinate and processed by the VR Source block.

Magnetic Levitation Model for Real-Time Windows Target Example (vrmaglev_rtwin)

In addition to the vrmaglev example, the vrmaglev_rtwin example works directly with the actual CE 152 scale model hardware in real time. The MathWorks created this model to work with Real Time Workshop, Real Time Windows Target, and the HUMUSOFT AD 512 data acquisition board. However, you can adapt this model for other targets and acquisition boards. A digital IIR filter, from the Signal Processing Toolbox, filters the physical system output. You can bypass the physical system by using the built-in plant model.

Running this model in real time is an example showing the capabilities of Simulink in control systems design and rapid prototyping. When you change the position of the virtual ball using your mouse, the real ball follows the position you set. When you push the real ball up or down against the coil force, the position of the virtual ball changes in the virtual world. If you remove the ball from the real system, the ball also disappears from the virtual world.

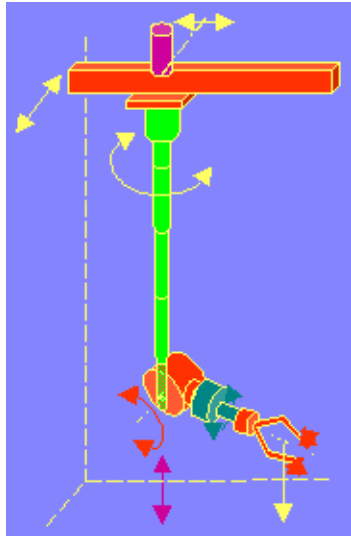
It is important to realize the power of this virtual reality operator control concept. Instead of saying, "I want the ball at the position of 0.4" you can say, "I want the object to be here," and it goes there.

Note that after enabling the remote view in the VR Sink block dialog box, you can control the Simulink model even from a client computer. This can be useful for distributing the computing power between a real-time Simulink model running on one machine and the rendering of a virtual reality world on another machine.

To work with this model, use as powerful a machine as possible or split the computing/rendering over two machines.

Manipulator with SpaceMouse Example (vrmanipul)

This example illustrates the use of the Virtual Reality Toolbox for virtual reality prototyping and testing the viability of designs before the implementation phase. Also, this example illustrates the use of the Magellan SpaceMouse for manipulating objects in a virtual world. Note that you must have the Magellan SpaceMouse in order to run this demo.



The VRML model represents a nuclear hot chamber manipulator. It is manipulated by a simple Simulink model containing the SpaceMouse Input block. This model uses all six degrees of freedom of the SpaceMouse for manipulating the mechanical arm, and this model uses mouse button 1 to close the grip of the manipulator jaws.

Magellan SpaceMouse is an input device with six degrees of freedom. It is useful for navigating and manipulating objects in a virtual world. SpaceMouse is also suitable as a general input device for Simulink models. This professional three-dimensional device greatly facilitates all the previously mentioned tasks. You can use the SpaceMouse for higher performance applications and user comfort. SpaceMouse is supported through the SpaceMouse Input block, which is included in the Virtual Reality Toolbox block library for Simulink.

The SpaceMouse Input block can operate in three modes to cover the most typical use of such a device in a three-dimensional context:

- Speeds
- Positions
- Viewpoint coordinates

Rotating Membrane Example (vrmemb1)

The vrmemb1 example is similar to the vrmemb example, but this time the associated virtual world is driven from a Simulink model.

Inverted Pendulum Example (vrpend)

The vrpend example illustrates the various ways a dynamic model in Simulink can interact with a virtual reality scene. It is the model of a two-dimensional inverted pendulum controlled by a PID controller. What distinguishes this model from “common” inverted pendulum models are the methods for setting the set point. You visualize and interact with a virtual world by using a Trajectory Graph and VR Sink blocks. The Trajectory Graph block allows you to track the history of the pendulum position and change the set point in three ways:

- **Mouse** — Click and drag a mouse pointer in the **Trajectory Graph** two-dimensional window
- **Input Signal** — External Trajectory Graph input in this model (driven by a random number generator)
- **VR Sensor** — Activates the input from a VRML **TouchSensor**

When the pointing device in the VRML viewer moves over an active **TouchSensor** area, the cursor shape changes. The triggering logic in this model is set to apply the new set point value with a left mouse button click.

Notice the pseudoorthographic view defined in the associated VRML file. This effect is achieved by creating a **Viewpoint** that is located far from the object of interest with a very narrow view defined by the VRML **FieldOfView** parameter. An orthographic view is useful for eliminating the panoramic distortion that occurs when you are using a wide-angle lens. The disadvantage of this technique is that locating the viewpoint at a distance makes the standard viewer navigation tricky or difficult in some navigation modes, such as the Examine mode. If you want to navigate around the virtual pendulum bench, you should use some other viewpoint.

Solar System Example (vrplanets)

The `vrplanets` example shows the dynamic representation of the first four planets of the Solar system, Moon orbiting around Earth, and Sun itself. The model uses the real properties of the celestial bodies. Only the relative planet sizes and the distance between the Earth and the Moon are adjusted, to provide an interesting view.

There are several viewpoints defined in the virtual scene, both static and attached to an observer on Earth. You can see that the planet bodies are not represented as perfect spheres. Using the VRML **Sphere** graphic primitive, which is rendered this way, simplified the model. If you want to make the planets more realistic, you could use the more complex **IndexedFaceSet** node type.

Mutual gravity accelerations of the bodies are computed using Simulink matrix-type data support.

Plane Takeoff Example (vrtkoff)

The `vrtkoff` example represents a simplified aircraft taking off from a runway. There are several viewpoints defined in this model, both static and attached to the plane, allowing you to see the takeoff from various perspectives.

The model demonstrates the technique of combining several objects imported or obtained from different sources (CAD packages, general 3-D modelers, and so on) into a virtual reality scene. Usually it is necessary for you to wrap such imported objects with an additional VRML **Transform** node. This wrapper allows you to set appropriately the scaling, position, and orientation of the objects to fit in the scene. In this example, the aircraft model from the V-Realm Builder Object Library is incorporated into the scene. The file `vrtkoff2.wrl` uses the same scene with a different type of aircraft.

MATLAB Interface Examples

The following table is a list of the MATLAB interface examples provided with the Virtual Reality Toolbox. Descriptions of the examples follow the table.

Example	RTW Ready	VR Source	VR Sink	Joystick	Space Mouse
vrcar			X		
vrheat			X		
vrmemb			X		

Car in the Mountains Example(vrcar)

This demonstration illustrates the use of the Virtual Reality Toolbox with the MATLAB interface. In a step-by-step tutorial, it shows commands for navigating a virtual car along a path through the mountains.

- 1 In the MATLAB Command Window, type

```
vrcar
```

A tutorial script starts running. Follow the instructions in the MATLAB Command Window.

Heat Transfer Example (vrheat)

This demonstration illustrates the use of the Virtual Reality Toolbox with the MATLAB interface for manipulating complex objects.

In this demonstration, matrix-type data is transferred between MATLAB and a virtual reality world. Using this feature, you can achieve massive color changes or morphing. This is useful for representing various physical processes. Pre-calculated data of time-based temperature distribution is used in an L-shaped metal block. The data is then sent to the virtual world. This forms an animation with relatively large changes.

This is a step-by-step demonstration. Shown are the following features:

- Reshaping the object
- Applying the color palette to represent distributed parameters across an object shape
- Working with VRML text objects
- Animating a scene using MATLAB interface
- Synchronization of multiple scene properties

At the end of this example, you can preserve the virtual world object in the MATLAB workspace, then save the resulting scene to a corresponding VRML file or carry out other subsequent operations on it.

Rotating Membrane with MATLAB GUI Example (vrmemb)

The `vrmemb` example shows how to use a MATLAB-generated 3-D graphic object with the Virtual Reality Toolbox. The membrane was generated by the `logo` function and saved in the VRML format using the standard `vrm1` function. You can save all Handle Graphics objects this way and use them with the Virtual Reality Toolbox as components of associated virtual worlds.

After starting the demo, you see a control panel with two sliders and three check boxes. Use the sliders to rotate and zoom the membrane while you use the check boxes to determine the axis to rotate around.

In the VRML scene, notice the text object. It is a child of the VRML **Billboard** node. You can configure this node so that its local z -axis turns to point to the viewer at all times. This can be useful for modeling virtual control panels and head-up displays (HUDs).

Implementation Notes

This section includes the following topics:

- **Virtual Reality Toolbox Server** — Accesses information about VRML scenes, provides an interface between MATLAB and Simulink, and communicates with clients
- **VRML Compatibility** — Limitations on support for VRML97 features

Virtual Reality Toolbox Server

The Virtual Reality Toolbox uses an internal HTTP Server for communication between a Web browser and the MATLAB/Simulink environment. It generates the main Virtual Reality Toolbox HTML page with the list of currently available virtual worlds and sends VRML and other requested files and data to clients (VRML viewers).

The server is started when the Virtual Reality Toolbox is loaded into MATLAB. This happens whenever you use a Virtual Reality Toolbox block in a Simulink block diagram, or whenever you open a vrworld object in the MATLAB interface. The HTTP Server is shut down when you close all Simulink models that contain Virtual Reality Toolbox blocks, or use the `vrclear` command.

When the HTTP Server is running, your browser can see a list of available virtual worlds at the following address:

```
http://localhost:port_number
```

Remote users can connect to the following address:

```
http://your_machine:port_number
```

You can set the port number of the server in the **Virtual Reality Toolbox Preferences** dialog box from the Simulink interface, or use `vrsetpref` in the MATLAB Command Window.

Depending on the status of served vrworld objects, the list of available virtual worlds can be empty.

VRML Compatibility

Virtual Reality Toolbox currently supports most features of VRML97, with the following limitations:

- The Virtual Reality Toolbox Server ignores the VRML Script node, but it passes the node to the VRML viewer. This allows you to run VRML scripts on the viewer side. You cannot run them on the Virtual Reality Toolbox Server.
- The Virtual Reality Toolbox Server ignores the Inline node, but it passes the node to the viewer. Therefore, the viewer sees the complete virtual world with all included substructures, but the included parts are not accessible from the toolbox. In some rare cases, this limitation can render the virtual world unusable with the Virtual Reality Toolbox. This happens under either of the following conditions:
 - The virtual world contains a USE reference to a node that is in the included part.
 - The virtual world contains an included part with a PROTO or EXTERNPROTO declaration that is referenced in the main virtual world file.

For a complete list of VRML97 nodes, refer to the VRML97 specification.

Installation

The Virtual Reality Toolbox Version 3.0 is distributed on the Release 13 CD. This CD has the files you need for installation on both your host computer and client computer.

System Requirements (p. 2-2)	Minimum hardware and software requirements to run the Virtual Reality Toolbox with MATLAB and Simulink
Installing the Virtual Reality Toolbox on the Host Computer (p. 2-9)	Install the Virtual Reality Toolbox on your desktop computer
Installing the VRML Viewer on the Host Computer (p. 2-15)	Install a viewer to view virtual worlds
Installing the VRML Editor on the Host Computer (p. 2-25)	Install VRML authoring tools to create virtual worlds
Removing Components (p. 2-31)	Uninstalling the Virtual Reality Toolbox and its components
Installation on the Client Computer (p. 2-33)	Install a viewer on another computer to view virtual worlds remotely
Testing the Installation (p. 2-35)	Open a Simulink model, display a virtual world, and run a simulation

System Requirements

The Virtual Reality Toolbox has the same hardware requirements as MATLAB. It is a multiplatform product that runs on PC-compatible computers with Windows or Linux. It also runs on SGI, Solaris, and Alpha hardware. For a list of supported operating systems, see “Supported Computer Platforms” on page 2-2.

This section includes the following topics:

- **Supported Computer Platforms** — Summary of the supported computer platforms and the viewer and editor that are provided for each of them.
- **Host Computer** — Run MATLAB, Simulink, the Virtual Reality Toolbox, VRML editor, and VRML viewer (Virtual Reality Toolbox viewer or Web browser with VRML plug-in).
- **Client Computer** — Run a Web browser with a VRML plug-in.

Supported Computer Platforms

The VR server is the part of the Virtual Reality Toolbox that interfaces with your Simulink models. It stores information about the current state of virtual worlds and manages connections to VR clients. The VR client is a VRML viewer that displays a virtual world. The VR client can be either the Virtual Reality Toolbox viewer or a Web browser with a VRML plug-in.

The following table summarizes the supported computer platforms and the viewer and editor that are provided for each of them.

Platform/Product	VR Server	Virtual Reality Toolbox Viewer	VRML Editor	VRML Browser Plug-In
Microsoft Windows 98, Windows NT 4.0, Windows XP, Windows ME, or Windows 2000	Yes	Yes	V-Realm Builder*	blaxxun Contact*
Linux 2.2.x and 2.4.x kernels	Yes	Yes	MATLAB editor*	No
SGI IRIX and IRIX64 6.5.8 (minimum)	Yes	Yes	MATLAB editor*	No (Cosmo Player)
Sun Solaris 2.6, 2.7, 2.8	Yes	Yes	MATLAB editor*	No
Compaq Alpha Tru64 UNIX 4.0f (minimum), 5.0, 5.1	Yes	Yes	MATLAB editor*	No

* Distributed on the MathWorks Release 13 product CD.

Host Computer

The host computer is a desktop computer where you install MATLAB, Simulink, the Virtual Reality Toolbox, a VRML editor and, optionally, a Web browser with a VRML plug-in. You can also install Real-Time Workshop with Real-Time Windows Target or xPC Target to run and view a real-time application.

The following table lists the minimum resources the Virtual Reality Toolbox requires on the host computer.

Hardware Requirements

Hardware	Description
CPU	Pentium, Athlon or higher (PC)
Graphics card	Graphics card with hardware 3-D acceleration
RAM	128 Mbytes or more
Peripherals	Hard disk drive with 45 Mbytes of free space CD-ROM drive
TCP/IP Communication	If you want to allow a connection from a client computer, you need a network connection between the host computer and the client computer.

The following table lists the minimum software the Virtual Reality Toolbox requires on your host computer. For a list of optional software products related to the Virtual Reality Toolbox, see “Related Products” in the Preface.

Software Requirements

Software	Description
Operating system	Windows 98, Windows NT 4.0, Windows ME, Windows XP, or Windows 2000 Sun Solaris 2.6, 2.7, 2.8 SGI IRIX and IRIX64 6.5.8 (minimum) Compaq Alpha Tru64 UNIX 4.0f (minimum), 5.0, 5.1 Linux 2.2.x or 2.4.x kernels The TCP/IP protocol needs to be installed.
MATLAB	Version 6.5 on the Release 13 CD.
Simulink	Version 5.0 on the Release 13 CD. Simulink is not required, but we highly recommend that you install it.
Virtual Reality Toolbox	Version 3.0 on the Release 13 CD.
VRML editor	For Windows platforms, you can install the VRML editor (V-Realm Builder 2.0) provided on the MathWorks CD. For UNIX/Linux the default editor is the MATLAB editor. When you create VRML worlds on these operating systems, you can use any 3-D modeling tool with the VRML97 export capability.

Software Requirements (Continued)

Software	Description
Web browser	<p>On PC platforms, you can use a Web browser and the blaxxun Contact plug-in to view virtual worlds. This is an alternative to using the Virtual Reality Toolbox viewer.</p> <p>Use Microsoft Internet Explorer 4.0 or higher, or Netscape Navigator 4.0 or higher with Java enabled.</p>
VRML plug-in	<p>If you are using a Web browser instead of the Virtual Reality Toolbox viewer, you need to install a VRML97 plug-in with External Authoring Interface (EAI) support. If you have blaxxun Contact (Windows) or Cosmo Player (SGI) on your computer, you have already installed a VRML plug-in.</p> <p>Windows platforms — You can install the blaxxun Contact 4.4 plug-in provided on the MathWorks CD, or you can download it from the MathWorks Web site at http://www.mathworks.com/support/product/VR/</p> <p>For information on how to install the blaxxun Contact plug-in, see “Installing a VRML Plug-In (Windows)” on page 2-16.</p>

Client Computer

You can use a client computer to view and control a virtual world. Because MATLAB or Simulink does not run on this computer, you need to connect to a host computer running a simulation or executable code. The host computer, through the VR Server, provides the values needed to animate a virtual world.

The client computer communicates with the host computer over TCP/IP, and it displays the virtual world using a VR client. In this case, the VR client is a VRML-enabled Web browser. You can verify the TCP/IP connection between the host and client computers by using the ping command from a command-line prompt. If there are problems, you must first fix the TCP/IP protocol settings according to the documentation for your operating system.

The following table lists the minimum hardware resources the Virtual Reality Toolbox needs on the client computer.

Hardware Requirements

Hardware	Description
Graphics card	Graphics card with hardware 3-D acceleration.
TCP/IP Communication	If you want to allow a connection from a client computer, you need a network connection between the host computer and the client computer.

The following table lists the software the Virtual Reality Toolbox requires on the client computer. You do not need to install the Virtual Reality Toolbox on the client computer.

Because the only component required for the client computer is standard VRML97 viewing software, it is possible that different configurations will work. For example, you might be able to run an operating system not listed in the table “Supported Computer Platforms” on page 2-2. However, these configurations have not been tested and they are not supported.

Software Requirements

Software	Description
Operating system	Windows 98, Windows NT 4.0, Windows ME, Windows XP, or Windows 2000 SGI IRIX and IRIX64 6.5.8 (minimum) The TCP/IP protocol needs to be installed.
Web browser	Use Microsoft Internet Explorer 4.0 or higher, or Netscape Navigator 4.0 or higher with Java enabled.
VRML plug-in	VRML97 plug-in with External Authoring Interface support. If you have blaxxun Contact (Windows) or Cosmo Player (SGI) on your computer, you have already installed a VRML plug-in. Windows platforms — You can install the blaxxun Contact 4.4 plug-in provided on the MathWorks CD, or you can download it from the MathWorks Web site at http://www.mathworks.com/support/product/VR/ For information on how to install the blaxxun Contact plug-in, see “Installing a VRML Plug-In (Windows)” on page 2-16.

Installing the Virtual Reality Toolbox on the Host Computer

The Virtual Reality Toolbox Version 3.0 is distributed on the Release 13 CD. For Web downloads, you need your MATLAB Access Number. Before you install the Virtual Reality Toolbox, you need to get a valid license file and/or personal license password. For detailed information about the installation process, see the installation documentation for your platform.

This section contains the following topics:

- **Getting or Updating Your License** — Valid license file and personal license password (PLP)
- **Components on a Host Computer** — Description of the individual components used with the Virtual Reality Toolbox
- **Installing from CD (Windows)** — PC installation procedure
- **Installing from CD (UNIX/Linux)** — UNIX/Linux installation procedure
- **Downloading from the Web**— Downloading the product from the Web

Getting or Updating Your License

Before you install the Virtual Reality Toolbox, you must have a valid license file and/or personal license password (PLP). The license file and/or personal license password identify the products you purchased from The MathWorks. These are the products you are permitted to install and use.

When you purchase a product, The MathWorks sends you a license file and/or personal license password (PLP) in an e-mail message. If you have not received a PLP number, contact The MathWorks.

Internet <http://www.mathworks.com/mla>

Log in to MATLAB Access using your last name and Access number. Follow the license links to determine your PLP number.

E-mail <mailto:service@mathworks.com>. Include your license number.

Telephone 508-647-7000. Ask for Customer Service.

Fax 508-647-7001. Include your license number.

Components on a Host Computer

This section introduces you to the individual components of the Virtual Reality Toolbox: what they are, what they are used for, and when they should or should not be installed. If you are not interested, you can skip this section, or you can simply accept the defaults at the component selection screen, and the recommended default components are installed:

- **Virtual Reality Toolbox** — This component contains the core files that interconnect MATLAB and Simulink to VRML. This component is required for the Virtual Reality Toolbox to operate, and you must install it on the host computer. This component is *not* used on a client computer.
- **Virtual Reality Toolbox viewer** — This is a multiplatform VRML viewer that is included with the Virtual Reality Toolbox, and it is set as the default viewer for displaying virtual worlds.
- **VRML plug-in** — Optionally, you can use a VRML plug-in for a Web browser to view virtual reality worlds. The blaxxun Contact plug-in is included with the Virtual Reality Toolbox for Windows platforms. However, you can also use the Virtual Reality Toolbox viewer. A VRML plug-in is the only component that you need to install on a client computer.
- **VRML editor** — If you are going to create and modify virtual worlds, you need a VRML97-compatible editor. V-Realm Builder is included on the MathWorks CD for Windows platforms. If you do not plan to edit virtual reality worlds or if you prefer to use a different VRML editor, you do not need to install it on your computer. For UNIX/Linux platforms, the MATLAB editor is the default VRML editor. This component is *not* used on a client computer.
- **Example Models** — These are MATLAB and Simulink programs and models connected to prebuilt virtual reality worlds. You can use these models and virtual reality worlds both for discovering the capabilities of the Virtual Reality Toolbox and as templates for building your own projects. This component is not used on the client computer.
- **Online Documentation** — This component contains the manual you are reading now. You can access the online version through the MATLAB Help browser. An Adobe Acrobat PDF file is available on the Release 13 CD. This documentation can be read using the Adobe Acrobat Reader. If you do not have this reader installed on your computer, you can download it from <http://www.adobe.com>.

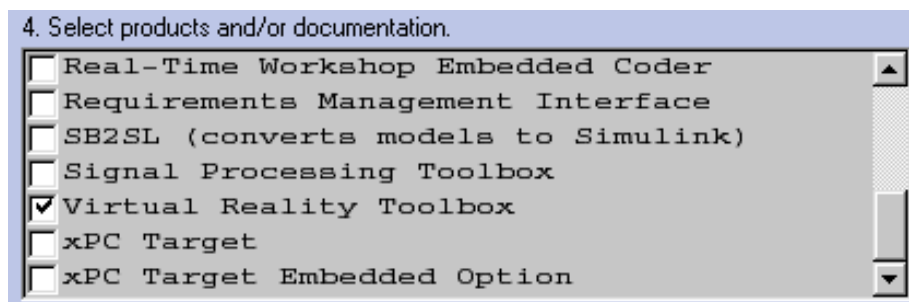
Installing from CD (Windows)

You can install the Virtual Reality Toolbox from The MathWorks Release 13 CD:

- 1 Insert the Release 13 CD into your host CD-ROM drive.

The installation program should start automatically after a few seconds. If the installation program does not start automatically, run `setup.exe` on the CD.

During the installation process, a screen similar to the following allows you to select the products to install.



- 2 Select the **Virtual Reality Toolbox** check box, then click **Next**.
- 3 Follow the instructions on each of the remaining screens.

Installation for the Virtual Reality Toolbox is complete.

The Virtual Reality Toolbox viewer is installed with the Virtual Reality Toolbox. For PC platforms, you have the option of installing a VRML plug-in for your browser as an alternative to the viewer. See “Installing a VRML Plug-In (Windows)” on page 2-16.

If you are on a PC platform, you need to complete additional steps for installing the VRML editor. See “Installing VRML Editor (Windows)” on page 2-25.

Installing from CD (UNIX/Linux)

The following is an overview of how to install the Virtual Reality Toolbox on a UNIX/Linux platform. If you have not installed any MathWorks products before, consult the installation guide for your platform for a more comprehensive explanation of the installation process:

- 1 Log in to your system.
- 2 Mount the CD-ROM drive.
- 3 Create a directory to be the mount point for the CD-ROM drive. For example:

```
mkdir /cdrom
```

- 4 Create the installation directory and move into it using the `cd` command. For example, to install into the location `/usr/local/matlab6p5`, use these commands:

```
cd /usr/local
mkdir matlab6p5
cd matlab6p5
```

Subsequent instructions in this book refer to this directory as `$MATLAB`.

Note This installation directory might already exist if you have installed MATLAB on your system. In this case, move into the already existing directory using the `cd` command.

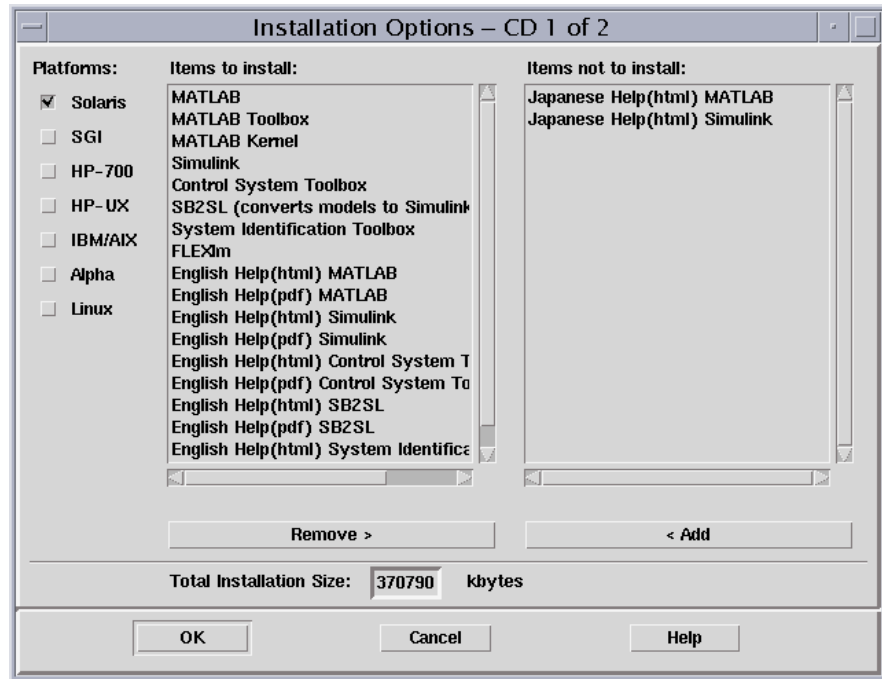
- 5 Move your license file, named `license.dat`, into the `$MATLAB` directory.

If you are upgrading an existing MATLAB installation, rename the license file in `$MATLAB/etc` directory. The installer does not process the new license file if it finds an existing license file in `$MATLAB/etc`.

- 6 Run the appropriate installation script for your platform.

```
/cdrom/install* & (Sun, Alpha, SGI, and Linux platforms)
```

- 7 During the installation process, a screen similar to the following allows you to select the products to install.



This dialog box lists all the products you are licensed to install in the **Items to Install** box. Make sure the Virtual Reality Toolbox is listed in this box.

- 8 Follow the instructions on each of the remaining screens.

Installation for the Virtual Reality Toolbox is complete.

The Virtual Reality Toolbox viewer is the default viewer for UNIX platforms. For more information, see “Virtual Reality Toolbox Viewer” on page 2-15.

If you are on a UNIX platform, the MATLAB editor is your default VRML editor. For more information, see “VRML Editor (UNIX/Linux)” on page 2-26.

Downloading from the Web

Version 3.0 of the Virtual Reality Toolbox is available for Web download. You download products from the Web when you want to obtain a demo, product update, or any product available on a MATLAB installation CD:

- 1 Open your Web browser and navigate to <http://www.mathworks.com>.
- 2 From the list on the right side of the page, select **Downloads**.
- 3 Under **MATLAB Access Members**, select **download products**.

The **Access Login** page appears.

- 4 Enter your **Last name** and **MATLAB Access Number**.
- 5 Click **login**.

The **downloads** page appears.

- 6 Select your platform and click **Continue**.
- 7 Select the **Virtual Reality Toolbox** and click **Continue**.
- 8 Follow the instructions on the **Download and Install** page in order to download and install the Virtual Reality Toolbox successfully. For more specific information relating to the installation of the Virtual Reality Toolbox, see the installation guide for your platform.

Note The most recent PDF documentation file is not always included in the product download. To get the latest PDF file for a product, go to <http://www.mathworks.com/access/helpdesk/help/helpdesk.shtml> and select the product's name. The Roadmap Page for the selected product appears. This Roadmap page contains a link to the latest version of the PDF documentation.

Installing the VRML Viewer on the Host Computer

You can use the Virtual Reality Toolbox viewer or VRML-enabled Web browser to view virtual worlds. The Virtual Reality Toolbox viewer is the only viewer that can be used on all supported platforms. The blaxxun Contact plug-in is available for PC platforms only.

This section includes the following topics:

- **Virtual Reality Toolbox Viewer** — Preferred method of viewing virtual scenes.
- **Installing a VRML Plug-In (Windows)** — Install the blaxxun Contact plug-in.
- **Installing a VRML Plug-in (UNIX/Linux)** — Install a VRML97 plug-in with External Authoring Interface support.
- **Setting the Default Viewer of Virtual Scenes** — View virtual scenes with the Virtual Reality Toolbox viewer or your VRML-enabled Web browser.

Virtual Reality Toolbox Viewer

The Virtual Reality Toolbox viewer is the preferred method of viewing a virtual scene. The viewer can be used on any supported operating system. It is installed and set as the default viewer when you install the Virtual Reality Toolbox. You can view virtual scenes as soon as the Virtual Reality Toolbox is installed on your machine.

Note It is possible to view virtual scenes with a Web browser that contains a VRML plug-in. Every VRML plug-in installs Java classes into the Web browser. It is best to limit the number of plug-ins you install on your machine in order to avoid Java errors and conflicts. For this reason, use only the Virtual Reality Toolbox viewer and the blaxxun Contact VRML plug-in on PC platforms. On UNIX and Linux platforms, use only the Virtual Reality Toolbox viewer.

Installing a VRML Plug-In (Windows)

When you install the Virtual Reality Toolbox, the Virtual Reality Toolbox viewer is set as the default viewer. If you want to use a Web browser as a VRML viewer, use the following procedure to install the blaxxun Contact plug-in. You can use this plug-in with either Microsoft Internet Explorer or Netscape Navigator. The blaxxun Contact plug-in is the only supported VRML plug-in.

Note The blaxxun Contact installer installs the plug-in for the current default browser only. If you change the default browser, you need to complete the install procedure a second time. The blaxxun Contact executable files are located at C:\<MATLAB root>\toolbox\vr\blaxxun.

