



V-Realm™ Builder

User's Guide and Reference

Ligos Corporation
 6001 Chatham Center Drive, Suite 300
 Savannah, GA 31405
 912/236-4374
<http://www.ligos.com>

Support: 912/232-7969
support@ids-net.com

Copyright © 1996-1997 Ligos Corporation

All rights Reserved. *V-Realm* is a trademark of Ligos Corporation. All other trademarks and trade names or registered trademarks and registered trade names are the properties of their respective holders. No part of this publication may be reproduced, transmitted, stored in a retrieval system, or translated into any language in any form by any means, without the written permission of Ligos Corporation. Information in this document is subject to change without notice and does not represent a commitment on the part of Ligos Corporation. The software described in this document is furnished under a license agreement. The software may be used or copied only in accordance with the terms of the agreement. Ligos Corporation cannot attest to the accuracy of this information. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark. Portions of this software are based in part on the VRMLScript Interpreter written by Silicon Graphics, Inc., Mountain View, California, USA.





Table of Contents

[Documentation Release Note](#)



[List of Exercises](#)

[\[0\] Introduction to the V-Realm Builder](#)




[\[1\] V-Realm Builder Setup and Installation](#)

-  Getting Started
-  System Requirements
-  Installing the V-Realm Builder
-  Starting V-Realm Builder

[\[2\] V-Realm Builder Interface](#)




-  V-Realm Builder Features
-  Quick Start Tutorial

[V-Realm Windows and Menus](#)




-  Main Menu
-  File Option
-  Edit Option

[VRML Coordinate System](#)





[\[3\] The Node Tree](#)

-  The Hierarchy
-  Nodes and Fields
-  Basic Grouping











[\[4\] Basic Worldbuilding](#)

-  Geometry and Appearance
-  Primitive Geometry Nodes
-  Appearance Nodes




[\[5\] Intermediate Worldbuilding](#)

-  Instancing with DEF and USE
-  Scene Binding Nodes
-  Light Sources
-  Other Intermediate Nodes









[\[6\] More On Grouping](#)

-  View Option
-  Nodes Option
-  Libraries Option
-  Add To Option
-  Edit Option
-  Import From Option
-  Manipulators Option
-  Mode Option
-  Window Option
-  Help Option

V-Realm Pop-Up Menus

-  Tree View Pop-Up Menu
-  Main View Window Pop-Up Menu
-  Orthographic View Pop-Up Menu












V-Realm Builder Toolbars

-  The Standard Toolbar
-  Mode Toolbar
-  Common Toolbar
-  Group Toolbar
-  Geometry Toolbar
-  Sensor Toolbar
-  Main Window Toolbar
-  Orthographic Windows Toolbar

Use of Mouse




Dialog Boxes

Editing VRML 2.0 Fields



-  Integer Editing
-  Floating Point Editing
-  Boolean Editing
-  String Editing
-  URL Editing
-  Vector (2) Editing
-  Vector (3) Editing
-  Rotation Editing
-  Color Editing
-  Time Editing
-  Multiple Field Editing

Color Mode Dialog






Toolbar Configuration Dialog

-  More Grouping Nodes
-  More Sophisticated Grouping Nodes
-  Web based Nodes



[7] Libraries

-  Drag and Drop
-  V-Realm Builder Library Icons Listed









[8] Specialized Editors

-  Complex Geometry Nodes
-  V-Realm Editors
-  Elevation Grid Editor
-  Extrusion Editor
-  Face Set Editor

[9] Moving Worlds

-  Interpolators
-  KeyFrame Animation






[10] Advanced Worldbuilding

-  Sensor Nodes
-  Time Dependent Nodes
-  The Sound Node
-  Manually Connecting Routes
-  Other Advanced Nodes
-  Scripting
-  PROTO and External PROTO nodes
-  Instancing with Prototypes

Appendix A - Keyboard Shortcuts

Appendix B - Hints and Tips

Appendix C - Manipulator Reference

-  Universal Manipulator
-  Centerball Manipulator
-  PointLight Manipulator
-  SpotLight Manipulator
-  DirectionalLight Manipulator

Appendix D - Glossary

Documentation Release Note

This HTML version of the V-Realm Builder manual is distributed with Virtual Reality Toolbox for MATLAB and Simulink. It is provided as-is, without modifications of the original text that is a part of standard distribution of V-Realm Builder by Ligos Corporation. Therefore the text contains some information that is not relevant to the Virtual Reality Toolbox distribution. Most of the outdated issues are related to product installation and contents of installation media. For information on these issues, please refer to the Virtual Reality Toolbox documentation.

List of Exercises

[Exercise 1: Basic Worldbuilding](#)

[Exercise 2: Intermediate Worldbuilding](#)

[Exercise 3: Libraries](#)

[Exercise 4: Elevation Grid](#)

[Exercise 5: Extrusion Editor](#)

[Exercise 6: IndexedFaceSet](#)

[Exercise 7: KeyFrame Animation Editor](#)

[Exercise 8: PROTO](#)

[0] Introduction to the V-Realm Builder

V-Realm Builder is a powerful three dimensional authoring package for the creation of 3D objects and "worlds" to be viewed with V-Realm Browser or any other VRML 2.0 compliant browser. **It is critical to remember that VRML 2.0 was developed as a "network" friendly virtual file specification.** V-Realm Builder and VRML were not intended to replace the modeling tools we have today. VRML cannot hope to cope with modeling packages that create incredibly realistic objects with 5 or 10 million polygons. These objects may be fine for stand-alone computers or applications, but to expect a file of this size to be easily transportable over the net in real time is not feasible. Because of this, V-Realm Builder using VRML was designed to give us tools to minimize the size of files and to provide a means of modeling more complex objects with primitives, without having to overload the net with large files.

For your convenience we have included a copy of the VRML 2.0 specification on the Installation CD that came with your V-Realm Builder software. It can be found in the main directory called SPEC. There should be a /SPEC directory that contains a directory called /VRML2 with HTML files that make up the VRML 2.0 Specification, and a README.TXT file. Please read the README.TXT file first.

Ligos Corporation designed V-Realm Builder with both expert and novice users in mind. Its point and click capability delivers the same intimate control of the 3D environment you get by hand-coding. Yet, in automating VRML development and delivering instant visual feed-back, it dramatically speeds 3D world construction. V-Realm Builder's GUI is tailored specifically to VRML and even an experienced VRML developer in creating 3D worlds. V-Realm Builder provides powerful editing capabilities while simplifying the 3D authoring process.

As with our earlier versions of the V-Realm Builder, this version has a powerful intuitive interface for world modeling. We have now added interactive behaviors as introduced in the VRML 2.0 Specification. If you are familiar with the earlier versions of the V-Realm Builder, you will already be familiar with most of the controls used. If you have never used any of our products, you will be impressed with the thought and simplicity we have designed into the interfaces that are employed by our Builder.

Control is now at your finger tips. Every node's fields are directly editable with immediate feedback in the scene for the modifications made to any variable or node. This control puts the V-Realm Builder in a class of its own. None of the builders available on the market today make it this simple and easy to add, edit, and modify nodes. Many of the edit functions are completed with just a click of the mouse, the use of the keyboard is limited to just the functions that require input. Now hand-coding of VRML worlds is truly a thing of the past.

This manual is designed to show you how to get started with V-Realm Builder and how to use its highly advanced features. The text enhancements listed below will assist you in comprehending the use of the V-Realm Builder application.

Text that is not enhanced in any way refers to standard information explaining the application.

Terms that are **bold** refer to variable names, titles for dialog boxes, or important terms to remember.

Terms that are in all CAPS like this refer to constant values such as TRUE and FALSE, or button titles in dialog boxes such as OK and CANCEL.

Items that are enhanced-shaded like this will refer to V-Realm Builder examples provided to demonstrate the use of the V-Realm Builder application.

Items that are enhanced-shaded like this will refer to information essential for understanding how to use the V-Realm Builder. This type of shading will be used to point out special uses of buttons or other tools. **Very important information will be shaded this way as well as bolded.**

Boxed items will refer to helpful tips that explain why and how to perform particular tasks. This section will sometimes give you shortcuts provided by the V-Realm Builder application for performing more complex tasks.

Exercises

Exercises provided to help the user learn VRML and the V-Realm Builder will be separated with a bold line like this sentence is.

[1] V-Realm Builder Setup and Installation

Getting Started

Welcome. You are about to embark upon an exciting and fascinating journey into three dimensional world building. To begin, let's review what you already have. You should have received the following with this product:

- User's Guide
- CD-ROM
- Registration Card

Please read the Ligos Corporation License Agreement that is printed on the CD-ROM tray card insert. It contains important information that you should know before using the product.

Don't forget to register your software.

Support provided only for users who register their software.

System Requirements

The minimum software and hardware requirements your computer system needs in order to run V-Realm Builder are as follows.

- A personal computer with a 486-66 (minimum) processor. A system based on a Pentium 120 MHz processor or greater is recommended.
- Either Microsoft Windows NT 3.51/4.0 or Microsoft Windows 95.
- A system with at least 16 MB of RAM (32 MB is recommended).
- A mouse pointing device is required.
- A sound card is recommended.
- A hard disk drive with approximately 40MB free disk space.

- Minimum video requirements are VGA video adapter with 1 MB RAM, capable of displaying 256 colors at 640x480 resolution. However, a VGA adapter with 2 MB RAM, capable of running at 800x600 high color resolution is recommended. Performance can be greatly improved with any of the OpenGL cards available on the market.

Installing the V-Realm Builder

Builder is easy to install, and setup. To begin the installation process, put your CD-ROM in the drive.

If you are using Windows 95 or Windows NT 4.0, and the install does not launch automatically, choose the **Run** option from the **Start** menu.

If you are using Windows NT 3.51, switch to **Program Manager** and choose the **Run** option from the **File** menu.

On the screen, you will see the **Run** dialog box. In the edit box provided type:

x:setup.exe

where **x:** is the drive with the install CD-ROM. Choose **OK** or press **Enter**, and the automatic installation procedure will begin.

You will be prompted to enter a location on your hard disk in which to install the program files. You may select the default location or a new location of your choice. **Default location is recommended.**

Starting V-Realm Builder

To start the application, in Windows NT 4.0 or Windows 95, go to the **Start** menu to locate the V-Realm Builder 2.0 folder and click the V-Realm Builder 2.0 Program icon. If you are using Windows NT 3.51, switch to the **Program Manager** and locate the V-Realm Builder 2.0 Icon group and double click on the V-Realm Builder Program icon.

[2] V-Realm Builder Interface

V-Realm Builder Features

V-Realm Builder possesses a number of important features that extend your ability to construct efficient 3D worlds. These include:

- **Intuitive interface:** The interface for V-Realm Builder removes the need for hand-coding VRML, without taking away the power. Its ease of use allows you to quickly master its operation and build VRML worlds.
- **Open standards:** V-Realm Builder produces files that can be read by all compliant VRML 2.0 browsers... there are no proprietary extensions that will limit your audience.
- **Extended 3D Manipulation Controls:** V-Realm Builder includes powerful 3D shape manipulation tools for the transformation of objects and lighting.
- **Customizable Libraries:** V-Realm Builder provides VRML objects, textures and materials through customizable Object, Texture, and Material Libraries. A number of example worlds are also included as examples of nearly every feature available in VRML.
- **Instant Visual Feedback with Multiple Views:** A Node Tree and multiple views of 3D objects can be simultaneously displayed, heightening developer interaction with the 3D environment under construction. V-Realm Builder displays up to four simultaneous object views that are independently adjustable for optimal viewing.
- **Texture mapping:** Objects can be textured with GIF (including transparency), JPEG, and RGB file formats. Movie Textures are supported with the MPEG1, MPEG2, and AVI file formats.
- **3D Object Importing:** V-Realm Builder lets you import any standard VRML 1.0 file as well as 3D objects created in the following file formats: Wavefront (*.obj), RAW (*.raw), 3D Studio (*.3ds), trueSpace Object (*.cob).
- **Specialized Editors:** Extrusion, Key Frame Animation, Elevation Grid, and Polygon Modeling Editors.
- **VRML 2.0 Compliance:** V-Realm Builder reads all VRML 2.0 compliant worlds.


- **Audio and Video Support:** V-Realm Builder allows the user to incorporate many of the commonly used audio and video file formats into the worlds they build.
- **Definable Viewpoints:** The user has the ability to save as many positions in a world as desired. These positions can be used as guideposts to navigating the world or as a virtual tour.
- **Customizable Toolbar Control:** All toolbars are moveable, dockable, and displayable. User interface allows user to control all available toolbars.
- **Smart Node Tree:** When editing a VRML file by adding or inserting nodes the "smart" node tree will automatically place the node object in the most efficient spot for VRML coding. In many instances, it acts as a syntax checker to ensure that nodes are placed appropriately.
- **Search Feature:** Items in the Node Tree can be quickly located by typing the word for which to search. The cursor will traverse the Node Tree until it finds the appropriate item automatically.
- **Smart Toolbars:** Toolbar icons will "gray out" when they do not relate to the function being performed. Icons that are related to the task being performed will stay solid.
- **Streamlined File Output:** V-Realm Builder writes out compact world files because it does not write out any values that are "default". In addition, the Builder provides the option of automatically saving a file in the g-zipped compressed binary file format.

Quick Start Tutorial

As you can guess, building 3D interactive environments can be a daunting task, especially on a 2D-oriented tool like a PC. But the V-Realm Builder gives you all the tools you need to create these worlds with a graphic interface that is powerful and easy to use. Before we get into the details, let's build a simple interactive scene to get familiar with the interface. VRML can be used as a platform to create anything from a spinning logo for a web page to a full-blown 3D game. We're going to start small, however, and create an animated ball that you can "push" from a platform. This tutorial is written to be followed step by step by a novice V-Realm Builder world builder. In this tutorial remember that to click on a button or icon means to left-click unless otherwise instructed.

If some of the terminology or functions don't make sense at the moment, don't worry. Just follow each step, using the illustrations as a guide.

1. *Open up V-Realm Builder 2.0 from the Windows Start Menu.* If it isn't already, maximize the application to full screen.

2. *Start by clicking on the New Document button* . What you see is the three-part interface that makes up V-Realm Builder. On the left is a blank, white area called the **Node Tree window**. As you build your world, a tree-like representation of your world will be displayed here. On the right is a large black area referred to as a **View Pane**. This is where you'll see your actual world as it is being constructed, and where you'll do the majority of your work. Along the top and the sides are the **Toolbars** that give you access to the different modes of the Builder, and the building blocks of a VRML world known as Nodes (see Figure 1):

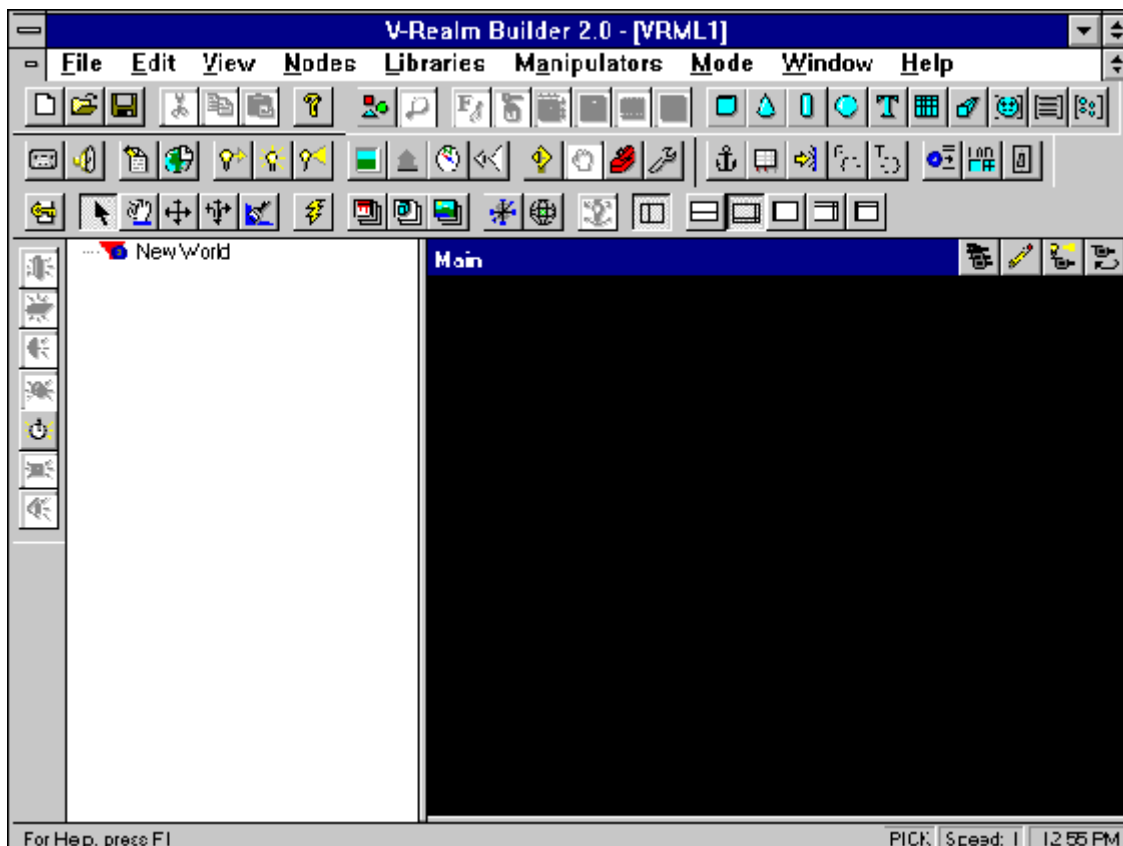




Figure 1: Screen Image

3. We'll start with a *Toolbar* function to add sky and ground to our world. Left click once on the **Insert Background** icon . Note how the *Node Tree* window is updated to include the *Background* node. (Please note that a default sky and ground color are provided by default to assist you.)

4. Return to the toolbar and click on the **Insert Sphere** icon . The *Node Tree* Window is updated with the group representing the sphere and its possible properties. The *View Pane* is updated to show a sphere and the background environment. The box and handles surrounding the sphere indicate that the sphere is selected, and can be manipulated (also known as **transformed**).

5. Click and hold your left mouse button on the sphere, and the *manipulator box* will change. At this point, you can drag the sphere to a new location and release it. For now, move the sphere to the upper left of the *View Pane* (see figure 2):

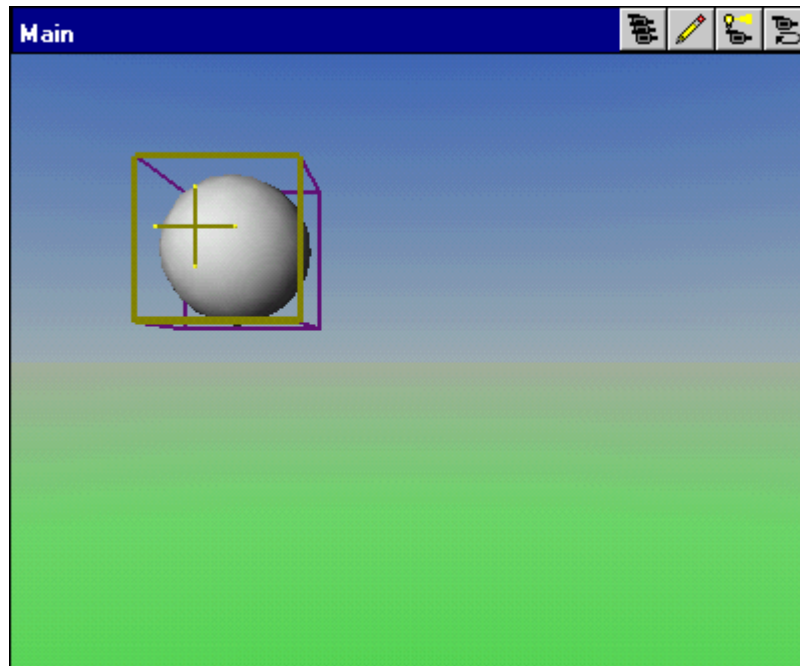









Figure 2: Main View Window Image

6. It's important in VRML to name (or "DEF", as in define) your object to create interaction later on. In the Node Tree, *click once on the top level of the sphere group, on the word Transform, to select it. Click again to change it. Change the word "Transform" to "Ball", and hit your Enter key.*


7. It's always a good idea to periodically save your file, so *click on the Save icon* , and save your file with the name "ball.wrl".


8. Now we'll create a box on which the ball can rest. *Click on the **Insert Box** icon* , and a box is created and selected. We can rescale the size of the box using the manipulator handles (the small white blocks on the corners). *Click and hold the cursor down on one of these handles, and drag to rescale the box.* Let's make it a little bit smaller, about half its current size, and release the mouse button. Now let's make it taller. *On your keyboard, hold down the Shift key, and with the mouse select a handle and drag "up" (forward on your mousepad) to rescale the height of the box.* You should now have a tall, thin pedestal.

9. During the worldbuilding process, you may find it helpful to move around your world and view it from different angles. Right now, you're in **Pick Mode** , indicated by the arrow cursor. To move your viewpoint around the world, *click on the **Navigation Mode** icon* . Now, as you *click and drag inside the View Pane*, you'll "walk" forward and backward, and be able to turn in place. If you ever want to return to your original position, *click on the **Reset Viewer** icon*  in the upper right of your pane. When you're done moving, *click on the Pick Mode button* to return to object manipulation.

10. As you did with the sphere in Step 5, *move the box under the sphere.* Your View Pane provides a perspective view of your scene, so aligning objects in 3D space can be a bit tricky. The Builder also provides panes for other views (top, side, front, etc.) to help you out, so you might want to temporarily *switch to multiple views with the **Equalize View Panes** icon* . Clicking and dragging action is the same as before, but now you can check your progress in 4 views simultaneously. Once you have the box under the sphere, you can *switch back to **View Main Pane*** .

11. *Change the name of the box's Transform group to "Pedestal" as you did in Step 6 with the sphere.*

12. Time to add some more color to our world. *Open the **Material Library***  *by clicking on its toolbar icon.* You may want to drag the Select Material dialog box to the side so that you can still see your objects in the View Pane underneath. *Scroll down to Blue Plastic on the Item List, and click on it.* The window should display a shiny blue sphere. *Click and hold on this sphere, and drag and drop the color onto the Ball in your View Pane.* The ball will turn blue. *Close the Select Material dialog.*

13. Textures are like materials, but are based on 2D images. Open the **Texture Library**  and scroll down to **Brick (Small)** (See Figure 3). As in Step 12, drag and drop the texture onto the **Pedestal**, and close the dialog:

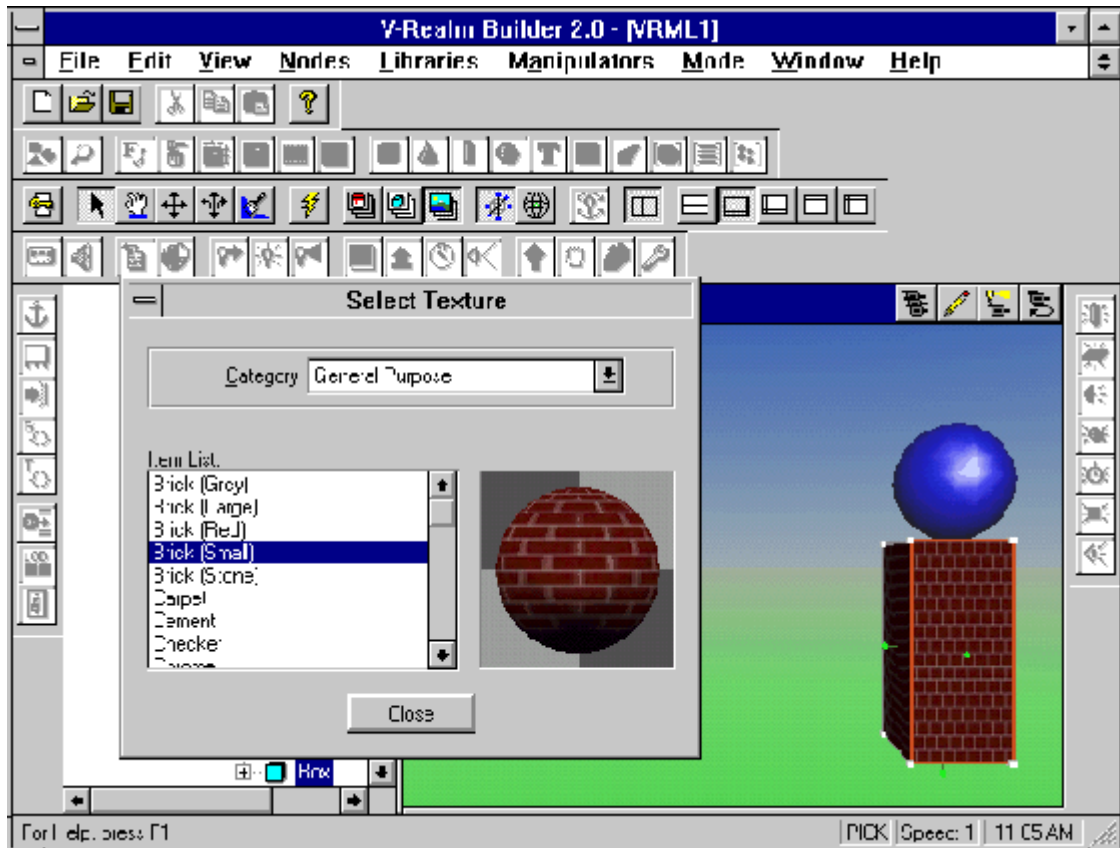




Figure 3: Select Texture for Object

14. Save your file!

15. VRML is also known as "Moving Worlds", so now we'll add some animation and interaction. Default settings must not have been changed. Click on the ball to select it, and switch to **KeyFrame Mode** . The interface will change to make room for a timeline indicator. Animations in VRML are created by specifying a series of events (known as KeyFrames) over a period of time. You can specify as few or as many KeyFrames as you wish, and the Builder (or a VRML browser) will interpolate the frames in between to create a smooth animation. For this exercise, we want to make the Ball roll off the Pedestal and bounce to the ground when we "push" on the Pedestal. First, we have to set the first KeyFrame, where the Ball currently rests, so click on the Ball and release. The timeline updates to show the properties of the Ball at the "0" seconds mark, indicated by red tickers.

16. Now we'll set the next KeyFrame at the 1-second mark. You can "jump" to next set of tickers by clicking on the Next Frame icon  at the top of the timeline... the marker jumps to the 1-second tickers. Click and drag to move the Ball to the right and down just a bit, about two units (the Ball currently has a diameter of 2 units).


17. Right now, the timeline is only set to one second, so change the timeline duration to 4 seconds by clicking on the up arrow next to the **Duration** box; (see Figure 4).

18. Click at the 2-second mark to set new KeyFrame tickers. Drag the Ball down and over to make it appear as if it has landed on the ground. If you're feeling really adventurous, you can rescale the height of the Ball to make it look like it "squished" on impact!



19. Click a new set of KeyFrames at about 2.6 seconds, and move the Ball right and up to the top of its next bounce point. "Unsquish" it if necessary.

20. Click a set of KeyFrames at 3.2 seconds, and move the Ball to the right and down to the ground again.

21. Click a set of KeyFrames at 4.0 seconds, and move the Ball to its final stop at the far right side of the View Pane.

22. That's it! You've created a VRML animation! To see how it looks, click on the Play icon . Don't worry if it doesn't look perfect... it may take a little practice, but you'll soon get the hang of creating 3D animation.

23. One of the other key aspects of VRML is interaction with the visitor to your world. One way to create interaction is by assigning a "trigger" that starts an event such as animation. For this world, we want to set it up so that when the viewer clicks/nudges the Pedestal, it disturbs the Ball causing it to roll off. To do this,

click on the **Insert Animation Trigger** icon  to assign a sensor to the event. By default, the radio button defaults to Touch Sensor, which is exactly what we want (see Figure 4). Assign the **Touch Sensor** to the Pedestal. Select OK. Return your cursor to the beginning of the animation by clicking on the First Frame icon .

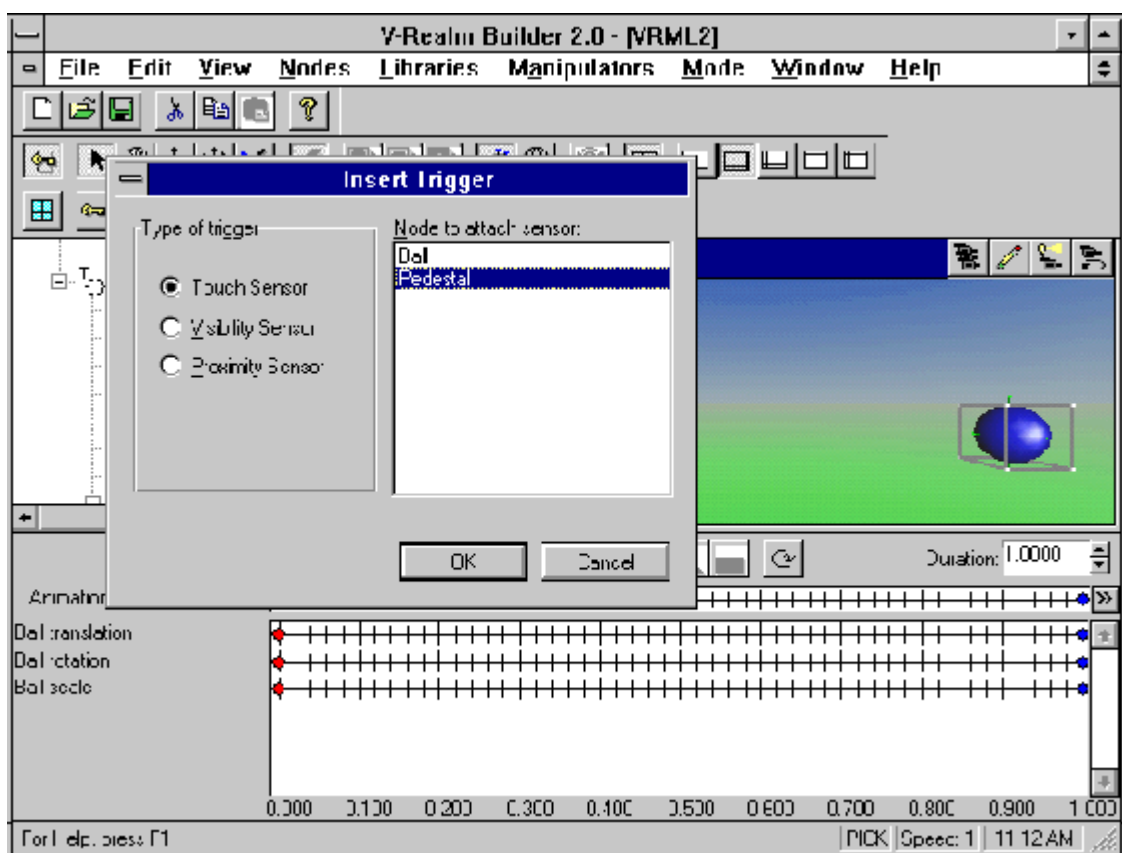



Figure 4: Attach Pedestal Trigger

24. Save your file. Leave KeyFrame Mode by clicking on the KeyFrame Mode toolbar button. You are returned to Build Mode.

25. Before we test our world, you may want to switch to Navigation Mode and "step back" from the scene by dragging down with the cursor. When you're positioned, switch back to your Pick tool.

26. You can test your world without ever leaving the Builder by switching to **Test Mode** . This simulates a VRML browser environment in which you can see how your world actually runs. Now click on the Pedestal, and the Ball will roll off, fall to the ground, bounce and roll to a stop. Leave Test Mode by clicking on its icon again.

Congratulations! You've created your first interactive VRML 2.0 world in the V-Realm Builder! And this is only the beginning. As you continue your work with the lessons in this manual, you'll find that there are many ways to animate objects, create more complex scenes, add more interaction, and incorporate other media

into your worlds.

The Quick Start Tutorial and all Exercises included in this manual have been implemented and are available for you to check your work against. Please refer to the sub-directory of your V-Realm Builder application called, "Worlds\Tutorials\" for the worlds.

V-Realm Windows and Menus

It is important to keep in mind while reading this section of the manual that we will be using terms that will describe actions on nodes in the VRML 2.0 Specification. We will make comments like, "This will add a IndexedFaceSet node to the node tree.", this will not seem to have sufficient description for what is being done. What we are attempting to accomplish here is a straightforward description of what the different items do. If you need further information, please refer to the appropriate section that follows for complete details.

Main Menu

<u>F</u> ile	<u>E</u> dit	<u>V</u> iew	<u>N</u> odes	<u>L</u> ibraries	<u>M</u> anipulators	<u>M</u> ode	<u>W</u> indow	<u>H</u> elp
--------------	--------------	--------------	---------------	-------------------	----------------------	--------------	----------------	--------------

Figure 5: Main Menu

The Main Menu gives the user access to all available commands and options. We will now discuss all the commands and options available through this menu. For each menu we will explore its sub-menu and so forth.

File Option

<u>F</u> ile	<u>E</u> dit	<u>V</u> iew	<u>N</u> odes	<u>L</u> ibraries	<u>M</u> anipulators	<u>M</u> ode	<u>W</u> indow	<u>H</u> elp
New Ctrl+N								
Open... Ctrl+O								
Close								
Save Ctrl+S								
Save As...								
1 C:\PROGRA~1\...\TETRA.WRL								
2 C:\worlds\reddots.wrl								
3 C:\worlds\tony13.wrl								
4 C:\worlds\tony1.wrl								
5 C:\worlds\VRML5.WRL								
6 C:\worlds\VRML4.WRL								
7 C:\worlds\VRML3.WRL								
Exit								

Figure 6: File Option

When selected, the **File** option from the Main Menu displays a list of file options to act upon the current file.

New CTRL+N opens a new data file.

Open CTRL+O opens a selected file. This option will open a dialog box that will allow the user to choose the file to be opened.

Close closes current file.

Save CTRL+S saves current file.

Save As saves selected file under a specific filename. This option will open a dialog box for the user to input the filename desired for the current file.

Save As Gzip... saves selected file as a gzipped file.

Recent File refers to recently open files.

Exit exits V-Realm Builder.

Edit Option

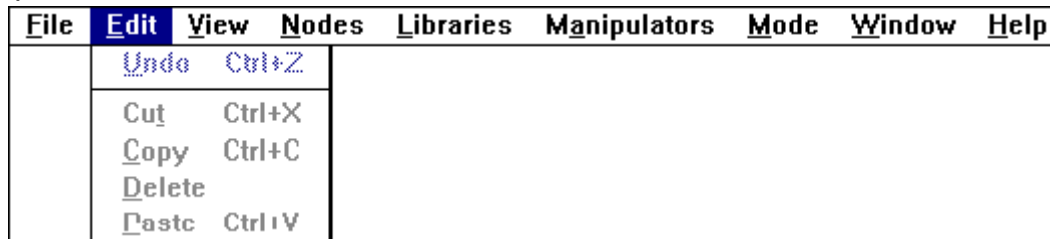


Figure 7: Edit Option

When selected, the **Edit** option from the Main Menu displays a list of edit options to act upon the current file.

Undo CTRL+Z will undo the last changes made to the file.

Cut CTRL+X cuts selected lines from the file, and places them on the clipboard.

Copy CTRL+C copies selected lines to the clipboard.

Delete deletes currently selected node.

Paste CTRL+V pastes the clipboard to the current world after the selected node.

View Option

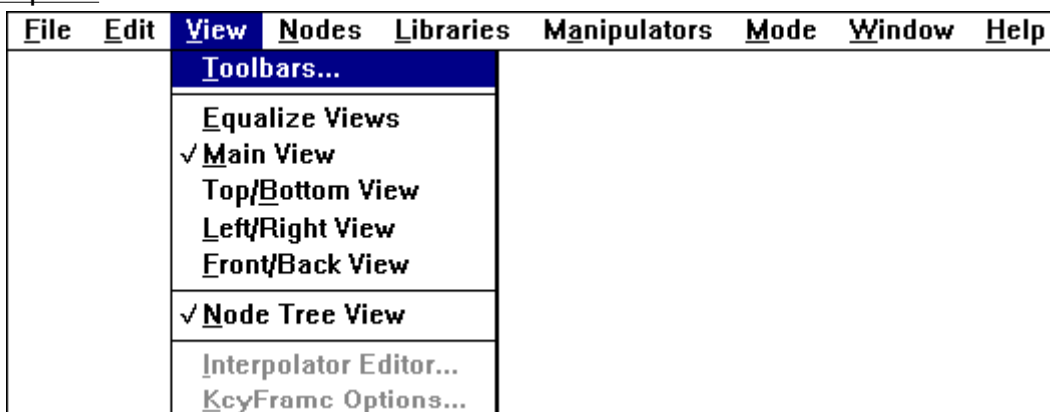


Figure 8: View Option

When selected, the **View** option from the Main Menu displays a list of items to be checked. The items checked will be active during your world building session. Listed below are the options and what they refer to.

Toolbars... allows the user to select the toolbars to display on the screen.

Equalize Views displays all available views.

Main View displays just the main perspective view.

Top/Bottom View displays just the top/bottom orthographic view.

Left/Right View displays just the left/right orthographic view.

Front/Back View displays just the front/back orthographic view.

Node Tree View displays the node tree window.

Interpolator Editor displays the Interpolator Editor.

KeyFrame Options allows user to set KeyFrame Options.

Nodes Option

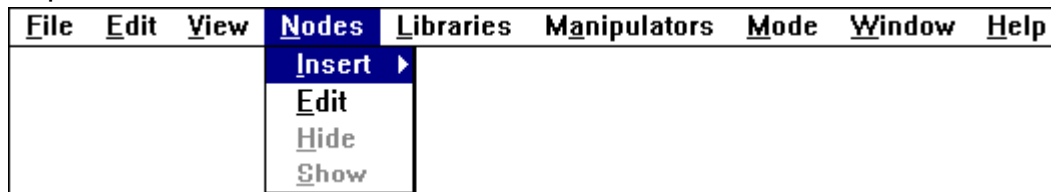


Figure 9: Nodes Option

When the **Nodes** option is chosen from the Main Menu, the following list of options are displayed for user selection. Please familiarize yourself with all the VRML nodes that can be used to build files.

A scene is the visual representation of the rendered .wrl file.

Insert There are many VRML nodes that can be inserted into the scene. They are divided into groups here. For an individual node select the appropriate group.

Edit Allows the user to edit the currently selected node.

Hide Hides the currently selected node. This has the effect of removing it from the scene graph without actually deleting it.

Show Shows the currently selected hidden node. This has the effect of returning the hidden node to the scene graph.

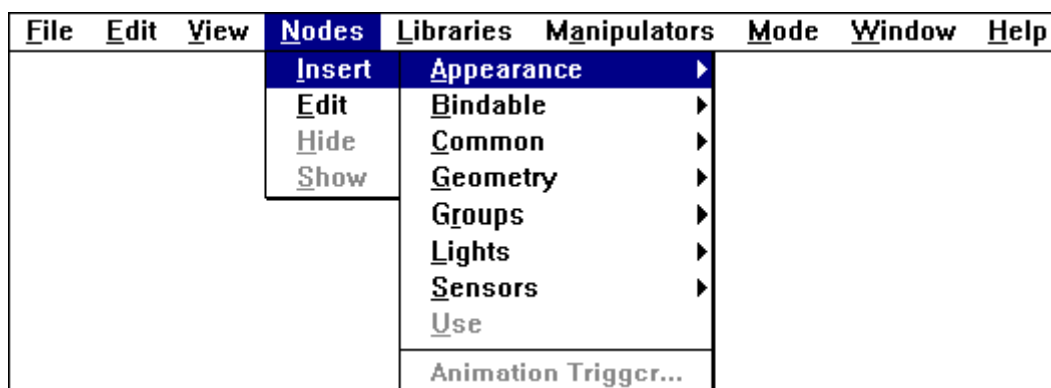


Figure 10: Nodes - Insert Option

When the Insert option is chosen under the Nodes option a list of node groups will be displayed for selection. At this point the user can choose any of the listed nodes for insertion into the scene. Below we will go into further detail listing the nodes included in each group. The USE and Animation Trigger options are also listed.

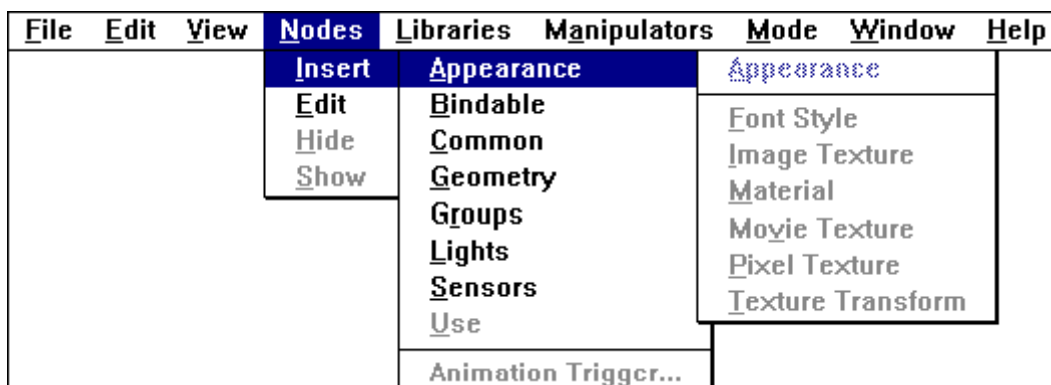


Figure 11: Appearance Option

Appearance Displays a list of appearance enhancing nodes that can be inserted into the scene.

Appearance Inserts an Appearance node into the scene.

FontStyle Inserts a FontStyle node into the scene.