You must use blaxxun Contact 4.4 with Version 3.0 of the Virtual Reality Toolbox. This version of the blaxxun Contact VRML plug-in is distributed with the Virtual Reality Toolbox. You can download blaxxun Contact 4.4 from <http://www.mathworks.com/support/product/VR/>.

If you have the MATLAB Web Server installed on your machine, make sure that the Web Server is stopped before you install the blaxxun Contact plug-in. Also, verify that you are connected to the Internet before starting this installation procedure:

1 Start MATLAB.

2 In the MATLAB Command Window, type

```
vrinstall -install viewer
```

MATLAB displays the message

```
Do you want to use OpenGL or Direct3d acceleration? (o/d)
```

3 Check the graphic card manual to determine the acceleration method to select. If you are not sure, select Direct 3d by typing

```
d
```


The blaxxun installer starts running and displays the following dialog box.



- 4 Follow the instructions on the remaining screens.
- 5 In the MATLAB Command Window, type

```
vrinstall -check
```

If the viewer installation was successful, MATLAB displays the following message:

```
VRML viewer:    installed
```

If the viewer installation was unsuccessful, MATLAB displays the message

```
VRML viewer:      not installed
```

Known Issue with the blaxxun Contact Plug-In

The blaxxun Contact VRML plug-in can fail to update the virtual scene when used with the Virtual Reality Toolbox 3.0 and Microsoft Internet Explorer 5.5 and above. Netscape users do not experience this problem.

If you are using Internet Explorer 5.5 or above, you must manually change a network security setting before you can use blaxxun Contact 4.4 with the Virtual Reality Toolbox Version 3.0. Upgrading your version of blaxxun Contact does not resolve this problem.

Changing the Default Network Security Setting

You must change your default network security setting before using the blaxxun Contact plug-in with Internet Explorer 5.5 and above to ensure that the virtual scene is updated appropriately:

- 1** Open Internet Explorer.
- 2** From the **Tools** menu, choose **Internet Options**.

The **Internet Options** dialog box opens.

- 3** Click the **Security** tab.
- 4** Select the **Custom Level** button.

The **Security Settings** dialog box opens.

- 5** Scroll down until you see **Microsoft VM**. The first subheading is **Java permissions**.
- 6** Select **Custom**.

The **Java Custom Settings** button appears in the lower left of the **Security Settings** dialog box.

- 7** Click **Java Custom Settings**.

The **Local intranet** dialog box opens.

- 8 Click the **Edit Permissions** tab.
- 9 Scan the main headings and subheadings (marked with a lock icon) until you see **Access to all Network Addresses**.
- 10 Under **Access to all Network Addresses**, select **Enable**.
- 11 Click **OK**.

The **Local intranet** dialog box closes.

- 12 In the **Security Settings** dialog box, click **OK**.

You are asked if you want to change the security settings for this zone.

- 13 Select **Yes**.

- 14 In the **Internet Options** dialog box, select **OK**.

Installing a VRML Plug-in (UNIX/Linux)

If you want to use a Web browser instead of the Virtual Reality Toolbox viewer to view virtual scenes, you need to install a VRML97 plug-in with External Authoring Interface (EAI) support. This requirement is met by blaxxun Contact for Windows platforms and Cosmo Player for SGI platforms. If you are using any other operating system, you need to use the Virtual Reality Toolbox viewer to view virtual worlds.

If you are on an SGI platform, you might already have the Cosmo Player plug-in installed in your Web browser:

- 1 At the MATLAB command prompt, type

```
vrsetpref('DefaultViewer','web')
```

This sets your default viewer of virtual scenes to your VRML-enabled Web browser.

2 Type

```
vrbounce
```

If the bouncing ball demo loads in your Web browser, Cosmo Player is already installed on your system. See “Setting the Default Viewer of Virtual Scenes” on page 2-20 for more information.

Go to <http://www.sgi.com/software/cosmo/player.html> for more information about Cosmo Player for SGI. If you do not have Cosmo Player installed on your system, follow the instructions provided on this Web site to install the Cosmo Player VRML plug-in into your default Web browser.

Note blaxxun Contact is the only supported VRML plug-in.

Setting the Default Viewer of Virtual Scenes

If you install a VRML plug-in in your Web browser, it is possible to view virtual scenes with either the Virtual Reality Toolbox viewer or your Web browser. You determine the viewer used to display your scene using the `vrsetpref` and `vrgetpref` commands. The following procedure assumes that you are working on a PC platform:

- 1 At the MATLAB command prompt, type

```
vrinstall -check
```

to determine whether blaxxun Contact is installed.

MATLAB displays

```
VRML viewer:    installed
VRML editor:    installed
```

The viewer and editor are installed. If the viewer is not installed, see “Installing a VRML Plug-In (Windows)” on page 2-16.

2 Determine your default viewer by typing

```
vrgetpref
```

MATLAB displays

```
ans =
```

```
DefaultFigurePosition: [5 25 400 320]  
DefaultPanelMode: 'halfbar'  
DefaultViewer: 'internal'  
Editor: [1x60 char]  
HttpPort: 8123  
TransportBuffer: 5  
VrPort: 8124
```

The `DefaultViewer` property is set to `'internal'`. The Virtual Reality Toolbox viewer is the default viewer for viewing virtual scenes. Any virtual scenes that you open are displayed in the viewer.

- 3 For example, at the MATLAB command prompt, type
`vrbounce`

The Bouncing Ball demo is loaded and the virtual scene is displayed in the Virtual Reality Toolbox viewer.



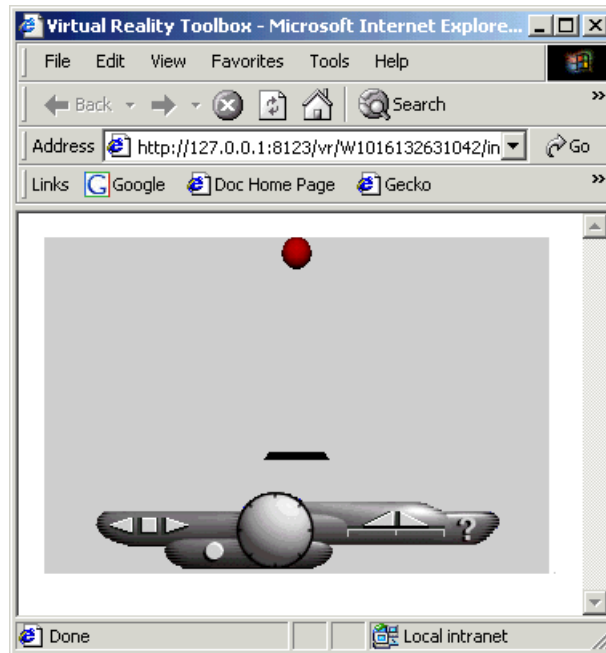
- 4 Change the default viewer to your Web browser by typing
`vrsetpref('DefaultViewer','web')`

The default Windows system VRML plug-in is used. The blaxxun Contact VRML plug-in sets itself as the default VRML plug-in during its installation.

- 5 At the MATLAB command prompt, type

```
vrbounce
```

The Bouncing Ball demo is loaded and the virtual scene is displayed in your Web browser.



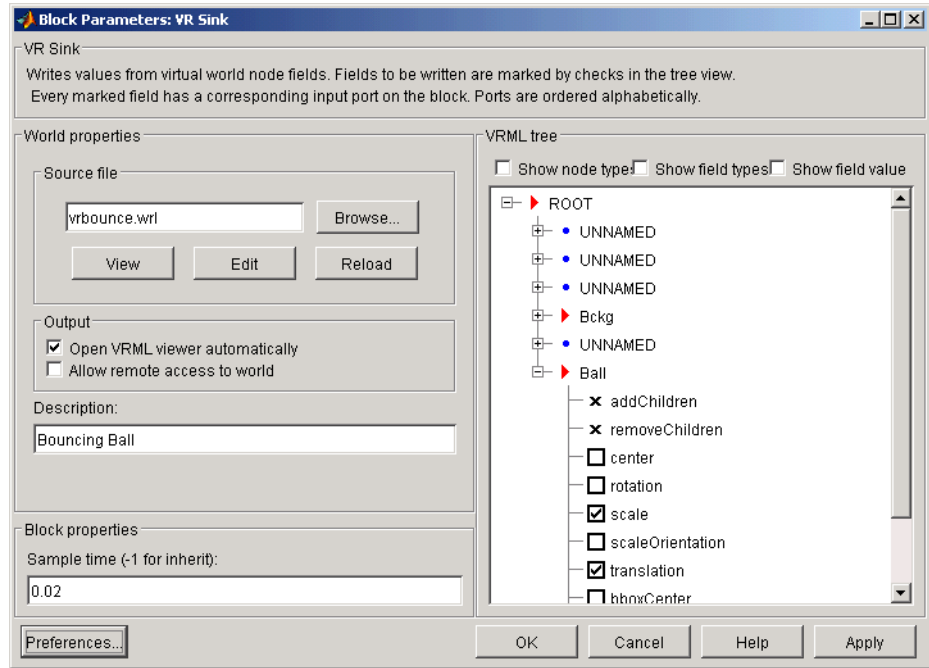
- 6 Reset the Virtual Reality Toolbox viewer as your default viewer by typing

```
vrsetpref('DefaultViewer','factory')
```

All virtual scenes are displayed by the Virtual Reality Toolbox viewer.

7 In the vrbounce model window, double-click the **VR Sink** block.

A **Block Parameters** dialog box opens.



The target of the **View** button is determined by the DefaultViewer property. If the DefaultViewer property is set to 'internal', clicking the **View** button opens the virtual world in the Virtual Reality Toolbox viewer. If the DefaultViewer property is set to 'web', clicking the **View** button opens the virtual world in your Web browser.

Installing the VRML Editor on the Host Computer

You can create virtual worlds with a VRML authoring tool or by writing VRML code in a text editor.

This section contains the following topics:

- **Installing VRML Editor (Windows)** — Install V-Realm Builder on your PC.
- **VRML Editor (UNIX/Linux)** — The MATLAB editor is the default VRML editor for UNIX platforms.
- **Setting the Default Editor of Virtual Scenes** — Edit virtual scenes with a VRML authoring tool or a text editor.

Installing VRML Editor (Windows)

When you install the Virtual Reality Toolbox, files are copied to your hard drive for V-Realm Builder, but the installation is not complete.

Installing the VRML editor writes a key to the Windows registry, making extra library files in V-Realm Builder available for you to use, and it associates the **Edit** button in Virtual Reality Toolbox blocks with this editor:

- 1 Start MATLAB.
- 2 In the MATLAB Command Window, type

```
vrinstall -install editor
```

or type

```
vrinstall('-install','editor')
```

MATLAB displays the following messages:

```
Starting editor installation...  
Done.
```

3 Type

```
vrinstall -check
```

If the editor installation was successful, MATLAB displays the following message:

```
VRML editor:    installed
```

VRML Editor (UNIX/Linux)

The MATLAB editor is the default VRML editor for UNIX platforms and no installation procedure is required. Currently, no VRML editor with the functionality of those available for Windows platforms exists for UNIX platforms. This means that you will need to understand and program the VRML language in order to create your virtual worlds. Alternatively, it is possible to use a general 3-D modeling tool with VRML97 export capabilities. For more information about the VRML modeling language, we recommend *Teach Yourself VRML2 in 21 Days* by Chris Marrin and Bruce Campbell.

Setting the Default Editor of Virtual Scenes

You can edit virtual scenes with a VRML authoring tool, such as V-Realm Builder, or with any text editor, as the VRML language is written in text files. You determine the editor that is used to edit your scene by using the `vrsetpref` and `vrgetpref` commands.

The following procedure demonstrates how to change your editor from V-Realm Builder to a text editor. It assumes that you are working on a PC platform:

1 At the MATLAB command prompt, type

```
vrinstall -check
```

to determine whether V-Realm Builder is installed.

MATLAB displays

```
VRML viewer:    installed
VRML editor:    installed
```

The viewer and editor are installed. If the editor is not installed, see “Installing VRML Editor (Windows)” on page 2-25.

2 Determine your default editor by typing

```
a = vrgetpref
```

MATLAB displays

```
a =  
DefaultFigurePosition: [5 25 400 320]  
DefaultPanelMode: 'halfbar'  
DefaultViewer: 'web'  
Editor: [1x60 char]  
HttpPort: 8123  
TransportBuffer: 5  
VrPort: 8124
```

The variable `a` is a structure array. You need to index into it to determine the `Editor` property.

3 To determine your default editor, type

```
a.Editor
```

MATLAB displays

```
ans =  
"%matlabroot\toolbox\vr\vrealm\program\vrbuild2.exe" "%file"
```

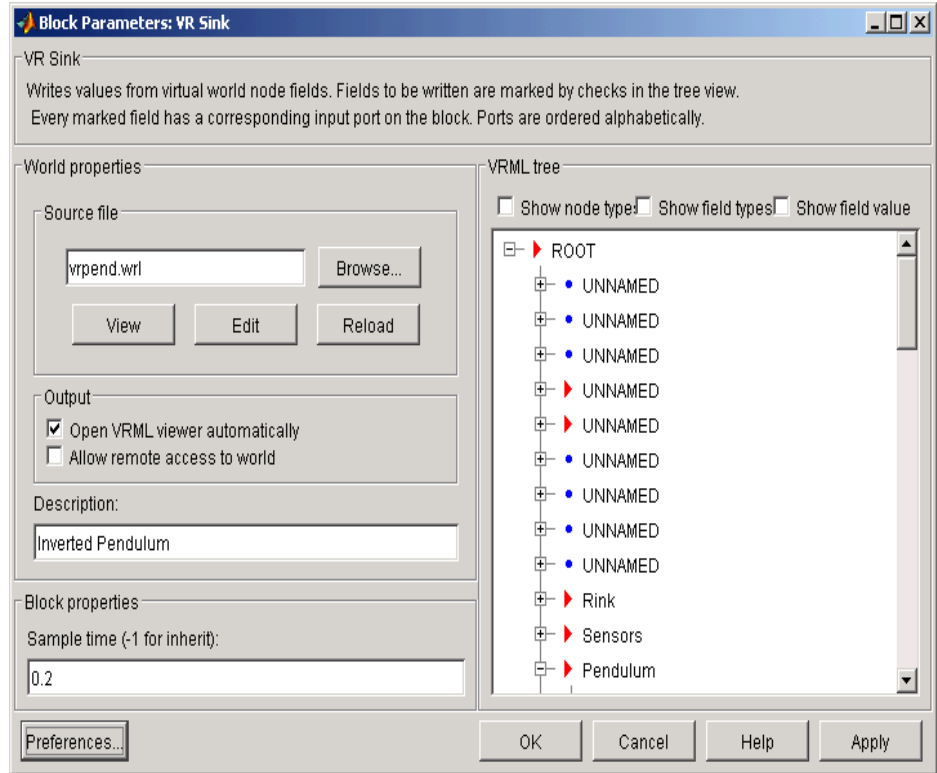
This is the path to the V-Realm Builder executable file. V-Realm Builder is the current VRML editor.

4 Verify that V-Realm Builder is your default editor. At the MATLAB command prompt, type

```
vrpend
```

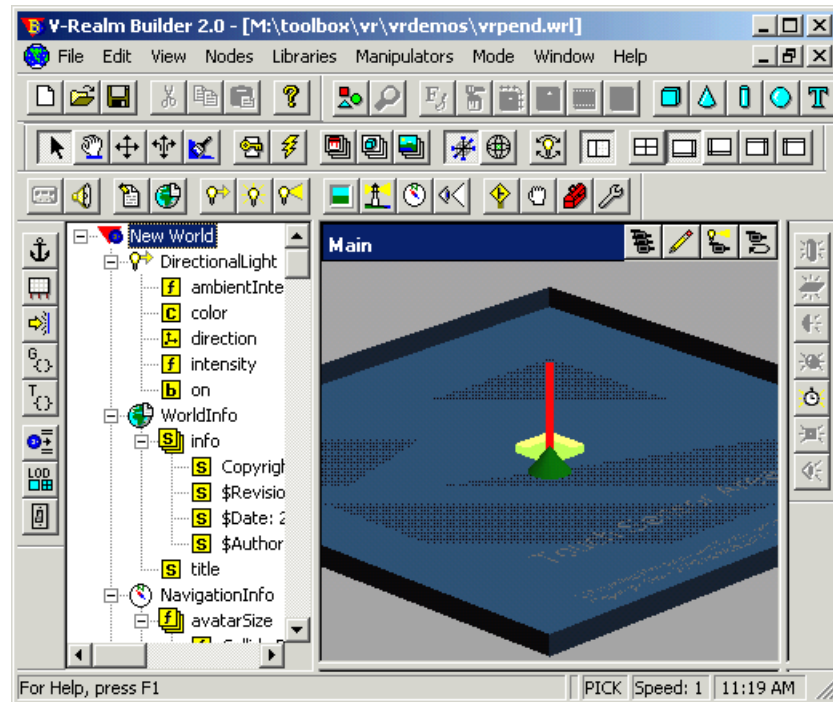
The Inverted Pendulum demo loads and the pendulum is visible in the viewer.

- 5 In the vrpend model window, double-click the **VR Sink** block.
The **Block Parameters** dialog box opens.



6 Click **Edit**.

The vrpemd model opens in the V-Realm Builder authoring tool.



7 Change the default editor to the MATLAB editor by typing

```
vrsetpref('Editor', '%matlabroot\bin\win32\meditor.exe %file')
```

You can set your editor to any text editor you want to use by specifying the path to the executable of the text editor.

8 Within the vrpemd demo, open the **VR Sink Block Parameters** dialog box.

- 9 Click the **Edit** button.

The MATLAB editor opens and is now set as your default VRML editor.

- 10 To reset the V-Realm Builder authoring tool as your default VRML editor, type

```
vrsetpref('Editor','factory')
```

Clicking the **Edit** button now launches V-Realm Builder.

Removing Components

Normally, you should not have to uninstall the Virtual Reality Toolbox, the blaxxun Contact plug-in, or V-Realm Builder. If you do, the following topics describe the appropriate procedures.

This section contains the following topics:

- **Removing the Virtual Reality Toolbox and V-Realm Builder** — Uninstalling the Virtual Reality Toolbox and V-Realm Builder
- **Removing the blaxxun Contact Plug-In** — Uninstalling the blaxxun Contact plug-in

Removing the Virtual Reality Toolbox and V-Realm Builder

Use the MathWorks uninstaller. Running this utility removes the Virtual Reality Toolbox and V-Realm Builder from your system. It also restores your previous system configuration:

- 1 On the task bar, click **Start**, point to **MATLAB**, and then click **R13 uninstaller**.

The MathWorks uninstaller begins running.

- 2 Clear the **Virtual Reality Toolbox** check box.
- 3 Follow the remaining uninstall instructions.

Note The blaxxun Contact plug-in is not uninstalled during the Virtual Reality Toolbox removal.

Removing the blaxxun Contact Plug-In

You can uninstall this VRML plug-in from the host computer by using the following procedure:

- 1 From the task bar, click **Start**, point to **Settings**, and click **Control Panel**.

- 2** In the **Control Panel** window, click **Add/Remove Programs**.
- 3** In the **Add/Remove Programs** dialog box, select **blaxxun Contact**, then click the **Add/Remove** button.

Installation on the Client Computer

In most configurations, you do not need to install a viewer on a client computer because you can perform all the tasks on a host computer. However, if you have very large models that take considerable computational resources, you might want to use a client computer to run and view the virtual world.

The client computer must have a VRML97 plug-in with External Authoring Interface (EAI) support. This means that your client computer must be a PC platform with the blaxxun Contact plug-in or an SGI platform with Cosmo Player. Of these two options, only blaxxun Contact is supported.

This section contains the following topics:

- **Installing a VRML Plug-In (Windows)** — Install the blaxxun Contact VRML plug-in on a computer running Microsoft Windows.
- **VRML Plug-In (UNIX/Linux)** — Install Cosmo Player on your SGI machine.

Installing a VRML Plug-In (Windows)

If you want to view a virtual world on a client computer, you need to use a Web browser with a VRML plug-in.

The blaxxun Contact plug-in is provided with the Virtual Reality Toolbox, but you cannot install the blaxxun Contact plug-in Version 4.4 on a client computer with the MathWorks installer. If you do not have this plug-in installed, use one of the following methods.

- Copy the file `blaxxuncontact44.exe` from your host computer to the client computer. This file is located at `C:\<MATLAB root>\toolbox\vr\blaxxun`.
- Download the blaxxun Contact plug-in from the MathWorks Web site at <http://www.mathworks.com/support/product/VR/>.

VRML Plug-In (UNIX/Linux)

Cosmo Player for SGI is currently the only VRML97 plug-in with External Authoring Interface (EAI) support available for UNIX/Linux platforms.

For more information about Cosmo Player for SGI, go to <http://www.sgi.com/software/cosmo/player.html>. If you do not have Cosmo Player installed on your system, follow the instructions provided on this Web site to install the Cosmo Player VRML plug-in in your default Web browser.

Note blaxxun Contact for PC platforms is the only supported method for viewing virtual worlds on a client computer.

Testing the Installation

The Virtual Reality Toolbox includes several Simulink models with the associated virtual worlds. These models are examples of what you can do with this toolbox. You can use one of these examples to test the installation of the Virtual Reality Toolbox, the VRML viewer, and the VRML editor.

This section contains the following topics:

- **Running a Simulink Interface Example** — Open a Simulink model for an inverted pendulum, start a simulation, and view the pendulum in a virtual world.
- **Running a MATLAB Interface Example** — View a virtual world of the MathWorks membrane.

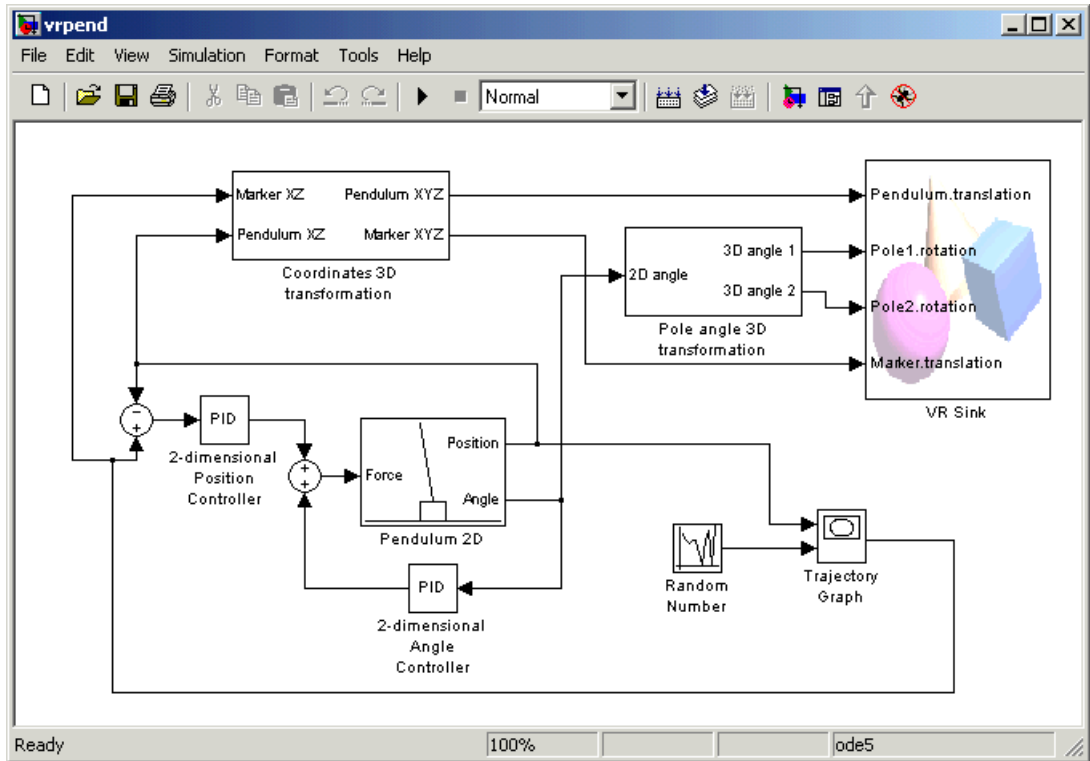
Running a Simulink Interface Example

In the demo directory for the Virtual Reality Toolbox, there is a Simulink model for a two-dimensional inverted pendulum. This model, which you can view in three dimensions with the toolbox, has an interactive set point and trajectory graph.

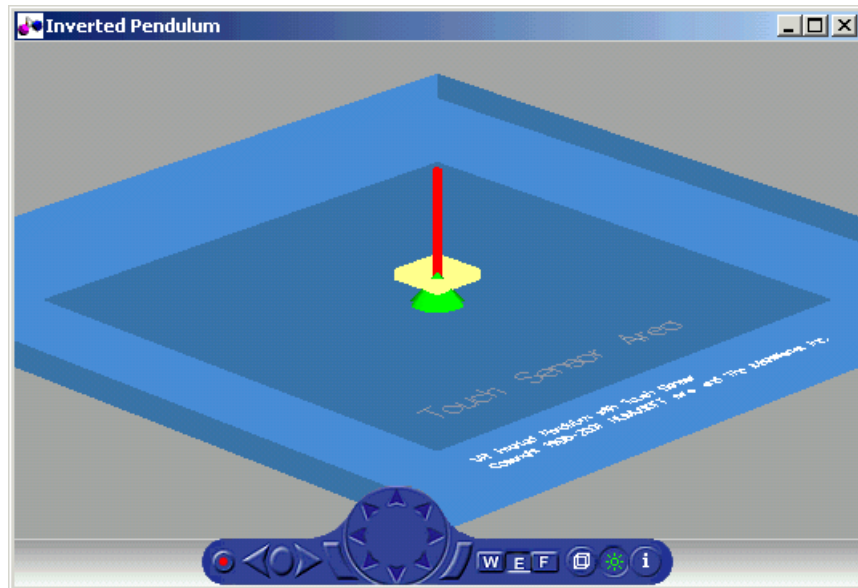
Before you can run this demo, you have to install MATLAB, Simulink, and the Virtual Reality Toolbox:

- 1 In the MATLAB Command Window, type
vrpend

A Simulink window opens with the model for an inverted pendulum.

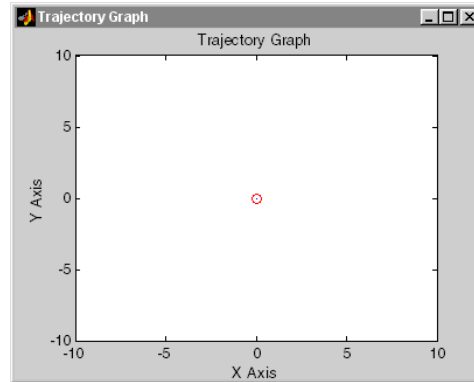


The Virtual Reality Toolbox viewer opens with a 3-D model of the pendulum.



Note Right-clicking in a virtual world displays a floating menu. From this menu, you can choose viewer settings and navigation modes. These settings include graphic quality and speed.

- 2 In the Simulink window, from the **Simulation** menu, click **Start**. A **Trajectory Graph** window opens, and a simulation starts running.



- 3 In the Virtual Reality Toolbox viewer window, point to a position on the blue surface and left-click.

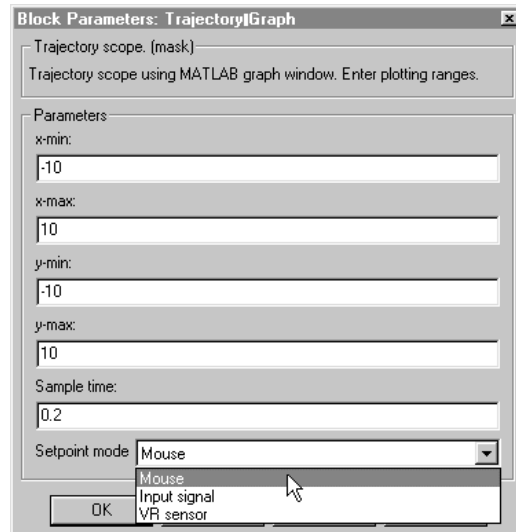
The pendulum set point, represented by the green cone, moves to a new location. Next, the path is drawn on the trajectory graph, and then the pendulum itself moves to the new location.

In the Virtual Reality Toolbox viewer window you see the animated movement of the pendulum. Use the viewer controls to navigate through the virtual world, change the viewpoints, and move the set point. For more information about using the Virtual Reality Toolbox viewer controls, see "Virtual Reality Toolbox Viewer" on page 5-21.

- 4 In the Simulink window, double-click the Trajectory Graph block.

The **Block Parameters: Trajectory Graph** dialog box opens.

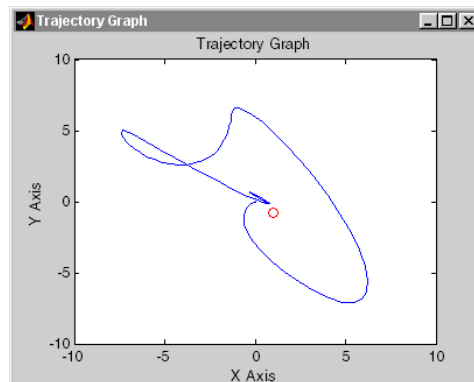
- From the **Setpoint mode** list, choose **Mouse**, then click **OK**.



You can now use the trajectory graph as a 2-D input device to set the position of the pendulum.

- Move the mouse pointer into the graph area and click.

The set point (red circle) for the pendulum position moves to a new location, and its trajectory is displayed as a blue line.



- 7 In the Simulink window, from the **Simulation** menu, click **Stop**. Close the Virtual Reality Toolbox viewer window and close the Simulink window.

You can try other examples in “Simulink Interface Examples” on page 1-14, or you can start working on your own projects.