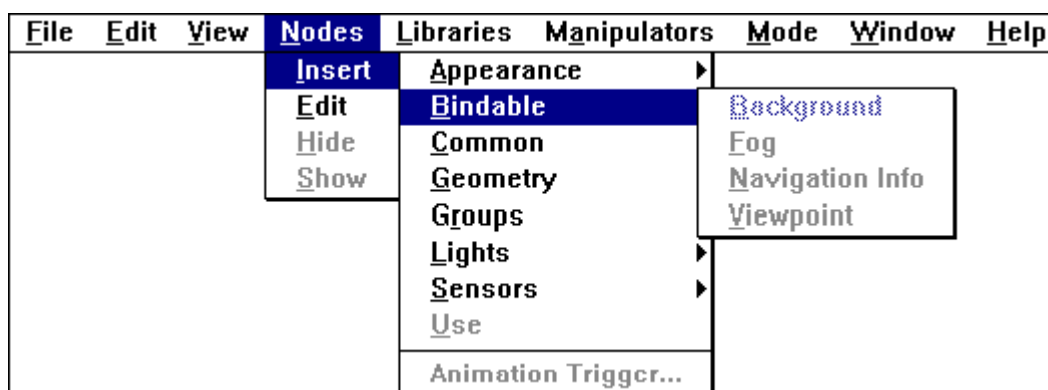
ImageTexture Inserts an ImageTexture node into the scene.

Material Inserts a Material node into the scene.

MovieTexture Inserts a MovieTexture node into the scene.

PixelTexture Inserts a PixelTexture node into the scene.

TextureTransform Inserts a TextureTransform node into the scene.

**Figure 12: Bindable Option**

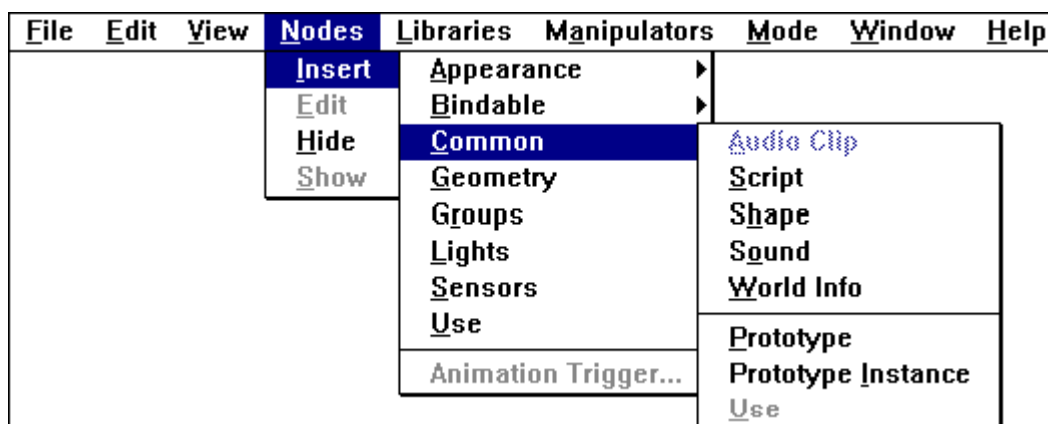
Bindable Displays a list of Bindable nodes that can be inserted into the scene.

Background Inserts a Background node into the scene.

Fog Inserts a Fog node into the scene.

NavigationInfo Inserts a NavigationInfo node into the scene.

Viewpoint Inserts a Viewpoint node into the scene.

**Figure 13: Common Option**

Common Displays a list of common nodes that can be inserted into the scene. **AudioClip** Inserts an AudioClip node into the scene.

Script Inserts a Script node into the scene.

Shape Inserts a Shape node into the scene.

Sound Inserts a Sound node into the scene.

WorldInfo Inserts a WorldInfo node into the scene.

Prototype Inserts a PROTO node into the scene.

Prototype Instance Inserts an instance to a PROTO node into the scene.

USE Inserts a copy of a node already **DEF'd** in the scene.

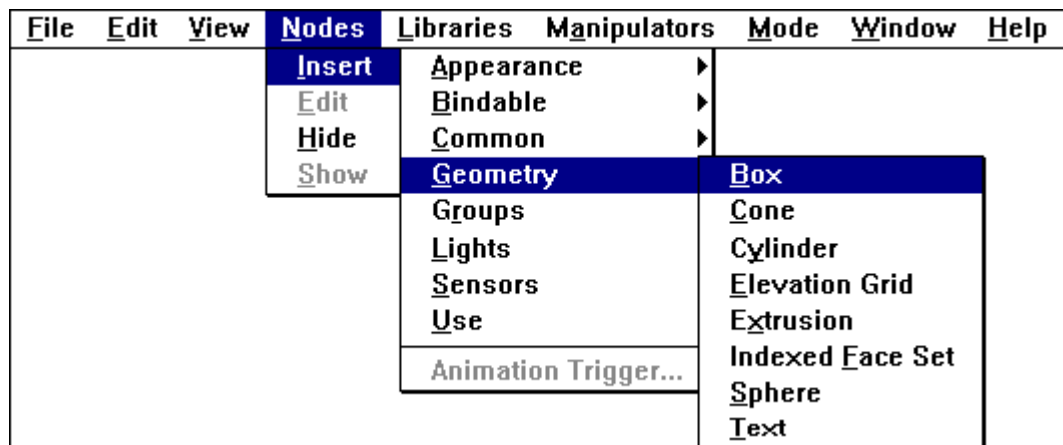


Figure 14: Geometry Option

Geometry Displays a list of geometry nodes that can be inserted into the scene.

Box Inserts a Box node into the scene.

Cone Inserts a Cone node into the scene.

Cylinder Inserts a Cylinder node into the scene.

ElevationGrid Inserts an ElevationGrid node into the scene.

Extrusion Inserts an Extrusion node into the scene.

IndexedFaceSet Inserts an IndexedFaceSet node into the scene.

Sphere Inserts a sphere node into the scene.

Text Inserts a Text node into the scene.

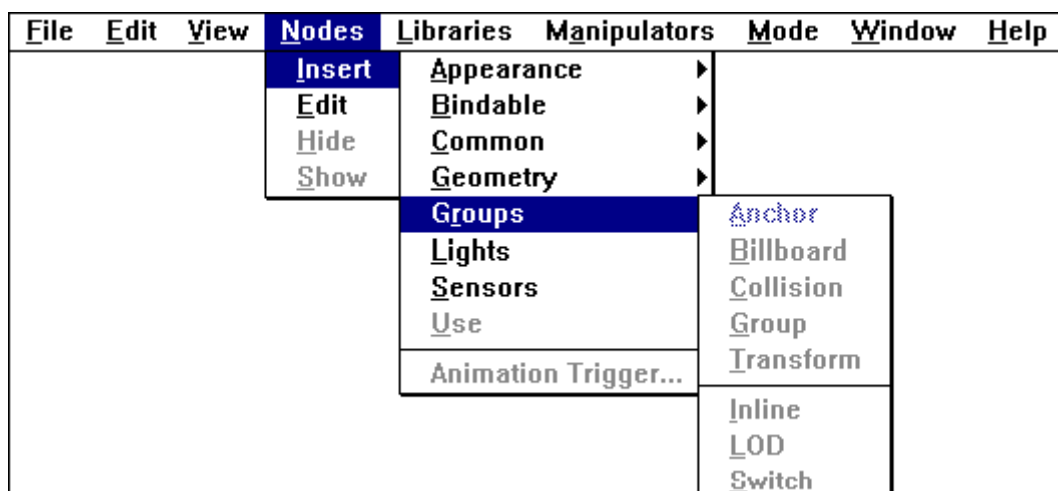


Figure 15: Groups Option

Groups Displays a list of grouping nodes that can be inserted into the scene.

Anchor Inserts an Anchor node into the scene.

Billboard Inserts a Billboard node into the scene.

Collision Inserts a Collision node into the scene.

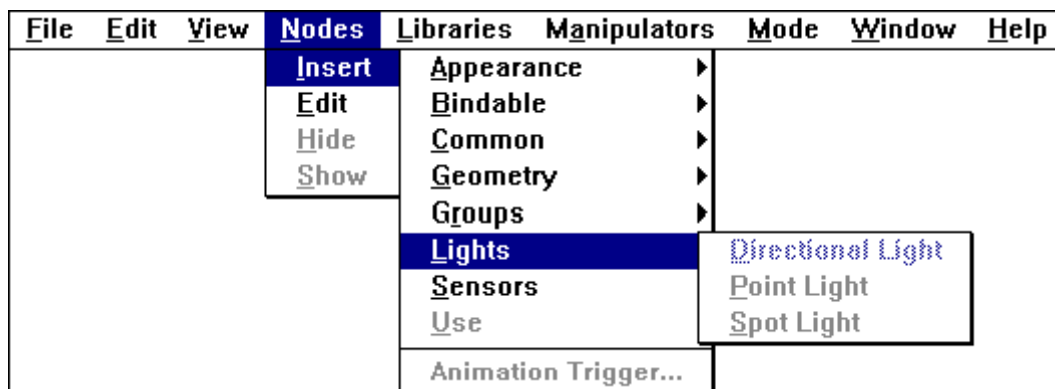
Group Inserts a Group node into the scene.

Transform Inserts a Transform node into the scene.

Inline Inserts an Inline node into the scene.

LOD Inserts a LOD node into the scene.

Switch Inserts a Switch node into the scene.

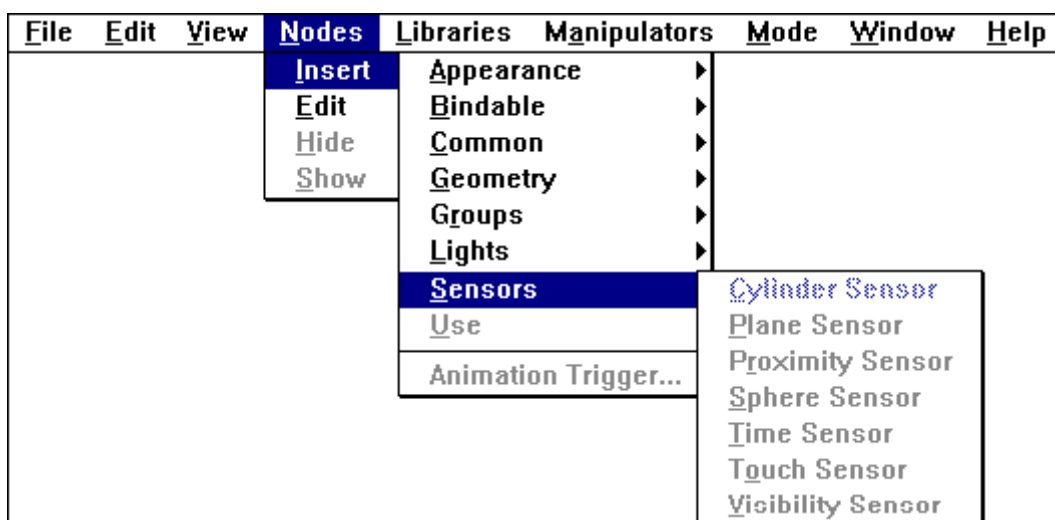
**Figure 16: Lights Option**

Lights Displays a list of lighting nodes that can be inserted into the scene.

DirectionalLight Inserts a DirectionalLight node into the scene.

PointLight Inserts a PointLight node into the scene.

SpotLight Inserts a SpotLight node into the scene.

**Figure 17: Sensors Option**

Sensors Displays a list of sensor nodes that can be inserted into the scene.

CylinderSensor Inserts a CylinderSensor node into the scene.

PlaneSensor Inserts a PlaneSensor node into the scene.

ProximitySensor Inserts a ProximitySensor node into the scene.

SphereSensor Inserts a SphereSensor node into the scene.

TimeSensor Inserts a TimeSensor node into the scene.

TouchSensor Inserts a TouchSensor node into the scene.

VisibilitySensor Inserts a VisibilitySensor node into the scene.

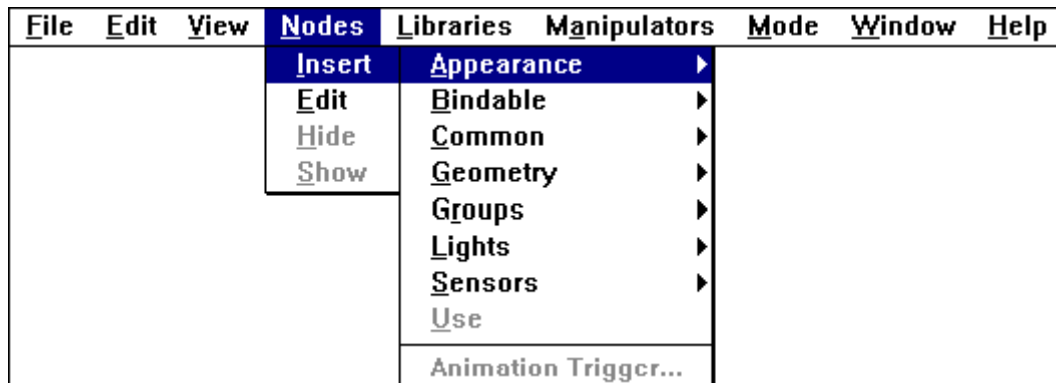


Figure 18: Use Option

Use Inserts a Use node into the scene.

Animation Trigger Inserts an Animation Trigger into a KeyFrame Animation.

Libraries Option



Figure 19: Libraries Option

Add To Allows the user to add items to a selected library.

Edit Allows the user to edit items in a selected library.

Import From Allows the user to import a library item into the scene.

Add To Option

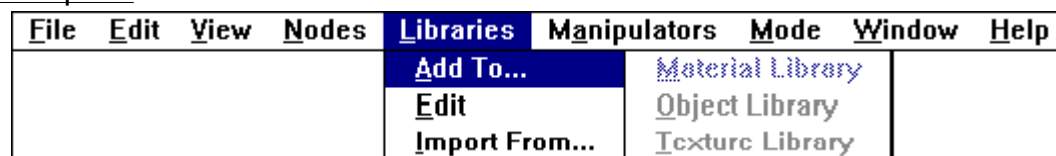


Figure 20: Add To Option

The user should select the library to which an item will be added. A dialog box will be displayed that will allow the item to be added. This option is used when an item selected in the Tree View is added to a library.

Edit Option

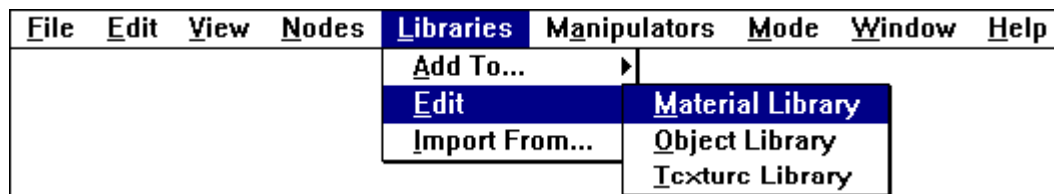


Figure 21: Edit Option

The user should select the library in which the item to be edited can be found. A dialog box will be displayed that will allow the user to edit items in the selected library.

Import From Option

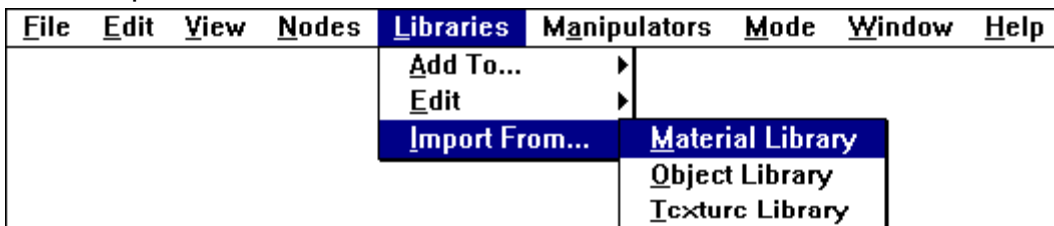


Figure 22: Import From Option

The user should select the library from which an item will be imported. A dialog box will be displayed that will allow the user to select an item for import to the scene graph.

Manipulators Option

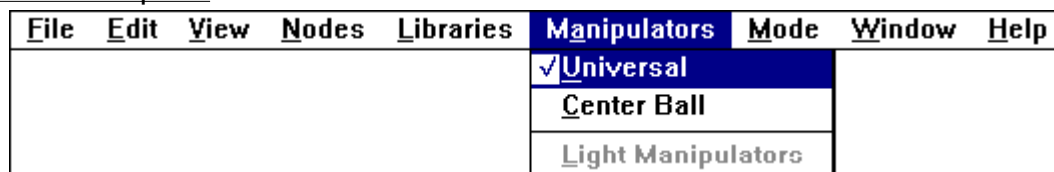


Figure 23: Manipulators Option

When the **Manipulators Option** is selected from the main menu, a list of manipulator options is displayed. In order for manipulators to work, the user must select an object for manipulation prior to selecting the manipulator option. Use the pick mode button to select the object, then select a manipulator to apply to the object.

Universal places a bounding box and tab box manipulator around a currently selected object. With these two manipulators and the use of the SHIFT, CONTROL, or CONTROL+SHIFT keys, you can produce the movement of any Open Inventor Manipulator. A table that shows the manipulation and the keystrokes for obtaining that manipulation follows. (See Appendix C) **This is the default Manipulator.**

Center Ball places a center ball manipulator around the currently selected object. The center ball manipulator allows the user to move or rotate an object about its center point. This manipulator maintains scale of objects being moved.

Light Manipulators displays the light manipulator in the scene. It allows the user to vary the position and direction of a light source in the scene. The Light Manipulator alters any light source in the scene. A light must be added to the scene before this option becomes active.

See Appendix C for complete description of Manipulators.

Mode Option

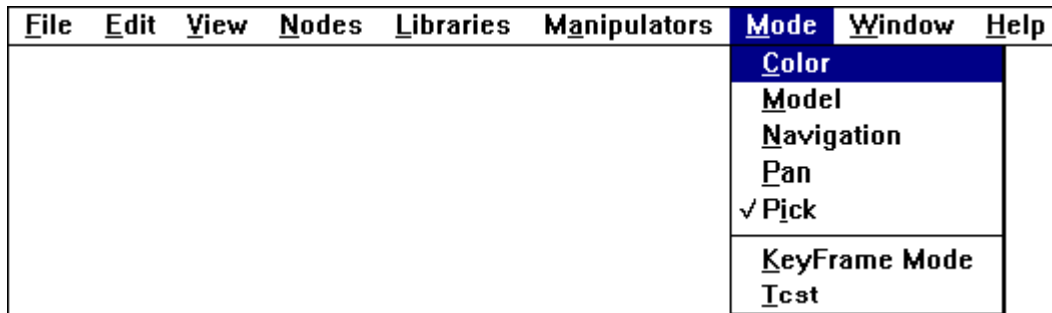


Figure 24: Mode Option

When the **Mode** option is chosen from the Main Menu the following items for selection will be displayed.

Color Sets the current Builder mode to color. This allows the user to edit/acquire color values for different kinds of geometry.

Model Sets the current Builder option to model. This allows the user to move and manipulate objects in the world being built.

Navigation Sets the current Builder option to navigation. This allows the user to walk around inside the world as it is being built.

Pan Allows the user to turn within the world and see the surrounding objects.

Pick Allows the user to select an object in the world to be manipulated.

KeyFrame Mode Opens the KeyFrame Animator for editing.

Test Allows an animation or route to be tested. Turns off manipulators.

Window Option

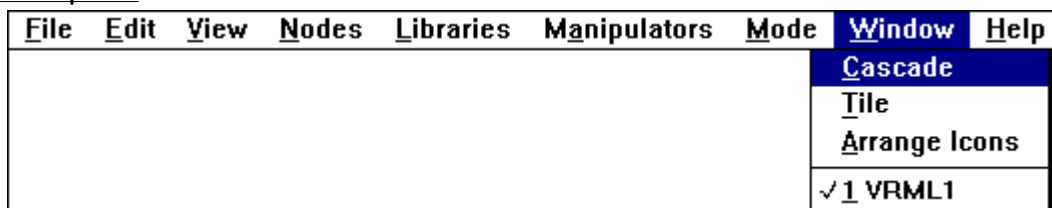


Figure 25: Window Option

When the **Window** option is chosen from the Main Menu the following items are displayed for user selection.

Cascade Cascades the currently open windows.

Tile Tiles the currently open windows.

Arrange Icons Arranges all the windows into icons on the screen.

List Lists up to eight open windows.

Help Option

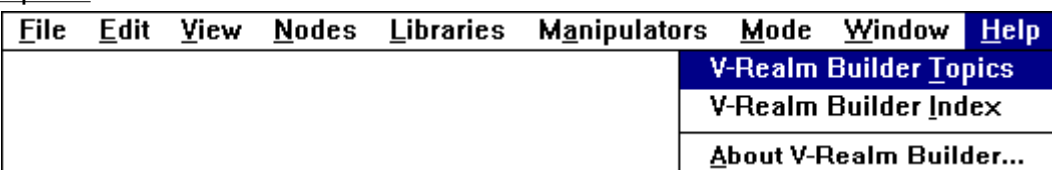


Figure 26: Help Option

The **Help Option** from the main menu displays the Help File options for the V-Realm Builder. The **V-Realm Builder Topics** and **V-Realm Builder Index** are specific help files for the V-Realm Builder. The **V-Realm Builder Topics** option displays the contents of the V-Realm Builder help file. The **V-Realm Builder Index** option displays all the sections in the V-Realm Builder in an alphabetically sorted list. Use the FIND option

from the **V-Realm Builder Index** or **V-Realm Builder Topics** to search for particular terms listed in the Help File. The **About V-Realm Builder** option lists current information regarding the V-Realm Builder.

V-Realm Pop-Up Menus

Tree View Pop-Up Menu

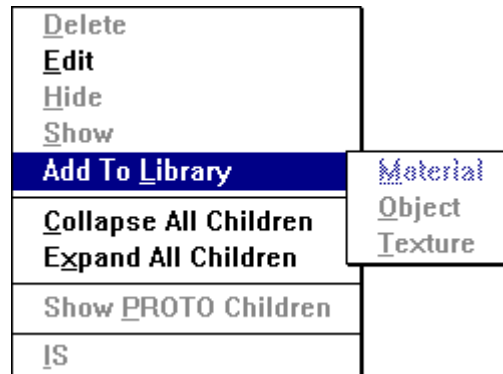


Figure 27: Tree View Pop-Up Menu

This menu is displayed when the right mouse button is clicked while in the Tree View Window. The Tree View must contain at least one node for the menu to appear. This menu gives the user access to the **Delete**, **Edit**, **Hide**, and **Show** functions, which all relate to a selected node in the Tree View Window. The Show PROTO Children item when activated will allow a PROTO to be seen in the view windows before it is instantiated. The **IS** item will allow fields to be linked through the **IS** function for PROTOs and External PROTOs. When an item has been selected, choosing the **Add To Library** option will allow the user to add the selected item to one of the displayed libraries. The options for **Collapse all children** and **Expand all children** will allow the user to either see all children nodes under a given node or have the node tree closed so that only the given node can be seen.

Main View Window Pop-Up Menu

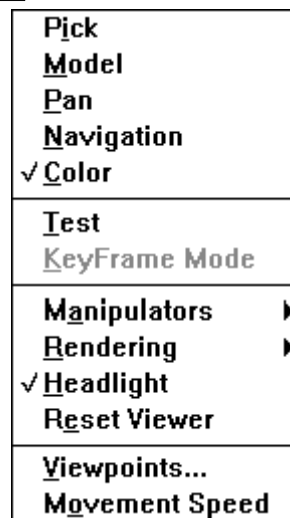


Figure 28: Main View Window Pop-Up Menu

This menu is displayed when the right mouse button is clicked while the cursor is in the Main View Window. This menu gives the user access to many commands that will affect the Main View Window.

Pick allows the user to select an object in the world to be manipulated.

Model sets the current Builder option to model. This allows the user to move and manipulate objects in the world being built.

Pan allows the user to turn within the world and see surrounding objects.

Navigation sets the current Builder option to navigation. This allows the user to walk around the world as it is being built.

Color sets the current mode to Color and opens the Material Painter Dialog.

Test allows the user to test a KeyFrame animation or route. Turns off manipulators.

KeyFrame sets the current mode to KeyFrame and opens the KeyFrame Animator.

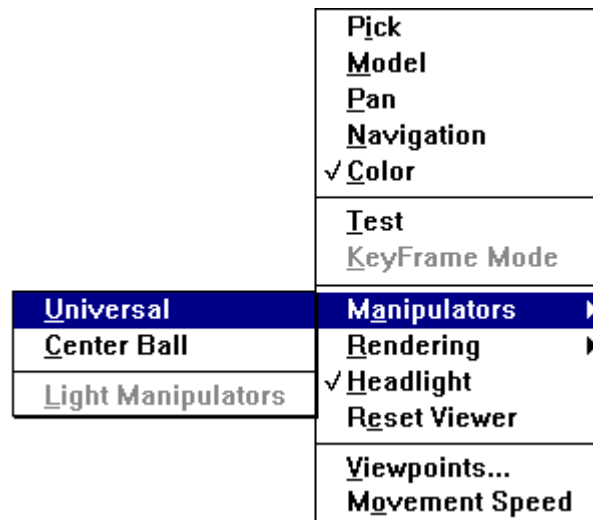


Figure 29: Manipulators Option

When the **Manipulators Option** is selected from the main Pop-Up menu, a list of manipulator options are displayed. In order for manipulators to work, the user must select an object for manipulation prior to selecting the manipulator option. Use the pick mode button to select the object, then select a manipulator to apply to the object. Clicking on an object to select it automatically applies the Universal manipulator or Centerball manipulator, if it is selected.

See Appendix C for complete description of Manipulators.

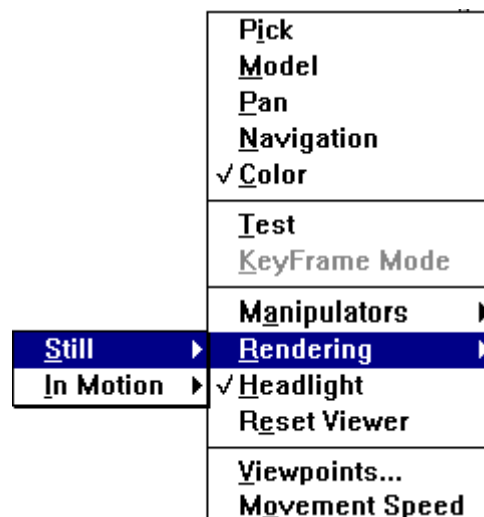


Figure 30: Rendering (Still and In Motion) Options

When either of the rendering options are chosen a list of rendering options are listed for selection.

Bounding Box sets the still or in motion rendering mode to bounding box and only shows the basic size and position of the object being rendered.

Default sets the still or in motion rendering mode to default and renders objects in the scene with as little or

as much detail as the scene graph specifies.

Flat Shading sets the still or in motion rendering mode to flat shading and renders objects with a limited shading quality. This renders most objects of a similar color or texture with a single color/shade, and limits the complexity of the scene.

Hidden line sets the still or in motion rendering mode to hidden line and renders objects so the basic shape and size can be determined, but extra lines hidden by the front of the object are not rendered.

Smooth Shading sets the still or in motion rendering mode to smooth shading and renders the objects as though the colors were evenly applied. This type of shading does not take into account shading of colors based on shadows or light.

Same as Still uses the same rendering method as Still Rendering.

Wire Frame sets the still or in motion rendering mode to wire frame and renders objects as though they were wire frame. This rendering method is similar to bounding box and smooth shading. It shows size, shape, and position as accurately as possible but does not retard rendering by placing textures or colors on the object.

Vertex sets the still or in motion rendering mode to vertex and basically renders only the vertices of a complex indexed shape.

Headlight This option turns on the headlight in the selected window but does not affect the headlight setting in any of the other windows. The default setting for **headlight** is **on**. Objects that emit color and light do not require the headlight to be on. Also use of directional, spot, and point lights can illuminate a scene.

Reset View This option places the viewer at the original coordinates when the world was opened and positions the orientation in the original position. This option opens the file from memory and does not make another call to the server for information.

Viewer Positions The user adds, deletes, or modifies viewer positions as needed.

Movement Speed The Movement Speed dialog box is displayed when this option is chosen. The user can set the current movement speed that will occur while moving around in a world being built from this dialog box. Press the OK button to accept the value entered or the CANCEL button to exit the dialog box without changing the value of the movement speed variable. The movement speed can be changed by the use of the F3 and F4 keys in the Main V-Realm Builder window. The F3 key will decrease the movement speed by a value of 1. The F4 key will increase the movement speed by a value of 1.

Orthographic View Pop-Up Menu

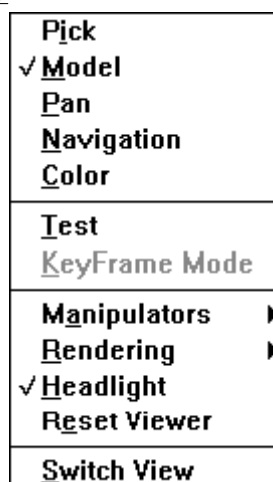


Figure 31: Orthographic View Pop-Up Menu

Pick allows the user to select an object in the world to be manipulated.

Model sets the current Builder option to model. This allows the user to move and manipulate objects in the world being built.

Pan allows the user to turn within the world and see surrounding objects.

Navigation sets the current Builder option to navigation. This allows the user to walk around inside the world as it is being built.

Color sets the current mode to Color and opens the Material Painter Dialog.

Test allows the user to test a KeyFrame animation or route. Turns off manipulators.

KeyFrame sets the current mode to KeyFrame and opens the KeyFrame Animator.

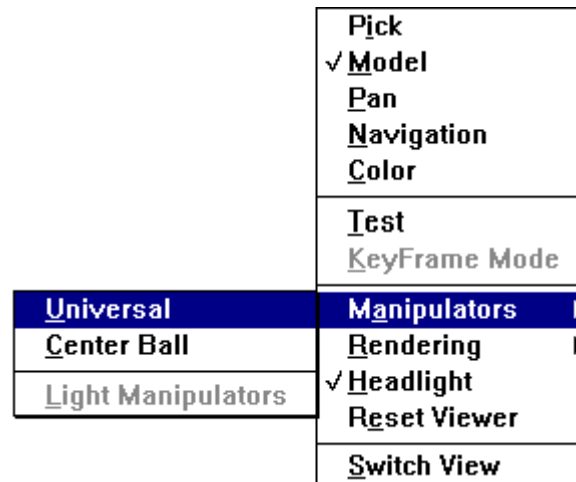


Figure 32: Manipulators Option

When the **Manipulators Option** is selected from one of the orthographic windows, a list of manipulator options is displayed. In order for manipulators to work, the user must select an object for manipulation prior to selecting the manipulator option. Use the pick mode button to select the object; then select a manipulator to apply to the object. Clicking on an object to select it automatically applies the Universal manipulator or Centerball manipulator, if it is selected.

See Appendix C for complete description of Manipulators.

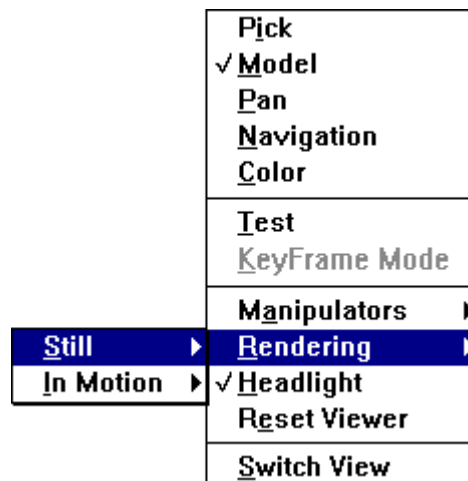


Figure 33: Rendering (Still and In Motion) option

When either of the rendering options is chosen, a list of rendering options is displayed for selection. Refer to the list of rendering styles listed on page 49.

Headlight This option turns on the headlight in one window but does not affect the headlight setting in any of the other windows. The default setting for **headlight** is **on**. Objects that emit color and light do not require the headlight to be on. Also use of directional, spot, and point lights can illuminate a scene.

Reset Viewer This option places the viewer at the original coordinates when the world was opened and positions the orientation in the original position. This option opens the file from memory and does not make

another call to the server for information.

Switch View Changes the current orthographic view to the opposite. Front to back, back to front, top to bottom, bottom to top, left to right, or right to left.

V-Realm Builder Toolbars

In discussing the toolbars available to the user in the V-Realm Builder application, we must first discuss why things were done the way they were. First, each toolbar was designed to give the user all functions relating to a specific task. Second, each toolbar was designed to automatically respond to the user's current choice in the node tree or view windows. Third, the user was made responsible for the position of each toolbar and whether or not a toolbar will be displayed at all. And lastly, the V-Realm Builder was designed to give you, the user, the ability to deal with each VRML node as "up-close-and-personal" as possible while giving you the ability to have many things done automatically for you. Let us discuss these issues and see how they may effect you in your world building process.

You will notice from the toolbars that are listed and explained below that each of them performs a specific function. The standard toolbar gives you access to all the editing functions. The grouping toolbar gives you access to all the VRML grouping nodes, and so on. By separating these functions into toolbars, you can activate only the toolbars that you need for the task you are performing, giving you more space on the screen to view the objects you are creating/modifying.

You will also notice that as you select objects or nodes in the node tree or view window, particular buttons on the toolbars will "gray out" or "stay solid". This feature was implemented to assist the user in knowing what kinds of nodes are related to each other. In other words, if you have selected the Texture node in order to apply a texture you will notice that only the MovieTexture, PixelTexture, and ImageTexture nodes will still be solid: also, you will notice that the Texture Library icon will also be solid. This acts like a VRML professor sitting beside you letting you know ahead of time if what you are doing is valid or not.

In making each toolbar the responsibility of the user we have made every toolbar movable and dockable. We have even made it possible to display just the desired toolbars by selecting just the ones you want to have displayed at any given time. Here we have given you, the user, all the control to make the creative environment conform to what is best for you.

The Standard Toolbar



Figure 34: The Standard Toolbar

The Standard Toolbar for the V-Realm application provides file, open, save, cut, paste, copy, and help buttons. These buttons give the user access to basic file functions.

Mode Toolbar



Figure 35: Mode Toolbar

The Mode Toolbar gives the user access to functions like select, model, move, Material Painter, KeyFrame Animator, Test, library control(object, material, texture), manipulators, and window control. The Mode Toolbar gives the user access to some of the V-Realm Builder's basic functions. The ability to select, manipulate and save objects to reusable libraries is one operation that can be accomplished using these tools. Another is the ability to control the layout of the screen with one or more view windows from which to inspect the world being created.

Common Toolbar



Figure 36: Common Toolbar

The Common Toolbar makes available to the user common nodes like AudioClip, Sound, Script, WorldInfo, lights, Background, Fog, NavigationInfo, Viewpoint, Route, Use, PROTO, and PROTO instance. The Common Toolbar gives the user access to some of the more powerful tools (nodes) available in VRML. These nodes provide the "bells and whistles" for any VRML file. They also provide the "mood" for VRML worlds through the use of Fog and lighting nodes.

Group Toolbar

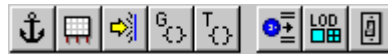


Figure 37: Group Toolbar

The Group Toolbar makes available grouping nodes. The Group Toolbar makes available to the user the specialized nodes that allow the enhancement of geometry, and nodes that allow the user to load alternate geometry from elsewhere. Nodes like Anchor, Billboard, Collision, Group, Transform, Inline, LOD and Switch are very powerful node types and are a basis for building interactive worlds.

Geometry Toolbar



Figure 38: Geometry Toolbar

The Geometry Toolbar makes available to the user all the geometry nodes available in VRML, from the most basic to the more complex Extrusion and IndexedFaceSet geometry nodes. Also available are the nodes that describe the geometry nodes. Those nodes relating to textures, materials, shape and appearance as well as the FontStyle node.

Sensor Toolbar



Figure 39: Sensor Toolbar

The Sensor Toolbar makes available to the user all the sensor nodes available. The Cylinder, Plane, Touch, Sphere, Time, Proximity, and Visibility Sensors are all available to the user.

Main Window Toolbar



Figure 40: Main Window Toolbar

The Main Window Toolbar consists of the Viewer Position button, Rendering Mode button, toggle Headlight button, and the Reset Viewer button. These buttons will affect only the window from which they are activated.

Orthographic Windows Toolbar



Figure 41: Orthographic Window Toolbar

The Orthographic Window Toolbar consists of the Flip View button, Rendering Mode button, toggle Headlight button, and the Reset Viewer button. These buttons will affect only the window from which they are activated.

Use of Mouse

The mouse is used extensively in the V-Realm Builder application for selecting items in the view windows, selecting functions from the menu or button toolbars, or calling up special pop-up menus.

Throughout the V-Realm Builder application you will find that, whenever possible, the mouse is used to select and position the cursor so that entering necessary data is made as painless as possible. Single click to highlight/choose - double click to activate/open. The mouse is by far the most important tool used in the V-

Realm Builder. It is a multi-purpose tool and any special uses of the mouse will be enhanced-shaded for your convenience.

One special function of the mouse is used in the Tree View Window. If the user left clicks on a node name in the window and types a name, the View Window's search engine will go directly to that node, even if that node is before the selected node in the node tree. Single click a second time on a node name and the Tree View Window will allow you to alter the currently selected node's **DEF** Name.

Left click on a node in the Tree View Window and type a desired **DEF** Name for another node and the desired node, will be displayed. (Search for **DEF** Name function)

The right-click of the mouse in most view panes will display the special pop-up menu for that view window.

Double left-click on a highlighted field will display a dialog box for input/edit.

Left click once on the selected field to enter a **DEF** name.

Dialog Boxes

Dialog boxes are used in the V-Realm Builder to allow the user to input specific data into the desired fields of nodes. The dialog boxes that are provided are specific to the field type and the data that is required for that field. The controls provided like slider bars and spin buttons are there to make the modification of some fields smoother.

Editing VRML 2.0 Fields

Integer Editing

Integer fields in the scene graph are preceded by the following icon:



Figure 42: Integer Field Icon

When the variable which is preceded by this icon is clicked on, an edit dialog will be displayed which will allow the user to edit this field. Shown below is the Integer Edit Dialog box.

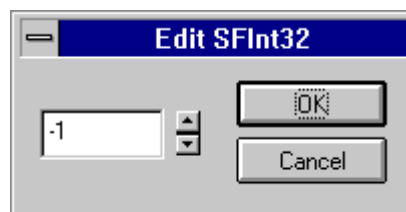


Figure 43: Integer Edit Dialog Box

Integer variables are edited through the Integer Edit Dialog box. This dialog allows for direct integer input from user or spin controls. An integer is a whole number, positive or negative, with no decimal part. 100 is an integer but 100.1 is not. The OK button accepts input data while the CANCEL button exits the dialog box without accepting the input.

Floating Point Editing

Floating point fields in the scene graph are preceded by the following icon:



Figure 44: Floating Point Field Icon

When the variable which is preceded by this icon is clicked on, an edit dialog will be displayed which will allow the user to edit this field. Shown below is the Floating Point Edit Dialog box.

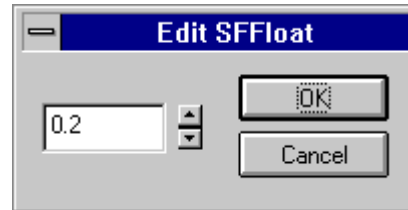


Figure 45: Floating Point Edit Dialog Box

Floating point numbers are described as numbers that use the format **number-decimal-number**. For instance the number 25.347 is a floating point number. Floating point variables are edited through the Floating Point Edit Dialog box. This dialog allows for direct integer input from user or spin controls. The OK button accepts input data while the CANCEL button exits the dialog without accepting the input.

Boolean Editing

Boolean fields in the scene graph are preceded by the following icon:



Figure 46: Boolean Field Icon

When the variable which is preceded by this icon is clicked on, an edit dialog will be displayed which will allow the user to edit this field. Shown below is the Boolean Edit Dialog box.

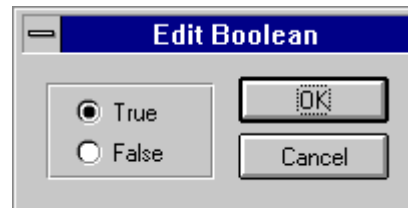


Figure 47: Boolean Edit Dialog Box

A Boolean value can only be TRUE or FALSE. Boolean variables are edited through the Boolean Edit Dialog box. This dialog allows for direct selection of the Boolean value with TRUE or FALSE radio buttons. The OK button accepts input data while the CANCEL button exits the dialog without accepting the input.

String Editing

String fields in the scene graph are preceded by the following icon:



Figure 48: String Field Icon

When the variable which is preceded by this icon is clicked on, an edit dialog will be displayed which will allow the user to edit this field. Shown below is the String Edit Dialog box.

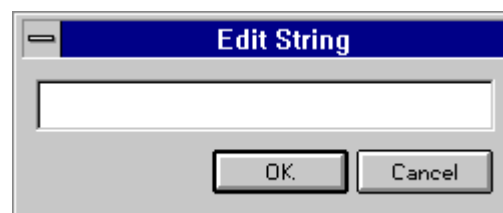


Figure 49: String Edit Dialog Box

A string is a single continuous list of alphanumeric or punctuation characters. String variables are edited through the Edit String Dialog box. This dialog allows for direct input of a string in an open edit field. The OK button accepts input data while the CANCEL button exits the dialog without accepting the input. Please note that VRML is case sensitive, so "Geometry" as a string is not the same as "geometry".

URL Editing

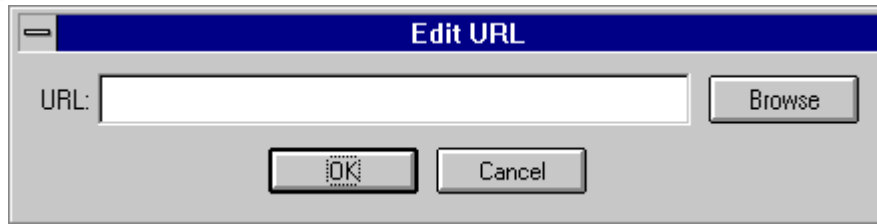


Figure 50: Edit URL Dialog Box

This dialog box is used to enter either an URL or a local filename. The BROWSE button is used to search the local drive system for a file or the user can enter the URL directly into the URL edit field. The OK and CANCEL buttons are used to accept or not accept the information input.

Vector (2) Editing

Vector (2) fields in the scene graph are preceded by the following icon:



Figure 51: Vector (2) Field Icon

When the variable which is preceded by this icon is clicked on, an edit dialog will be displayed which will allow the user to edit this field. Shown below is the Vector (2) Edit Dialog box.