Running a MATLAB Interface Example

This model, which can be viewed in three dimensions with the toolbox, has a MATLAB interface to control the figure in a VRML viewer window.

Additional examples are listed in the table “MATLAB Interface Examples” on page 1-21:

- 1 In the MATLAB window, type

```
vrmemb
```

MATLAB displays the following messages:

```
Loading...
```

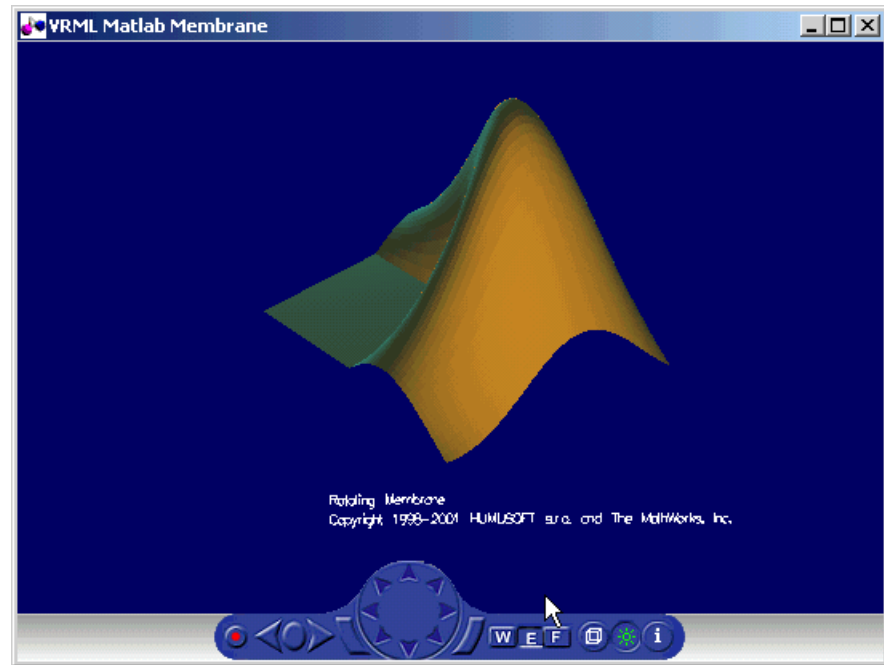
```
This example shows you how to use a MATLAB generated 3-D  
graphic object in the Virtual Reality Toolbox.
```

```
. . .
```

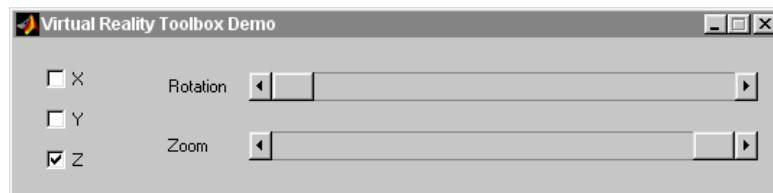
```
Press Enter to start the demonstration.
```


- 2 Press the **Enter** key.

The Virtual Reality Toolbox viewer opens with a 3-D model.



- 3 Use the viewer controls to move within the virtual world, or use the demo dialog box to rotate the membrane.



Note Sometimes the **Virtual Reality Toolbox Demo** dialog box is hidden behind the viewer window.

Simulink Interface

The Virtual Reality Toolbox works with both MATLAB and Simulink. However, the Simulink interface is the preferred way of working with the toolbox. It is more straightforward to use and all the toolbox features are easily accessible through a graphical user interface (GUI).

Associating a Virtual World with Simulink (p. 3-2)

Associate a Simulink model with a virtual world, and connect signals from the Simulink model to the virtual world

Using the Simulink Interface (p. 3-13)

Open a Simulink model, display the associated virtual world on a host computer or on a client computer, and observe the simulated process in the virtual world

Associating a Virtual World with Simulink

With the Virtual Reality Toolbox you can interface a Simulink block diagram with a virtual world. The example in this section explains how to display a simulated virtual world on a host computer. This is the recommended way to view associated virtual worlds on the host computer.

This section includes the following topics:

- **Adding a Virtual Reality Toolbox Block** — Connect a Simulink model to a virtual world
- **Changing the Virtual World Associated with a Simulink Block** — Change the virtual world associated with a Simulink model, and change the signals passed between Simulink and the virtual world

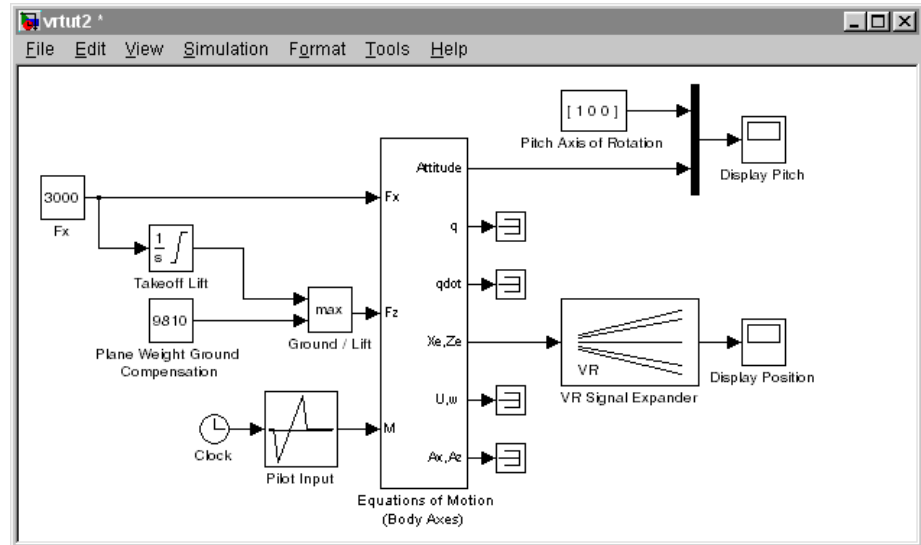
Adding a Virtual Reality Toolbox Block

Simulating a Simulink model generates signal data for a dynamic system. By connecting the Simulink model to a virtual world, you can use this data to control and animate the virtual world.

After you create a virtual world and a Simulink model, you can connect the two with Virtual Reality Toolbox blocks. The example in this procedure simulates a plane taking off and lets you view it in a virtual world:

- 1 In the MATLAB Command Window, type
`vrtut2`

A Simulink model opens without a Virtual Reality Toolbox block that connects the model to a virtual world.



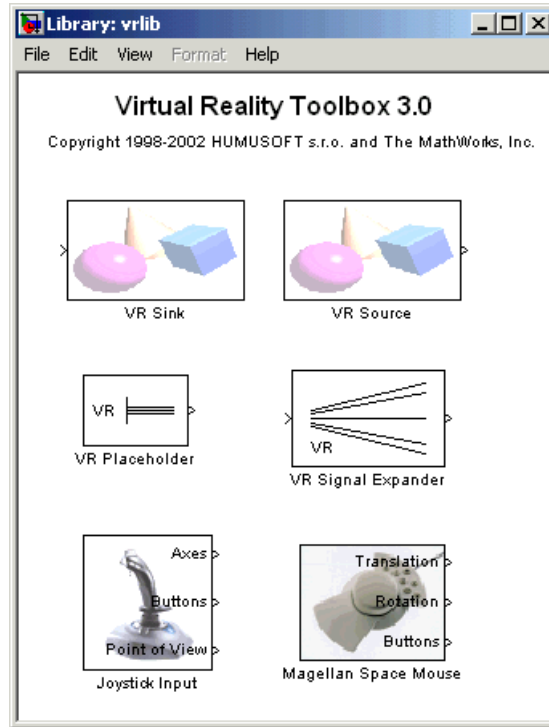
2 From the **Simulation** menu, select **Normal**, then click **Start**.

Observe the results of the simulation in the scope windows.

3 In the MATLAB Command Window, type

```
vrlib
```

The Virtual Reality Toolbox library opens.

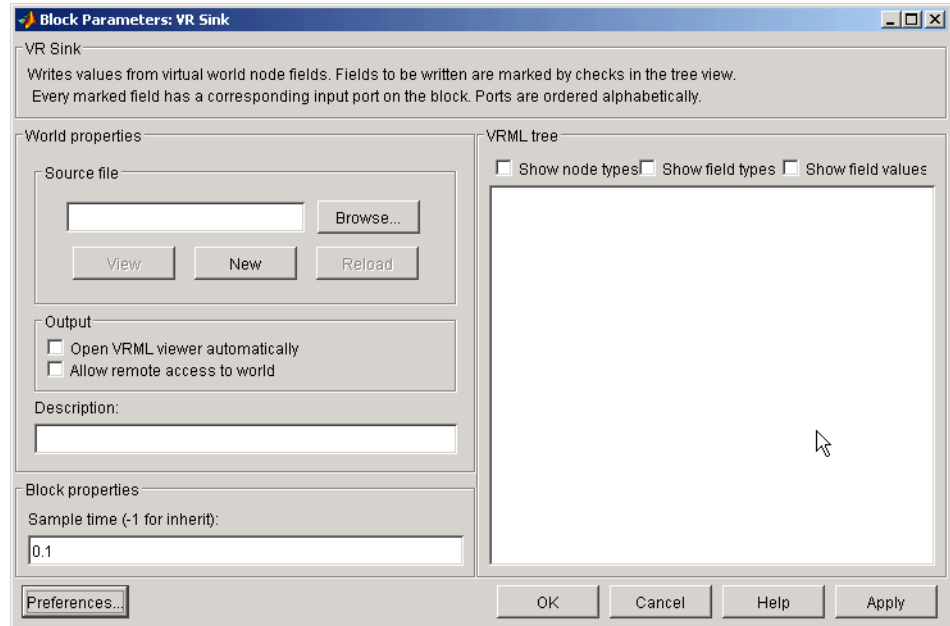


- 4 From the **Library** window, drag and drop the VR Sink block to the Simulink diagram. You can then close the **Library: vrlib** window.

Now you are ready to select a virtual world for the visualization of your simulation. A simple virtual world with a runway and a plane is in the VRML file `vrtkoff.wr1`.

- 5 In the Simulink model, double-click the block labeled VR Sink.

The **Block Parameters: VR Sink** dialog box opens.



- 6 In the **Description** text box, enter a brief description of the model. This description appears on the list of available worlds served by the Virtual Reality Toolbox server. For example, type
VR Plane taking off
- 7 Click the **Browse** button. The **Select World** dialog box opens. Find the directory `<matlab root>\toolbox\vr\vr demos`. Select the file `vr tkoff.wr1`, and click **Open**.
- 8 In the **Block Parameters: VR Sink** dialog box, click **Apply**.

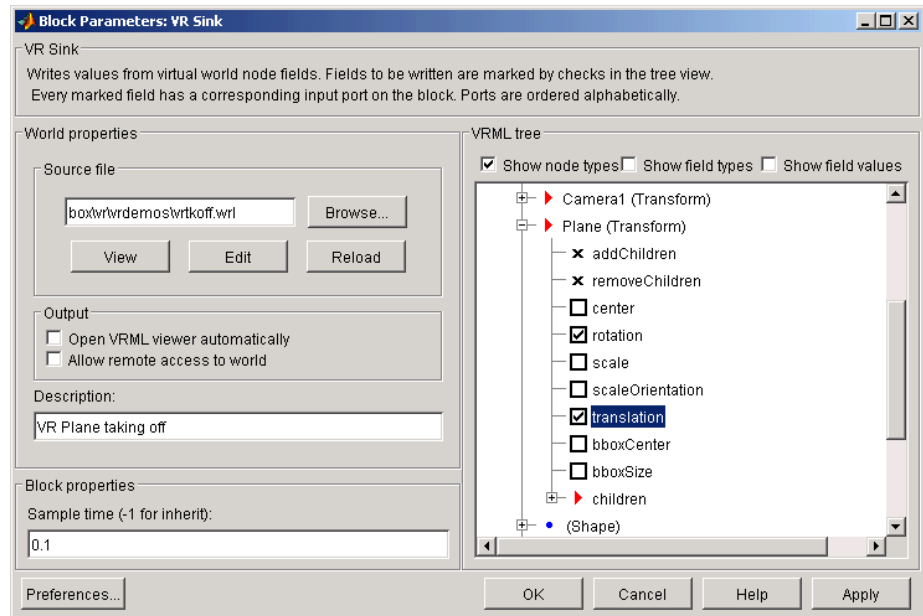
A VRML tree appears on the right side, showing the structure of the associated virtual reality scene.

- 9** Select **Show node types**. On the left of the **Plane Transform** node, click the + square.

The Plane Transform tree expands. Now you can see what characteristics of the plane can be driven from Simulink. This model computes the position and the pitch of the plane.

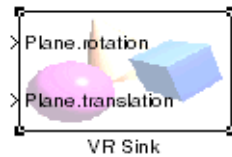
- 10** In the Plane Transform tree, select the **translation** and **rotation** fields.

The selected fields are marked with checks. These fields represent the position (translation) and the pitch (rotation) of the plane.



- 11** Click **OK**.

In the Simulink diagram, the VR Sink block is updated with two inputs.



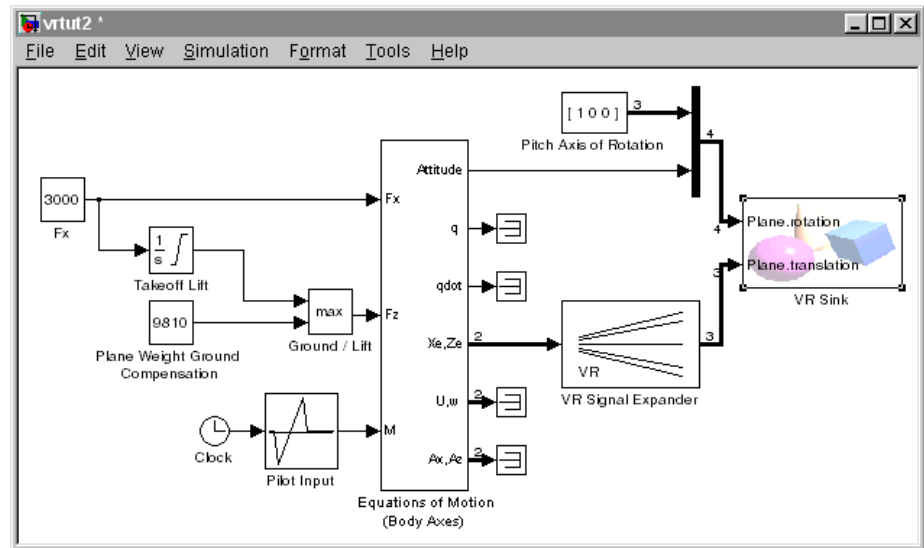
The first input is Plane rotation. The rotation is defined by a four-element vector. The first three numbers define the axis of rotation. In this example, it should be $[1 \ 0 \ 0]$ for the x -axis. The pitch of the plane is expressed by the rotation about the x -axis. The last number is the rotation angle around the x -axis, in radians.

- 12** In the Simulink model, connect the line going to the Scope block labeled Display Pitch to the Plane rotation input.

The second input is Plane translation. This input describes the plane's position in the virtual world. This position consists of three coordinates, x , y , z . The connected vector must have three values. In this example, the runway is in the x - z plane. The y -axis defines the altitude of the plane.

- 13** In the Simulink model, connect the line going to the Scope block labeled Display Position to the Plane translation input.

After connecting the signals and removing the Scope blocks, your model should look similar to the figure shown.



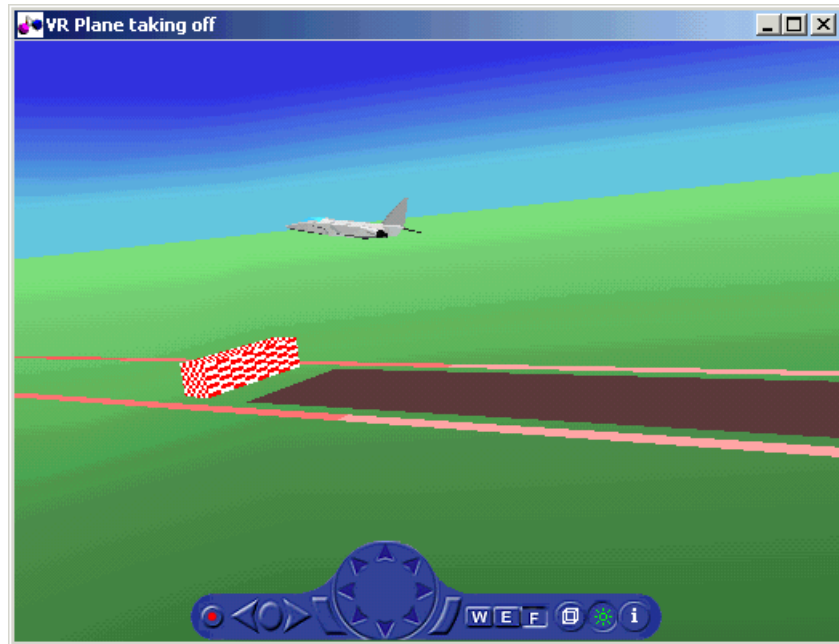
Note Virtual world degrees of freedom have different requested input vector sizes depending on the associated VRML field types. If the vector size of the connected signal does not match the associated VRML field size, an Incorrect input vector size error is reported when you start the simulation.

- 14** Double-click the VR Sink block in the Simulink model. Select the **View** button. A viewer window containing the plane's virtual world opens.



15 From the **Simulation** menu of the model file, click **Start**.

A plane, moving right to left, starts down the runway and takes off into the air.



Changing the Virtual World Associated with a Simulink Block

Sometimes you might want to associate a different virtual world with a Simulink model or connect different signals.

After you associate a virtual world with a Simulink model, you can select another virtual world or change signals connected to the virtual world. This procedure assumes that you have connected the Simulink model with a virtual world. See “Adding a Virtual Reality Toolbox Block” on page 3-2:

- 1 Double-click the VR Sink block in the model.

The **Block Parameters: VR Sink** dialog box opens.

- 2 Click the **Browse** button. The **Select World** dialog box opens. Find the directory `<matlab root>\toolbox\vr\vr demos`. Select the file `vr tkoff2.wrl`, and click **Open**.

- 3 In the **Block Parameters: VR Sink** dialog box, click **Apply**.

A VRML tree appears on the right side. Simulink associates a new virtual world with the model.

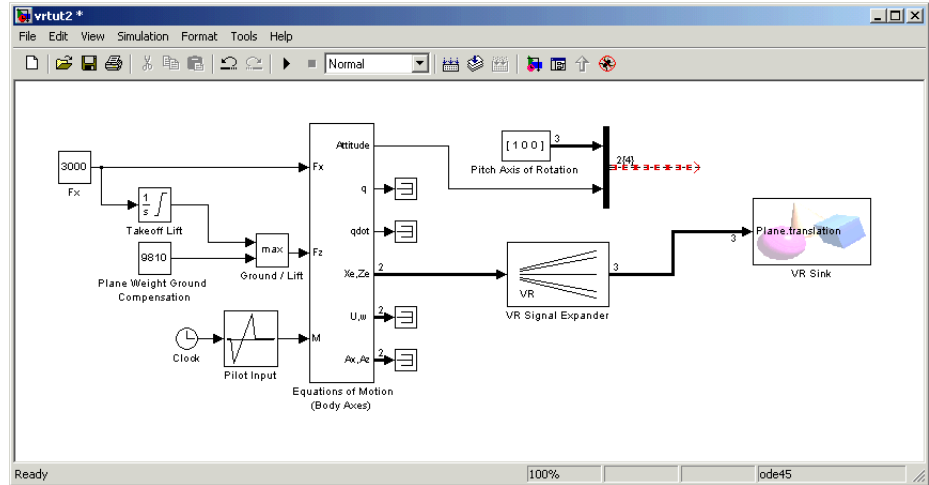
- 4 On the left of the **Plane Transform** node, click the plus sign square.

The **Plane Transform** tree expands. Now you can see what characteristics of the plane you can drive from Simulink. This model computes the position.

- 5 In the **Plane Transform** tree, select the **translation** field check box. Clear the **rotation** field check box. Click **OK**.

The VR Sink block updates and changes to just one input, the Plane translation. The Virtual Reality block is ready to use with the new parameters defined.

- 6 Verify that the correct output is connected to your VR Sink block. The output from the VR Signal Expander should be connected to the single input.



- 7 Run the simulation again and view the new model in the viewer.

Using the Simulink Interface

You can view a virtual world connected to a Simulink block diagram and make parameter changes from Simulink or the virtual world.

This section includes the following topics:

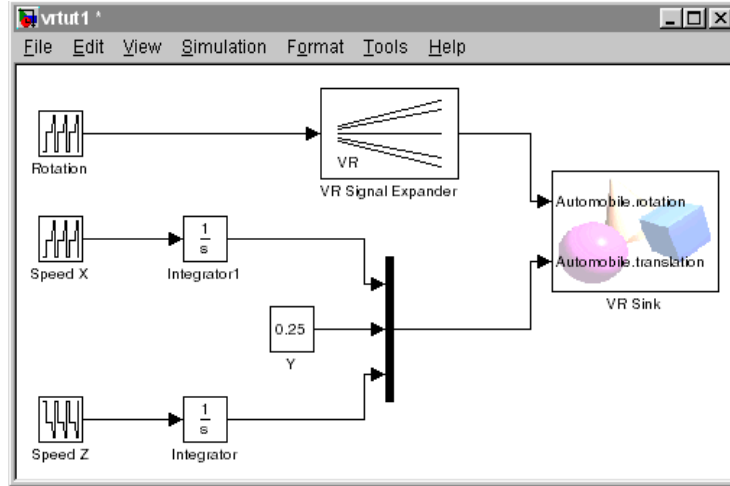
- **Displaying a Virtual World and Starting Simulation** — Display and interact with a virtual world on your host computer using the Virtual Reality Toolbox viewer.
- **View a Virtual World with a Web Browser on the Host Computer** — Connect to the Virtual Reality Toolbox host to access and view virtual worlds.
- **View a Virtual World with a Web Browser on the Client Computer** — Display and interact with a virtual world on a client computer.

Displaying a Virtual World and Starting Simulation

This example explains how to display a simulated virtual world using the Virtual Reality Toolbox viewer on your host computer. This is the default and recommended method for viewing virtual worlds. A Simulink window opens with the model of a simple automobile. Automobile trajectory (vehicle position and angle) is viewed in virtual reality:

- 1 In the MATLAB Command Window, type
vrtut1

A Simulink window opens with the model of an automobile.



A VRML viewer also opens with a three-dimensional model of the virtual world associated with the model.



- 2 Arrange the viewer and Simulink windows on your screen so that they are both visible at the same time.
- 3 In the Simulink window, from the **Simulation** menu, click **Start**.
The simulation starts and, in the VRML viewer window, a car moves along the mountain road.
- 4 Use the VRML viewer controls to move the camera within this virtual world while the simulation is running. For more information on the Virtual Reality Toolbox viewer controls, see "Virtual Reality Toolbox Viewer" on page 5-21.
- 5 In the Simulink window, from the **Simulation** menu, click **Stop**.

Opening a Viewer Window

If you close the viewer window, you might want to reopen it:

- 1 In the Simulink model window, double-click the VR Sink or VR Source block.

A **Block Parameters** dialog box opens.

- 2 Click **View**.

Your default viewer opens and displays the virtual scene. For more information on setting your default viewer, see “Setting the Default Viewer of Virtual Scenes” on page 2-20.

Multiple instances of the viewer can exist on your screen. A viewer appears each time you click **View** in the **Block Parameters** dialog box. This feature is particularly useful if you want to view one scene from many different viewpoints at the same time.

View a Virtual World with a Web Browser on the Host Computer

Normally, you view a virtual world by clicking the **View** button in a block parameters dialog box. The virtual world opens in the Virtual Reality Toolbox viewer or your VRML-enabled Web browser, depending on your `DefaultViewer` setting. For more information on setting your default viewer, see “Setting the Default Viewer of Virtual Scenes” on page 2-20.

Alternatively, you can view a virtual world in your Web browser by selecting an open virtual world from a list in your Web browser. You can display the HTML page that contains this list by connecting to the Virtual Reality Toolbox host. This is the computer on which the toolbox is currently running.

The following procedure describes how to connect to the Virtual Reality Toolbox host:

- 1 At the MATLAB command prompt, type

```
vrcrane
```

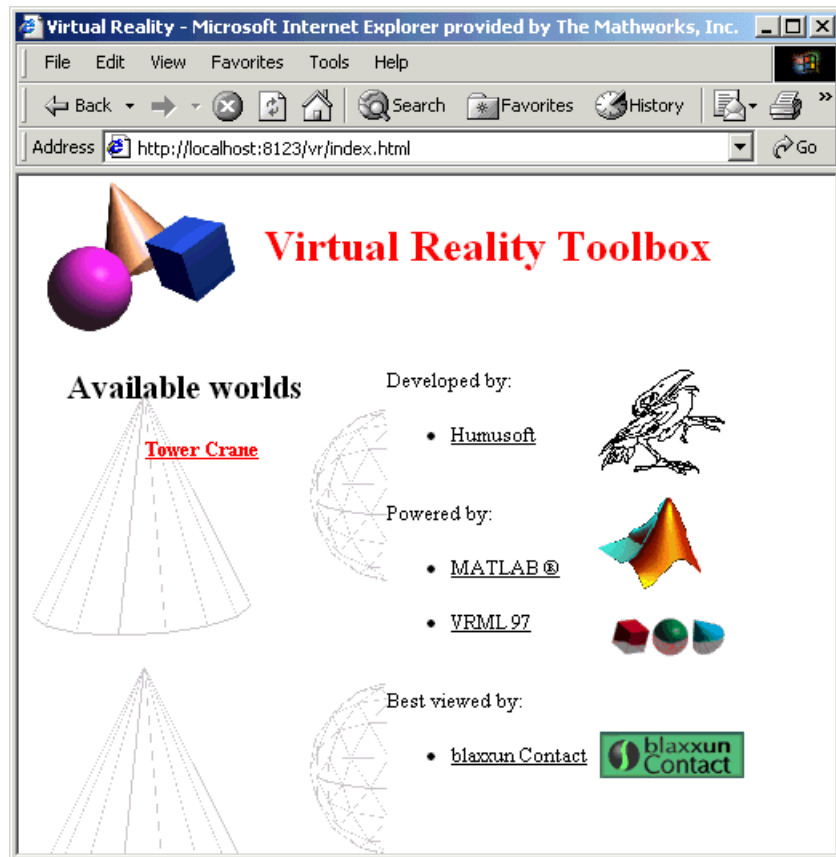
The Tower Crane demo is loaded and becomes active.

- 2 Open your VRML-enabled Web browser. In the address line of the browser, type

`http://localhost:8123`

Note To connect to the main HTML page from a client computer, type `http://hostname:8123`, where `hostname` is the name of the computer on which the toolbox is currently running.

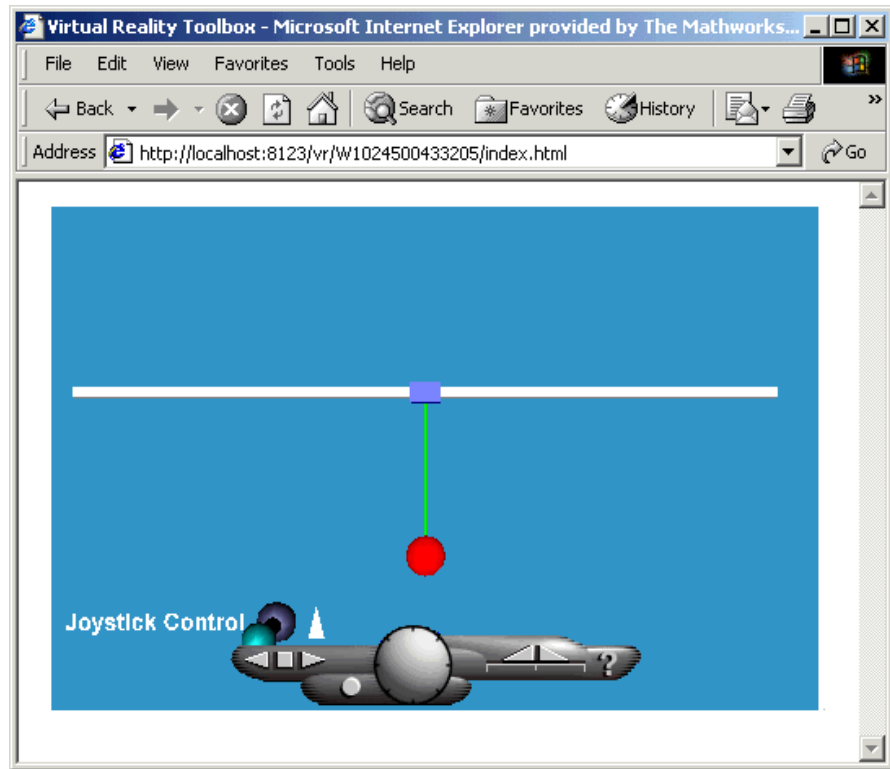
The following page is loaded and becomes active.



The main HTML page for the Virtual Reality Toolbox lists the currently available virtual worlds. The Tower Crane virtual world appears as a link.

3 Click **Tower Crane**.

The Tower Crane virtual world appears in your Web browser.



From the main HTML page, you can select one of the listed available worlds or click the **reload** link to update the status of the virtual worlds supported by the toolbox. This page does not require the VRML capabilities from the browser; it is a standard HTML page. Nevertheless, when you click one of the virtual world links in the list, the browser has to be VRML-enabled to display the virtual world correctly and to communicate with the Virtual Reality Toolbox.

View a Virtual World with a Web Browser on the Client Computer

The Virtual Reality Toolbox allows you to simulate a process on a host computer while running the visualization of the process on a client computer. You view the virtual world on the client computer using a Web browser. This client computer is connected to the host computer through a network using the TCP/IP protocol. This means you need to know the name or IP address of the host computer you want to access from the client computer.