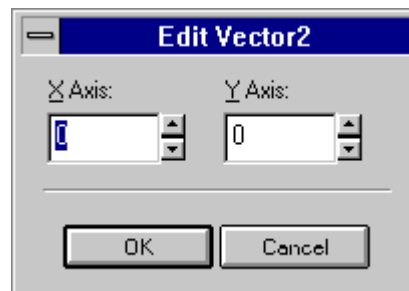


Figure 52: Vector (2) Edit Dialog Box

Vector (2) variables are defined as two floating point numbers used to specify x and y axis values. Vector (2) variables are edited through the Vector (2) Edit Dialog box. This dialog allows for the direct input of vector (2) values from user input or spin controls. The OK button accepts input data while the CANCEL button exits the dialog without accepting the input.

Vector (3) Editing

Vector (3) fields in the scene graph are preceded by the following icon:



Figure 53: Vector (3) Field Icon

When the variable which is preceded by this icon is clicked on, an edit dialog will be displayed which will allow the user to edit this field. Shown below is the Vector (3) Edit Dialog box.

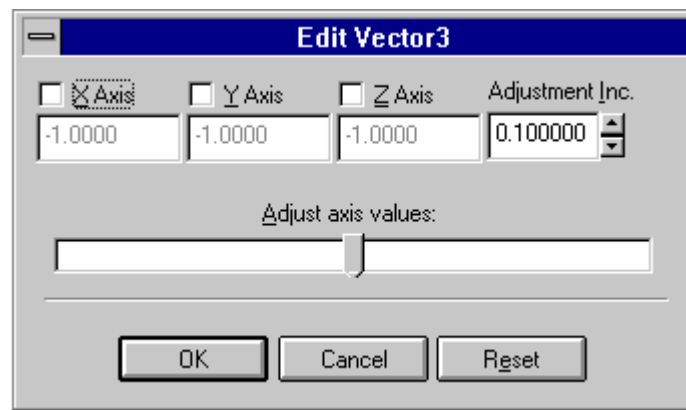


Figure 54: Vector (3) Edit Dialog Box

Vector (3) variables are defined as three floating point numbers used to specify x, y, and z axis values. Vector (3) variables are edited through the Vector (3) Edit Dialog box. This dialog allows for the direct input of vector (3) values from user input, spin controls, and/or slider bar control. Check boxes must be selected for the slider to alter individual values. The Adjustment increment is provided to allow the slider to alter the individual values by a selected amount. The OK button accepts input data while the CANCEL button exits the dialog without accepting the input.

Rotation Editing

Rotation fields in the scene graph are preceded by the following icon:



Figure 55: Rotation Field Icon

When the variable which is preceded by this icon is clicked on, an edit dialog will be displayed which will allow the user to edit this field. Shown below is the Rotation Edit Dialog box.

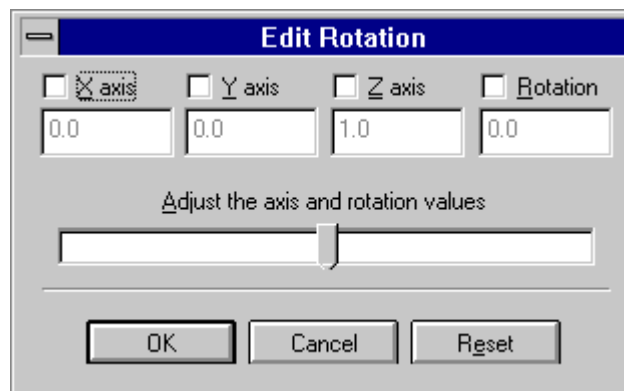


Figure 56: Rotation Edit Dialog Box

Rotation is defined as a vector of four floating point numbers, the first three for x, y, and z axis, and the last for a rotation factor around the selected axis or axes. Rotation variables are edited through the Rotation Edit Dialog box. This dialog allows for the direct input/change of rotation values from user input or slider bar. Check boxes must be selected for the slider to alter individual values. The OK button accepts input data while the CANCEL button exits the dialog without accepting the input.

Color Editing

Color fields in the scene graph are preceded by the following icon:



Figure 57: Color Field Icon

When the variable which is preceded by this icon is clicked on, an edit dialog will be displayed which will allow the user to edit this field. Shown below is the Color Edit Dialog box.

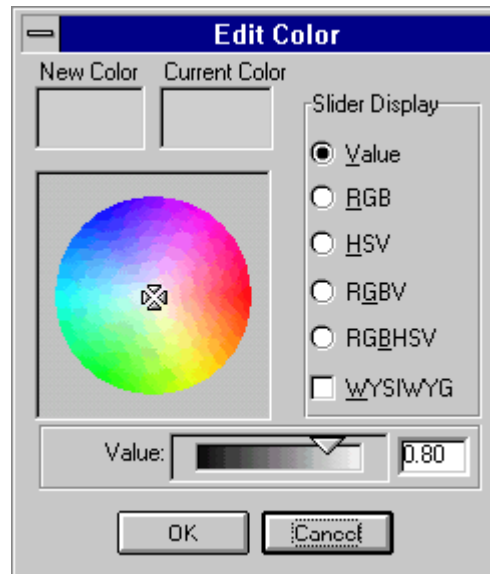


Figure 58: Color Edit Dialog Box

A color value can be defined in any of the standard color methods. The most common way of defining color is through the use of RGB, where R equals Red, G equals Green, and B equals Blue. These RGB color amounts would be mixed as the paint store technician would mix a can of paint. Color variables are edited through the Color Edit Dialog box. This dialog allows for direct selection of color values from the use of the mouse to select a color and/or from the input of appropriate color values in the color value fields displayed. Note that there are several color editing selections available. Choose the editor type you are most familiar with. The New Color and Current Color preview windows display color selections that have been made in the Color Edit Dialog or the color selection that was made before the Color Edit Dialog was opened. The OK button accepts input data while the CANCEL button exits the dialog without accepting the input.

Time Editing

Time fields in the scene graph are preceded by the following icon:



Figure 59: Time Field Icon

When the variable which is preceded by this icon is clicked on, an edit dialog will be displayed which will allow the user to edit this field. Shown below is the Time Edit Dialog box.

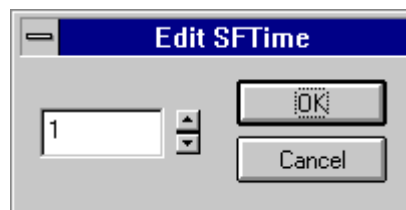


Figure 60: Time Edit Dialog Box

Time variables are edited through the Time Edit Dialog box. This dialog allows for direct input of time index values from the user or spin controls. Time is calculated as the number of seconds from the start time of the world being viewed. If Time has a value of 100 this would indicate that the time is 100 seconds after the start of the world. The OK button accepts input data while the CANCEL button exits the dialog without accepting the input.

Multiple Field Editing



Figure 61: Multiple Field Icons

When a Multiple Field icon is pressed the following dialog is displayed.

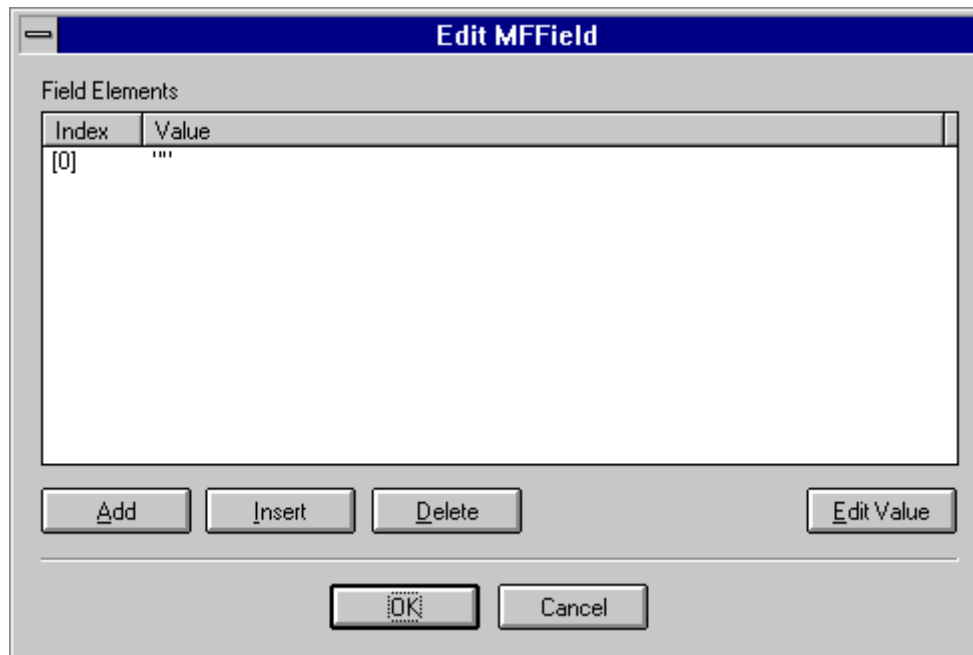


Figure 62: Multiple Field Edit Dialog Box

This dialog box keeps track of multiple data strings input for fields that require multiple values. The **Index** starts at 0 and counts up. As each value is edited the **Value** field for the particular Index will be updated. The ADD, INSERT, DELETE, and EDIT VALUE buttons give the user full access to value editing functionality so that the list of data strings can be maintained. The OK and CANCEL buttons are provided to accept or not accept the changes made during this editing session.

Color Mode Dialog

This dialog box is displayed when the Color Mode is turned on. This mode allows the user to edit and acquire colors from the scene to use elsewhere or to save to the library for later use.

Toolbar Configuration Dialog

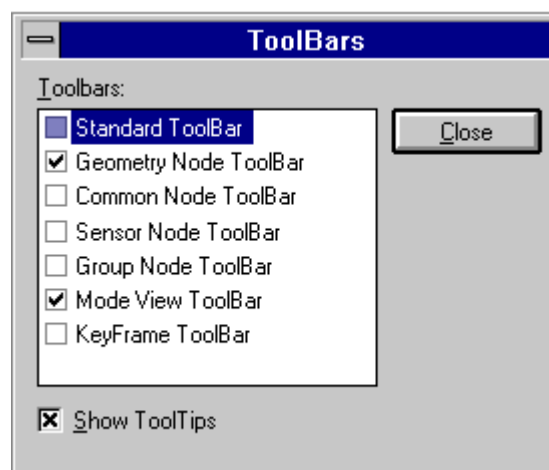


Figure 64: Toolbar Configuration Dialog Box

This dialog box is displayed when the user chooses the View Toolbars option from the Main Menu. This dialog box allows the user to choose the toolbars to be displayed at any given time. On the left, the user selects the toolbars to be displayed by checking the boxes. The **Show Tooltips** selection at the bottom determines if the tool tips will be displayed as the cursor moves over the toolbars. The CLOSE button accepts the selections and closes the dialog box.

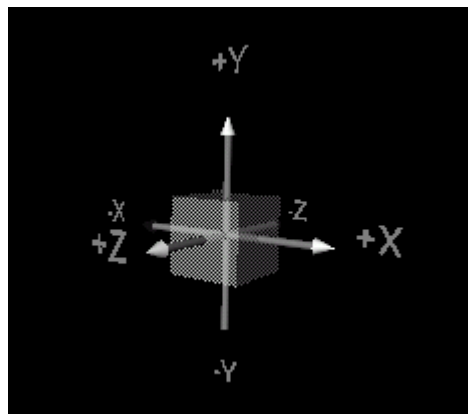
VRML Coordinate System

One of the fundamental attributes of any 3D object is its position in 3D space. VRML uses the Cartesian coordinate system, in which a point has an X coordinate denoting its position along a horizontal (width) axis, a Y coordinate denoting its position along a vertical (height) axis, and a Z coordinate representing object depth. To remember how the coordinate system works in VRML, remember that the X axis points to the right, the Y axis points up, and the Z axis points out towards the viewer.

An object created in this 3D space will have associated with it a set of X, Y, and Z values that correspond to the center of the object (primitives) or to the vertices that make up the object (PointSet, IndexedLineSet, IndexedFaceSet, or other complex objects).

By default, the center of all primitive objects is assumed to be (0,0,0). This can be changed by using a transform or translation node to alter its position in space. Since more complex objects are created by specifying the points that make up the surface of the object, there is no assumption as to the center of the object. This would be defined by the calculation made from wrapping a bounding box around the object and calculating the center point.

The following figure shows the box in the 3D space that we have been talking about. Notice how the axes are pointing.

**Figure 65: VRML Coordinate System**

[3] The Node Tree

The Hierarchy

A **group** node is a VRML node type that groups elements called **children** together. For example, a building can be thought of as a **group** node with windows, doors, and walls as **children** since they are a subset of the larger structure.

There is no limit to the number of **children** a **group** node can have, but all **children** share the characteristics of the parent (**group**) node.

Nodes and Fields

A node refers to any VRML node type listed in the VRML Specification. For example, the **Box** node has a field called **size**. The term **Box** refers to the name of the VRML node and the term **size** refers to a parameter that describes a characteristic of the VRML node **Box**. We could use this type of example to describe all the VRML nodes, but to make this discussion simple, let us say that fields describe characteristics of VRML nodes and the name of the node tells us the VRML node being described.

All nodes can be given a **DEF** name. This **DEF** name is used by the V-Realm Builder to perform many tasks. It is very important for you to give every node a descriptive name so that these nodes can be called upon later.

One of the tools that is driven by the **DEF** name is the search function. It is possible to select any node in the Node Tree and type a **DEF** name you want to "goto", and the node will be automatically searched for and displayed if found. If you do not give each node a distinct name the search function will not perform correctly.

Secondly, it is possible to reuse whole nodes in the Node Tree by the use of the **USE** function. Only nodes that have been given a **DEF** name can be reused in this fashion.

Thirdly, only nodes that have been given a **DEF** name will be displayed in the Route Dialog Box, which will be described later in this document. This function is very important if you wish to create animated or moving worlds. Only specifically named nodes can be used to accept sensors and interpolators.

Give every node a DEF name that describes the function of the node.

Basic Grouping

The Group and Transform nodes are the two most basic grouping nodes. The Group node acts as a parenthesis when it groups other nodes together. It does not impress any special restrictions on the nodes that it groups. The Transform node defines a coordinate system for its children and thus will require uniformity between its children. This means that children of a transform will all use the same local coordinate system.

Group



Figure 66: Group Node Icon

A Group node groups its children together but does not impose any special constraints upon them.

Transform



Figure 67: Transform Node Icon

A Transform node groups its children together and defines a coordinate system for them that is relative to the parent's coordinate system.

[4] Basic Worldbuilding

Geometry and Appearance

When discussing the basics of world building we must discuss two general principles: geometry and appearance. Geometry in its simplest state refers to the primitive objects that are made available in VRML - Box, Cone, Cylinder, and Sphere. These simple objects are created with default values that make them visible in the scene. These simple primitive objects can be altered in size, color, texture, position, and many other ways. When an object is altered, we are changing its appearance. Appearance, Color, ImageTexture, Material, MovieTexture, PixelTexture, FontStyle, TextureTransform, and Shape are the VRML nodes that allow us to alter some of the appearance characteristics of geometry whether primitive or not.

Primitive Geometry nodes:

Box



Figure 68: Box Node Icon

The Box node specifies a cube in the local coordinate system centered at (0,0,0) and aligned with the coordinate axes. By default, the box measures 2 units in each dimension, from -1 to +1. The Box's size field specifies the extent of the box along the X, Y, and Z axes respectively and must be greater than 0.0. Textures are applied individually to each face of the box; the entire untransformed texture goes on each face. On the front, back, right, top, bottom, and left faces of the box, when viewed from the outside with the Y axis up, the texture is mapped onto each face with the same orientation as if the image were displayed in normal 2D. On the top face of the box, when viewed from the outside along the +Y axis looking down with the -Z axis as the view up direction, the texture is mapped onto the face with the same orientation as if the image were displayed in normal 2D. In the bottom face of the box, when viewed from the outside along the -Y axis looking up with the +Z axis as the view up direction, the texture is mapped onto the face with the same orientation as if the image were displayed in normal 2D. The Box geometry is considered to be solid and thus requires outside faces only.

Cone



Figure 69: Cone Node Icon

The Cone specifies a cone which is centered in the local coordinate system and whose central axis is aligned with the local Y axis. The **bottomRadius** field specifies the radius of the cone's base, and the height field specifies the height of the cone from the center of the base to the apex. By default, the cone has a **radius** of 1.0 at the bottom and a **height** of 2.0, with its apex at y=1 and its bottom at y=-1. Both **bottomRadius** and **height** must be greater than 0.0. The **side** field specifies whether sides of the cone are created, and the **bottom** field specifies whether the bottom cap of the cone is created. When a texture is applied to the sides of the cone, the texture wraps counterclockwise (from above) starting at the back of the cone. The texture has a vertical seam at the back in the YZ plane. The bottom cap texture appears right side up when the top of the cone is rotated towards the -Z axis. The Cone geometry is considered to be solid and thus requires outside faces only.

Cylinder



Figure 70: Cylinder Node Icon

The Cylinder node specifies a capped cylinder centered at (0,0,0) in the local coordinate system with a central axis oriented at the local Y axis. By default, the cylinder **height** is sized from -1 to +1. **Radius** is 1. The radius field specifies the cylinder's **radius** and the **height** field specifies the cylinder's height along the central axis. Both **radius** and **height** must be greater than 0.0. The cylinder has three parts: the **side**, the **top** (Y= +height), and the **bottom** (Y= -height). Each part has an associated Boolean field that indicates whether the parts exist (TRUE) or do not exist (FALSE). If the parts do not exist, they are not considered during collision detection. When a texture is applied to the cylinder, it is applied differently to the **sides**, **top**, and **bottom**. On the **sides**, the texture wraps counterclockwise (from above) starting at the back of the cylinder. The texture has a vertical seam at the back, intersecting the YZ plane. For the **top** and **bottom** caps, a circle is cut out of the unit square centered at (0, height, 0) with dimensions (2 x radius) by (2 x radius). The **top** texture appears right side up when the top of the cylinder is tilted toward the +Z axis, and the **bottom** texture appears right side up when the top of the cylinder is tilted toward the -Z axis. The Cylinder geometry node is considered to be solid and thus requires outside faces only.

Sphere



Figure 71: Sphere Node Icon

The Sphere node specifies a sphere centered at (0,0,0) in the local coordinate system. The **radius** field specifies the radius of the sphere and must be ≥ 0.0 . When a texture is applied to a sphere, the texture covers the entire surface, wrapping counterclockwise from the back of the sphere. The texture has a seam at the back where the YZ plane intersects the sphere. The Sphere geometry is considered solid and thus requires outside faces only.

Appearance Nodes:

Along with all the other texture file formats available for application to a piece of geometry, the user also has the ability to apply transparent .GIF formatted textures. These textures act in a special way in that they make the base "color" or "material" transparent so that the object can be seen through. In this way, only the non-transparent part of the transparent .GIF will be seen on the object.

Appearance



Figure 72: Appearance Node Icon

The Appearance node specifies the visual properties of geometry by defining the material and texture nodes. The default value for each of the fields in this node is NULL. However, if the field is non-NULL, it must contain a node of the appropriate type.

ImageTexture



Figure 73: Image Texture Node Icon

The ImageTexture node defines a texture map by specifying an image file and general parameters for mapping image to geometry.

Material



Figure 74: Material Node Icon

The Material node specifies surface material properties for associated geometry nodes and is used by the VRML lighting equations during rendering. All fields in the Material node range from 0.0 to 1.0.

MovieTexture



Figure 75: Movie Texture Node Icon

The MovieTexture node defines a time dependent texture map (contained in a movie file) and parameters for controlling the movie and the texture mapping. This file can be AVI or MPEG, but the V-Realm Builder does not support compressed audio tracks in AVI files.

Shape




Figure 76: Shape Node Icon


The Shape node has two fields: **appearance** and **geometry**, which are used to create rendered objects in the world. The **appearance** field specifies an Appearance node that specifies the visual attributes (e.g. material and texture) to be applied to the geometry. The **geometry** field specifies a geometry node. The specified geometry node is rendered with the specified appearance nodes applied. If the geometry node is NULL the object is not drawn. The default is NULL.


Exercise 1: Basic Worldbuilding


In order to get a solid understanding of the interface and the building process, we'll use a variety of methods to build an object. We'll later see that there are simpler methods to accomplish some of these tasks. But for the time being, we want to get familiar with the three sections of the Builder interface: the Toolbars, the View Pane(s), and the Node Tree.

1. Open the Builder and create a new, blank file .

2. The **Toolbar** is actually made up of a number of dockable sub-toolbars, with icons (representing **nodes**) grouped together by common function. With your left mouse button, you can click in an empty area on the toolbar and drag those tools to a new location. Try it with the toolbar group that has an "Anchor" icon (the

Group toolbar) . Left click and drag on an empty area of the Group toolbar, and release. If the mouse button is released away from the edges of the application, the toolbar remains free floating. Click on the "x" button to close that toolbar. Go up to the "View" menu at the top and open the **Toolbars dialog**. This gives you a choice of which toolbars to display. Check off "Group Node Toolbar" and your toolbar reappears. Close the Toolbars Dialog box, then left-click and drag the toolbar to an empty space along the edges, release the mouse button, and the toolbar will dock.

3. Let's create our first object, a pillar. First, we'll specify that what we're about to create is all contained within a single group. Return to your Group Toolbar and find the icon representing the **Insert Group** . Left-click on the icon and you'll see a Group node is added to the **Node Tree** on the left-hand side of the screen. Nothing changes in the black **View Pane** because this is only a Grouping node. There's nothing visual to it yet. Note that in the Node Tree, the Group has a plus sign next to it. This means that it is expandable. Left click on the plus sign, and the node expands to reveal the **fields** associated with the node (nodes begin with a capital letter, fields in lowercase). Click on the **children** field to select it. Anything we add to the Node Tree at this moment will, in the hierarchy, be considered a child to the parent, Group. All components of the pillar we'll create will be arranged underneath this group.

4. On the Geometry toolbar, left click on the **Insert Cylinder** icon . The Node Tree is updated to show a **Transform group** containing the geometry and properties of a cylinder. The cylinder appears in the View Pane, with handles showing that it is selected. This cylinder will be the shaft of our pillar.

5. Let's make the cylinder taller. On your keyboard, hold down the **CTRL** and **SHIFT** keys. With your mouse in the View Pane, left click and hold on one of the small white manipulator boxes on the top of the cylinder. Now, keeping your mouse button down, drag the mouse forward (up along the Y-axis) to make the cylinder scale taller. Note how the status bar at the bottom reflects the changes you are making. Release the mouse button when you are finished (for a complete explanation of the use of manipulators, see "Appendix C - Manipulator Reference").

6. Return your attention to the Node Tree, and let's take a look at how our changes are affecting it. Underneath the Transform node, you'll see a scale field. Double click on scale, and the Edit Dialog pops-up. At default, the values here would usually read "1 1 1" for X, Y and Z axes. As you can see, your actions in Step 5 changed the "Y" value of scale.

7. Suppose you'd like to be more precise with your scale changes: click inside the box in the dialog for Y Axis, and change the number to 1.5. Note how the cylinder changes within the View Pane. You can also drag the slider to change the value. The increment of adjustment is controlled by the value within the **Adjustment Increment** box. Using either method, change the value of Y back to "1". Click on OK to close the dialog.


8. It's important to note that in the above example, we weren't changing the actual size of the pillar, but the scale of the group that included the cylinder. **Scale** is a type of **transformation**, which is why the cylinder appears underneath the Transform grouping node. Translation is another type of transformation, and another field within the Transform group. *Double click on translation*, and another dialog opens. *This time, select the X axis and use the slider bar to move the cylinder to the left side of the view pane. Click OK to close the dialog.*

9. Now let's try the same thing visually. *In the View Pane, left click directly onto the cylinder.* Yellow guides appear to allow you to move the object along X or Y. *Drag the cylinder back to the center.* The values of this translation are displayed in the status bar below and updated in the Node Tree automatically.


10. Since we undid the scale of the cylinder, we still need to make it taller to be the shaft of our pillar. *In the Node Tree, expand the Cylinder node to reveal its fields. Double-click on the height field and change the value to "6".* These values, by the way, don't actually represent any specific measurement system, but it's common to think of them as 1 VRML unit = 1 meter. The shaft is now tall again. *Double-click on radius and change the value to ".5".*





11. Be conscious of the fact that anything you add to a scene has to be rendered, so it's best to eliminate any unnecessary surfaces. *The cylinder has fields for both bottom and top. Double-click those and change the values to FALSE to eliminate these surfaces.*

12. It's also important to define, or **DEF**, your work as you create it. Any node can be DEF'd, but not everything must be. In this case, let's **DEF** three things -- the Group itself, the Transform group, and the Shape that represents the shaft. *In the Node Tree, click once on the label Group to select it, pause, then click once again to highlight it. Type the name PILLAR in its place, and hit Enter on the keyboard. Now label the node Transform as PILLARXFORM and Shape as SHAFT.* As we're done with SHAFT for the moment, you can *collapse it by clicking on the minus sign* to make some space in the Node Tree.



13. It's a good idea to periodically save your file. *On the toolbar, click on the Save icon  and name the file "pillar.wrl".* VRML files are named with the ".wrl" extension for easy identification over the Internet.

14. At this point, we've been moving pretty slowly, with detailed explanations. We'll move a little faster now to create the bases of our pillar, and you can use whichever method outlined above that you prefer. *Remember to periodically save your file.*

15. In order to apply transformations globally to our pillar from now on, we'll make the bases themselves children of the parent, PILLARXFORM. *Select the children field under PILLARXFORM in the Node Tree and, from the toolbar, click on Insert Box .* **A Box primitive appears in the view pane.**

16. *Using one of the methods described above, make the box about a quarter of its current height and align it with the bottom of the shaft.* You may find it helpful to either adjust your view within the scene by switching modes. You are currently in **Pick Mode** . By switching to **Pan Mode** , you can adjust your viewing position up/down and right/left. By switching to **Navigation Mode** , you can "walk" back/forth and turn in place. You may also want to either switch the View Pane you're working with to an orthographic (without perspective) view to line objects up more precisely, or **Equalize View Panes** . *Once you have positioned your view, you can return to Pick Mode to select and transform the Box.*


17. **DEF** the Box's Transform as BOX1XFORM.

18. To make it easier to create the second box for the top of the pillar, we can simply copy and paste the first one. *In the Node Tree, select the BOX1XFORM, and click on the Copy icon .* Now, select the children field above it (children of PILLARXFORM), and Paste . The Node Tree now shows another BOX1XFORM. *Change the DEF to BOX2XFORM for clarity.* Because the two Boxes share the same properties, they currently occupy the same space. *Select one of the boxes and transform it up to the top of the pillar shaft.*

19. You may wonder why we went to the trouble of being so precise about the hierarchy of the pillar. Try this experiment. In the View pane, *click on the SHAFT and move it.* Note how the whole pillar moves. That's

because the BOXES, as children, inherit the properties of the parent, the SHAFT. If they were all arranged as equals (or siblings) in the Node Tree, we might find ourselves having to transform each object individually. Later on, we'll see additional benefits to using the hierarchy wisely.

20. Let's add some color to our pillar. *Expand the SHAFT node to reveal appearance, and expand that until you see the Material node. Double-click on Material to bring up the **Material Editor**.* The Material editor has visual tools for each of the fields associated with the Material node. For the most part, you'll work with Diffuse Color, so *double-click on its palette.* The **Color Editor** dialog opens, and you can either visually select a color with the mouse in the color wheel, or enter precise values such as RGB, HSV, etc. *Choose a sandy, light brown color.* Note how your choice is reflected in the Material Editor to show you a shaded and lit version of the color. *Click on OK to exit the Color Editor and OK to leave the Material Editor.* The shaft is now painted.

21. You could repeat these steps for the boxes, but if you want them to be the same color, there is an easier way. *From the toolbar, switch to **Color Mode** , and the Material Painter appears.* It looks much like the Material Editor, but it has a "Paint" and "Acquire" function. *Choose "Acquire" and hold your cursor over the pillar. The cursor changes to an eyedropper, and by clicking you can "pick up" the material of the shaft. Switch to "Paint" and the cursor changes to a paintbrush. Click on the boxes to paint them with the new material then click the close button. Save the finished file.* Now, that may have seemed like a lot of steps to go through to create a simple pillar, but keep in mind that we explored nearly every aspect of the Builder interface while doing so. You may want to start a new file and try again, this time only sticking with the steps you're comfortable with. Some people like to be precise and use the Node Tree as much as possible. Others are more visual and tend to stick to the View Panes. You'll find over time that you'll often work in both, and that certain tasks are easier accomplished in one than the other. As you continue to work through these exercises, you'll use powerful features such as customizable libraries, drag and drop, and instancing DEF'd objects. Not only will these make worldbuilding easier, but they will speed up the process as well.

The Quick Start Tutorial and all Exercises included in this manual have been implemented and are available for you to check you work against. Please refer to the sub-directory of your V-Realm Builder application called, "Worlds\Tutorials\" for the worlds.

[5] Intermediate Worldbuilding

Instancing with DEF and USE

Instancing refers to the use of the VRML keywords **DEF** and **USE**. These keywords are used to simulate "copying" information from one place to another by defining some information with the use of the **DEF** keyword and giving it a name. At a later time in the scene, the user can use the **USE** keyword with the name used to define the earlier information. This has the effect of using all the earlier defined information as though it were all "copied" to the position where the **USE** keyword is located. Because the information is not actually copied but "referred to", the code that was defined by the **DEF** keyword is not reproduced and the scene graph and file are smaller.

Any node can be named and used again.

Scene Binding Nodes:

Background, Fog, NavigationInfo, and Viewpoint are scene binding nodes. Scene Binding nodes offer critical information about the current scene. There can be no overlapping of scene binding information. Any of the information from these nodes can stay the same from scene to scene, but if new information is provided the old information is replaced. Accumulation cannot occur when dealing with scene binding nodes. As you can tell from the node names, these nodes describe the environment we are viewing the scene in and how we view the scene.

Background



Figure 77: Background Node Icon

The Background node is used to specify a color backdrop that simulates ground and sky, as well as a background texture, or panorama, that is placed behind all geometry in the scene. Background nodes are specified in the local coordinate system and are affected by the accumulated rotation of their parents.

Fog



Figure 78: Fog Node Icon

The Fog node provides a way to simulate atmospheric effects by blending objects with the color specified in the **color** field and based on the objects' distance from the viewer. The distances are calculated in the coordinate space of the Fog node. The **visibilityRange** specifies the distance (in the Fog node's coordinate space) at which objects are totally obscured by the fog. A **visibilityRange** of 0.0 or less disables the Fog node. The **set_bind** variable is set to allow the browser to use the fog as default in the browser view or not. The **fogType** variable gives more information on how the Fog is rendered.

NavigationInfo



Figure 79: NavigationInfo Node Icon

The NavigationInfo node contains information describing the physical characteristics of the viewer and viewing model such as **avatarSize**, **headlight**, **speed**, **type**, and **visibilityLimit**.

Viewpoint



Figure 80: Viewpoint Node Icon

The Viewpoint node defines a specific location in the local coordinate system from which the user might view the scene. There can be any number of these viewpoints in the scene but only one can be active at any given time.

Light Sources:

In general, shape nodes are illuminated by the sum of all the lights in the scene that affect them. Both **direct** and **ambient** illumination are considered. DirectionalLight, PointLight, and SpotLight (direct illumination) are the types of light sources that are available in VRML 2.0. All light source nodes contain an **intensity**, **color**, and **ambientIntensity** field. The **intensity** field specifies the brightness of the direct transmission from the light; the **ambientIntensity** specifies the intensity of the ambient transmission of light; the **color** field specifies the spectral color properties of the light emission as an **RGB** value from 0.0 to 1.0. Ambient illumination comes from the light given off by objects in the scene.

A shape node is unlit if the shape's **appearance** field is NULL or if the **material** field in the appearance node is NULL (lighting is OFF). A shape node is lit if the shape's **appearance** field is specified and the **material** field of the appearance node is also specified (lighting is ON).

Light nodes add complexity to the world and increase rendering time.

DirectionalLight



Figure 81: Directional Light Node Icon

The DirectionalLight node defines a directional light source that illuminates along rays parallel to a given 3D vector. The use of a directional light will mimic sunlight by illuminating everything in a scene from the same angle.

PointLight



Figure 82: Point Light Node Icon

The PointLight node specifies a point light source emanating from a specified point in 3D space. A point source emits light in all directions; that is, it is omni-directional. A Point Light is similar to a lamp because it emits light evenly in all directions from a center location.

SpotLight



Figure 83: Spot Light Node Icon

The SpotLight node defines a light source that emits light from a specific point along a specific direction vector and is constrained within a solid conical angle. SpotLights may illuminate geometry nodes that respond to light sources and intersect the solid angle. The solid conical angle is the arch of light being created by the SpotLight. The headlamp of a car is a good example of a SpotLight. Its light starts at the lamp and illuminates a conic shape away from the light source.

Other Intermediate Nodes:

The FontStyle and Text nodes are the last of the intermediate nodes. They define text strings and their appearance in the scene.

FontStyle



Figure 84: Font Style Node Icon

The FontStyle node defines the **size**, font **family** and **style** of the text's font, as well as the direction of the text strings and any specific language rendering techniques that must be used for non-English text. This node can be used with the Text node only.

Text



Figure 85: Text Node Icon


The Text node specifies a two-sided, flat text string object positioned in the XY plane of the local coordinate system based on values defined in the **fontStyle** field. Text nodes may contain multiple text strings. Text strings are stored in visual order. Textures are applied to text as follows: the texture origin is at the origin of the first string, as determined by the justification. The texture is scaled equally in both S and T dimensions, with the font height representing 1 unit. S increases to the right, and T increases up.


Exercise 2: Intermediate Worldbuilding

Now we can move on from the creation of a simple object and start building a complete world. Through the use of Instancing, we can easily reuse objects and keep the file size very small. And nodes that control elements such as Background Color and Lighting allow us to provide an environment to our world. Finally, we'll see how the ability to establish Viewpoints in our world allows us to control the experience being given



to visitors.

1. *Start a new file...* we're going to create a temple-like structure;

2. Click on the **Insert Background** icon ... colors for sky and ground are added to the scene that can be customize later;

3. Click on **Insert Group**  and **DEF** it as `TEMPLE`. Expand it and highlight **children**, as everything we add for the temple at this point will fall under this group.

4. Insert a **Box** that will act as the floor of the temple... *in the Node Tree, expand Box, and set the size to be 20 units wide (X), 1 unit high (Y), and 30 units deep (Z). DEF the Shape Node as FLOOR, and DEF the Transform above it as MAINXFORM;*

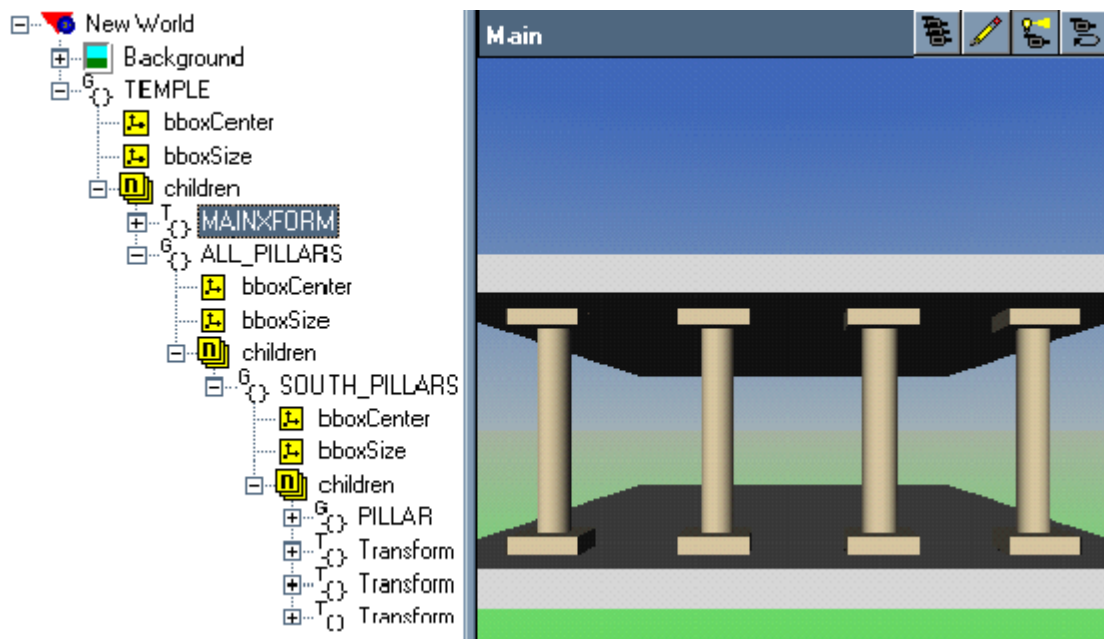
5. Add an "empty" Transform  group as a child to the `MAINXFORM`. Now we can try out our first example of instancing to create a roof. With the children field of this new Transform selected, click on the **Insert USE**  icon. A node is added that says `USE:NOTHING`. Double-click on the "hand" to bring up the "Edit USE" box. Click on the dropdown, and select `FLOOR`. You have now created a new instance of the floor without having to repeat the code. In order to see it, we'll have to raise it away from the floor. In the Node Tree, double-click on the **translation** above `USE:FLOOR`, check off "Y axis" in the "Edit Vector3" box, and use the slider control to raise the roof about 8 units. DEF the whole Transform as `ROOF`. Save your file as `Temple.wrl`.

6. Now let's extend grouping and instancing a little further to quickly build the rest of our temple. Create a new Group called (**DEF'd**) `ALL_PILLARS` as a child to the `TEMPLE` group. Expand the `ALL_PILLARS` group, and create a Group called `SOUTH_PILLARS` as a child to `ALL_PILLARS`. Expand the `SOUTH_PILLARS` group.

7. Open the file from the previous tutorial, `pillar.wrl` (don't worry, you can have more than one world open in the Builder at a time). Select the `PILLAR` group, and copy it to the clipboard. Switch back to your `Temple.wrl` file from the Window menu.

8. Paste the `PILLAR` into the `SOUTH_PILLARS` group as a child. The pillar will appear in the middle of your temple. Use any of the methods covered previously to position your pillar in the front left-hand corner of the temple. You may find it helpful to switch View Panes to position it visually. Adjust the floor and the roof as necessary so that they are aligned with the top and bottom of the pillar. Save the file.

9. As a child to the `SOUTH_PILLARS` group, insert a new Transform. As a child to this, insert a **USE** node. Edit it to `USE: PILLAR`. Adjust the translation field above it to slide the new pillar across the X axis about 4 ½ units to the right. Repeat this procedure two more times, each time sliding a **USE'd** pillar to space them out evenly. You should have a scene and a Node Tree that looks similar to this:






10. Create a Transform called NORTH_PILLARS as a child to ALL_PILLARS. You can now insert a USE node and USE: SOUTH_PILLARS to create the pillars on the far side of the temple. Use the translation field in the Node Tree to send them back along the Z-axis about -24 units.


11. Repeat the first part of the above procedure to create EAST_PILLARS. You'll need to adjust the rotation field 90 degrees around the Y (up and down) axis, so double-click on the rotation field for the EAST_PILLARS transform to bring up the Edit Rotation dialog. As we only need to adjust the rotation around the Y-axis, change the Y value to 1, and the Z value to zero. Set Rotation to 90 degrees (x=0, y=1, z=0, rotation=90). Now, adjust translation of this new group of pillars to position it along the right (East) side of the temple.

12. Create the WEST_PILLARS. This time, however, you can simplify the task with USE:EAST_PILLARS, since the EAST_PILLARS is already rotated. Align the WEST_PILLARS along the left (West) side of the temple. Save the file.

As you can see, Instancing is very powerful. Anything (even another instance) that has previously been DEF'd can be USE'd again. The advantages are the ability to make global changes to groups of objects and a much smaller file size (at this point, the entire Temple.wrl file is only 2.5 kb!).



13. Now we can add some features that allow us to control the experience of a visitor to our world. First, we'll add the **Navigation Info** node near the top of our file. Expand this, and double-click on the **headlight** field. Set it to false. Headlights is a feature in VRML browsers (and in the Builder) which illuminate a scene that has no light settings, but they often overexpose the scene. To see the effect, turn off the headlight  in your View Pane.


14. We'll obviously need to add some light, so insert a **Point Light**  above the NavigationInfo node. To see a representation of the light itself, you can toggle the **Light Manipulators**  to "on". The sphere in the center of the temple is the light you just added. By clicking with your left mouse button onto the "pole" that sticks out of the top of the light, you can drag it up into the middle of the room. Expanding the Point Light node in the Node Tree reveals other settings for this type of light. Double-click on **color**, and set it to a mild, reddish glow (like candle light).

15. Another type of light is **Directional Light**. This is a good way to represent sunlight or moonlight in a scene. Insert a **Directional Light**  below the PointLight, and drag it above the roof for easier manipulation. Note the arrowhead manipulator that extends from the light. Clicking and dragging on it changes the direction the light will radiate. When you've positioned it to your liking, use the Node Tree to adjust the color and intensity to something like moonlight. Save the file, and toggle off the light manipulators (light manipulators never actually appear in a finished VRML file).

16. In our first step, we added a background. Let's customize the sky to make it look more like a nighttime scene. *Expand the Background node, and scroll down to `skyColor`. Expand `skyColor` and double click on the [3] field to show the color of the sky at the horizon. Change it in the color wheel to an orange color, like the sky after sunset. Move up to [2], and change it to a deep purple. Move up to [1], and change it to a dark, purple-blue. Move up to [0], and change it to a light blue, like the moon.* All of these changes are reflected in real-time in your View Pane. If you wish, you can experiment with the fields for `groundColor`, `skyAngle`, and `groundAngle` as well. Background is a node that is best designed visually.

17. Finally, we'll set up a **Viewpoint** that determines where the person entering our world will "land".

Adding a custom Viewpoint in the Builder is very easy. Simply switch to Navigation Mode , and position yourself anywhere in the scene in the Main View Window that you'd like to be the initial view. When you're ready, click on the **Access/Edit Viewpoint**  tool to insert the viewpoint. You must click on a group node for this to be added. To test it, move yourself to a new position in the Main View and open **the Viewer**

Position list  in the Main View Pane. [Viewpoint 0] has been added. Click on Go To, and you'll be taken back to that point. Save your file.