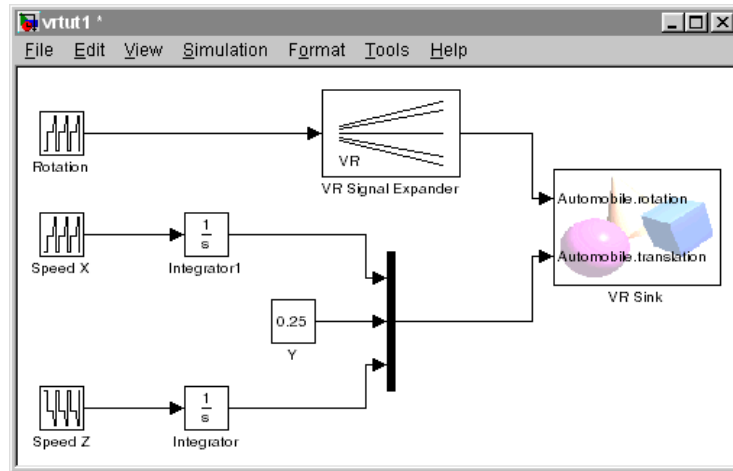
Viewing a virtual world on a client computer might be useful for remote computing, presentation of the results over the Web, or in situations where it is desirable to distribute computing and graphical power.

This example explains how to display a simulated virtual world on a client computer. In this case, the client computer is a PC platform with the blaxxun Contact plug-in. The same procedure can be used to view a virtual world remotely on an SGI platform with Cosmo Player. However, blaxxun Contact is the only supported VRML plug-in. In this example, a Simulink window opens with the model of a simple automobile. The automobile trajectory (vehicle position and angle) is viewed in virtual reality:

- 1 On the host computer, in the MATLAB Command Window, type

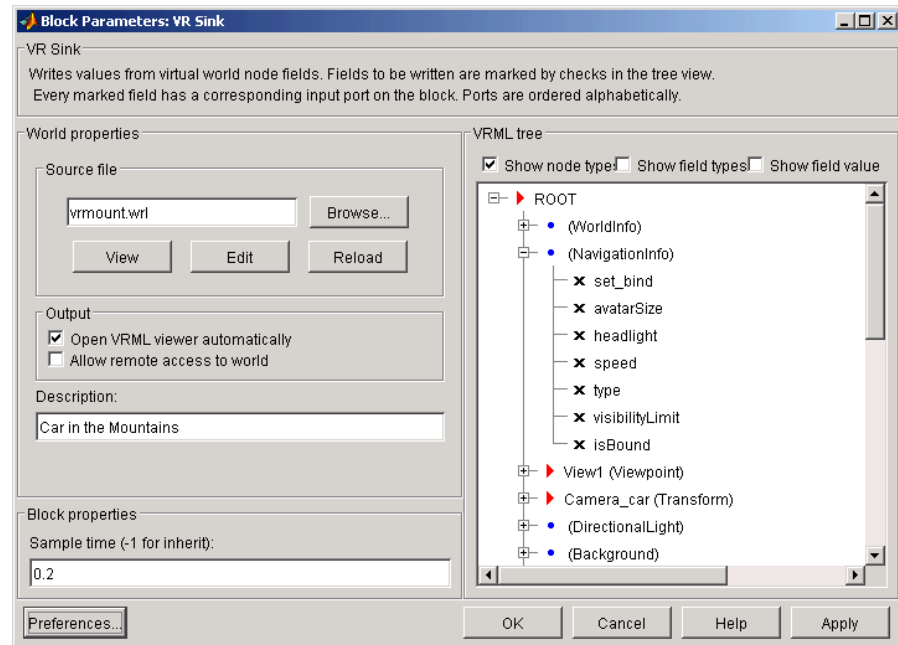
```
vrtut1
```

A Simulink window opens with the model of an automobile.



- 2 Double-click the VR Sink block. This block is in the right part of the model window.

A **Block Parameters: VR Sink** dialog box opens.



3 Select the **Allow remote access to world** check box.

Note This option allows any computer connected to the network to view your model. You should never select this box when you want your model to be private or confidential.

4 Click **OK**.

5 On the client computer, open your VRML-enabled Web browser. In the **Address** line, enter the address and Virtual Reality Toolbox port number for the host computer running Simulink. For example, if the IP address of the host computer is 192.168.0.1, enter

```
http://192.168.0.1:8123
```

To determine your IP address on a Windows system,

- Click **Start**, click **Run**, type cmd, and enter ipconfig (Windows 2000).

- Click **Start**, click **Run**, in the Open box enter `wntipcfg` (Windows NT).
- Click **Start**, click **Run**, and in the Open box enter `winipcfg` (Windows 98).

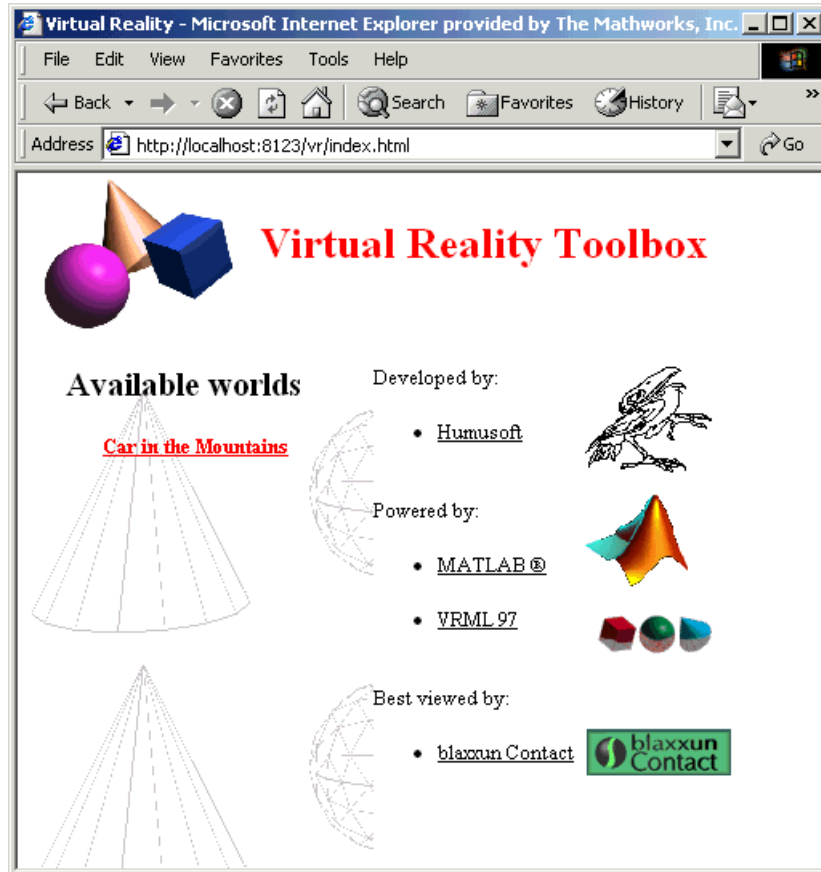
To determine your IP address on a UNIX system, type the command

```
ifconfig device_name
```

Click **OK**. An **IP Configuration** dialog box opens with a list of your IP, mask, and gateway addresses.

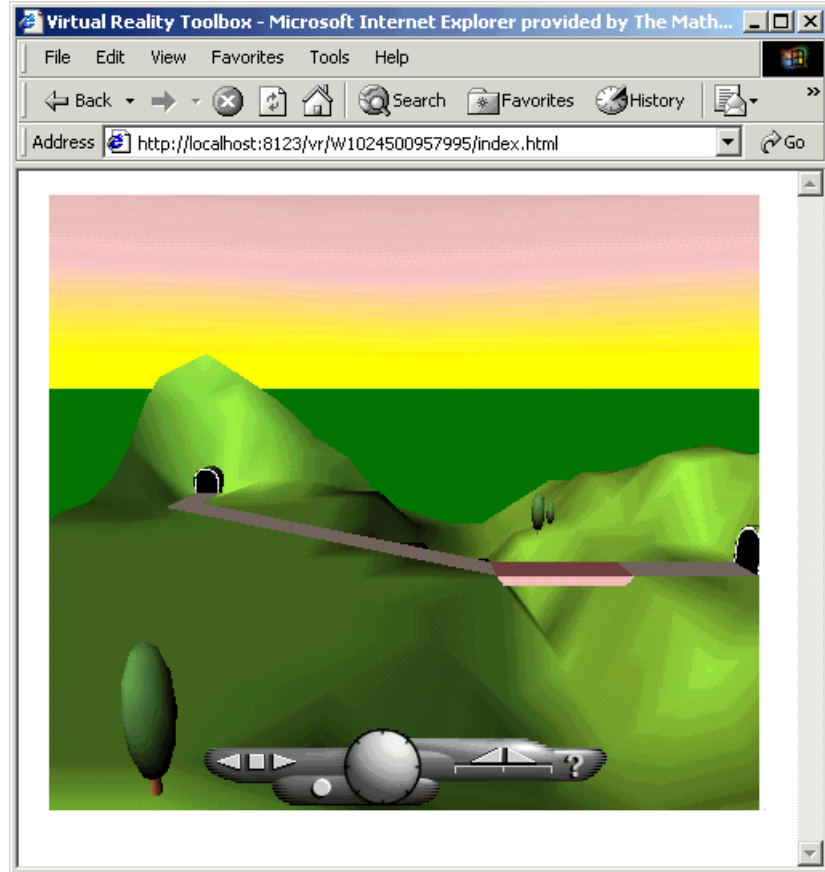
Alternatively, for Windows platforms, you can open a DOS shell and type `ipconfig`.

The Web browser displays the main Virtual Reality Toolbox HTML page. There is only one virtual world in the list because you only have one Simulink model open.



6 Click Car in the Mountains.

The Web browser displays a 3-D model of the virtual world associated with the model.



- 7** On the host computer, in the Simulink window, from the **Simulation** menu, click **Start**.

On the client computer, the animation of the scene reflects the process simulated in the Simulink diagram on the host computer.

You can tune communication between the host and the client computer by setting the **Sample time** and **Transport buffer size** parameters.

- 8** Use the Web browser controls to move within this virtual world while the simulation is running.
- 9** On the host computer, in the Simulink window, from the **Simulation** menu, click **Stop**. On the client computer, close the Web browser window.

MATLAB Interface

Although using The Virtual Reality Toolbox with the Simulink interface is the preferred way of working with the toolbox, you can also use the MATLAB interface. Enter commands directly in the MATLAB Command Window or use M-files to control virtual worlds.

Creating Virtual Reality Toolbox
Objects (p. 4-2)

Create `vrworld` and `vrnode` objects so you can interact with a virtual world through the MATLAB command-line interface

Using the MATLAB Interface (p. 4-4)

Control virtual worlds by entering commands directly in the MATLAB Command Window or by using M-files

Creating Virtual Reality Toolbox Objects

You need to create vrworld objects before you can interact with a virtual world through the MATLAB command-line interface.

This section includes the following topic:

- **Creating a vrworld Object** — Create a vrworld object to connect MATLAB with a virtual world

Note The Simulink interface and the MATLAB interface share the same virtual world objects. This makes it possible for you to use the MATLAB interface to change the properties of vrworld objects originally created by Simulink with Virtual Reality Toolbox blocks.

Creating a vrworld Object

To connect MATLAB to a virtual world, you need to create a vrworld object. A virtual world is defined by a VRML file with the extension `.wrl`. For a complete list of virtual world objects, see Chapter 8, “vrworld Object Reference,” Chapter 9, “vrnode Object Reference,” and Chapter 10, “vrfigure Object Reference.”

After you create a virtual world, you can create a vrworld object. This procedure uses the virtual world `vrmount.wrl` as an example:

- 1 Open MATLAB. In the MATLAB Command Window, type

```
myworld = vrworld('vrmount.wrl')
```

MATLAB displays

```
myworld =  
vrworld object: 1-by-1
```

- 2 Type

```
vrwhos
```

MATLAB displays the messages

```
(untitled)
  Closed, associated with
  'C:<matlab root>\toolbox\vr\vr demos\vrmount.wrl'.
  Visible for local viewers.
  No clients are logged on.
  World id is 'W995480635991'.
```

The `vrworld` object `myworld` is associated with the virtual world `vrmount.wrl`. You can think of the variable `myworld` as a handle to the `vrworld` object stored in the MATLAB workspace.

Your next step is to open a virtual world using the `vrworld` object. See “Opening a Virtual World” on page 4-4.

Using the MATLAB Interface

This section includes the following topics:

- **Opening a Virtual World** — Open a virtual world and scan its structure
- **Interacting with a Virtual World** — Set new values for the available virtual world nodes and their fields
- **Closing and Deleting a vrworld Object** — Close open virtual worlds and remove them from memory

Opening a Virtual World

Opening a virtual world lets you view the virtual world in a VRML viewer, scan its structure, and change virtual world properties from the MATLAB Command Window.

After you create a `vrworld` object, you can open the virtual world by using the `vrworld` object associated with that virtual world. This procedure uses the `vrworld` object `myworld` associated with the virtual world `vrmount.wrl` as an example:

- 1 In the MATLAB Command Window, type

```
open(myworld);
```

MATLAB opens the virtual world `vrmount.wrl`.

- 2 Type

```
set(myworld, 'Description', 'My first virtual world');
```

The description property is changed to `My first virtual world`. This is the description that is displayed in all Virtual Reality object listings, in the title bar of the Virtual Reality Toolbox viewer, and in the list of virtual worlds on the Virtual Reality Toolbox HTML page.

- 3 Open a Web browser. In the **Address** box, type

```
http://localhost:8123
```

The browser displays the Virtual Reality Toolbox HTML page with a link to **My first virtual world**. The number 8123 is the default Virtual Reality Toolbox port number. If you set a different port number on your system,

enter that number in place of 8123. For more information on the Virtual Reality Toolbox HTML page, see “View a Virtual World with a Web Browser on the Host Computer” on page 3-16.

4 In the browser window, click **My first virtual world**.

Your default VRML-enabled Web browser displays the virtual world `vrmount.wrl`.

Alternatively, you can display the virtual world by using the command `view(myworld)`, which displays the virtual scene in your default viewer. For more information on changing your default viewer, see “Setting the Default Viewer of Virtual Scenes” on page 2-20.

Interacting with a Virtual World

In the life cycle of a `vrworld` object you can set new values for all the available virtual world nodes and their fields using `vrnode` object methods. This way, you can change and control the degrees of freedom for the virtual world from within the MATLAB environment.

An object of type `vrworld` contains nodes named in the VRML file using the DEF statement. These nodes are of type `vrnode`. For more information, see Chapter 8, “`vrworld` Object Reference” and Chapter 9, “`vrnode` Object Reference” for the full description of these objects.

After you open a `vrworld` object, you can get a list of available nodes in the virtual world. This procedure uses the `vrworld` object `myworld` and the virtual world `vrmount.wrl` as an example:

1 In the MATLAB Command Window, type

```
nodes(myworld);
```

MATLAB displays a list of the vrnode objects and their fields that are accessible from the Virtual Reality Toolbox.

```
View1 (Viewpoint) [My first virtual world]
Camera_car (Transform) [My first virtual world]
VPfollow (Viewpoint) [My first virtual world]
Automobile (Transform) [My first virtual world]
Wheel (Shape) [My first virtual world]
Tree1 (Group) [My first virtual world]
Wood (Group) [My first virtual world]
Canal (Shape) [My first virtual world]
ElevApp (Appearance) [My first virtual world]
River (Shape) [My first virtual world]
Bridge (Shape) [My first virtual world]
Road (Shape) [My first virtual world]
Tunnel (Transform) [My first virtual world]
```

2 Type

```
mynodes = get(myworld, 'Nodes')
```

MATLAB creates an array of vrnode objects corresponding to the virtual world nodes and displays

```
mynodes =
vrnode object: 13-by-1
```

3 Type

```
whos
```

MATLAB displays the messages

Name	Size	Bytes	Class
mynodes	13x1	3344	vrnode object
myworld	1x1	120	vrworld object

Now you can get node characteristics and set new values for certain node properties. For example, you can change the position of the automobile by using **Automobile**, which is the fourth node in the virtual world.

4 Access the fields of the Automobile node by typing

```
fields(mynodes(4));
```

or

```
fields(myworld.Automobile)
```

Note that dot notation is the preferred method for accessing the nodes of the vrworld object myworld.

MATLAB displays the following table.

Field	Access	Type	Sync
translation	exposedField	SFVec3f	off
center	exposedField	SFVec3f	off
bboxCenter	field	SFVec3f	off
children	exposedField	MFNode	off
scale	exposedField	SFVec3f	off
bboxSize	field	SFVec3f	off
removeChildren	eventIn	MFNode	off
scaleOrientation	exposedField	SFRotation	off
rotation	exposedField	SFRotation	off
addChildren	eventIn	MFNode	off

The **Automobile** node is of type Transform. This VRML node allows you to change its position by changing its **translation** field values. From the list, you can see that **translation** requires three values, representing the [x y z] coordinates of the object.

5 Type

```
view(myworld)
```

Your default viewer opens and displays the virtual world vrmount.wrl.

6 Move the MATLAB window and the browser window side by side so you can view both at the same time. In the MATLAB Command Window, type

```
myworld.Automobile.translation = [15 0.25 20];
```

MATLAB sets a new position for the **Automobile** node, and you can observe that the car is repositioned in the VRML browser window.

It is possible to change the node fields listed by using the function `vrnode/setfield`.

Closing and Deleting a vrworld Object

After you are finished with a session, you must close all open virtual worlds and remove them from memory:

- 1 In the MATLAB Command Window, type

```
close(myworld);  
delete(myworld);
```

The virtual world representation of the `vrworld` object `myworld` is removed from memory. All possible connections to the viewer and browser are closed and the virtual world name is removed from the list of available worlds.

Note Closing and deleting a virtual world does not delete the `vrworld` object handle `myworld` from the MATLAB workspace.

Virtual Worlds

The Virtual Reality Toolbox includes the Virtual Reality Toolbox viewer for all supported platforms. For PC platforms, it includes a VRML plug-in (blaxxun Contact) and a VRML editor (V-Realm Builder). For UNIX/Linux platforms, the default VRML editor is the MATLAB editor. A basic understanding of these tools and how to use them will help you to get started quickly.

VRML Editing Tools (p. 5-2)

Description of the differences between general and native editors

Deformation of a Sphere Example
(p. 5-5)

Tutorial for creating a simple virtual world with V-Realm Builder and associating this virtual world with Simulink blocks from the Virtual Reality Toolbox

Viewing a Virtual World (p. 5-21)

Description of the Virtual Reality Toolbox viewer and the blaxxun Contact VRML plug-in

VRML Data Types (p. 5-37)

VRML data types are data types used by VRML nodes to define objects and types of data that can appear in the VRML node fields and events.

VRML Editing Tools

There is more than one way to create a virtual world described with the VRML code. For example, you can use a text editor to write VRML code directly, or you can use a VRML editor to create a virtual world without having to know anything about the VRML language. However, you need to understand the structure of a VRML tree to connect your virtual world to Simulink blocks and signals.

This section includes the following topics:

- **Editors for Virtual Worlds** — General 3-D and native VRML editors
- **V-Realm Builder** — Native VRML editor shipped with the PC version of the Virtual Reality Toolbox

Editors for Virtual Worlds

A VRML file uses a standard text format that you can read with any text editor. Reading the text is useful for debugging, automated processing, and directly changing VRML code. Also, if you use the correct VRML syntax, you can use any common text editor to create virtual scenes similar to the way you create HTML pages.

Many people prefer to create simple virtual worlds using their favorite text editor. However, the primary way for you to create a virtual world is with a 3-D editing tool. These tools allow you to create complex virtual scenes without a deep understanding of the VRML language.

These 3-D editing tools offer the power and versatility necessary for creating many types of practical and technical models. For example, you can import 3-D objects from some CAD packages to make the authoring process easier and more efficient. For VRML authoring, there are basically two types of 3-D editing tools:

- General 3-D authoring packages that can export into VRML format
- Native VRML authoring tools

General 3-D Editors — General 3-D editors do not use VRML as their native format. They export their formats to VRML. There are many commercial packages, such as 3D Studio, that can do this. These tools have many features and are very easy to use. General 3-D editing tools target specific types of work. For example, they can target visual art, animation, games, or technical

applications. They offer different working environments depending on the application area for which they are designed. Some of these general 3-D editing tools can be very powerful, expensive, and complex to learn, but others are relatively inexpensive and might satisfy your specific needs.

It is interesting to note that the graphical user interfaces for many of the general commercial 3-D editors use features typical of the native VRML editing tools. For example, in addition to displaying 3-D scenes in various graphical ways, they also offer hierarchical tree-style views that provide a good overview of the model structure and a convenient shortcut to 3-D element definitions.

Native VRML Editors — Native VRML editors use VRML as their native format. This guarantees that all the features in the editor are compatible with VRML. Also, native VRML editors can use features that are unique for the VRML format, like interpolators and sensors.

Unfortunately, there are currently few advanced VRML editors of commercial quality. Most native VRML editors are in the development stage and are harder to use than a general 3-D editor. V-Realm Builder by Ligos Corporation is one of the exceptions. It is one of the most advanced VRML editing tools currently available for personal computers. V-Realm Builder is available only for Windows operating systems.

For PC, the Virtual Reality Toolbox includes V-Realm Builder as a native 3-D editor. For more information, see “V-Realm Builder” on page 5-4 and “Deformation of a Sphere Example” on page 5-5.

V-Realm Builder

V-Realm Builder is a flexible, graphically oriented tool for 3-D editing and is available for Windows operating systems only. It is a native VRML authoring tool that provides a convenient interface to the VRML syntax. Its primary file format is VRML. Its graphical interface (GUI) offers not only the graphical representation of a 3-D scene and tools for interactive creation of graphical elements, but also a hierarchical tree-style view (tree viewer) of all the elements present in the virtual world.

These structure elements are called nodes. V-Realm Builder lists the nodes and their properties according to their respective VRML node types, and it supports all 54 VRML97 types. For each type of node there is a specific tool for convenient modification of the node parameters. You can access node properties in two ways:

- Using dialog boxes accessible from the tree viewer
- Directly, using a pointing device

In many cases, it is easier to use the tree viewer to access nodes because it can be difficult to select a specific object in a 3-D scene. The tree view also lets you easily change the nesting levels of certain nodes to modify the virtual world according to your ideas. In the tree viewer, you can give the nodes unique names — a feature necessary for working with Virtual Reality Toolbox.

Deformation of a Sphere Example

The example in this section shows you how to create a simple virtual world using V-Realm Builder. It does not describe everything you can do with V-Realm Builder, but it does describe the basics to get you started.

This example assumes you finished the installation of V-Realm Builder using the function `vrinstall`. See “Installing VRML Editor (Windows)” on page 2-25.

This section includes the following topics:

- “Defining the Problem” on page 5-5
- “Adding a Virtual Reality Toolbox Block” on page 5-6
- “Creating a Sphere in a Virtual World” on page 5-8
- “Creating a Box in a Virtual World” on page 5-13
- “Connecting a Simulink Model to a Virtual World” on page 5-17

Defining the Problem

Suppose you want to simulate and visualize in virtual reality the deformation of a sphere. In your virtual world, you want to have two boxes representing rigid plates (B1, B2) and an elastic sphere (S) between them. All three of the objects are center-aligned along the X axis. The boxes B1 and B2 move towards S with identical velocities, but they move in opposite directions. As they reach the sphere S, they start to deform it by reducing its X dimension and stretching it in both its Y and Z dimensions.

Positions and dimensions of the objects are listed in the following table.

Object	Center Position	Dimensions
B1	[3 0 0]	[0.3 1 1]
B2	[-3 0 0]	[0.3 1 1]
S	[0 0 0]	$r = 0.9$

Your first task is to open a Simulink model and add a Virtual Reality Toolbox block to your model. See “Adding a Virtual Reality Toolbox Block” on page 5-6.

Simulink model — The Virtual Reality Toolbox includes the tutorial model `vrutut3.mdl`. This is a simplified model in which the deformation of an elastic

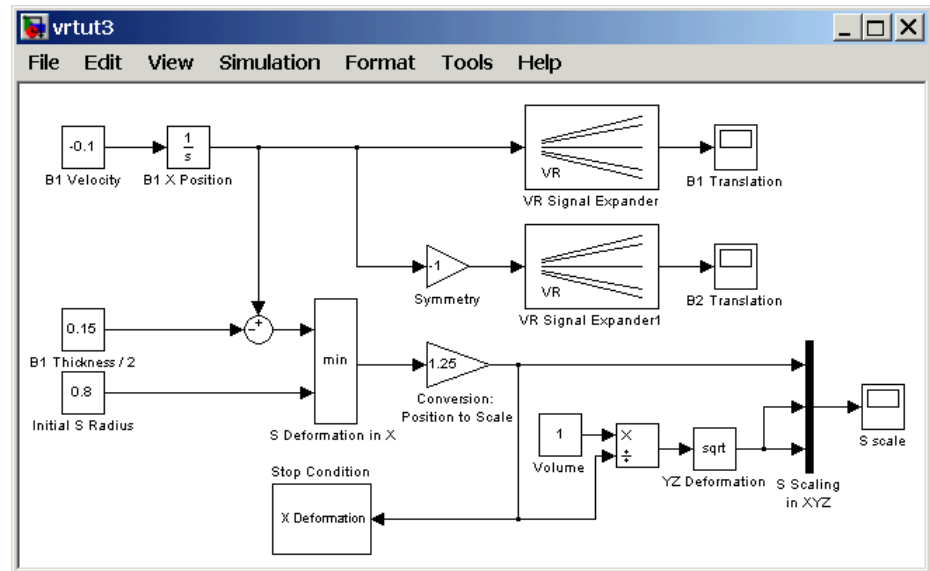
sphere is simulated. After collision with the rigid blocks, the sphere's X dimension is decreased by a factor from 1 to 0.4, and the Y and Z dimensions are expanded so that the volume of the deformed sphere-ellipsoid remains constant. There are also additional blocks in the model to supply the correctly sized vectors to the Virtual Reality Toolbox block. The simulation stops when the sphere is deformed to 0.4 times its original size in the X direction.

Adding a Virtual Reality Toolbox Block

This procedure uses the Simulink model `vrtut3.mdl` as an example to explain how to add a Virtual Reality Toolbox block to your model. The model generates the values for the position of B1, the position of B2, and the dimensions of S for the problem previously defined. See “Defining the Problem” on page 5-5.

- 1** From the directory `C:\<matlab root>\toolbox\vr\vr demos\`, copy the file `vrtut3.mdl` to your MATLAB working directory.
- 2** Start MATLAB, and then change the current directory to your MATLAB working directory.
- 3** In the MATLAB Command Window, type
`vrtut3`

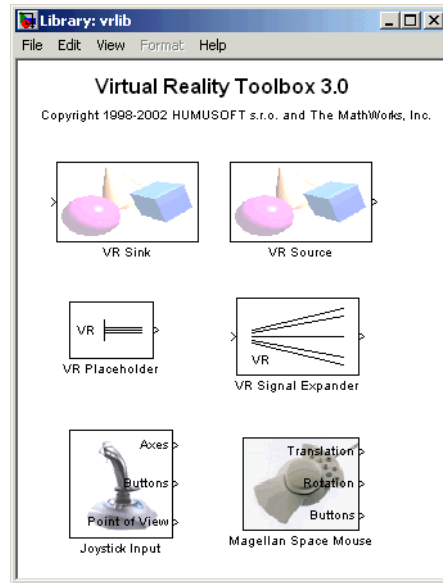
A Simulink window opens with a model that does not contain a Virtual Reality Toolbox block. Instead, this model uses Scope blocks to temporarily monitor the relevant signals.



4 From the MATLAB Command Window, type

```
vrlib
```

The Virtual Reality Toolbox library opens.



- From the Library window, drag-and-drop the VR Sink block to the Simulink diagram. You can then close the **Library: vrlib** window.

Your next task is to create a virtual world that you will associate with the VR Sink block. See “Creating a Sphere in a Virtual World” on page 5-8.

Creating a Sphere in a Virtual World

You need to create a virtual world before you can connect it to a Simulink model and visualize signals.


After you add a VR Sink block to your Simulink model, you can create a virtual world using V-Realm Builder. This procedure uses the model `vrtut3.mdl` as an example and assumes that you have opened the model and that you have added a VR Sink block. See “Adding a Virtual Reality Toolbox Block” on page 5-6.

- From the task bar, click **Start**, and then click **Run**.

- 2 In the **Run** dialog box, enter

```
C:\<matlab root>\toolbox\vr\vrealm\program\vrbuild2.exe
```

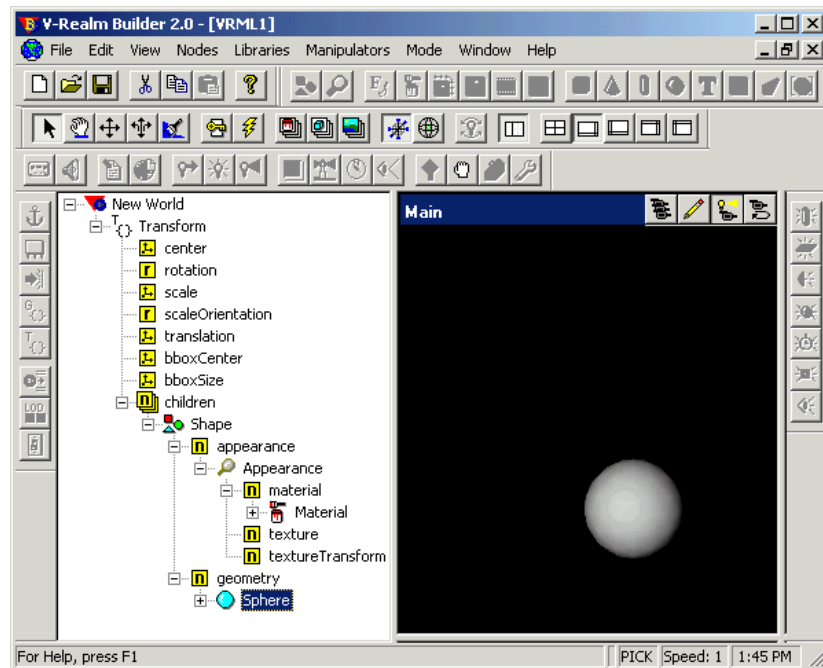
The V-Realm Builder application window opens.