Worlds can have multiple Viewpoints that browsers will interpret. If you're going to have many Viewpoints, you should use the description field of the Viewpoint node to label them. In most VRML browsers, these descriptions will appear in a navigation menu. Note that Viewpoints that are placed beneath a Transform will be affected by any translations, rotations, etc. above the Viewpoint. This can be very powerful when you want to attach a Viewpoint to a particular object, especially an animated one moving through VRML space!

[6] More On Grouping

More Grouping Nodes

As discussed earlier, grouping nodes combine other nodes called children into a cohesive group. Listed in this section are grouping nodes that are more sophisticated than the simple grouping nodes Group and Transform in that they can combine node alteration capability and external or file reference capability.

More Sophisticated Grouping Nodes:

These nodes in some way can change the way the grouped nodes act and appear in the scene. Some change coordinates of child nodes, some replace one child with another child, while some calculate collisions with children.

Billboard



Figure 86: Billboard Node Icon

The Billboard node is a grouping node which modifies its coordinate system so that the billboard node's local Z axis turns to point at the viewer. This will cause the Billboard to always be visible to the viewer. The Billboard node has children which may be other grouping or leaf nodes.

Collision



Figure 87: Collision Node Icon

By default, all objects in the scene can be collided with. Browsers can detect geometric collisions between the user's movement and the scene's geometry, and prevent the user from entering the geometry unless this is desired. The Collision node is a grouping node that may turn off collision detection for its descendants,

specify alternative objects to use for collision detection, and send events signaling that a collision has occurred between the user and the Collision group's geometry or alternate. If there are no Collision nodes specified in the scene, browsers will detect collision with all objects during navigation.

LOD



Figure 88: LOD Node Icon

The LOD node specifies various levels of detail or complexity for a given object, and provides hints for browsers to automatically choose the appropriate version of the object based on the distance from the user. Using LOD allows the author to increase details, such as a more complex shape or texture, only when the user comes within a specified range. This helps keep down the amount of rendering the browser must perform, helping to increase movement speed through the world.

Switch



Figure 89: Switch Node Icon

The Switch grouping node traverses zero or one of the nodes specified in the **choice** field. The Switch grouping node is unique in that it allows one or none of its children be displayed in the scene. The **whichChoice** field of this node specifies which child will be displayed in the scene when the Switch node is reached. It is possible to give this variable a -1 value. This tells the Switch node to display NONE of its children. Child nodes of a Switch node still send and receive information (routes and sensors) even if they are not displayed.

Web based Nodes:

These nodes are mainly used to attach external or local files to a child or scene. They are listed here as web based nodes because their primary use is to give the user the ability to "import" information without having to include all the information directly into the current world file. Like the use of the **DEF** and **USE** keywords earlier, these nodes help to keep the world file size small. Sometimes the spreading out of net requests for information will improve the performance of a file.

Anchor



Figure 90: Anchor Node Icon

The Anchor grouping node causes a URL to be fetched over the network when the viewer activates (e.g. clicks) some geometry contained within the Anchor's children. If the URL pointed to is a legal VRML world, then that world replaces the geometry which the anchor is a member of. The information retrieved from the URL can be any type of Audio/Visual/Text file. If the file cannot be directly displayed, an appropriate interface will be activated to review the information. For instance, if a text file is fetched, the browser will open a text window, if an audio file is fetched the browser will open an audio window, and so forth...

Inline



Figure 91: Inline Node Icon

The Inline node is a grouping node that reads its children's data from a location in the World Wide Web. This node allows the user to input data in his/her world that is not directly input into the .wrl file. This node allows the user to specify a file location on the world wide web or local file for inclusion into the .wrl file. In this way the .wrl file can stay small, and the inline files can be loaded at world startup time instead of having to require all the appropriate information to be coded into the .wrl file. This not only saves time but gives the user a very powerful tool in including objects that have already been created. With the Inline node, the user need just refer to the file containing the information needed to load it into the scene.

[7] Libraries

Drag and Drop

A powerful feature of the V-Realm Builder is its ability to allow the user to select materials, textures, or geometry from either the node tree, view window, or from any of the libraries and essentially "drop" them into place. Imagine having the ability to walk through a furniture store and being able to select furniture to just drop into your home.

The following is true for Materials only and while using the Material Painter. Highlight the material you wish to "acquire", either in the view window or in the library, and just "apply" it to the view window where needed.

One thing to remember is that when you drag geometry into the scene you must drop it into an appropriate node type (like shape, or children node). The same is true for materials and textures (select and drop into an appropriate node type). When you drag objects into the view window, the V-Realm Builder calculates an appropriate place for the node to be inserted. Geometry cannot be dragged and dropped if the node tree is empty, nor can an object be dropped into the view window if it is empty.

This may seem to be a basic explanation of how "drag and drop" works but it is just that simple. You will need to use this tool just a couple of times before you will see how easy it works. The node tree to the left of the screen is a **smart** node tree. It automatically determines if the selected node can or cannot be inserted or dropped at a given point. If you release the object anyway the node tree will insert the object at the closest possible point from where you released the object. Having this ability saves time in debugging world files because the node tree "self-debuggs".

Another **cool** feature brought to you by the "drag and drop" facility is the ability to cut and paste objects. Let us say that you dragged an object from one of the libraries but where you dropped it does not function in the desired way. You can now cut the object from where it was dropped and move it elsewhere and test its functionality there. Not everyone can be a Mozart and create something from scratch with perfection, but with the tools given to you by use of the libraries, "drag and drop", cut and paste, and a smart node tree, you can create worlds that will seem to be perfect the first time.

V-Realm Builder Library Icons Listed



Figure 92: Material Library Icon

The Material Library allows users to include sample materials directly into their worlds. It is also possible to add to the Material Library for use in future worlds. There are several types of libraries available. Users should ensure the proper type of information has been added to each library. Within the Material Library are simple materials or colors. These simple materials can be combined to create more complex materials, or custom materials can be created and added to the library for future use. The benefit of this library concept is that once a material has been created, it can be imported into a new world. It is not necessary to go to another file to open, copy to clipboard, open new file, and paste from clipboard. These steps are eliminated by simply choosing the material from the library and adding it to the scene, using the import or direct drag and drop option.

When pressed the Material Library Icon will open the following dialog box:

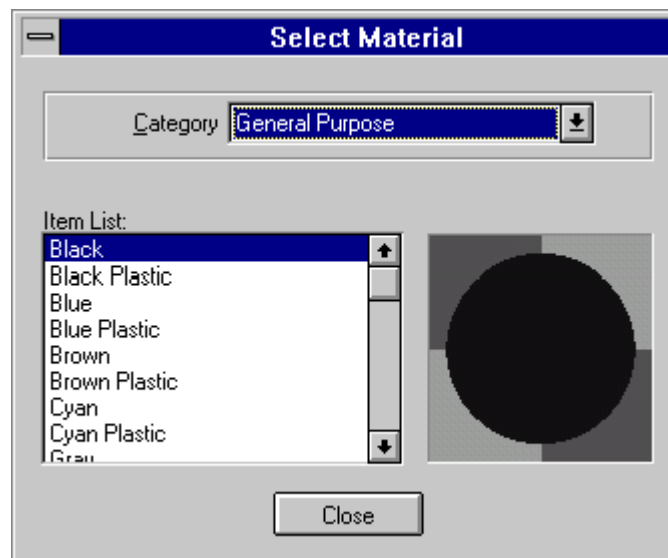


Figure 93: Select Material Dialog Box

This dialog box gives the user access to all the categories and items contained in them. Any of these items can be chosen for application to the scene. The CLOSE button will close the window without applying the material.

When the Edit Material Library option is chosen from the Main Menu the following dialog box is displayed:

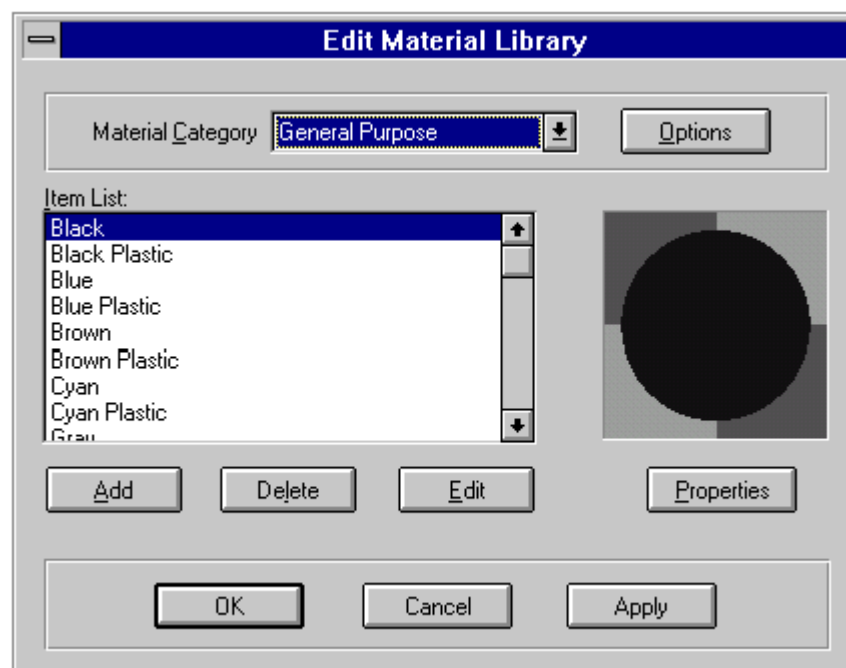


Figure 94: Edit Material Library Dialog Box

The user chooses a category and a material from the scrollable list in the Edit Material Library dialog box for editing. The mouse is used to click on the selection, or the cursor keys can be used to scroll through the list until the desired material is highlighted. As you scroll through or select different materials, the preview window to the right of the dialog box will display the material. When the desired material is selected, the user has the ability to ADD, DELETE, or EDIT the material. The OPTIONS button will open the Material Library Options dialog box and allow the user to maintain the categories used in the Material Library. The PROPERTIES button will open the Material Library Properties dialog box and allow the user to edit the properties of a particular item that has been selected from the library. The APPLY button saves any changes. This dialog box gives total control of all aspects of the material library, and should be used when dealing with the material library directly. When you finish with the library, press the OK button to return to the previous task. If you wish to exit without editing a material, press the CANCEL button.

When the ADD or EDIT buttons are pressed on the Edit Material Library Dialog the following dialog box is

displayed:

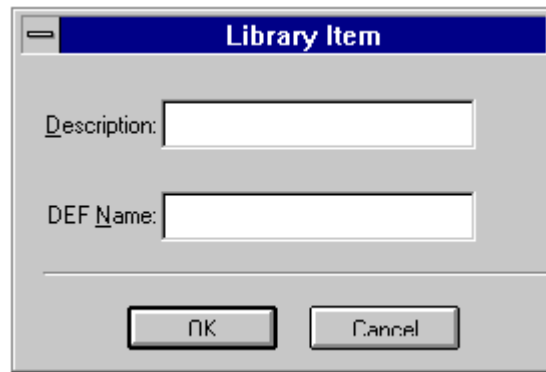


Figure 95: Add Material Library Item Dialog Box

When adding information to a library, the dialog box requires the user to enter a **description** of the item being added. Remember the material will be "created" when the properties of the material have been set. This is done by pressing the PROPERTIES button on the main Edit Material Library dialog box. The dialog box will also have a field for **DEF Name**. This field must contain a name that can be used to identify the node when it is called up from the library. Press the OK button to accept the information and close the dialog box. Press the CANCEL button if you wish to cancel the operation.

When the OPTIONS button is pressed on the Edit Material Library Dialog box the following dialog is displayed:

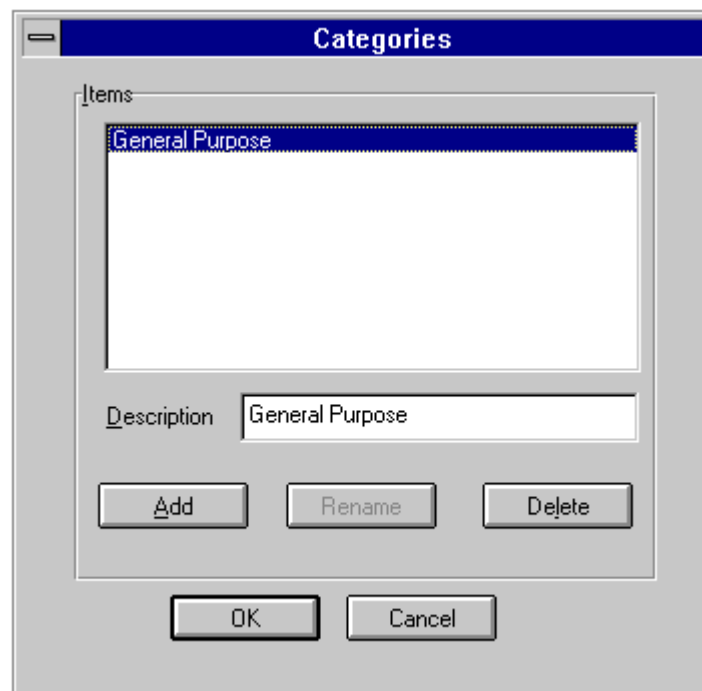


Figure 96: Material Library Categories Dialog Box

The user can maintain the category list for a library from the Material Library Categories Dialog box. The ADD button adds a new category to the list of categories. The DELETE button deletes the highlighted category. **This also deletes all files that are associated with the category.** The RENAME button is used to change the name of any of the categories.

When you are finished with the Categories Dialog box, press the OK button to return to the previous task. If you wish to exit without editing, press the CANCEL button.

When the PROPERTIES button is pressed on the Edit Material Library Dialog box the following dialog box is displayed:

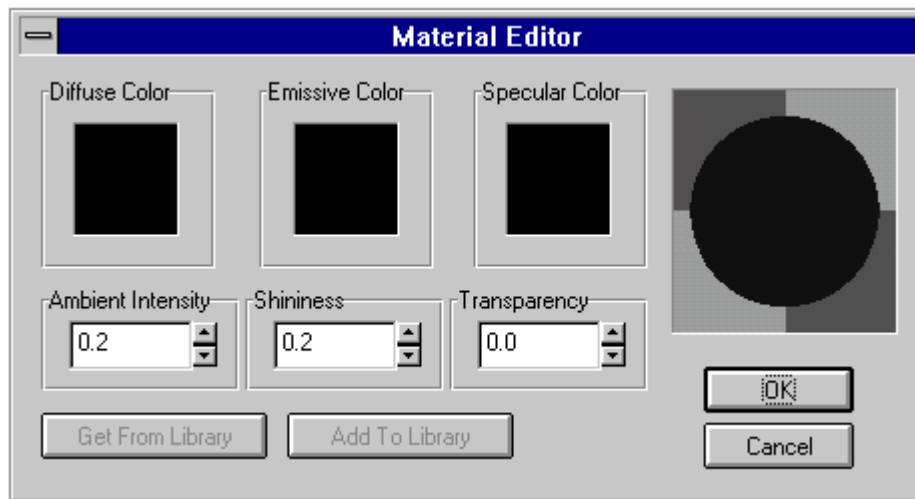


Figure 97: Material Editor Dialog Box

The user can set any of the properties needed to achieve any material needed from the Material Editor Dialog box. The chosen material will be displayed on the colorball to the right of the dialog. The OK button is used to accept the material chosen and the CANCEL button is used to ignore the material that was chosen and exit the dialog.



Figure 98: Object Library Icon

The Object Library allows users to include sample objects directly into his/her world. It is also possible to add your own objects to the object library. There are several types of libraries available: users should ensure the proper type of information has been added to each library. Simple objects such as steps, columns, walls, and tables are found within the Object Library. These simple objects are used to enhance the worlds you create. However, you may wish to add to the library your own special objects for future worlds. The benefit of this library is that once the time has been spent to create an object you only have to "import" the object into a new world. You no longer need to go to another file to open, copy to clipboard, open new file, and paste from clipboard. These steps are eliminated by simply choosing the object from the library and adding it to the scene using the import or direct drag and drop option.

When pressed the Object Library Icon will open the following dialog box:

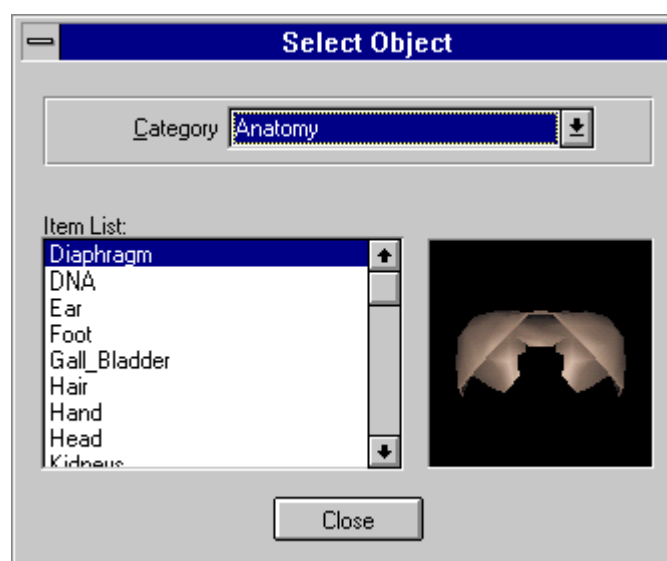


Figure 99: Select Object Dialog Box

This dialog box gives the user access to all the categories and items contained in them. Any of these items can be chosen for application to the scene. The CLOSE button will close the window without applying the material.

When the Edit Object Library option is chosen from the Main Menu the following dialog box is displayed:

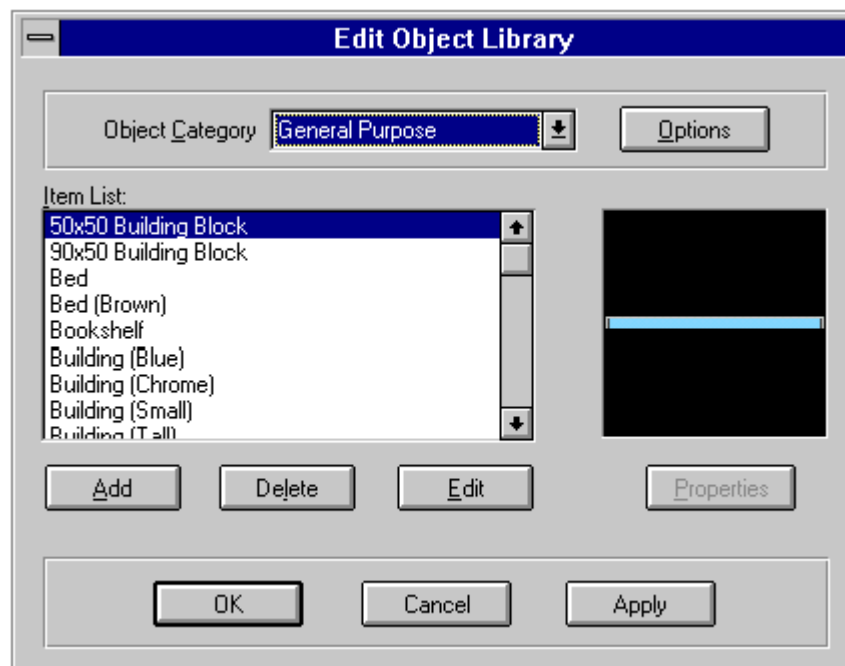


Figure 100: Edit Object Library Dialog Box

The user can choose an object category and an object from the scrollable list of objects from the Edit Object Library dialog box for editing. The mouse is used to click on the selection, or the cursor keys can be used to scroll through the list until the desired object is highlighted. As you scroll through and select different objects, the preview window to the right of the dialog box will display the object. When the desired object is selected, the user has the ability to ADD, DELETE, or EDIT an object. The OPTIONS button is used to open the Categories dialog box. It is used to control the categories and their names within the Object Library. The APPLY button saves any changes. This dialog box gives total control of all aspects of the Object Library, and should be used when dealing with the Object Library directly. When you finish with the library, press the OK button to return to the previous task. If you wish to exit without editing an object press the CANCEL button.

When the ADD or EDIT buttons are selected from the Edit Object Library dialog box the following dialog box is displayed:

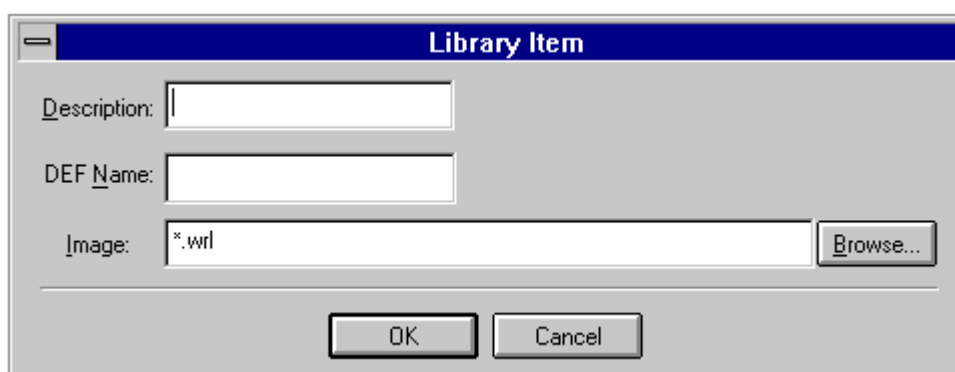


Figure 101: Add/Edit Object Library Item Dialog Box

When adding information to a library, the dialog box requires the user to enter a **Description** of the item being added. This dialog box is also used to edit existing objects. The dialog box also requires a full path name to the file (**Image** field) that contains the item. This is accomplished by typing the full path name into the field provided, or the user can use the BROWSE button to initiate a search for a path to the item. This field has a default (*.wrl) file extension. This is used to display only files of this type. The other required field in the dialog box is for the **DEF Name** variable input. This option allows the user to apply the same node definition repeatedly in the scene graph without having to reenter the appropriate data. This also allows the author to reuse the object by simply providing a **USE** node with the object's **DEF Name**.