- 3 From the **File** menu, click **New** or click the blank page icon .

In the left pane, V-Realm Builder displays an empty VRML tree, and in the right pane it displays an empty virtual world.

- 4 On the toolbar, click the sphere icon .

In the left pane you can see the VRML syntax tree for a sphere. This tree includes the following nodes: **Transform**, **Shape**, **Appearance**, **Material**, and **Sphere**. A yellow icon indicates the field of a node.

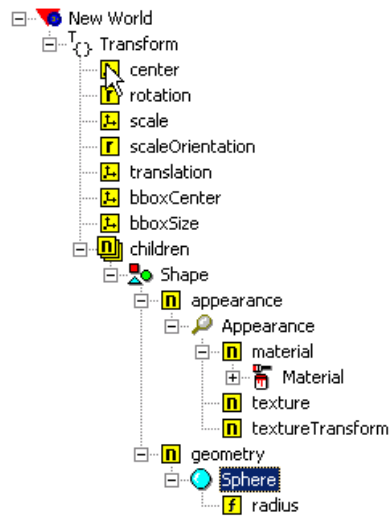


The top-level node is the **Transform** node. This grouping node allows you to change the position and scale of objects (children) that are part of this node.

Its subtree consists of one object, which is described in the **Shape** node. The **Shape** node contains the **appearance** and **geometry** fields.

5 Expand the **Sphere** node.

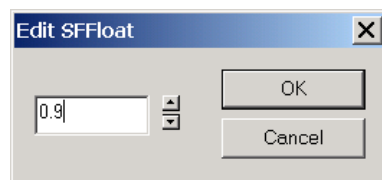
The **radius** field appears. The yellow icon indicates the type of value. In this case, **f** indicates a value with the type SFFloat. SFFloat is a 32-bit floating-point value.



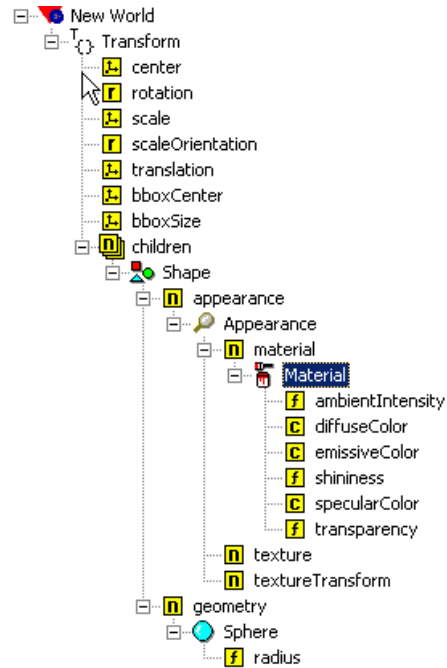
6 Double-click the **radius** field.

The **Edit SFFloat** dialog box opens.

7 In the text box, enter 0.9, and then click **OK**. In the right pane, the sphere appears smaller.



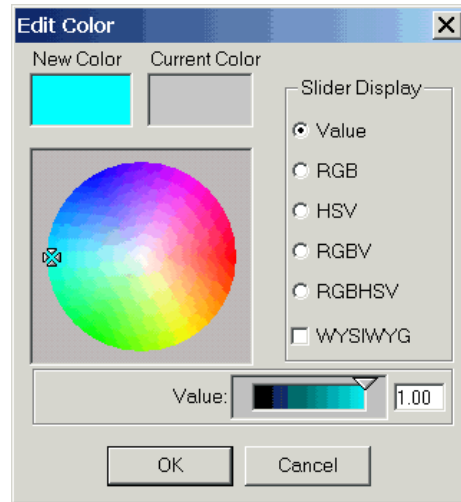
- 8 Under the **Shape** node, expand the **appearance** field. Under the appearance field, expand the **Appearance** node. Under the **Appearance** node, expand the **material** field. Under the **material** field, expand the **Material** node.



- 9 Under the **Material** node, double-click the **diffuseColor** field.

The **Edit Color** dialog box opens.

- 10** Set the color to blue or any other color you would like, and then click **OK**.



- 11** If you want to check or modify the position of the sphere, double-click the **translation** field under the **Transform** node. You do not need to change the default values from [0 0 0].

In this exercise, you want to deform the sphere. You can apply deformation by changing the **scale** field of the sphere's **Transform** node. The Virtual Reality Toolbox requires you to assign a name to this node so that it can access it. In VRML syntax, the named nodes are indicated by the "DEF Name Node" statement. V-Realm Builder lists node names next to their icons in the tree viewer.


- 12** Click the **Transform** node, and then click the node a second time.

The text appears in edit mode.

- 13** Enter a name for the node. For example, enter the letter S, and then click anywhere to exit the text mode.

Your next task is to create two boxes in the virtual world. See "Creating a Box in a Virtual World" on page 5-13.

Creating a Box in a Virtual World

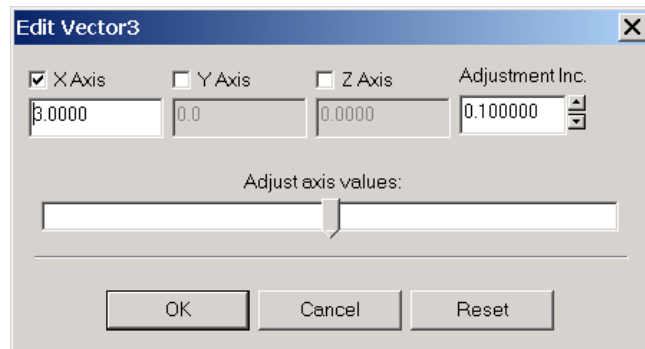
- 1 In the tree, click **New World** (the topmost item). On the toolbar, click the box icon .

A new box object appears at the same position as the sphere centered in the origin of the coordinate system. Note that the sphere is hidden behind the box and currently is not visible.

- 2 Double-click the **translation** field under the **Transform** node.

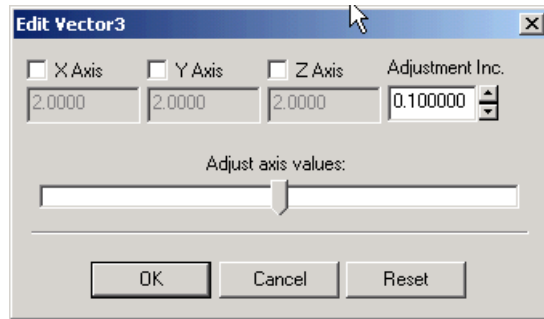
The **Edit Vector 3** dialog box opens. Notice that there are two **Transform** nodes. Use the one with the **Box** node in its subtree.

- 3 Select the **X Axis** check box. In the text box below, enter **3**, then click **OK**.



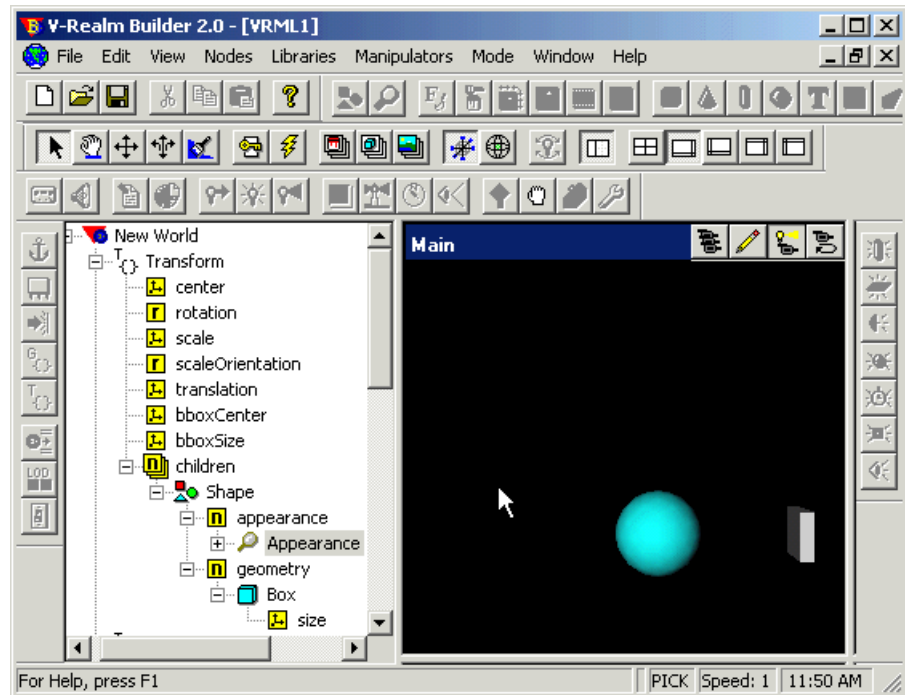
The position of the box is set to [3 0 0].

- 4 Expand the **Box** node. Double-click the **size** field under the **Box** node.
The **Edit Vector 3** dialog box opens.



- 5 Set the values to [0.3 1 1], and then click **OK**.

The figure below shows the tree view with the expanded branch of the box.

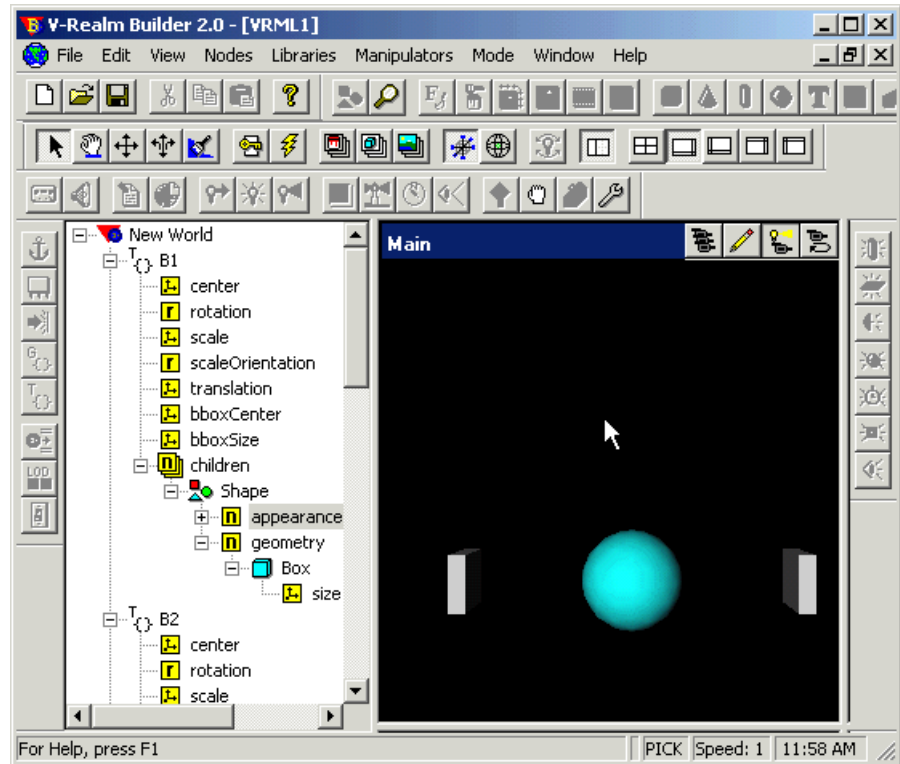


- 6 Create a second box the same way you created the first box.
- 7 To move the second box to its correct place, double-click the **translation** field of the second **Transform** node, and change its position to $[-3 \ 0 \ 0]$.
- 8 Double-click the **size** field under the **Box** node. Set the values to $[0.3 \ 1 \ 1]$, and then click **OK**.

The scene is now complete.

- 9 To access the positions of the boxes from a Virtual Reality Toolbox block, give each **Transform** node a name. For example, set the name of the first **Transform** node to B1 and the second **Transform** node to B2. The Virtual

Reality Toolbox allows you to access fields of only those nodes that are named in virtual worlds.



10 Save the virtual world as `vrtut3.wr1` in the same working directory where the file `vrtut3.mdl` resides, and then exit V-Realm Builder.

Caution If you want to use your virtual worlds with the Virtual Reality Toolbox, do not save them in a compressed Gzip format.

Your next task is to connect the model outputs to the Virtual Reality Toolbox block in your Simulink model. See “Connecting a Simulink Model to a Virtual World” on page 5-17.

Connecting a Simulink Model to a Virtual World

After you create a virtual world, a Simulink model, and add a Virtual Reality Toolbox block to your model, you can define the associations between the model signals and the virtual world. This procedure uses the model `vrtut3.mdl` as an example. It assumes that you have opened the model and that you have added a VR Sink block. See “Adding a Virtual Reality Toolbox Block” on page 5-6.

- 1 In the Simulink window, double-click the **VR Sink** block.

The **Block Parameters: VR Sink** dialog box opens again.

- 2 Click **Browse**.

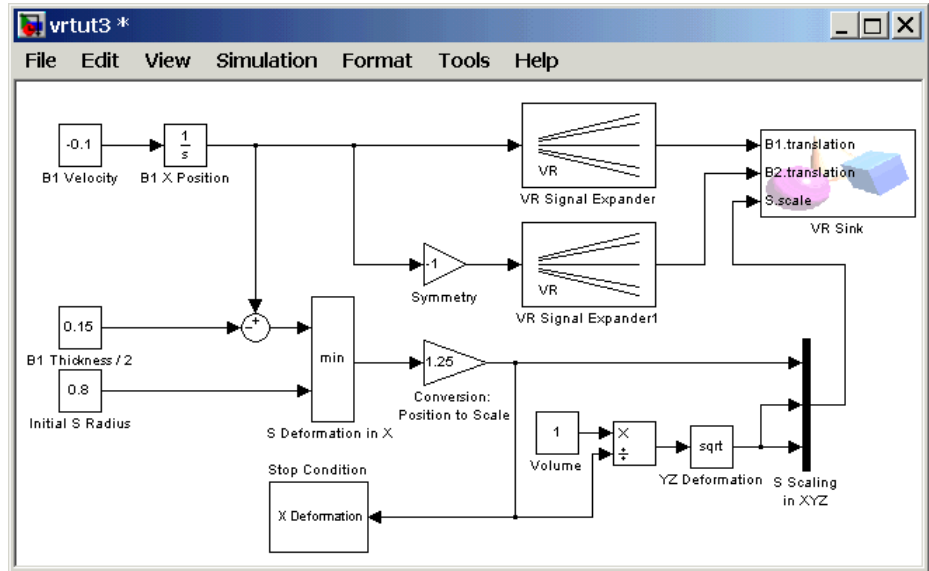
The **Select World** dialog box opens.

- 3 Select `vrtut3.wr1`, and then click **Open**.

- 4 In the tree viewer, select the **S** scale, **B1** translation, and **B2** translation check boxes as the nodes you want to connect to your model signals. Click **OK** to close the dialog box.

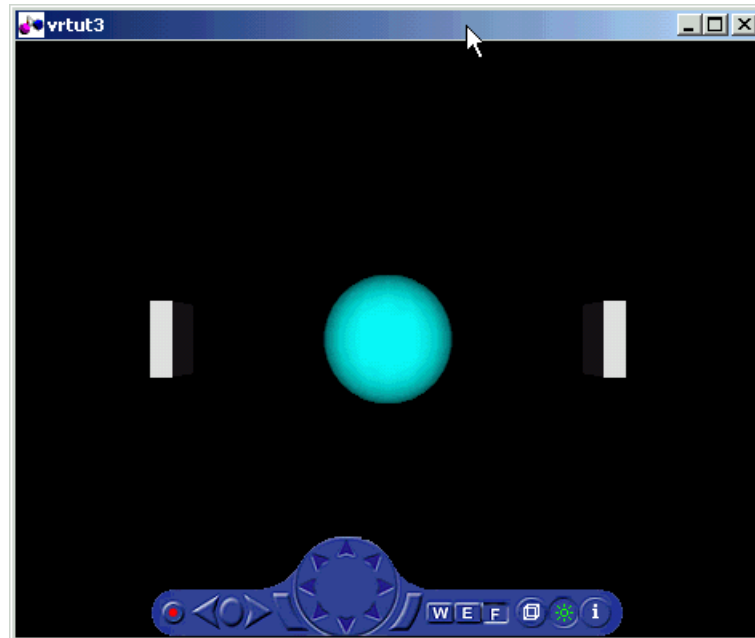
The Virtual Reality Toolbox block appears with corresponding inputs.

- 5 Connect these input lines to the matching signals in the model. These signals were originally connected to Scope blocks.



- 6 Double-click the **VR Sink** block. In the **Block Parameters: VR Sink** dialog box, click the **View** button.

Your default viewer opens and displays the virtual world. For more information on changing your default viewer, see “Setting the Default Viewer of Virtual Scenes” on page 2-20.



- 7** In the Simulink window, from the **Simulation** menu, click **Start**.

In your default viewer, you see a 3-D animation of the scene. Using the viewer controls you can observe the action from various viewpoints.

When the width of the sphere is reduced to 0.4 of its original size, the simulation stops running.



This example shows you how to create and use a very simple virtual reality model. Using the same method, you can create more complex models for solving your particular problems.

Viewing a Virtual World

After you create a virtual world with VRML code, you can visualize that virtual world with the Virtual Reality Toolbox viewer or with a VRML-enabled Web browser. The Virtual Reality Toolbox viewer can be used on all supported platforms and is the preferred method of viewing virtual worlds.

This section includes the following topics:

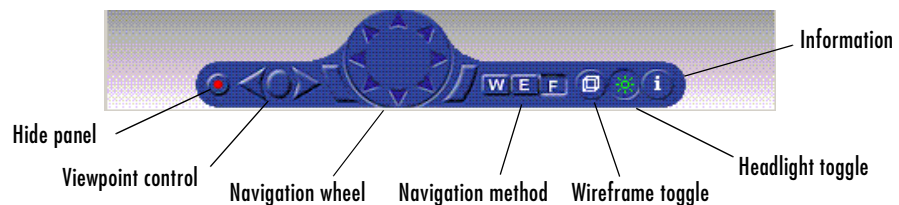
- “Virtual Reality Toolbox Viewer” on page 5-21
- “blaxxun Contact VRML Plug-in” on page 5-33

Virtual Reality Toolbox Viewer

The Virtual Reality Toolbox contains a viewer as the default method for viewing virtual worlds. You can use this viewer on any supported operating system. For a list of supported operating systems, see “System Requirements” on page 2-2. The following topic provides an overview of the features and controls of the viewer.

- 1 Open a virtual world by selecting a Virtual Reality demo or by clicking the **View** button in the **Block Parameters** dialog box of a Virtual Reality block.

The Virtual Reality Toolbox viewer is loaded and becomes active. It displays the virtual scene with a control panel at the bottom.



Note The Virtual Reality Toolbox viewer settings are saved when you save your model file.

Viewpoint Control

There are three buttons on the control panel that affect the viewpoint of the scene. The center circular button resets the camera to the current viewpoint. This button is useful when you have been moving about the scene and need to reorient yourself. The keyboard equivalents of this button are **Home** and **R**. The **Esc** key resets the camera to the default viewpoint.

You can use the right and left arrows associated with viewpoint control to browse through predefined viewpoints. These buttons are inactive if other viewpoints are not specified by the author. You can also use the **Page Up** and **Page Down** keys to navigate through these viewpoints.

Control Menu

Access the control menu by right-clicking in the viewer window. You can use the control menu to specify a predefined viewpoint or change the appearance of the control panel. You can also control the navigation method, speed, and rendering of the virtual world. For more information about navigation methods, see “Navigation” on page 5-25. For more information about rendering, see “Rendering” on page 5-23.

Changing the Navigation Speed

- 1** In the viewer window, right-click.
- 2** Point to **Navigation**.
- 3** Point to **Speed**, then select **Very Slow**.

Your navigation speed within the virtual world is much slower than before.

Note Your navigation speed controls the distance you move with each keystroke. It does not affect rendering speed.

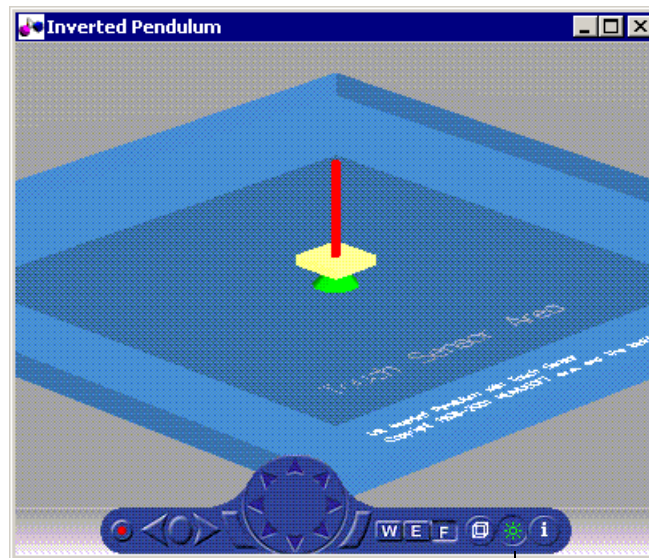
Consider setting a higher speed for large scenes and a slower speed for more controlled navigation in smaller scenes.

Rendering

You can change the rendering of the scene through the control panel or the control menu. The vrpemd demo is used to demonstrate the viewer's functionality.

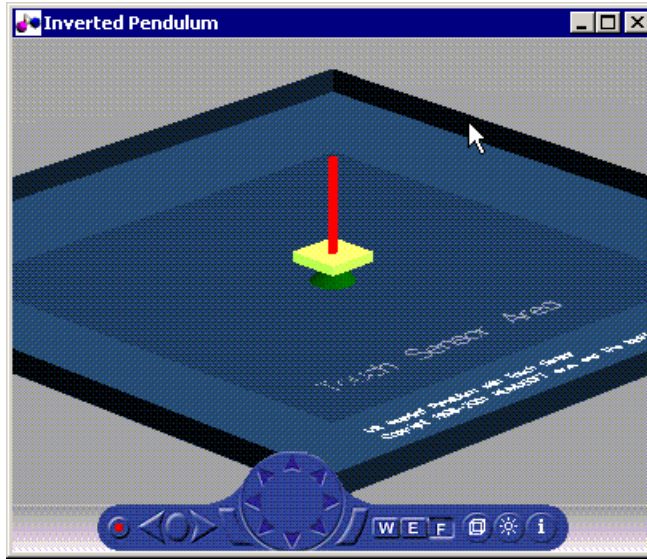
You can turn the antialiasing of the scene on or off. Antialiasing is a technique that attempts to smooth the appearance of jagged lines. These jagged lines are the result of a printer or monitor's not having enough resolution to represent a line smoothly. When **Antialiasing** is on, the jagged lines are surrounded by shades of gray or color. Therefore, the lines appear fuzzy rather than jagged.

You can turn the camera headlight and the lighting of the scene on or off. When **Headlight** is off, the camera does not emit light. Consequently, the scene can appear dark. For example, the following figure depicts the vrpemd demo with **Headlight(on)**.



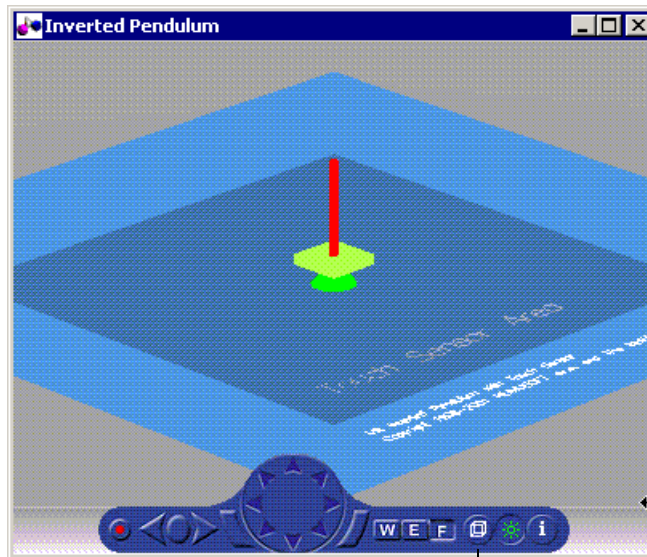
Headlight toggle

The scene looks darker when **Headlight** is set to off.



Note It is helpful to define enough lighting within the virtual scene so that it is lit regardless of the **Headlight** setting.

When **Lighting** is off, the virtual world appears as if lit in all directions. Shadows disappear and the scene loses some of its three-dimensional quality. The following is the vrpend demo with **Lighting(off)**.



Wireframe toggle

If **Textures** is off, objects do not have texture in the virtual scene.

If **Transparency** is off, transparent objects are rendered as solid objects.

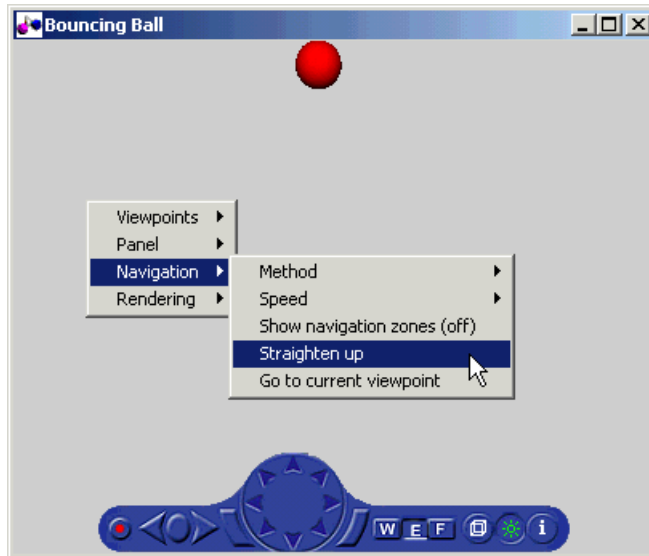
Turning **Wireframe** on changes the scene's objects from solid to wireframe renderings.

Navigation

You can navigate around a virtual scene using the control panel, control menu, mouse, and keyboard. The vrbounce demo is used to demonstrate the viewer's functionality.

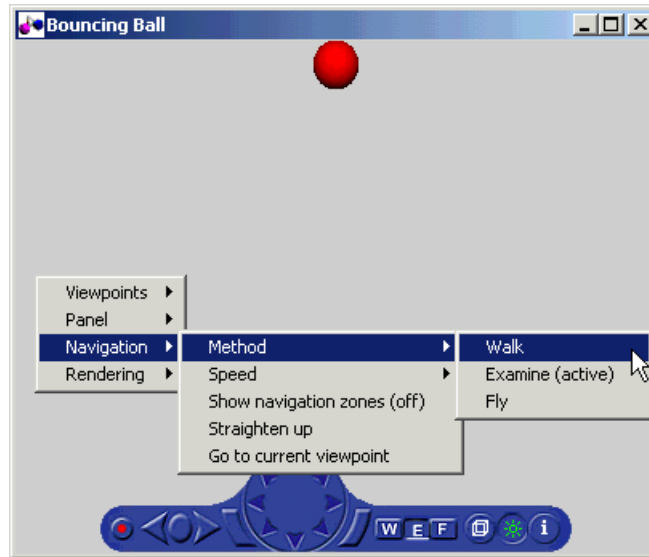
Control Panel — The center navigation wheel and two curved buttons on either side move you about the scene. Experiment by moving backward and forward and side to side until you become comfortable with the controls.

Control Menu — Right-click in the viewer window to access the control menu. Point to **Navigation** and a new menu is displayed.



From this menu, you can reset the camera so that it is pointed straight ahead by choosing **Straighten up**. You can also return to the current viewpoint by choosing **Go to current viewpoint**. This option is useful if you are navigating about the scene and want to reorient yourself.

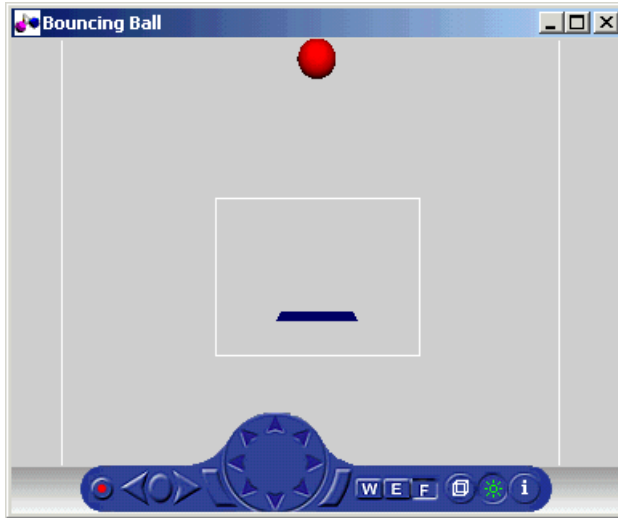
Mouse — Navigation with the mouse depends on the navigation method you select and the navigation zone you are in when you first click and hold down the mouse button. You can set the navigation method using the **W**, **E**, or **F** buttons on the control panel or by right-clicking in the viewer. Point to **Navigation**. Next, point to **Method** and click **Walk**, **Examine**, or **Fly**.



Navigation Method

To view the navigation zones for a scene, right-click in the viewer and point to **Navigation**. Then click **Show navigation zones (off)**. The navigation zones are toggled on and appear in the virtual scene.

For example, using the vrbounce demo with **Method** set to **Fly**, there are three navigation zones.



The following table summarizes the behavior associated with the movement modes and navigation zones when you use your mouse to navigate through a virtual world. Turn the navigation zones on and experiment by clicking and dragging your mouse in the different zones of a virtual world.

Mouse Navigation

Movement Mode	Zone and Description
Walk	<p>Outer — Click and drag the mouse up, down, left, or right to move the camera in any of these directions in a single plane.</p> <p>Inner — Click and drag the mouse up and down to move forward and backward. Drag the mouse left and right to turn left or right.</p>
Examine	<p>Outer — Click and drag the mouse up and down to move forward and backward. Drag the mouse left and right to pan left or right.</p> <p>Inner — Click and drag the mouse to rotate the viewpoint about the origin of the scene.</p>
Fly	<p>Outer — Click and drag the mouse to tilt the view either left or right.</p> <p>Inner — Click and drag the mouse to move the camera up, down, left, or right within the scene.</p> <p>Center — Click and drag the mouse up and down to move forward and backward. Move the mouse left or right to turn in either of these directions.</p>

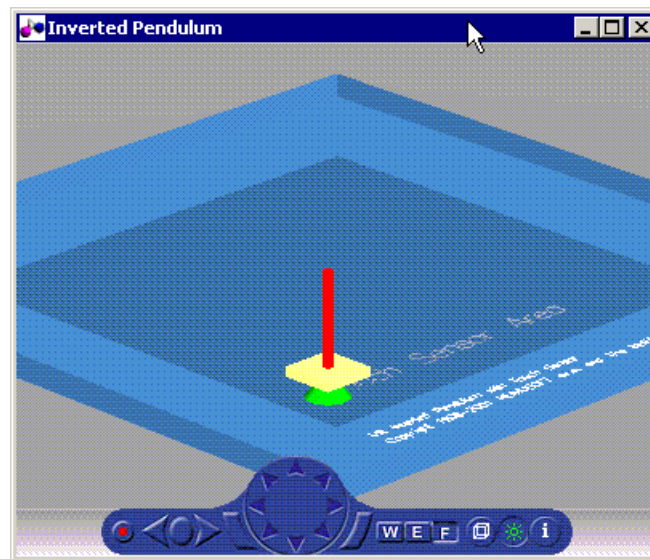
If your virtual world contains sensors, these sensors take precedence over mouse navigation at the sensor's location.

Example of How Sensors Affect Mouse Navigation

- 1 In the MATLAB Command Window, type

```
vrpend
```

at the MATLAB command prompt. The inverted pendulum demo starts, and the viewer displays the following scene.



- 2 In the Simulink model window, from the **Simulation** menu, choose **Start**.

The demo starts running.

- 3 Click in the viewer window inside and outside the sensor area. Notice that the sensor takes precedence over navigation with the left mouse button. The shape of your pointer changes when it is located over the sensor area.

If the sensor covers the entire navigable area, mouse navigation is effectively disabled. In this case, use the control panel or the keyboard to move about the scene. For a three-button mouse or a mouse with a clickable wheel, you can

always use the middle button or the wheel to move about the scene. The middle mouse button and wheel do not trigger sensors within the virtual world.

Keyboard — It is also possible to use the keyboard to navigate through a virtual world. It can be faster and easier to issue a keyboard command, especially if you want to move the camera repeatedly in a single direction. The following table summarizes the keyboard commands and their associated navigation functions. Note that the letters presented do not need to be capitalized in order to perform their intended function.

Keyboard Navigation

Keyboard Command	Navigation Function
0	Reset the camera so that it is pointed straight ahead
B	Toggle the alpha blending on/off. If alpha blending is on, objects that have a set alpha value are displayed as transparent.
F/G	Zoom in/out
H	Toggle the headlight on/off
I	Show/hide information on viewpoint name and camera position
L	Toggle the lighting on/off
T	Toggle the object textures on/off
V	Toggle the navigation zones on/off
W	Toggle the wireframe option on/off
Esc	Return to default viewpoint
R, Home	Return to current viewpoint
Page Up, Page Down	Move between preset viewpoints
S,X, <, >	Rotate the viewpoint about the origin of the scene

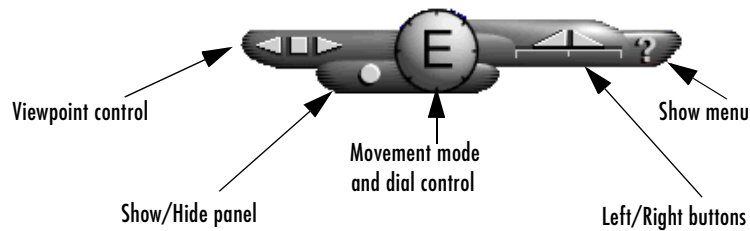
Keyboard Navigation (Continued)

Keyboard Command	Navigation Function
Left Arrow, Right Arrow	Move the camera left and right
Up Arrow, Down Arrow	Move the camera forward and backward
N,M	Tilt the camera right and left
\	Camera is bound/unbound from the viewpoint
Shift-W	Set the navigation method to Walk
Shift-E	Set the navigation method to Examine
Shift-F	Set the navigation method to Fly
A, Z	Pan up and down
Shift-A, Shift-Z	Slide up and down
O,P	Pan right and left
Shift-O, Shift-P	Slide right and left

blaxxun Contact VRML Plug-in

The Virtual Reality Toolbox includes the blaxxun Contact VRML plug-in. This is a popular VRML plug-in for either Microsoft Internet Explorer or Netscape Navigator on a Windows platform. This section provides a quick overview of the functions and controls of the blaxxun Contact VRML plug-in.

When you open a VRML file with a Web browser, the blaxxun Contact VRML plug-in is used to display a virtual scene. A control panel is located at the bottom of the scene.



Viewpoint Control

There are three buttons on the control panel for controlling the viewpoint. The square button in the middle resets the current viewpoint to its initial position. This is the most useful viewpoint control button until you gain enough experience with the viewer to explore worlds using navigation. The keyboard shortcut for the square button is the **Esc** key.

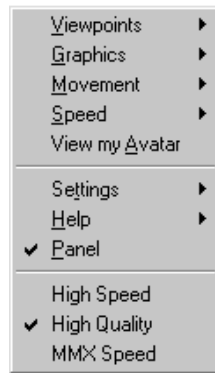
You use the other two triangular buttons to browse forward and backward through author-defined viewpoints of the virtual world. If the author does not define other viewpoints, these buttons are inactive. The keyboard shortcuts for the triangular buttons are the **Page Up** and **Page Down** keys.

Control Menu

You use the control menu to review or select viewer settings and navigation methods. To access the control menu, use the following procedure:

- 1 On the control panel, click the question mark, or place your mouse pointer anywhere in the browser window, and then right-click.

If you selected DirectX with the blaxxun Contact installation, a menu similar to the one shown appears.



- 2 From the menu, you can make changes to the navigational mode, graphic quality, and graphic speed.

Depending on the complexity of the virtual world and the required speed and rendering quality, you can choose the settings that best meet your needs.

Because the viewer's graphical performance is strongly dependent on several factors, you might want to experiment to find a reasonable compromise between the quality and speed for your system.

Navigation

The dial control and left/right buttons give you direct access to the movement mode for walking through a virtual world. However, the movement behavior of your mouse pointer changes depending on the movement mode you select.

When you select a different movement mode, clicking your left mouse button causes your viewpoint to move differently. Practice changing the movement mode and navigating through a virtual world until you get satisfactory results.

To select a movement mode, use the following procedure:

- 1 Place your mouse pointer over a virtual world, then right-click. A menu appears.
- 2 On the menu, point to **Movement**. A submenu appears.
- 3 Select **Walk**, **Slide**, **Rotate**, **Examine**, **Fly**, **Pan**, or **Jump**.

A letter in the center of the dial indicates the current movement mode. For example, in the preceding illustration, the large E stands for Examine mode.

Initially, you should use Examine mode, which is for examining objects from various angles. You will find that the functions of the left/right button controls in Examine mode are the easiest for beginners to master.

Movement Modes

The following table lists the movement modes.

Movement Mode	Description
Walk	Drag the mouse toward the top or the bottom of the screen to move forward or backward, and drag toward the left or right to turn left or right.
Slide	Drag the mouse to move up, down, left, or right within a plane that is perpendicular to your view.
Rotate	Press the left mouse button to select a rotation point within the scene. Then drag the mouse toward the top or bottom to move forward or back, or drag the mouse toward left or right to rotate around the fixed point.
Examine	Press the left mouse button to select a rotation point within the scene. Then drag the mouse up, down, left, or right to rotate the object.
Fly	Press the left mouse button to start flying. Drag the mouse toward the top or bottom to rise or sink, and drag toward the left or right.
Pan	Drag the mouse toward the top or bottom of the scene to loop up and down, and drag toward the left or right to turn left or right.
Jump	Place your mouse pointer over an object, then left-click. Your view moves to that point.

blaxxun Contact Settings

For PC, the Virtual Reality Toolbox includes the blaxxun Contact VRML plug-in for Web browsers. The viewer allows you to select several working configurations, and its performance depends on several factors:

- The speed of your hardware
- System display driver settings
- Method of 3-D rendering
- blaxxun Contact parameters
- The size of the window in which you display the 3-D visualization

You might want to test the various combinations possible on your system to find an optimal configuration for the best performance in 3-D visualization.

With respect to the 3-D rendering method, you can install blaxxun Contact with two basic configurations using OpenGL and Direct3-D drivers. You can tune the viewer performance by setting the parameters in the **Settings-Preferences** dialog box of the viewer floating menu, accessible by right-clicking when viewing a virtual scene.

In Direct3D configuration, you can select the speed and quality on the fly from the top level of the menu. You can, depending on the system capabilities, select one of the options on the menu. For example, you can select High Speed, High Quality, Hardware Acceleration, and MMX Speed.

In the OpenGL configuration, you can set similar rendering properties. From the floating menu, choose **Settings**, and then choose **Preferences**.

VRML Data Types

VRML data types are data types used by VRML nodes to define objects and types of data that can appear in the VRML node fields and events.

This section includes the following topics:

- **VRML Field Data Types**
- **VRML Data Class Types**

VRML Field Data Types

The following table shows the VRML data types and how they are converted to MATLAB types.

For a detailed description of the VRML fields, refer to the *VRML97 Standard*.

VRML Type	Description	VR Toolbox Type
SFBool	The Boolean value true or false.	'on' or 'off'
SFFloat	A 32 bit floating-point value.	Double
SFInt32	A 32 bit signed-integer value. SFInt32_value = floor(double_value)	Double
SFTime	An absolute or relative time value.	Double
SFVec2f	Vector of two floating-point values that you usually use for 2-D coordinates. For example, texture coordinates.	Double array (1-by-2)
SFVec3f	Vector of three floating-point values that you usually use for 3-D coordinates	Double array (1-by-3)

VRML Type	Description	VR Toolbox Type
SFColor	Vector of three floating-point values you use for RGB color specification.	Double array (1-by-3)
SFRotation	Vector of four floating-point values you use for specifying rotation coordinates (x,y,z) of an axis plus rotation angle around that axis.	Double array (1-by-4)
SFImage	Two-dimensional array represented by a sequence of floating-point numbers.	N/A
SFString	String in UTF-8 encoding. Compatible with ASCII, allowing you to use Unicode characters.	String
SFNode	Container for a VRML node.	N/A
MFFloat	Array of SFFloat values.	Double array (n-by-1)
MFInt32	Array of SFInt32 values.	Double array (n-by-1)
MFVec2f	Array of SFVec2f values.	Double array (n-by-2)
MFVec3f	Array of SFvec3f values.	Double array (n-by-3)
MFColor	Array of SFColor values.	Double array (n-by-3)
MFRotation	Array of SFRotation values.	Double array (n-by-4)
MFString	Array of SFString values.	Cell array of strings
MFNode	Array of SFNode values.	N/A

VRML Data Class Types

A node can contain four classes of data: `field`, `exposedField`, `eventIn`, and `eventOut`. These classes define the behavior of the nodes, the way the nodes are stored in the computer memory, and how they can interact with other nodes and external objects.

VRML Data Class	Description
<code>eventIn</code>	An event that can be received by the node
<code>eventOut</code>	An event that can be sent by the node
<code>field</code>	A private node member, holding node data
<code>exposedField</code>	A public node member, holding node data

eventIn

Usually, `eventIn` events correspond to a `field` in the node. Node fields are not accessible from outside the node. The only way you can change them is by having a corresponding `eventIn`.

Some nodes have `eventIn` events that do not correspond to any `field` of that node, but provide additional functionality for it. For example, the **Transform** node has an `addChildren` `eventIn`. When this event is received, the child nodes that are passed are added to the list of children of a given transform.

You use this class type for fields that are exposed to other objects.

eventOut

This event is sent whenever the value of a corresponding node `field` that allows sending events changes its value.

You use this class type for fields that have this functionality.

field

A field can be set to a particular value in the VRML file. Generally, the field is private to the node and its value can be changed only if its node receives a corresponding eventIn. It is important to understand that the field itself cannot be changed on the fly by other nodes or via the external authoring interface.

You use this class type for fields that are not exposed and do not have the eventOut functionality.

exposedField

This is a powerful VRML data class that serves many purposes. You use this class type for fields that have both eventIn and eventOut functionality. The alternative name of the corresponding eventIn is always the field name with a set_ prefix. The name of the eventOut is always the field name with a _changed suffix.

The exposedField class defines how the corresponding eventIn and eventOut behave. For all exposedField classes, when an event occurs, the field value is changed, with a corresponding change to the scene appearance, and an eventOut is sent with the new field value. This allows the chaining of events through many nodes.

The exposedField class is accessible to scripts, whereas the field class is not.

Block Reference

The Virtual Reality Toolbox includes Simulink blocks to output and input signals from a virtual world, control input devices, and interface VRML fields with Simulink signals.

This chapter is a reference for the following blocks:

Joystick Input (p. 6-2)	Asynchronous joystick input device driver
Magellan SpaceMouse (p. 6-4)	Magellan SpaceMouse input device driver
VR Placeholder (p. 6-6)	Generate a vector of signals for masking the unused or unaffected components of VRML fields
VR Signal Expander (p. 6-7)	Expand an input vector into a fully qualified VRML field signal by filling the blank positions in the output port with VR Placeholder signals
VR Sink (p. 6-9)	Provide the GUI interface to output signals from Simulink to virtual worlds
VR Source (p. 6-12)	Provide the GUI interface to input signals from virtual worlds to Simulink

Joystick Input

Purpose Asynchronous joystick input

Library Virtual Reality Toolbox

Description



The Joystick Input block provides a convenient interaction between a Simulink model and the virtual world associated with a Virtual Reality Toolbox block. It works only on Windows operating systems.

The Joystick Input block uses axes, buttons and, if present, the point-of-view selector. You can use this block as you would use any other Simulink source block. Its output ports reflect the status of the joystick controls for axes and buttons.

Block Parameters Dialog Box

Joystick ID — The system ID assigned to the given joystick device. You can find the properties of the joystick connected to the system in the Game Controllers section of the system Control Panel.

Adjust output ports according to joystick capabilities — If you select this check box, the output ports do not have the full width provided by the Windows Game Controllers interface. Instead, the Virtual Reality Toolbox dynamically adjusts the output ports to correspond to the capabilities of the connected joystick each time the model is opened.

Output Ports — Depending on the check box setting previously described, output ports either have fixed maximum width provided by the system Game Controllers interface or the output ports change to correspond to the actual capabilities of the connected joystick.

Output Port	Value	Description
Axes	Vector of doubles in the range < -1; 1 >	Outputs correspond to the current position of the joystick in the given axis. Values are normalized to the range from -1 to 1.
Buttons	Vector of doubles 0 — Button released 1 — Button pressed	Outputs correspond to the current status of joystick buttons.
Point of view	-1 — selector inactive <0;360) — the angle of the POV selector, in degrees	Output corresponds to the current status of the joystick Point of View selector.

Magellan SpaceMouse

Purpose Process input from Magellan SpaceMouse device

Library Virtual Reality Toolbox

Description



Magellan Space Mouse

The Magellan SpaceMouse is a device similar to a joystick in purpose, but it also provides movement control with six degrees of freedom. This block reads the status of the SpaceMouse and provides some commonly used transformations of the input. The SpaceMouse block supports all models of SpaceMouse and PuckMan devices manufactured by 3Dconnexion. It is only supported on Windows operating systems.

Data Type Support A Magellan SpaceMouse block outputs signals of type double.

Block Parameters Dialog Box

Port — Serial port to which the Magellan SpaceMouse is connected. Possible values are COM1...COM4 and USB.

Output Type — This field specifies how the inputs from the device are transformed:

- **Speed** — No transformations are done. Outputs are translation and rotation speeds.
- **Position** — Translations and rotations are integrated. Outputs are position and orientation in the form of roll/pitch/yaw angles.
- **Viewpoint Coordinates** — Translations and rotations are integrated. Outputs are position and orientation in the form of an axis and an angle. You can use these values as viewpoint coordinates in VRML.

Dominant mode — If this check box is selected, the mouse accepts only the prevailing movement and rotation and ignores the others. This is very useful for beginners using the Magellan SpaceMouse.

Disable position movement — Fixes the positions at the initial values, allowing you to change rotations only.

Disable rotation movement — Fixes the rotations at initial values, allowing you to change positions only.

Normalize output angle — Determines whether the integrated rotation angles should wrap on a full circle (360°) or not. This is not used when you set the output mode to **Speed**.

Position sensitivity — Mouse sensitivity for translations. Lower values correspond to higher sensitivity.

Rotation sensitivity — Mouse sensitivity for rotations. Lower values correspond to higher sensitivity.

Initial position — Initial condition for integrated translations. This is not used when you set the output mode to **Speed**.

Initial rotation — Initial condition for integrated rotations. This is not used when you set the output mode to **Speed**.

VR Placeholder

Purpose Send an unspecified value to a Virtual Reality Toolbox block

Library Virtual Reality Toolbox

Description



The VR Placeholder block sends out a special value that is interpreted as “unspecified” by the VR Sink block. When this value appears on the VR Sink input, whether as a single value or as an element of a vector, the appropriate value in the virtual world stays unchanged. Use this block to change only one value from a larger vector. For example, use this block to change just one coordinate from a 3-D position.

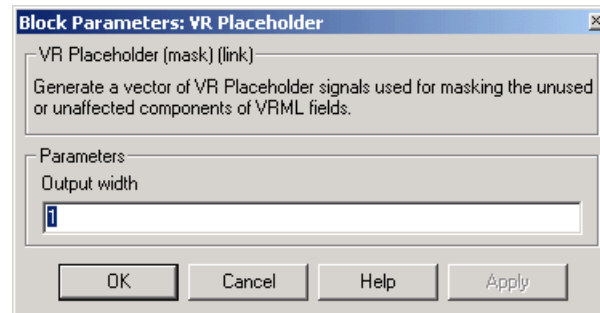
The value output by the VR Placeholder should not be modified before being used in other VR blocks.

Data Type Support

A VR Placeholder block outputs signals of type `double`.

Block Parameters Dialog Box

Output Width — Length of the vector containing placeholder signal values.



Purpose Expand input vectors into fully qualified VRML field vectors

Library Virtual Reality Toolbox

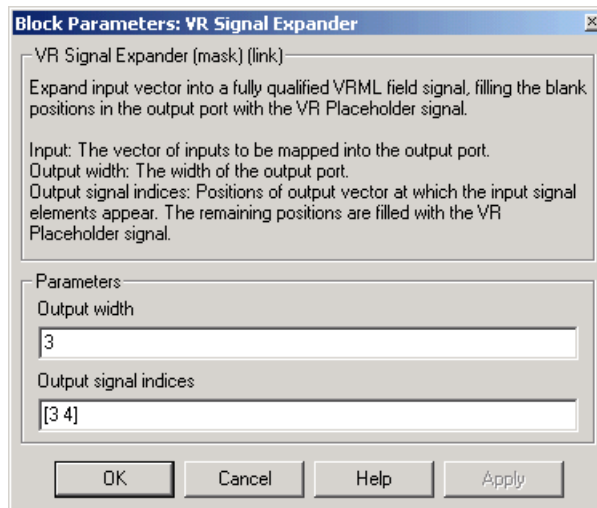
Description The VR Signal Expander block creates a vector of predefined length, using some values from the input ports and filling the rest with placeholder signal values.



VR Signal Expander

Data Type Support A VR Signal Expander block accepts and outputs signals of type double.

Block Parameters Dialog Box



Output width — How long the output vector should be.

Output signal indices — Vector indicating the position at which the input signals appear at the output. The remaining positions are filled with VR Placeholder signals.

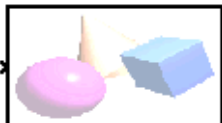
VR Signal Expander

For example, suppose you want an input vector with two signals, and you want an output vector with three signals, and you want the first input signal in position 3 and the second input signal in position 2. In the **Output width** box, enter [3] and in the **Output signal indices** box, enter [3,2]. The first output signal is unspecified.

Purpose Write data from the Simulink model to the virtual world

Library Virtual Reality Toolbox

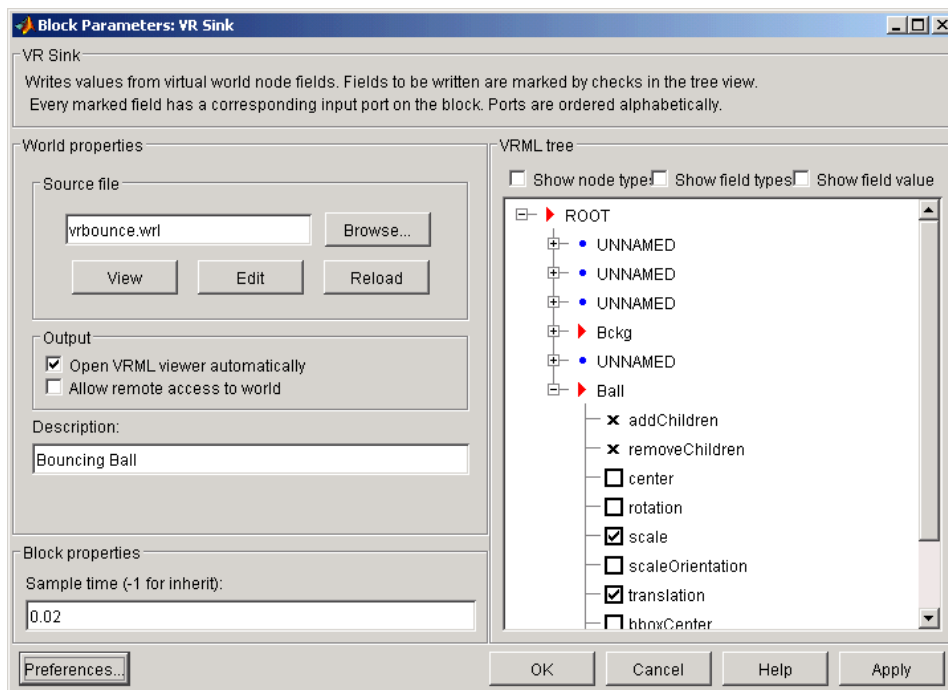
Description The VR Sink block writes values from its ports to virtual world fields specified in the **Block Parameters** dialog box.



VR Sink

Data Type Support A VR Sink block accepts signals of type double.

Block Parameters Dialog Box



Source file — VRML file name specifying the virtual world this block is connected to. The **View** button allows you to view the world in the Virtual Reality Toolbox viewer or a Web browser. The **Edit** button launches an external VRML editor, and the **Reload** button reloads the world after you change it. By default, the full path to the associated .wrl file appears in this text box. If you enter only the filename in this box, the Virtual Reality Toolbox assumes that the .wrl file resides in the same directory as the model file.

Open VRML viewer automatically — If you select this check box, the default VRML viewer displays the virtual world after loading the Simulink model.

Allow remote access to world — If you select this check box, the virtual world is accessible for viewing on a client computer. If it is not selected, the world is visible only on the host computer. This is equivalent to the RemoteView property of a vrworld object. See Chapter 4, “MATLAB Interface.”

Description — Description that is displayed in all virtual reality object listings, in the title bar of the Virtual Reality Toolbox viewer, and in the list of virtual worlds on the Virtual Reality Toolbox HTML page. This is equivalent to the Description property of a vrworld object. See Chapter 4, “MATLAB Interface.”

Sample time — Enter the sample time or -1 for inherited sample time.

VRML Tree — This box shows the structure of the VRML file and the virtual world itself.

Nodes that have names are marked with red arrows and can be accessed from MATLAB. Nodes without names, but whose children are named, are also marked with red arrows. This marking scheme makes it possible for you to find all accessible nodes by traversing the tree using arrows. Other nodes have a blue dot before their names.

Fields with settable values have check boxes. Use these check boxes to select the fields you want Simulink to output values to. For every selected field, an input port is created in the block. Input ports are assigned to the selected nodes and fields in the order corresponding to the VRML file.

Fields whose values cannot be written (because their parent nodes do not have names, or because they are not of VRML type eventIn or exposedField), have an X-shaped icon.

Show node types — If you select this check box, node types are shown in the VRML tree.

Show field types — If you select this check box, field types are shown in the VRML tree.

Show field values — If you select this check box, the dialog box shows, in the VRML tree, the current numeric values of the fields.

VR Source

Purpose

Read data from the virtual world to a Simulink model

Library

Virtual Reality Toolbox

Description

The VR Source block reads values from virtual world fields specified in the **Block Parameters** dialog box and inputs their values.

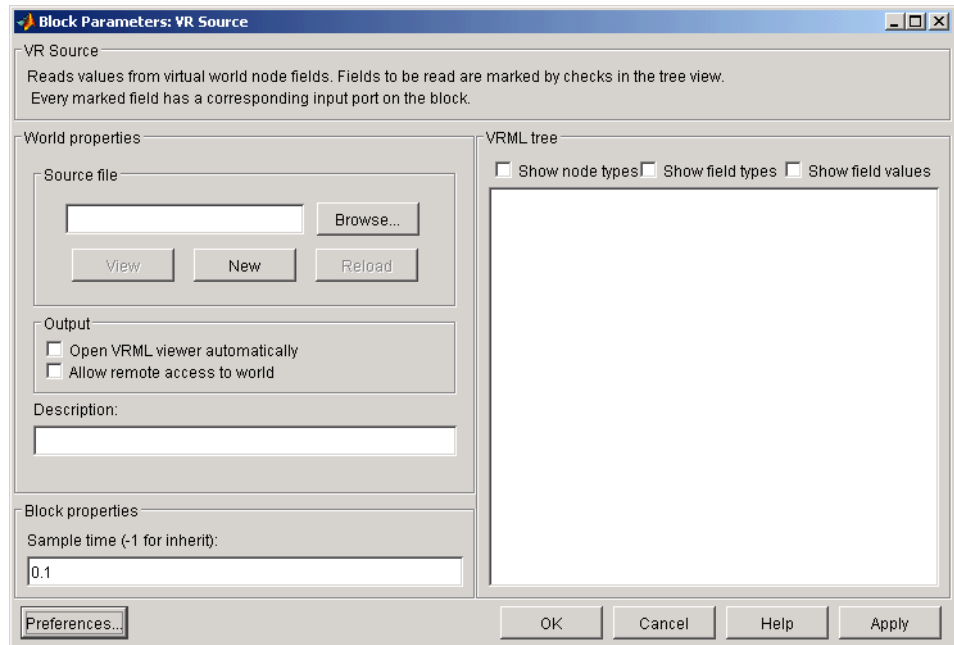


VR Source

Data Type Support

A VR Source block outputs signals of type double.

Block Parameters Dialog Box



Source file — VRML file name specifying the virtual world this block is connected to. The **View** button allows you to view the world in a viewer, the **Edit** button launches an external VRML editor, and the **Reload** button reloads the world after you change it.

Open VRML viewer automatically — If you select this check box, the default VRML viewer displays the virtual world after loading the Simulink model.

Allow remote access to world — If this check box is selected, the virtual world is accessible for viewing on a client computer. If it is not selected, the world is visible only on the host computer. This is equivalent to the `RemoteView` property of a `vrworld` object. See Chapter 4, “MATLAB Interface.”

Description — Description that is displayed in all virtual reality object listings, in the title bar of the Virtual Reality Toolbox viewer, and in the list of virtual worlds on the Virtual Reality Toolbox HTML page. This is equivalent to the `Description` property of a `vrworld` object. See Chapter 4, “MATLAB Interface.”

Sample time — Enter the sample time or -1 for inherited sample time.

VRML Tree — This box shows the structure of the VRML file and the virtual world itself.

Nodes that have names are marked with red arrows and can be accessed from MATLAB. Nodes that do not have names, but whose children are named, are also marked with red arrows. This marking scheme makes it possible for you to find all accessible nodes by traversing the tree using arrows. Other nodes have a blue dot before their names.

Fields with readable values have check boxes. Use these check boxes to select the fields you want Simulink to monitor and to input values from. For every selected field, an output port is created in the block. Output ports are assigned to the selected nodes and fields in the order corresponding to the VRML file.

Fields whose values cannot be read (because their parent nodes do not have names, or because their values cannot be imported to Simulink), have an X-shaped icon.

Show node types — If this check box is selected, node types are shown in the VRML tree.

VR Source

Show field types — If this check box is selected, field types are shown in the VRML tree.

Show field values — If you select this check box, the dialog box shows the current numeric values of the fields in the VRML tree.

Function Reference

This chapter is a reference for the MATLAB interface functions. Some of the Virtual Reality Toolbox features, such as setting VRML text values or arrays with variable size, are accessible from the MATLAB interface only.

This chapter is a reference for the following functions.

<code>vrclear</code> (p. 7-2)	Delete all closed virtual worlds from memory
<code>vrclose</code> (p. 7-3)	Close Virtual Reality figure windows
<code>vrdrawnow</code> (p. 7-4)	Update the virtual world
<code>vrgetpref</code> (p. 7-5)	Read values of Virtual Reality Toolbox preferences
<code>vrinstall</code> (p. 7-8)	Install and check Virtual Reality Toolbox components
<code>vrlib</code> (p. 7-10)	Open Simulink block library for the Virtual Reality Toolbox
<code>vrsetpref</code> (p. 7-11)	Change Virtual Reality Toolbox preferences
<code>vrview</code> (p. 7-12)	View a virtual world using either the Virtual Reality Toolbox viewer or the Web browser
<code>vrwho</code> (p. 7-13)	Provide list of virtual worlds in memory
<code>vrwhos</code> (p. 7-14)	Provide detailed list of virtual worlds in memory

vrclear

Purpose Delete all closed virtual worlds from memory

Syntax `vrclear`
`vrclear('force')`

Description The `vrclear` function removes from memory all virtual worlds that are closed and invalidates all `vrworld` objects related to them. This command does not affect open virtual worlds. Open virtual worlds include those loaded from Simulink. You use this command to

- Ensure that the maximum amount of memory is free before a memory-consuming operation takes place
- Perform a general cleanup of memory

The `vrclear('force')` command removes all virtual worlds from memory, including worlds opened from Simulink.

See Also `vrworld/delete`, `vrworld`

Purpose	Close virtual reality figure windows
Syntax	<code>vrclose</code> <code>vrclose all</code>
Description	<code>vrclose</code> and <code>vrclose all</code> close all the open virtual reality figures.
Example	<p>Open a series of virtual reality figure windows by typing</p> <pre>vrend vrbounce vrlights vrcrane</pre> <p>Arrange the viewer windows so they are all visible. Type</p> <pre>vrclose</pre> <p>All the virtual reality figure windows disappear from the screen.</p>
See also	<code>vrfigure/close</code>

vrdrawnow

Purpose Update the virtual world

Syntax `vrdrawnow`

Description `vrdrawnow` removes from the queue pending changes to the virtual world and makes these changes to the scene in the viewer.

Changes to the scene are normally queued and the views are updated when

- MATLAB is idle for some time (no Simulink model is running and no M-file is being executed).
- A Simulink step is finished.

Purpose Read values of Virtual Reality Toolbox preferences

Syntax

```
x = vrgetpref
x = vrgetpref('preference_name')
x = vrgetpref('preference_name','factory')
x = vrgetpref('factory')
```

Arguments *preference_name* Name of the preference to read.

Description `x = vrgetpref` returns the values of all the Virtual Reality Toolbox preferences in a structure array.

`x = vrgetpref('preference_name')` returns the value of the specified preference. If *preference_name* is a cell array of preference names, a cell array of corresponding preference values is returned.

`x = vrgetpref('preference_name','factory')` returns the default value for the specified preference.

`x = vrgetpref('factory')` returns the default values for all the preferences. The following preferences are defined.

DefaultFigurePosition	Sets the position and size of the Virtual Reality Toolbox viewer window.
DefaultPanelMode	Determines the appearance of the control panel in the viewer. Valid values are 'opaque', 'translucent', 'off', 'halfbar', 'bar', and 'factory'. Default is 'halfbar'.
DefaultViewer	Specifies which viewer is used to view a virtual scene. The Virtual Reality Toolbox viewer is used when the preference is set to 'internal'. The Web browser is used when this preference is set to 'web'. Default is 'internal'.
Editor	Path to the VRML editor. If this path is empty, the MATLAB editor is used.

HttpPort	IP port number used to access the VR server over the Web via HTTP. If you change this preference, you must restart MATLAB before the change takes effect.
TransportBuffer	Length of the transport buffer (network packet overlay) for communication between the VR server and its clients.
VrPort	IP port used for communication between the VR server and its clients. If you change this preference, you must restart MATLAB before the change takes effect.

Note that the `HttpPort`, `VrPort`, and `TransportBuffer` preferences affect Web-based viewing of virtual worlds. `DefaultFigurePosition` and `DefaultPanelMode` affect the Virtual Reality Toolbox viewer.

DefaultPanelMode — The `DefaultPanelMode` preference controls the appearance of the control panel in the Virtual Reality Toolbox viewer. For example, setting this value to 'translucent' causes the control panel to appear translucent.

DefaultViewer — The `DefaultViewer` preference determines whether the virtual scene appears in the Virtual Reality Toolbox viewer or in your Web browser. If the preference is set to 'internal', the Virtual Reality Toolbox viewer is the default viewer. If it is set to 'web', the default Web browser with the VRML plug-in is the default viewer.

Editor Preference — The `Editor` preference contains a path to the VRML editor executable file. When you use the `edit` command, the Virtual Reality Toolbox runs the VRML editor executable with all parameters required to edit the VRML file.

When you run the editor, the Virtual Reality Toolbox uses the `Editor` preference value as if you typed it into a command line. The following tokens are interpreted:

<code>%matlabroot</code>	Refers to the MATLAB root directory
<code>%file</code>	Refers to the VRML file name

For instance, a possible value for the Editor preference is

```
'%matlabroot\bin\win32\meditor.exe %file'
```

Note the quotation marks around the executable filename and the VRML filename. If this preference is empty, the MATLAB editor is used.

HttpPort Preference — The HttpPort preference specifies the network port to be used for Web access. The port is given in the Web URL as follows:

```
http://server.name:port_number
```

The default value of this preference is 8123.

Transport Buffer — The TransportBuffer preference defines the size of the message window for client-server communication. This value determines how many messages, at a maximum, can travel between the client and the server at one time.

Generally, higher values for this preference make the animation run more smoothly, but with longer reaction times. (More messages in the line create a buffer that compensates for the unbalanced delays of the network transfer.)

The default value is 5, which is optimal for most purposes. You should change this value only if the animation is significantly distorted or the reaction times are very slow. On fast connections, where delays are introduced more by the client rendering speed, this value has very little effect. Viewing on a host computer is equivalent to an extremely fast connection. On slow connections, the correct value can improve the rendering speed significantly but, of course, the absolute maximum is determined by the maximum connection throughput.

VrPort Preference — The VrPort preference specifies the network port to use for communication between the Virtual Reality Toolbox server (host computer) and its clients (client computers). Normally, this communication is completely invisible to the user. However, if you view a virtual world from a client computer, you might need to configure the security network system (firewall) so that it allows connections on this port. The default value of this preference is 8124.

See Also

vrsetpref

vrinstall

Purpose Install and check Virtual Reality Toolbox components

Syntax

```
vrinstall('action')  
vrinstall action  
vrinstall('action','component')  
vrinstall action component  
x = vrinstall('action', 'component')
```

Arguments

action Type of action for this function. Values are -interactive, -selftest, -check, -install, and -uninstall.

component Name of the component for the action. Values are viewer and editor.

Description You use this function to manage the installation of optional software components related to the Virtual Reality Toolbox. Currently there are two such components: VRML plug-in and VRML editor.

Action Value	Description
-selftest	Checks the integrity of the Virtual Reality Toolbox. If this function reports an error, you should reinstall the Virtual Reality Toolbox. The function <code>vrinstall</code> automatically does a self-test with any other actions.
-interactive	Checks for the installed components, and then displays a list of uninstalled components you can choose to install.
-check	Checks the installation of optional components. If the given component is installed, returns 1. If the given component is not installed, returns 0. If you do not specify a component, displays a list of components and their status.

Action Value	Description (Continued)
-install	Installs optional components. This action requires you to specify the component name. All components can be installed using this command, but some of them (currently only the plug-in) need to be uninstalled using the system standard uninstallation procedure.
-uninstall	<p>Uninstalls optional components. This option is currently available for the editor only. Note that this action does not remove the files for the editor from the installation directory. It removes the editor registry information.</p> <p>If you want to uninstall the VRML plug-in, exit MATLAB and, from the Control Panel window, select Add/Remove Programs.</p>

Examples

Install the VRML plug-in. This command starts the blaxxun Contact install program and installs the plug-in to your default Web browser.

```
vrinstall -install viewer
```

Install the VRML editor. This command associates V-Realm Builder with the **Edit** button in the **Block parameters** dialog boxes.

```
vrinstall -install editor
```

vrlib

Purpose	Open the Simulink block library for the Virtual Reality Toolbox
Syntax	<code>vrlib</code>
Description	<p>The Simulink library for the Virtual Reality Toolbox has six blocks: VR Sink, VR Source, VR Placeholder, VR Signal Expander, Joystick Input, and Magellan SpaceMouse.</p> <p>Alternatively, you can access these blocks from a Simulink block diagram. In the Simulink window, from the View menu, click Show Library Browser.</p>

Purpose Change Virtual Reality Toolbox preferences

Syntax `vrsetpref('preference_name', preference_value)`
`vrsetpref('factory')`

Arguments

<i>preference_name</i>	Name of the preference.
<i>preference_value</i>	New value of the preference.

Description This command sets the given Virtual Reality Toolbox preference to a given value. For detailed description of the preferences, see `vrgetpref` on page 7-5. Changes to the `HttpPort` or `VrPort` preferences only take effect after you restart MATLAB.

When you use 'factory' as a single argument, all preferences are reset to their default values. If you use 'factory' for a preference value, that single preference is reset to its default.

See Also `vrgetpref`

vrview

Purpose View a virtual world using either the Virtual Reality Toolbox viewer or a Web browser

Syntax

```
vrview
x = vrview('filename')
x = vrview('filename', '-internal')
x = vrview('filename', '-web')
```

Description `vrview` opens the default Web browser and loads the Virtual Reality Toolbox Web page containing a list of virtual worlds available for viewing.

`x = vrview('filename')` creates a virtual world associated with the `.wrl` file, opens the virtual world, and displays it in the Virtual Reality Toolbox viewer or the Web browser depending on the value of the `DefaultViewer` preference. The handle to the virtual world is returned.

`x = vrview('filename', '-internal')` creates a virtual world associated with the `.wrl` file, opens the virtual world, and displays it in the Virtual Reality Toolbox viewer.

`x = vrview('filename', '-web')` creates a virtual world associated with the `.wrl` file, opens the virtual world, and displays it in your Web browser.

See Also `vrworld`, `vrworld/open`, `vrworld/view`

Purpose Provide a list of virtual worlds in memory

Syntax `vrwho`
`x = vrwho`

Description If you do not specify an output parameter, `vrwho` displays a list of virtual worlds in memory in the MATLAB Command Window.

If you specify an output parameter, `vrwho` returns a vector of handles to existing `vrworld` objects, including those opened from Simulink.

See Also `vrwhos`, `vrworld`, `vrclear`

vrwhos

Purpose	Provide a detailed list of virtual worlds in memory
Syntax	<code>vrwhos</code>
Description	Displays a list of virtual worlds currently in memory, with a description, in the MATLAB Command Window. The relation between <code>vrwho</code> and <code>vrwhos</code> is similar to the relation between <code>who</code> and <code>whos</code> .
See Also	<code>vrwho</code> , <code>vrclear</code>

vrworld Object Reference

While the Simulink interface is the preferred method for using the Virtual Reality Toolbox, you can access virtual worlds through the MATLAB interface. To use this interface, you create objects in the MATLAB workspace and associate those objects with your virtual worlds.

MATLAB functions and the Simulink interface share the same Virtual Reality Toolbox objects. It is possible to access these objects from both the MATLAB and Simulink interfaces simultaneously.

This chapter includes the following sections:

vrworld Object Properties (p. 8-2)

vrworld object properties allow you to control the behavior of objects

vrworld Object Methods (p. 8-4)

vrworld object methods allow you to access and manipulate objects

vrworld Object Properties

A vrworld object is a handle of a virtual scene. It allows you to interact with and control the scene.

The vrworld object properties allow you to control the behavior of objects. The following table lists the properties for vrworld objects.

Property	Value	Description
Clients	Scalar	Number of clients currently viewing the virtual world. Read-only
ClientUpdates	off on Default: on	Client cannot or can update the virtual scene. Read/write.
Description	String Default: automatically taken from the VRML file property Title	Description of the world. Read/write.
Figures	Vector of vrfigure objects	Vector of handles to Virtual Reality Toolbox viewer windows currently viewing the virtual world. Read-only.
FileName	String	Name of the associated VRML file. Read/write.
Nodes	Vector of vrnode objects	Named vrnode objects in the virtual world. Read-only.
Open	off on Default: off	Indicates a closed or open virtual world. Read-only.

Property	Value	Description (Continued)
RemoteView	off on Default: off	Remote access flag. Read/write.
View	off on Default: on	Indicates an unviewable or viewable virtual world. Read/write.

vrworld Object Methods

The vrworld object methods allow you to access and manipulate objects. The following is a list of vrworld object methods. A reference page for each vrworld object method follows the table.

Method	Description
vrworld	Create a new vrworld object associated with a virtual world.
vrworld/close	Close a virtual world.
vrworld/delete	Delete virtual world from memory.
vrworld/edit	Open a virtual world file in external VRML editor.
vrworld/get	Read the property value of a vrworld object.
vrworld/isvalid	Return 1 if the vrworld object is valid, and 0 if it is not.
vrworld/nodes	List nodes available in the virtual world.
vrworld/open	Open virtual world.
vrworld/reload	Reload virtual world from the associated VRML file.
vrworld/save	Write virtual world to VRML file.
vrworld/set	Change property value of a vrworld object.
vrworld/view	View a virtual world.

Purpose	Create a new vrworld object associated with a virtual world		
Syntax	<pre>myworld = vrworld('vrm1_file.wrl') myworld = vrworld myworld = vrworld([])</pre>		
Arguments	<table><tr><td><i>vrm1_file.wrl</i></td><td>Name of the VRML file from which the virtual world is loaded. If no file extension is specified, the file extension <code>.wrl</code> is assumed.</td></tr></table>	<i>vrm1_file.wrl</i>	Name of the VRML file from which the virtual world is loaded. If no file extension is specified, the file extension <code>.wrl</code> is assumed.
<i>vrm1_file.wrl</i>	Name of the VRML file from which the virtual world is loaded. If no file extension is specified, the file extension <code>.wrl</code> is assumed.		
Description	<p><code>myworld = vrworld('vrm1_file.wrl')</code> creates a handle to a vrworld object that is associated with the specified VRML file.</p> <p><code>myworld = vrworld</code> creates an empty vrworld handle that does not refer to any virtual world.</p> <p><code>myworld = vrworld([])</code> returns an empty array of vrworld handles.</p> <p>A vrworld object identifies a virtual world in a way very similar to a handle. All functions that affect virtual worlds accept a vrworld object as an argument to identify the virtual world.</p> <p>If the given virtual world already exists in memory, the new vrworld object is associated with the existing virtual world. That is, a second virtual world is not loaded into memory. If the virtual world does not exist in memory, it is loaded from the associated VRML file. The newly loaded virtual world is closed and must be opened before you can use it.</p> <p>The vrworld object associated with a virtual world remains valid until you use either <code>delete</code> or <code>vrclear</code>.</p>		
Examples	<pre>myworld = vrworld('vrpend.wrl')</pre>		
See Also	<code>vrworld/open</code> , <code>vrworld/delete</code> , <code>vrworld/close</code>		

vrworld/close

Purpose Close a virtual world

Syntax `close(vrworld_object)`

Arguments `vrworld_object` Name of a vrworld object representing the virtual world.

Description This method changes the virtual world from an opened to a closed state:

- If the world was opened more than once, you must use an appropriate number of `close` calls before the virtual world closes.
- If `vrworld_object` is a vector of vrworld objects, all associated virtual worlds close.
- If the virtual world is already closed, `close` does nothing.

Opening and closing virtual worlds is a mechanism of memory management. When the system needs more memory and the virtual world is closed, you can discard its contents at any time.

Generally, you should close a virtual world when you no longer need it. This allows you to reuse the memory it occupied. The vrworld objects associated with this virtual world stay valid after it is closed, so the virtual world can be opened again without creating a new vrworld object.

Examples

```
myworld = vrworld('vrpend.wrl')
open(myworld)
close(myworld)
```

See Also `vrworld`, `vrworld/open`, `vrworld/delete`

Purpose Delete a virtual world from memory

Syntax `delete(vrworld_object)`

Arguments *vrworld_object* Name of a vrworld object representing a virtual world.

Description The `delete` function removes from memory the virtual world associated with a vrworld object. The virtual world must be closed before you can delete it.

Deleting a virtual world frees the virtual world from memory and invalidates all existing vrworld objects associated with the virtual world.

You do not commonly use this method. One of the possible reasons to use this method is to ensure that a large virtual world is removed from memory before another memory-consuming operation starts.

See Also `vrworld/close`, `vrclear`

vrworld/edit

Purpose Open a virtual world file in an external VRML editor

Syntax `edit(vrworld_object)`

Arguments `vrworld_object` Name of a vrworld object representing a virtual world.

Description The `edit` function opens the VRML file associated with the `vrworld` object in a VRML editor. The Editor preference specifies the VRML editor to use.

The VRML editor saves any changes you make directly to a virtual world file. If the virtual world is open,

- Use the save command in the VRML editor to save the changes to a virtual world file. In MATLAB, the changes appear after you reload the virtual world.
- Use the save method in MATLAB to replace the modified VRML file. Any changes you made in the editor are lost.

See Also `vrworld/reload`, `vrworld/save`

Purpose Read the property value of a vrworld object

Syntax

```
get(vrworld_object)
x = get(vrworld_object, 'property_name')
```

Arguments

<i>vrworld_object</i>	Name of a vrworld object representing a virtual world.
<i>property_name</i>	Name of the property.

Description If no property name is given, the `get(vrworld_object)` method displays all the virtual world properties and their values. When a property name is given, the value of that property is returned.

The following are properties of vrworld objects. Names are not case sensitive.

Field Name	Description
Clients	Number of clients viewing this world.
ClientUpdates	This is 'on' if the clients are allowed to update the viewed scene and 'off' if they are not.
Description	Description of the virtual world, as it appears on the main Web page.
Figures	Handles of figures currently open for this world.
FileName	Name of the VRML file associated with this world.
Nodes	A vector of vrnode objects for all nodes in the virtual world.
Open	If the virtual world is open, value is 'on'. If the virtual world is closed, value is 'off'.
RemoteView	If virtual world is enabled for remote viewing, it is set to 'on'; otherwise, it is set to 'off'.
View	If virtual world is enabled for viewing, it is set to 'on'; otherwise, it is set to 'off'.

The `ClientUpdates` property is set to 'on' by default and can be set by the user. When it is set to 'off', the viewers looking at this virtual world should not update the view according to the virtual world changes. That is, the view is frozen until this property is changed to 'on'. This is useful for preventing tearing effects with complex animations. Before every animation frame, set `ClientUpdates` to 'off', make the appropriate modifications to the object positions, and then switch `ClientUpdates` back to 'on'.

The `Description` property defaults to '(untitled)' and can be set by the user. If the virtual world is loaded from a VRML file containing a **WorldInfo** node with a `title` property (see the VRML reference), the `Description` property is loaded from the VRML file instead.

The `Nodes` property is valid only when the virtual world is open. If the virtual world is closed, `Nodes` always contains an empty vector.

The `RemoteView` property is set to 'off' by default and can be set by the user. If it is set to 'on', all viewers can access the virtual world through the Web interface. If it is set to 'off', only host viewers can access it.

The `View` property is set to 'on' by default and can be set by the user. When it is set to 'off', the virtual world is not accessible by the viewer. You rarely use this property.

See Also

`vrworld/set`, `vrworld`

Purpose Return 1 if the vrworld object is valid, and 0 if it is not

Syntax `x = isvalid(vrworld_object)`

Arguments *vrworld_object* Name of a vrworld object representing a virtual world.

Description A vrworld object is considered valid if its associated virtual world still exists. `x = isvalid(vrworld_object)` returns an array that contains a 1 where the elements of *vrworld_object* are valid vrworld objects, and returns a 0 where they are not.

You use this method to check whether the vrworld object is still valid. Using a `delete` or `vrclear` command can make a vrworld object invalid.

See Also `vrfigure/isvalid`, `vrnode/isvalid`

vrworld/nodes

Purpose	List nodes available in the virtual world		
Syntax	<pre>nodes(<i>vrworld_object</i>) x = nodes(<i>vrworld_object</i>)</pre>		
Arguments	<table><tr><td><i>vrworld_object</i></td><td>Name of a vrworld object representing a virtual world.</td></tr></table>	<i>vrworld_object</i>	Name of a vrworld object representing a virtual world.
<i>vrworld_object</i>	Name of a vrworld object representing a virtual world.		
Description	<p>If you give an output argument, the method <code>nodes</code> returns a cell array of the names of all available nodes in the world. If you do not give an output argument, the list of nodes is displayed in the MATLAB window.</p> <p>You can use the <code>'-full'</code> switch to obtain a detailed list that contains not only the nodes, but also all their fields. This switch affects only the output to the MATLAB Command Window.</p> <p>The virtual world must be open for you to use this method.</p>		
See Also	<code>vrworld</code> , <code>vrworld/open</code>		

Purpose Open a virtual world

Syntax `open(vrworld_object)`

Arguments `vrworld_object` Name of a vrworld object representing a virtual world.

Description The open method changes the associated virtual world from a closed to an opened state. This ensures that the virtual world is loaded into memory and you cannot remove it from memory until you close it. If the input argument is an array of virtual world handles, all the virtual worlds associated with those handles are opened. The virtual world must be open for you to use it. You can close the virtual world by using the method `close`.

You can call the method `open` more than once, but you must use an appropriate number of `close` calls before the virtual world returns to a closed state.

Example Create two vrworld objects by typing

```
myworld1 = vrworld('vrmount.wr1')
```

```
myworld2 = vrworld('vrpend.wr1')
```

Next, create an array of virtual world handles by typing

```
myworlds = [myworld1 myworld2];
```

```
open(myworlds) opens both of these virtual worlds.
```

See Also `vrworld`, `vrworld/close`

vrworld/reload

Purpose	Reload a virtual world from VRML file
Syntax	<code>reload(vrworld_object)</code>
Arguments	<i>vrworld_object</i> A vrworld object representing a virtual world.
Description	<p>The <code>reload</code> method reloads the virtual world from the VRML file associated with the <code>vrworld</code> object. If the input argument is an array of virtual world handles, all the virtual worlds associated with those handles are reloaded. The virtual world must be open for you to use this method.</p> <p><code>reload</code> forces all the clients currently viewing the virtual world to reload it. This is useful when there are changes to the VRML file.</p>
See Also	<code>vrworld/edit</code> , <code>vrworld/save</code> , <code>vrworld/open</code>

Purpose Write a virtual world to VRML file

Syntax `save(vrworld_object, 'vrml_file')`

Arguments

<code>vrworld_object</code>	Name of a vrworld object representing a virtual world.
<code>vrml_file</code>	Name of the VRML file to save the virtual world to.

Description The save method saves the current virtual world to a VRML97 file. The virtual world must be open for you to use this method.

The resulting file is a VRML97 compliant UTF-8 encoded text file. Lines are indented using spaces. Line ends are encoded as CR-LF or LF according to the local system default. Values are separated by spaces.

See Also `vrworld/edit`, `vrworld/reload`, `vrworld/open`

vrworld/set

Purpose Change property values of a vrworld object

Syntax `set(vrworld_object, 'property_name', property_value)`

Arguments

<i>vrworld_object</i>	Name of a vrworld object representing a virtual world.
<i>property_name</i>	Name of the property.
<i>property_value</i>	New value of the property.

Description You can change the Description, View, RemoteView, and ClientUpdates properties. For a detailed discussion of these properties, see [vrfigure/get](#).

See Also [vrworld/get](#), [vrworld](#)

Purpose	View a virtual world		
Syntax	<pre>view(vrworld_object) x = view(vrworld_object) x = view(vrworld_object, '-internal') x = view(vrworld_object, '-web')</pre>		
Arguments	<table><tr><td><i>vrworld_object</i></td><td>Name of a vrworld object representing a virtual world.</td></tr></table>	<i>vrworld_object</i>	Name of a vrworld object representing a virtual world.
<i>vrworld_object</i>	Name of a vrworld object representing a virtual world.		
Description	<p>The <code>view</code> method opens the default VRML viewer on the host computer and loads the virtual world associated with the <code>vrworld</code> object into the viewer window. You specify the default VRML viewer using the <code>DefaultViewer</code> preference. The virtual world must be opened for you to use this method.</p> <p><code>x = view(vrworld_object)</code> opens the default VRML viewer on the host computer and loads the virtual world associated with the <code>vrworld</code> object into the viewer window. If the Virtual Reality Toolbox viewer is used, <code>view</code> also returns the <code>vrfigure</code> handle of the viewer window. If a Web browser is used, <code>view</code> returns an empty array of <code>vrfigure</code> handles.</p> <p><code>x = view(vrworld_object, '-internal')</code> opens the virtual world in the Virtual Reality Toolbox viewer.</p> <p><code>x = view(vrworld_object, '-web')</code> opens the virtual world in the Web browser.</p> <p>If the virtual world is disabled for viewing (that is, the <code>View</code> property for the associated <code>vrworld</code> object is set to <code>'off'</code>), the <code>view</code> method does nothing.</p>		
Examples	<pre>myworld = vrworld('vrpend.wrl') open(myworld) view(myworld)</pre>		
See Also	<code>vrworld</code> , <code>vrview</code>		

vrnode Object Reference

While the Simulink interface is the preferred method for using the Virtual Reality Toolbox, you can access virtual worlds through the MATLAB interface. To use this interface, you create objects in the MATLAB workspace and associate those objects with your virtual worlds.

MATLAB functions and the Simulink interface share the same Virtual Reality Toolbox objects. It is possible to access these objects from both the MATLAB and Simulink interfaces simultaneously.

This chapter includes the following sections:

vrnode Object Properties (p. 9-2)

vrnode object properties allow you to control the behavior of objects.

vrnode Object Methods (p. 9-3)

vrnode object methods allow you to access and manipulate objects

vrnode Object Properties

A vrnode object is a handle of a VRML node. It allows you to get and set the node properties. A vrnode object is a child object of a vrworld object.

The vrnode object properties allow you to control the behavior of objects. The following table lists the properties for vrnode objects.

Property	Value	Description
Fields	Cell array	Valid field names for the VRML node. Read-only.
Name	String	Name of the node. Read-only.
Type	String	VRML type of the node. Read-only.
World	Handle	Handle of the parent vrworld object. Read-only.

vrnode Object Methods

A vrnode object is a handle of a VRML node. It allows you to get and set the node properties. A vrnode object is a child object of a vrworld object.

The vrnode object methods allow you to access and manipulate objects. The following is a list of the vrnode object methods. A reference page for each vrnode object method follows the table.

Method	Description
vrnode	Create a new node or a handle to an existing node.
vrnode/delete	Delete a vrnode object.
vrnode/fields	Return summary of VRML fields of a node object.
vrnode/get	Read property value of vrnode object or VRML field.
vrnode/getfield	Get a field value of a vrnode object.
vrnode/isvalid	Return 1 if the vrnode object is valid, and 0 if it is not.
vrnode/set	Change a property or a VRML field of a virtual world node.
vrnode/setfield	Change a field value of a vrnode object.
vrnode/sync	Enable or disable synchronization of VRML fields with clients.

vrnode

Purpose

Create a new node or a handle to an existing node

Syntax

```
mynode = vrnode
mynode = vrnode([])
mynode = vrnode(vrworld_object, 'node_name')
mynode = vrnode(vrworld_object, 'node_name', 'node_type')
mynode = vrnode(parent_node, 'parent_field', 'node_name',
'node_type')
```

Arguments

<i>vrworld_object</i>	Name of a vrworld object representing a virtual world.
<i>node_name</i>	Name of the node.
<i>node_type</i>	Type of the node.
<i>parent_node</i>	Name of the parent node that is a vrnode object.
<i>parent_field</i>	Name of the field of the parent node.

Description

`mynode = vrnode` creates an empty vrnode handle that does not reference any node.

`mynode = vrnode([])` creates an empty array of vrnode handles.

`mynode = vrnode(vrworld_object, 'node_name')` creates a handle to an existing named node in the virtual world.

`mynode = vrnode(vrworld_object, 'node_name', 'node_type')` creates a new node called *node_name* of type *node_type* on the root of the virtual world. It returns the handle to the newly created node.

`mynode = vrnode(parent_node, 'parent_field', 'node_name', 'node_type')` creates a new node called *node_name* of type *node_type* that is a child of the *parent_node* and resides in the field *parent_field*. It returns the handle to the newly created node.

A vrnode object identifies a virtual world node in a way very similar to a handle. If the vrnode method is applied to a node that does not exist, the node is created, the vrnode object is created, and the handle to the vrnode object is returned. If the vrnode method is applied to an existing node, the handle to the vrnode object associated with this node is returned.

See Also

vrworld, vrnode/get, vrnode/set, vrnode/getfield, vrnode/setfield,
vrnode/delete

vrnode/delete

Purpose Delete a vrnode object

Syntax `delete(vrnode_object)`
`delete(n)`

Arguments `vrnode_object` Name of a vrnode object.

Description `delete(vrnode_object)` deletes the virtual world node.

`delete(n)` deletes the vrnode object referenced by the vrnode handle *n*. If *n* is a vector of vrnode handles, multiple nodes are deleted.

As soon as a node is deleted, it and all its child objects are removed from all clients connected to the virtual world.

See Also `vrworld/delete`

Purpose

Return a summary of VRML fields of a node object

Syntax

```
fields(vrnode_object)
x = fields(vrnode_object)
```

Arguments

vrnode_object Name of a vrnode object representing the node to be queried.

Description

`fields(vrnode_object)` displays a list of VRML fields and subfields of the node associated with the vrnode object in the MATLAB Command Window.

`x = fields(vrnode_object)` returns the VRML fields of the node associated with the vrnode object in a structure array. The resulting structure contains the VRML fields and the following subfields:

- **Type** is the name of the VRML field type, for example, 'MFString', 'SFColor'.
- **Access** is the accessibility description, for example, 'eventIn', 'exposedField'.
- **Sync** is the synchronization status 'on' or 'off'. See also `vrnode/sync` on page 9-14.

See Also

`vrnode/get`, `vrnode/set`

vrnode/get

Purpose Read property value of vrnode object or VRML field

Syntax

```
get(vrnode_object)
x = get(vrnode_object, 'property_name')
x = get(vrnode_object, 'field_name')
```

Arguments

<i>vrnode_object</i>	Name of a vrnode object representing the node to be queried.
<i>property_name</i>	Name of the property to be read.
<i>field_name</i>	Name of the VRML field to be read.

Description If a vrnode object is the only argument, a list of all properties and VRML fields is displayed in the MATLAB Command Window.

If a property name is given as an argument, the method returns the value of that property. All these properties are read-only.

Vrnode objects are case sensitive and have the following properties.

Property	Description
Fields	Valid field names for this type of VRML node.
Name	Node name.
Type	Node type string (for example, 'Transform', 'Shape').
World	A vrworld object representing the node's parent world.

Node fields queried by get can be eventOut or exposedField. EventIn does not have values; therefore, you cannot read these values.

If a VRML field name is given as an argument, the method returns the value of that VRML field. The type of the result depends on the type of the field.

Field Name	Field and Result Type
SFInt32, SFFloat, SFTime	Double
SFVec2f	Two doubles
SFVec3f, SFColor	Three doubles
SFRotation	Four doubles
MFFloat	n doubles
MFVec2f	n-by-2 doubles
MFVec3f, MFColor	n-by-3 doubles
MFRotation	n-by-4 doubles
SFString	String
MFString	Cell array of strings
SFBool	'on' or 'off'

Values that consist of more than one double are returned in the form of a row vector.

VRML fields with types SFNode and MFNode are not accessible in this version of the Virtual Reality Toolbox. These fields always return an empty vector.

See Also

`vrnode/set`, `vrnode/getfield`, `vrnode/setfield`, `vrnode`

vrnode/getfield

Purpose Get a field value of a vrnode object

Syntax

```
getfield(vrnode_object)
x = getfield(vrnode_object)
x = getfield(vrnode_object, 'fieldname')
```

Arguments

<i>vrnode_object</i>	Name of a vrnode object representing the node to be queried.
<i>fieldname</i>	Name of the vrnode object field whose values you want to query.

Description `getfield(vrnode_object)` displays all the field names and their current values for the vrnode object.

`x = getfield(vrnode_object)` returns a structure array whose fields are the fields of the vrnode object. Each field contains the values of a vrnode object field.

`x = getfield(vrnode_object, 'fieldname')` returns the values in the vrnode object's specified field. If *fieldname* is a 1-by-N or N-by-1 cell array of strings containing field names, `getfield` returns a 1-by-N or N-by-1 cell array of field values.

See Also `vrnode/get`, `vrnode/set`, `vrnode/setfield`, `vrnode`

Purpose Return 1 if the vrnode object is valid, and 0 if it is not

Syntax `x = isvalid(vrnode_object_vector)`

Arguments *vrnode_object_vector* Name of an array of vrnode objects to be queried.

Description This function detects whether the vrnode objects contained in the vector are valid, and returns a vector containing a 1 for the valid vrnode objects and a 0 for the invalid ones.

The vrnode object is considered valid if the following three conditions are met:

- The parent world of the node still exists.
- The parent world of the node is open.
- The node still exists in the parent world.

See Also `vrworld/isvalid`, `vrfigure/isvalid`

vrnode/set

Purpose Change a property or a VRML field of a virtual world node

Syntax

```
x = set(vrnode_object, 'property_name', 'property_value')
x = set(vrnode_object, 'fieldname', 'fieldvalue')
```

Arguments

<i>vrnode_object</i>	Name of a vrnode object representing a node in the virtual world.
<i>property_name</i>	Name of a property.
<i>fieldname</i>	Name of a field.

Description `x = set(vrnode_object, 'property_name', 'property_value')` changes the specified property of the vrnode object to the specified value.

`x = set(vrnode_object, 'fieldname', 'fieldvalue')` changes the specified field of the vrnode object to the specified value. Note that you can specify multiple field names and field values in one line of code by grouping them in pairs. For example, `x = set(vrnode_object, 'fieldname', 'fieldvalue', 'fieldname', 'fieldvalue', ...)`

If a vrnode object is the only argument, a list of all settable VRML fields along with allowed value types is displayed in the MATLAB Command Window.

Because all node properties are read-only, you can write values only to VRML fields. VRML field names are case sensitive, while property names are not.

See Also `vrnode/get`, `vrnode/getfield`, `vrnode/setfield`, `vrnode`

Purpose Change a field value of a vrnode object

Syntax `x = setfield(vrnode_object, 'fieldname', 'fieldvalue')`

Arguments

<code>vrnode_object</code>	Name of a vrnode object representing the node to be queried.
<code>fieldname</code>	Name of the vrnode object field whose values you want to query.

Description `x = setfield(vrnode_object, 'fieldname', 'fieldvalue')` changes the specified field of the vrnode object to the specified value. You can specify multiple field names and field values in one line of code by grouping them in pairs. For example, `x = setfield(vrnode_object, 'fieldname', 'fieldvalue', 'fieldname', 'fieldvalue', ...)`. Note that VRML field names are case sensitive, while property names are not.

See Also `vrnode/get`, `vrnode/set`, `vrnode/getfield`, `vrnode`

vrnode/sync

Purpose Enable or disable synchronization of VRML fields with clients

Syntax `sync(vrnode_object, 'field_name', 'action')`

Arguments

<i>vrworld_object</i>	Name of a vrworld object representing a virtual world.
<i>field_name</i>	Name of the VRML field to be changed.
<i>action</i>	The action parameter determines what should be done: <ul style="list-style-type: none">• 'on' enables synchronization of this field.• 'off' disables synchronization of this field.

Description The sync method controls whether the value of a VRML field is synchronized.

When the field is marked 'on', the field value is updated every time it is changed on the client computer. If the field is marked 'off', the host computer ignores the changes on the client computer.

Synchronized fields add more traffic to the network line because the value of the field must be resent by the client any time it is changed. Because of this, you should mark for synchronization only the fields you need to scan for user changes. By default, fields are not synchronized.

Synchronization is meaningful only for readable fields. For example, readable fields are eventOuts and exposedFields. You cannot enable synchronization for the field eventIns or the nonexposed fields.

See Also `vrnode/get`, `vrnode`

vrfigure Object Reference

While the Simulink interface is the preferred method for using the Virtual Reality Toolbox, you can access virtual worlds through the MATLAB interface. To use this interface, you create objects in the MATLAB workspace and associate those objects with your virtual worlds.

MATLAB functions and the Simulink interface share the same Virtual Reality Toolbox objects. It is possible to access these objects from both the MATLAB and Simulink interfaces simultaneously.

This chapter includes the following sections:

- | | |
|--------------------------------------|---|
| vrfigure Object Properties (p. 10-2) | vrfigure object properties allow you to control the behavior of objects |
| vrfigure Object Methods (p. 10-5) | vrfigure object methods allow you to access and manipulate objects |

vrfigure Object Properties

A vrfigure object is a handle to the Virtual Reality Toolbox viewer window that allows you to get and set the viewer properties. A vrfigure object is a child object of a vrworld object.

The vrfigure object properties allow you to control the behavior of objects. The following table lists the properties for vrfigure objects.

Property	Value	Description
Antialiasing	'off' 'on' Default: off	Determine whether antialiasing is used when rendering scene. Antialiasing smoothes textures by interpolating values between texture points. Read/write.
CameraBound	'off' 'on' Default: on	Control whether or not the camera moves with the current viewpoint. Read/write.
CameraDirection	Vector of three doubles	Specify the camera direction relative to the direction of the current viewpoint. Read/write.
CameraDirectionAbs	Vector of three doubles	Specify the camera direction in world coordinates. Read-only.
CameraPosition	Vector of three doubles	Specify the camera position relative to the position of the current viewpoint. Read/write.
CameraPositionAbs	Vector of three doubles	Specify the camera position in world coordinates. Read-only.
CameraUpVector	Vector of three doubles	Specify the camera up vector relative to the up vector of the current viewpoint. Read/write.

Property	Value	Description (Continued)
CameraUpVectorAbs	Vector of three doubles	Specify the camera up vector in world coordinates. Read-only.
DeleteFcn	String	Callback invoked when closing the vrfigure object. Read/write.
Headlight	'off' 'on' Default: on	Turn the headlight on or off. Read/write.
Lighting	'off' 'on' Default: on	Turn the lighting of the scene on or off. Without lighting, the scene loses its three-dimensional quality. Read/write.
Name	String	Name of this vrfigure object. Read/write.
PanelMode	'opaque' 'translucent' 'off' 'halfbar' 'bar' Default: 'halfbar'	Control the appearance of the control panel in the Virtual Reality Toolbox viewer window. Read/write.
Position	Vector of four doubles	Screen coordinates of this vrfigure object. Read/write.
Textures	'off' 'on' Default: on	Turn texture rendering on or off. Read/write.

Property	Value	Description (Continued)
Transparency	'off' 'on' Default: on	Specify whether or not transparency information is taken into account when rendering. Read/write.
Viewpoint	String If active viewpoint does not have a name, value is empty.	Vrfigure object's active viewpoint. Read/write.
Wireframe	'off' 'on' Default: off	Specify whether objects are drawn as solids or wireframes. Read/write.
World	Vrworld object	World this vrfigure object is displaying. Read-only
ZoomFactor	Double	Camera zoom factor. Read/write.

vrfigure Object Methods

A vrfigure object is a handle to the Virtual Reality Toolbox viewer window that allows you to get and set the viewer properties. A vrfigure object is a child object of a vrworld object.

The vrfigure object methods allow you to access and manipulate objects. The following is a list of the vrfigure object methods. A reference page for each vrfigure object method follows the table.

Method	Description
vrfigure	Create a new virtual reality figure.
vrfigure/capture	Create a RGB image from a virtual reality figure.
vrfigure/close	Close a virtual reality figure.
vrfigure/get	Read property value of vrfigure object.
vrfigure/isvalid	Return 1 if the vrfigure object is valid, and 0 if it is not.
vrfigure/set	Change property value of vrfigure object.

vrfigure

Purpose Create a new virtual reality figure

Syntax

```
f = vrfigure(world)
f = vrfigure(world,position)
f = vrfigure
f = vrfigure([])
```

Description `f = vrfigure(world)` creates a new virtual reality figure showing the specified world and returns an appropriate `vrfigure` object. The input argument `world` must be a `vrworld` object.

`f = vrfigure(world,position)` creates a new virtual reality figure at the specified position.

`f = vrfigure` returns an empty `vrfigure` object that does not have a visual representation.

`f = vrfigure([])` returns an empty vector of type `vrfigure`.

Example Create a `vrworld` object. At the MATLAB command prompt, type

```
myworld = vrworld('vrmount.wrl')
```

The `vrworld` object `myworld` is associated with the virtual world `vrmount.wrl`.

Next, open the virtual world using the `vrworld` object. You must open the virtual world before you can view it. At the MATLAB command prompt, type

```
open(myworld)
```

You can now view the virtual world in the Virtual Reality Toolbox viewer by typing

```
f = vrfigure(myworld)
```

Your viewer opens and displays the virtual scene.

See also `vrworld`, `vrworld/open`

Purpose Create an RGB image from a virtual reality figure

Syntax `image_capture = capture(vrfigure_object)`

Description `image_capture = capture(vrfigure_object)` captures a virtual reality figure into a TruColor RGB image that can be displayed by the `image` command.

Example Create a `vrworld` object. At the MATLAB command prompt, type

```
myworld = vrworld('vrmount.wr1')
```

The `vrworld` object `myworld` is associated with the virtual world `vrmount.wr1`.

Next, open the virtual world using the `vrworld` object. You must open the virtual world before you can view it. At the MATLAB command prompt, type

```
open(myworld)
```

You can now view the virtual world in the Virtual Reality Toolbox viewer by typing

```
f = vrfigure(myworld)
```

Your viewer opens and displays the virtual scene. Next, create an RGB image by typing

```
image_capture = capture(f);
```

Lastly, view the image

```
image(image_capture)
```

The scene from the viewer window is displayed in a MATLAB figure window.

See also `vrfigure`

vrfigure/close

Purpose Close a virtual reality figure

Syntax `close(vrfigure_object)`

Arguments `vrfigure_object` Name of a figure object.

Description `close(vrfigure_object)` closes the virtual reality figure referenced by `vrfigure_object`. If `vrfigure_object` is a vector of `vrfigure` handles, then multiple figures are closed.

Examples

```
myworld = vrworld('vrpend.wr1')
open(myworld)
f = vrfigure(myworld)
close(f)
```

See Also `vrworld`, `vrworld/open`, `vrfigure`

Purpose Read a property value of a vrfigure object

Syntax

```
get(vrfigure_object)
x = get(vrfigure_object, 'property_name')
```

Arguments

<i>vrfigure_object</i>	Name of a vrfigure object.
<i>property_name</i>	Name of the property.

Description The `get(vrfigure_object)` method returns the object properties of the vrfigure object. This method is useful when you want to determine the current values of these properties.

The following are properties of vrfigure objects.

Property	Value	Description
Antialiasing	'off' 'on' Default: off	Determine whether antialiasing is used when rendering scene. Antialiasing smoothes textures by interpolating values between texture points. Read/write.
CameraBound	'off' 'on' Default: on	Control whether or not the camera moves with the current viewpoint. Read/write.
CameraDirection	Vector of three doubles	Specify the camera direction relative to the direction of the current viewpoint. Read/write.
CameraDirectionAbs	Vector of three doubles	Specify the camera direction in world coordinates. Read-only.

vrfigure/get

Property	Value	Description (Continued)
CameraPosition	Vector of three doubles	Specify the camera position relative to the position of the current viewpoint. Read/write.
CameraPositionAbs	Vector of three doubles	Specify the camera position in world coordinates. Read-only.
CameraUpVector	Vector of three doubles	Specify the camera up vector relative to the up vector of the current viewpoint. Read/write.
CameraUpVectorAbs	Vector of three doubles	Specify the camera up vector in world coordinates. Read-only.
DeleteFcn	String	Callback invoked when closing the vrfigure object. Read/write.
Headlight	'off' 'on' Default: on	Turn the headlight on or off. Read/write.
Lighting	'off' 'on' Default: on	Turn the lighting of the scene on or off. Without lighting, the scene loses its three-dimensional quality. Read/write.
Name	String	Name of this vrfigure object. Read/write.

Property	Value	Description (Continued)
PanelMode	'opaque' 'translucent' 'off' 'halfbar' 'bar' Default: 'halfbar'	Control the appearance of the control panel in the Virtual Reality Toolbox viewer window. Read/write.
Position	Vector of four doubles	Screen coordinates of this vrfigure object. Read/write.
Textures	'off' 'on' Default: on	Turn texture rendering on or off. Read/write.
Transparency	'off' 'on' Default: on	Specify whether or not transparency information is taken into account when rendering. Read/write.
Viewpoint	String If active viewpoint does not have a name, value is empty.	Vrfigure object's active viewpoint. Read/write.
Wireframe	'off' 'on' Default: off	Specify whether objects are drawn as solids or wireframes. Read/write.
World	Vrworld object	World this vrfigure object is displaying. Read only
ZoomFactor	Double	Camera zoom factor. Read/write.

vrfigure/get

Example

Create a vrworld object:

```
myworld = vrworld('vrmount.wrl');
```

The vrworld object `myworld` is associated with the virtual world `vrmount.wrl`.
Open the world:

```
open(myworld)
```

Create a vrfigure object:

```
f = vrfigure(myworld);
```

You can now get the object properties of the vrfigure object `f`:

```
get(f)
```

This returns the following object properties:

```
Antialiasing = 'off'  
CameraBound = 'on'  
CameraDirection = [0 0 -1]  
CameraDirectionAbs = [0 -0.198669 -0.980067]  
CameraPosition = [0 0 0]  
CameraPositionAbs = [20 4 50]  
CameraUpVector = [0 1 0]  
CameraUpVectorAbs = [0 0.980067 -0.198669]  
Headlight = 'on'  
Lighting = 'on'  
Name = 'VR Car in the Mountains'  
PanelMode = 'opaque'  
Textures = 'on'  
Transparency = 'on'  
Viewpoint = 'View1'  
Wireframe = 'off'  
ZoomFactor = 1
```

See Also

[vrfigure/set](#), [vrfigure](#)

Purpose	Return 1 if the vrfigure object is valid, and 0 if it is not
Syntax	<code>x = isvalid(vrfigure_object_vector)</code>
Arguments	<i>vrfigure_object_vector</i> Name of an array of vrfigure objects.
Description	This function detects whether the vrfigure handles are valid and returns an array that contains a 1 where the vrfigure handles are valid and returns a 0 where they are not.
See Also	<code>vrworld/isvalid</code> , <code>vrnode/isvalid</code>

vrfigure/set

Purpose Change a property value of vrfigure object

Syntax `set(vrfigure_object, 'property_name', property_value)`

Arguments

<code>vrfigure_object</code>	Name of a vrfigure object.
<code>property_name</code>	Name of the property you want to set.
<code>property_value</code>	New value of the property.

Description The `set(vrfigure_object)` method allows you to set the property value of a vrfigure object. This method is useful when you want to change the value of a property.

Example Create a vrworld object.

```
myworld = vrworld('vrmount.wrl');
```

The vrworld object `myworld` is associated with the virtual world `vrmount.wrl`.
Open the world:

```
open(myworld)
```

Create a vrfigure object:

```
f = vrfigure(myworld);
```

The VR Car in the Mountains virtual world opens in the Virtual Reality Toolbox viewer. You can now set the object properties of the vrfigure object `f`:

```
set(f, 'Name', 'Car on a Mountain Road')
```

You can see that the name of the virtual world has changed in the viewer.

See Also `vrfigure/get`, `vrfigure`

A

- adding
 - Virtual Reality Toolbox blocks 3-2
- associating virtual worlds with Simulink blocks
 - 3-10

B

- blaxxun Contact
 - creating virtual worlds 5-8
 - installing 2-16
 - known issue 2-18
 - VRML viewer 5-33
- bouncing ball
 - Simulink example 1-15

C

- capture
 - Virtual Reality Toolbox `vrfigure` method 10-7
- car
 - MATLAB interface example 1-21
- changing default network security setting 2-18
 - See also* blaxxun Contact
- changing virtual world associated with Simulink
 - block 3-10
- client computer
 - installation of VRML viewer (UNIX) 2-34
 - installation of VRML viewer (Windows) 2-33
 - system requirements 2-7
- close
 - Virtual Reality Toolbox `vrfigure` method 10-8
 - Virtual Reality Toolbox `vrworld` method 8-6
- closing virtual worlds 4-8
- components on client computer 2-33
- components on host computer 2-10

- connecting
 - Simulink model to a virtual world 5-17
- control menu 5-22
- coordinate system
 - MATLAB 1-9
 - VRML 1-9
- creating `vrworld` object 4-2

D

- default editor
 - setting 2-26
- default viewer
 - setting 2-20
- deformation of a sphere example
 - adding Virtual Reality Toolbox blocks 5-6
 - connecting Simulink to a virtual world 5-17
 - creating a box in a virtual world 5-13
 - creating a sphere in a virtual world 5-8
 - defining the problem 5-5
- delete
 - Virtual Reality Toolbox `vrnode` method 9-6
 - Virtual Reality Toolbox `vrworld` method 8-7
- deleting virtual worlds 4-8
- displaying virtual worlds 3-13

E

- edit
 - Virtual Reality Toolbox `vrworld` method 8-8
- editor
 - uninstalling 2-31
- editors 5-2
 - general 3-D 5-2
 - native VRML 5-2

examples

- bouncing ball 1-15
- car 1-21
- deformation of a sphere 5-5
- heat transfer 1-21
- inverted pendulum 1-19
- lighting 1-16
- magnetic levitation 1-16
- magnetic levitation for Real-Time Windows Target 1-17
- manipulator with SpaceMouse 1-17
- MATLAB interface 1-14
- plane taking off 1-20
- rotating membrane 1-22
- Simulink interface 1-14
- solar system 1-20
- tower crane 1-16
- using MATLAB interface 1-21

F**features**

- Virtual Reality Toolbox 1-3

fields

- Virtual Reality Toolbox `vrnode` method 9-7

file format

- VRML 1-10

functions

- MATLAB interface 7-1
- `vrclear` 7-2
- `vrgetpref` 7-5
- `vrinstall` 7-8
- `vrlib` 7-10
- `vrsetpref` 7-11
- `vrview` 7-12
- `vrwho` 7-13
- `vrwhos` 7-14

G**get**

- Virtual Reality Toolbox `vrfigure` method 10-9
- Virtual Reality Toolbox `vrnode` method 9-8
- Virtual Reality Toolbox `vrworld` method 8-9

getfield

- Virtual Reality Toolbox `vrnode` method 9-10

H**heat transfer**

- MATLAB example 1-21

history

- VRML 1-8

host computer

- installing Virtual Reality Toolbox 2-9
- installing VRML editor (Windows) 2-25
- installing VRML viewer (UNIX) 2-19
- installing VRML viewer (Windows) 2-16
- required components 2-10
- system requirements 2-4
- Virtual Reality Toolbox viewer 2-15
- VRML editor (UNIX) 2-26

- I**
- installation
 - blaxxun Contact 2-16
 - client computer 2-33
 - components on host computer 2-10
 - host computer 2-9
 - supported platforms 2-2
 - system requirements 2-2
 - testing 2-35
 - viewer on host computer 2-15
 - Virtual Reality Toolbox 2-9
 - VRML editor (UNIX) 2-26
 - VRML editor (Windows) 2-25
 - VRML viewer (UNIX) 2-19
 - VRML viewer (Windows) 2-16
 - interacting with a virtual world 4-5
 - inverted pendulum
 - Simulink example 1-19
 - isvalid
 - Virtual Reality Toolbox `vrfigure` method 10-13
 - Virtual Reality Toolbox `vrnode` method 9-11
 - Virtual Reality Toolbox `vrworld` method 8-11
- L**
- license
 - getting or updating 2-9
 - lighting
 - Simulink example 1-16
- M**
- magnetic levitation
 - Simulink example 1-16
 - Simulink example for Real-Time Windows Target 1-17
 - manipulator with Space Mouse
 - Simulink example 1-17
 - MATLAB
 - coordinate system 1-9
 - interface examples 1-21
 - MATLAB interface
 - creating a `vrworld` object 4-2
 - interacting with a virtual world 4-5
 - opening a virtual world 4-4
 - table of general functions 7-1
 - `vrfigure` object methods 10-5
 - `vrnode` object methods 9-3
 - `vrworld` object methods 8-4
 - methods
 - capture 10-7
 - close
 - Virtual Reality Toolbox `vrfigure` method 10-8
 - Virtual Reality Toolbox `vrworld` method 8-6
 - delete
 - Virtual Reality Toolbox `vrnode` method 9-6
 - Virtual Reality Toolbox `vrworld` method 8-7
 - edit 8-8
 - fields 9-7
 - get
 - Virtual Reality Toolbox `vrfigure` method 10-9
 - Virtual Reality Toolbox `vrnode` method 9-8
 - Virtual Reality Toolbox `vrworld` method 8-9
 - `getfield` 9-10

methods (continued)

invalid

Virtual Reality Toolbox `vrfigure` method
10-13Virtual Reality Toolbox `vrnode` method
9-11Virtual Reality Toolbox `vrworld` method
8-11

nodes 8-12

open 8-13

reload 8-14

save 8-15

set

Virtual Reality Toolbox `vrfigure` method
10-14Virtual Reality Toolbox `vrnode` method
9-12Virtual Reality Toolbox `vrworld` method
8-16

setfield 9-13

sync 9-14

view 8-17

`vrfigure` 10-6`vrnode` 9-4`vrworld` 8-5**N**

native VRML 5-2

navigation

about a virtual scene 5-25

example of navigation 5-30

keyboard 5-31

using the control menu 5-26

using the control panel 5-25

using the mouse 5-27

navigation speed

changing 5-22

nodes

Virtual Reality Toolbox `vrworld` method 8-12

non-preferred-term

See preferred-term**O**

objects

`vrfigure` methods 10-5`vrnode` methods 9-3`vrworld` methods 8-4*See also* methods

open

Virtual Reality Toolbox `vrworld` method 8-13

opening a viewer window 3-15

opening virtual worlds 4-4

overview

associating virtual worlds with Simulink 3-2

Simulink interface 3-2

viewing a virtual world 5-21

virtual worlds 5-2

`vrfigure` objects 10-1

VRML 1-8

VRML editing tools 5-2

`vrnode` objects 9-1`vrworld` objects 8-1**P**

plane taking off

Simulink example 1-20

platforms

supported 2-2

R

- reload
 - Virtual Reality Toolbox `vrworld` method 8-14
- rendering of a virtual scene 5-23
- rotating membrane
 - Simulink example 1-19
 - Virtual Reality Toolbox example 1-22
- running Simulink example 2-35

S

- save
 - Virtual Reality Toolbox `vrworld` method 8-15
- server
 - Virtual Reality Toolbox 1-23
- set
 - default editor 2-26
 - default viewer 2-20
- set
 - Virtual Reality Toolbox `vrfigure` method 10-14
 - Virtual Reality Toolbox `vrnode` method 9-12
 - Virtual Reality Toolbox `vrworld` method 8-16
- setfield
 - Virtual Reality Toolbox `vrnode` method 9-13
- simulation
 - displaying virtual worlds 3-13
 - starting 3-13
- Simulink
 - associating with virtual worlds 3-2
 - interface examples 1-14
 - See also* examples
- Simulink blocks
 - adding Virtual Reality Toolbox blocks 3-2
 - changing virtual world association 3-10
 - VR Placeholder 6-6
 - VR Signal Expander 6-7

- VR Sink 6-2
 - VR Source 6-4
 - Simulink interface
 - overview 3-2
 - Simulink interface examples
 - bouncing ball 1-15
 - deformation of a sphere 5-6
 - inverted pendulum 1-19
 - lighting 1-16
 - magnetic levitation 1-16
 - magnetic levitation with Real-Time Windows Target 1-17
 - manipulator with SpaceMouse 1-17
 - plane taking off 1-20
 - rotating membrane 1-19
 - running and viewing 2-35
 - solar system 1-20
 - tower crane 1-16
 - solar system
 - Simulink example 1-20
 - SpaceMouse
 - Simulink examples 1-17
 - supported platforms 2-2
 - sync
 - Virtual Reality Toolbox `vrnode` method 9-14
 - system requirements
 - client computer 2-7
 - host computer 2-4
-
- T**
 - testing
 - installation 2-35
 - MATLAB example 2-40
 - Simulink example 2-35
 - tower crane
 - Simulink example 1-16

U

uninstalling

editor 2-31

Virtual Reality Toolbox 2-31

V-Realm Builder 2-31

VRML viewer (Windows) 2-31

V

view

Virtual Reality Toolbox `vrworld` method 8-17

view a virtual world

using a Web browser on the client computer
3-19using a Web browser on the host computer
3-16

viewer

installation on client computer 2-33

installation on host computer 2-15

opening 3-15

See also VRML viewer

viewpoint control 5-22

Virtual Reality Toolbox

description 1-2

Virtual Reality Toolbox blocks

VR Placeholder 6-6

VR Signal Expander 6-7

VR Sink 6-2

VR Source 6-4

Virtual Reality Toolbox example

car 1-21

heat transfer 1-21

rotating membrane 1-22

running and viewing 2-40

Virtual Reality Toolbox viewer 5-21

changing navigation speed 5-22

control menu 5-22

navigation 5-25

rendering 5-23

viewpoint control 5-22

virtual worlds

associating with Simulink 3-2

closing 4-8

deleting 4-8

displaying 3-13

interacting with 4-5

opening 4-4

overview 5-2

VR Placeholder

Simulink block 6-6

VR Signal Expander

Simulink block 6-7

VR Sink

Simulink block 6-2

VR Source

Simulink block 6-4

`vrclear`

Virtual Reality Toolbox function 7-2

V-Realm Builder

installing 2-25

uninstalling 2-31

VRML editor 5-4

`vrfigure`

Virtual Reality Toolbox method 10-6

`vrfigure` object

Virtual Reality Toolbox methods 10-5

`vrgetpref`

Virtual Reality Toolbox function 7-5

`vrinstall`

Virtual Reality Toolbox function 7-8

`vrlib`

Virtual Reality Toolbox function 7-10

VRML

- coordinate system 1-9
- file format 1-10
- history 1-8
- overview 1-8

VRML editor 5-2

- general 3-D 5-2
- installing on host computer (Windows) 2-25
- on UNIX platforms 2-26
- V-Realm Builder 5-4

VRML object reference

- vrfigure objects 10-1
- vrnode objects 9-1
- vrworld objects 8-1

VRML objects

- overview of vrfigure objects 10-1
- overview of vrnode objects 9-1
- overview of vrworld objects 8-1

VRML viewer

- blaxxun Contact 5-33
- changing navigation speed 5-22
- control menu 5-22
- installing on client computer (UNIX) 2-34
- installing on client computer (Windows) 2-33
- installing on host computer (UNIX) 2-19
- installing on host computer (Windows) 2-16
- known issue (blaxxun Contact) 2-18
- navigation 5-25
- rendering 5-23
- uninstalling 2-31
- viewpoint control 5-22
- Virtual Reality Toolbox 5-21

vrnode

- Virtual Reality Toolbox method 9-4

vrnode object

- Virtual Reality Toolbox methods 9-3

vrsetpref**Virtual Reality Toolbox function 7-11****vrview****Virtual Reality Toolbox function 7-12****vrwho****Virtual Reality Toolbox function 7-13****vrwhos****Virtual Reality Toolbox function 7-14****vrworld****Virtual Reality Toolbox method 8-5****vrworld object****creation 4-2****Virtual Reality Toolbox methods 8-4****W****Web browser**

- view a virtual world on a client computer 3-19
- view a virtual world on the host computer 3-16