The OK button accepts entered data and the CANCEL button exits the editing mode without accepting any entered data.

When the OPTIONS button is pressed on the Edit Object Library dialog box the following dialog box is displayed:

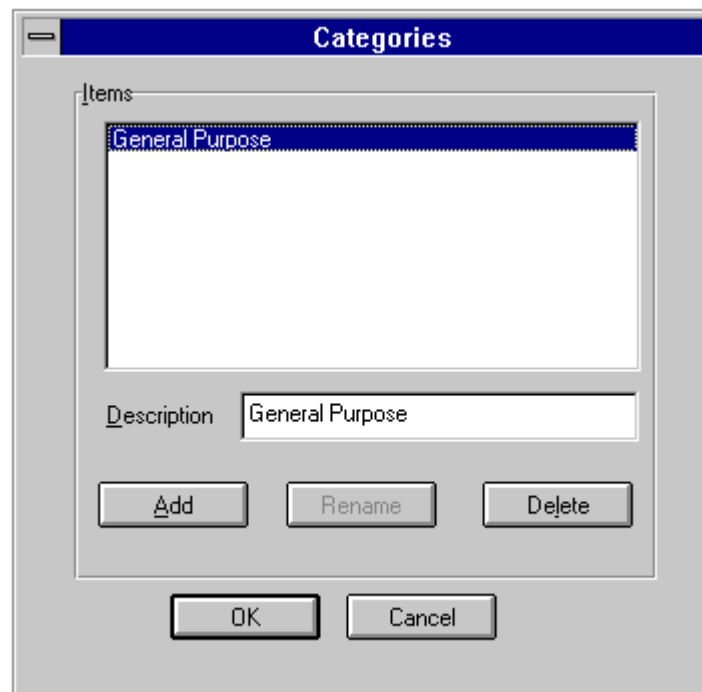


Figure 102: Object Library Categories Dialog Box

In the Object Library Categories Dialog box the user maintains the category list for the library. The ADD button adds a new category to the list of categories. The DELETE button deletes the highlighted category. **This deletes the category and all files that are associated with it you cannot delete the General Purpose Category.** The RENAME button is used for changing the name of any category. When you finish with the dialog box, press the OK button to accept changes and return to the previous task. If you wish to exit without editing the dialog box press the CANCEL button.



Figure 103: Texture Library Icon

The Texture Library allows the user to include sample textures into his/her world. The user can add his/her own textures to the Texture Library for later use. There are several library types available; users should ensure that the proper information has been added to each library. Simple textures and images are found in the Texture Library. These textures can be applied to objects or the user can create custom textures. The benefit is that once time has been spent creating textures, the user only has to "import" the texture into a world. It is not necessary to go to another file to open, copy to clipboard, open new file, and paste from clipboard. These steps are eliminated by simply choosing the texture from the library and adding it to the scene, using the import or direct drag and drop option. Creating textures is time consuming; this allows users to spend time developing good textures that can be reused. Please note that very large textures (150K+) may cause the program to fail.

When pressed the Texture Library Icon will open the following dialog box:

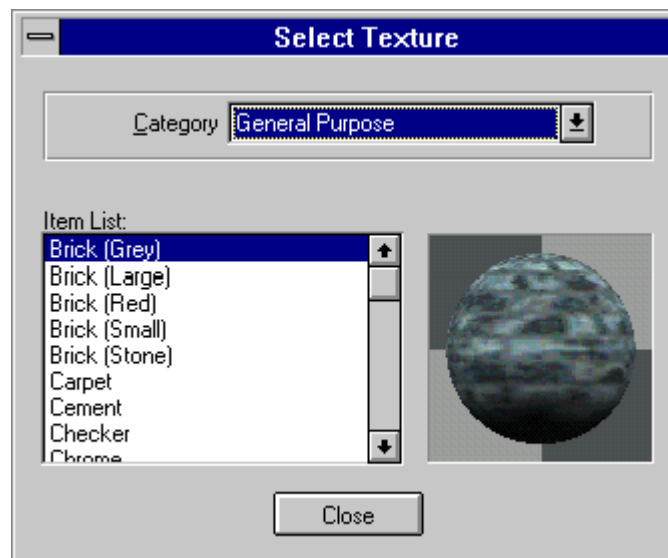


Figure 104: Select Texture Dialog Box

This dialog box gives the user access to all the categories and items contained in them. Any of these items can be chosen for application to the scene. The CLOSE button will close the window without applying the texture.

When the Edit Texture Library option is chosen from the Main Menu the following dialog box is displayed:

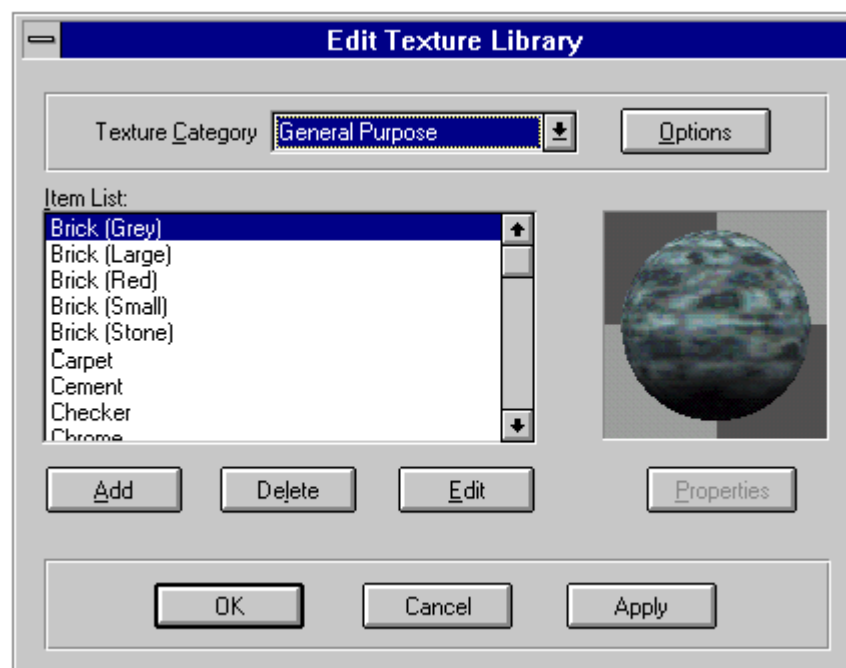


Figure 105: Edit Texture Library Dialog Box

In the Edit Texture Library dialog box you are expected to choose a category and a texture from a scrollable list of textures. The mouse is used to select an item, or the cursor keys can be used to scroll through the list. As you scroll through textures, the preview window to the right of the dialog box will display the texture. When the desired texture is selected you have the ability to ADD, DELETE or EDIT the texture. The OPTIONS button opens the Categories Dialog and gives the user the ability to maintain Texture Library categories. The APPLY button saves any changes. This dialog box gives total control of all aspects of the texture library and should be used when dealing with the texture library. When you finish with the library, press the OK button to accept changes and return to the previous task. If you wish to exit without editing a texture press the CANCEL button.

When the ADD or EDIT buttons are chosen from the Edit Texture Library dialog box the following dialog box is displayed:

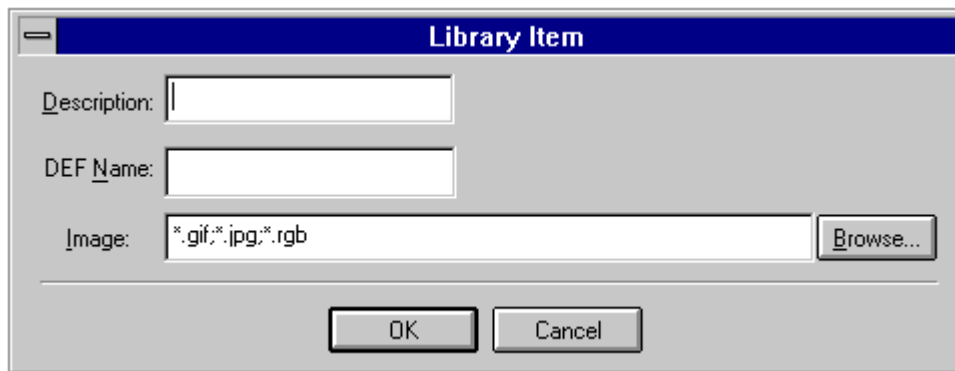


Figure 106: Add/Edit Texture Library Item Dialog Box

When adding information to a library, this dialog box requires the user to enter a **Description** for the item being added. This dialog is also used for editing existing textures. The dialog box will also require a full path name to the file (**Image** field). This is accomplished by typing the full path name into the field provided, or the user can browse for a path to the item by clicking the BROWSE button. This field has a default (*.gif; *.jpg; *.rgb) file extension. These are used to display only these file types during the browse operation. The other required field in the dialog box is for the **DEF Name** variable input. This option allows the user to apply the same node definition repeatedly in the scene graph without having to reenter the appropriate data. This also allows the author to reuse the texture by simply providing a **USE** node with the texture's **DEF Name**. The OK button accepts entered data, and the cancel button exits the editing mode without accepting any entered data.

When the OPTIONS button is chosen from the Edit Texture Library dialog box the following dialog box is displayed:

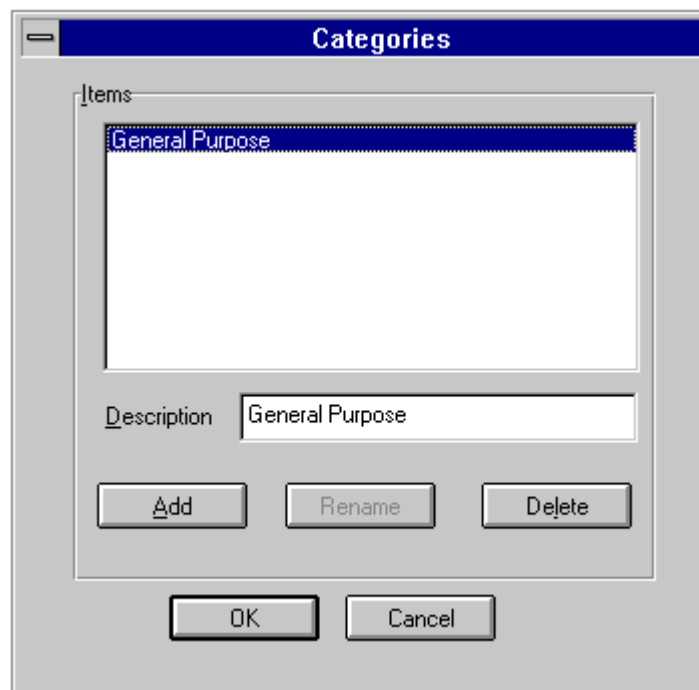










Figure 107: Texture Library Categories Dialog Box

In the Texture Library Categories Dialog box the user maintains the category list for the library. The ADD button adds a new category to the list of categories. The DELETE button deletes the highlighted category. **This deletes the category and all files that are associated with it.** The RENAME button is used for changing the name of any category. When you finish with the dialog box, press the OK button to accept changes and return to the previous task. If you wish to exit without editing the dialog box press the CANCEL button.

Exercise 3: Libraries

In this exercise, we're going to create a not-so-simple tree! The purpose here will be to focus on the special grouping nodes listed in the previous chapter and to use the V-Realm Builder libraries to simplify our work. In the process, we'll see how an object can be highly optimized to render faster in any VRML browser.

1. Start a new file... create a **Collision**  group named `TREE` to hold our entire tree. This will keep visitors to your world from walking through the tree. Expand the node by clicking on the plus-sign.
2. As a child of the collision node, insert a **LOD (Level of Detail)**  grouping node. We'll actually be creating two trees that occupy the same space... a "high resolution" version for when we're close to the tree, and a "low resolution" version for when we're far away. Expand the LOD node, and double-click on the range field. The **Edit MFField** dialog appears. Click *Insert*, then click *Edit Value*. Change the value to 20 units, and click on *OK* to exit out of the *SFFloat* dialog and click *OK* again to exit out of the *MFField* dialog.
3. Select the level field, and add an **Anchor**  group. Anchoring allows any child object of the Anchor to be hyperlinked to another file on the web. We'll return to this concept later in the exercise.
4. Expand the Anchor, highlight children, and insert a **Cone** . This will be the trunk of our tree. Expand the Cone, and set the bottom radius to .4. Set the height to 5.3. Double-click the bottom field, and set to `FALSE` (we'll never see the bottom of the tree, so no need to render it). You should now have a tall, thin cone.
5. Open up the **Material Library** , and select "Brown". Left click inside the color window, and drag and drop it onto the cone in the View Pane. Close the Select Material box.
6. Open up the **Texture Library** , and select "Wood (Brown)". Left click inside the color window, and drag and drop it into the Node Tree onto the field texture. Note that we could have dropped it directly onto the cone itself as we did with Material; in both cases, you can do it whichever way is more convenient. Close the Select Texture box. Note how the items you took from the library have updated in the Node Tree with their own **DEF** names.
7. Before we get too far, we should do some **DEF**'ing of our own. Name the Shape node `TRUNK` and the Transform `TREEHIXFORM`.
8. Now to create some leaves. Instead of creating complex geometry for branches and individual leaves, we can do it all with texturing. First, select children for the `TREEHIXFORM` node. Now, insert an **Indexed Face Set** . An empty manipulator box appears. In the Node Tree, double-click on Indexed Face Set, and a special editor for creating polygons appears. We'll go into more detail later on this tool, but for the time being, just select *Circle* and click on *OK*.
9. In the View Pane, click down on the front of the circle and drag it up so that the top of the trunk is centered with the circle, and release. Now, click on the corner handles for the circle to scale it up larger (about two times as big), and release. Now, holding the Shift key, rescale the circle so that it's wider than it is tall (about three units wide). Note that you can accomplish the same thing by adjusting the scale field of the Transform within the Node Tree.
10. Open up the Texture Library, and select "Tree (Transparent)". Left click inside the color window, and drag and drop it onto the circle. Close the Select Texture dialog. This particular texture is a transparent GIF file. That is, the file has a color selected within it that is set to render as transparent. In VRML, you can either "decals" a texture onto an object (as it is now), or "mask" an object. In this case, we want the transparent section to mask out the polygon underneath. In order to do so, you want to delete the Material node currently associated with the polygon. In the Node Tree, find the Material node for the Indexed Face Set, select it, and delete it with the keyboard Delete key. The polygon disappears, but the leaves texture remains! (Not until the file is saved.)
11. **DEF** the Transform as `TOP`, **DEF** the Shape as `LEAVES`, and save your file.

12. Before we do any further work on this "hi-rez" tree, we'll use it to create our "low-rez" version. *Select the level field of the LOD, and create a **Billboard**  grouping node.* Anything within a Billboard node can be set to face any direction along an axis (or axes) at all times. By default, items within a Billboard group are set always to spin around their Y axis and face the viewer on the ground. Now, *copy the entire group for TREEHIXFORM, and paste it as a child of the Billboard.* Now, whenever we are 20 or more units away from the tree, this simple version of the tree will be the one rendered, and it will always face us. In this way, the browser never has to expend a lot of resources rendering a tree to which we never get very close. *Rename this version of the tree TREELOXFORM.*

13. Right now, our hi-rez and low-rez versions are nearly identical, so let's make the hi-rez version more complex. We'll do this by adding some more leaves facing in other directions to give the tree some dimension. *Add a Transform as a child of TOP, and as a child of this Transform, add a **USE** node... set it to USE:LEAVES. Now, change the rotation field in the Node Tree of this new set of leaves to rotate around the Y axis about 60 degrees (x=0, y=1, z=0, rotation=60). Repeat this step one more time, this time rotating it about 120 degrees. Save your file.*

You now have a tree that is very efficient! If you back away from the tree very slowly, you'll notice the switch to the "lo-rez" version at 20 units away. No matter where you are in the scene, the tree is always facing you. And when you're up close, the tree looks fairly realistic. You could easily build an entire forest with trees like this (USE:TREE), without worrying about file size or polygon count. In fact, the best thing to do at this point is add it to our V-Realm Builder Object Library so that we can use it again in the future.

14. *In the Node Tree, select the TREE collision group at the top of the file. Right click on the Node, and select "Add to Library" from the pop-up menu. Give it a description like LOD_TREE, and put it in the "Landscapes" category.* Now when you access the object library in the future, you can retrieve this tree from the Landscapes category, and drag and drop it into your scene whenever needed.

15. Last step. Earlier on, we put the hi-rez version of the tree inside an Anchor group. Now we can set up the hyperlink we want for the tree. *Expand the URL field, double-click on S, and type in <http://www.1stresource.com/t/treedoc/>. For description, put in The Tree Doctor. Parameter is for special browser information, but most VRML plugins will recognize the HTML "target" command. In this case, enter target=_new, and the VRML browser will open up a new instance of the web browser to display the website (this is primarily useful with HTML frames).*

Now you can test the file out in your web browser. *Open up the file, walk around the tree to see how LOD works, and when you're up close, you can click on the tree to take you to an actual website.*

Needless to say, not all of the steps are necessary to make a tree! But keep in mind the use of these techniques, especially when you're looking to create complex scenes that could get too large or difficult to render. These are some of the same techniques that 3D game designers have to keep in mind every day. And remember that the libraries are yours and completely customizable. Save your own objects, textures and materials in the libraries so that with a click and a drag and drop, you have instant access to your creations in the future.

[8] Specialized Editors

Complex Geometry Nodes:

Geometry nodes must be contained within a shape node in order to be visible by the user. The shape node contains exactly one node in its geometry field. This node must be one of the following nodes: ElevationGrid, Extrusion, IndexedFaceSet, or one of the primitives. Several of these nodes contain Coordinate, Color, Normal, and TextureCoordinate nodes as geometric property nodes. The application of materials, textures, and colors are different for each geometric node.

The ElevationGrid, Extrusion, and IndexedFaceSet nodes each have fields that provide hints about the shape -- whether it contains ordered vertices, whether the shape is solid, and whether it contains convex faces. These fields are **ccw**, **solid**, and **convex**. The **ccw** field indicates whether the vertices are ordered in a counterclockwise direction when viewed from outside (TRUE). If the order is clockwise the value will be

FALSE. The **solid** field indicates whether the shape encloses a volume (TRUE). In this case the hint allows the browser to perform backface culling. If nothing is known about the shape, the solid field will be FALSE and backface culling cannot be performed. Backface culling is the action of the rendering engine where unseen polygons are not rendered because they are not in view. The **convex** field indicates whether all faces in the shape are convex (TRUE). If nothing is known about the faces then this field value will be FALSE.

The **creaseAngle** field affects how default normals are generated. The angles between edges are compared to the **creaseAngle**, and if they are less they are smoothed; if not, the edge will have a sharp crease.

Double-left clicking on many of these nodes will open an editor for that node type.

ElevationGrid



Figure 108: Elevation Grid Node Icon

The ElevationGrid node specifies a uniform rectangular grid of varying **height** in the XZ plane of the local coordinate system. The geometry is described by a scalar array of **height** values that specify the height of a rectangular surface above each point of the grid.

Extrusion



Figure 109: Extrusion Node Icon

The Extrusion node specifies geometric shapes based on a 2D cross section extruded along a 3D spine. The cross section can be scaled or rotated at each spine point to produce a wide variety of shapes.

IndexedFaceSet



Figure 110: Indexed Face Set Node Icon

The IndexedFaceSet node represents a 3D shape formed by constructing faces (polygons) from vertices listed in the **coord** field.

IndexedLineSet



Figure 111: Indexed Line Set Node Icon

The IndexedLineSet node represents a 3D geometry formed by constructing polylines from 3D points specified in the **coord** field. Normally the lines produced here will not be interconnected. V-Realm Builder can read, display, and write out this node type, but cannot create or edit it.

PointSet



Figure 112: Point Set Node Icon

The PointSet node specifies a set of 3D points in the local coordinate system with associated colors at each point. The **coord** field specifies a Coordinate node (or instance of a Coordinate node). If the **coord** field is NULL, then the PointSet is empty. PointSets are not lit, not texture mapped, or collided with during collision detection. V-Realm Builder can read, display, and write out this node type, but cannot create or edit it.

V-Realm Editors

Double-clicking on some of the nodes like ElevationGrid, Extrusion, IndexedFaceSet, IndexedLineSet, and PointSet will open a special editor which will assist you in inputting and modifying data for these nodes.

Elevation Grid Editor

The ElevationGrid is an adjustable grid extending along the XZ plane that is useful for creating such objects as platforms and landscapes.

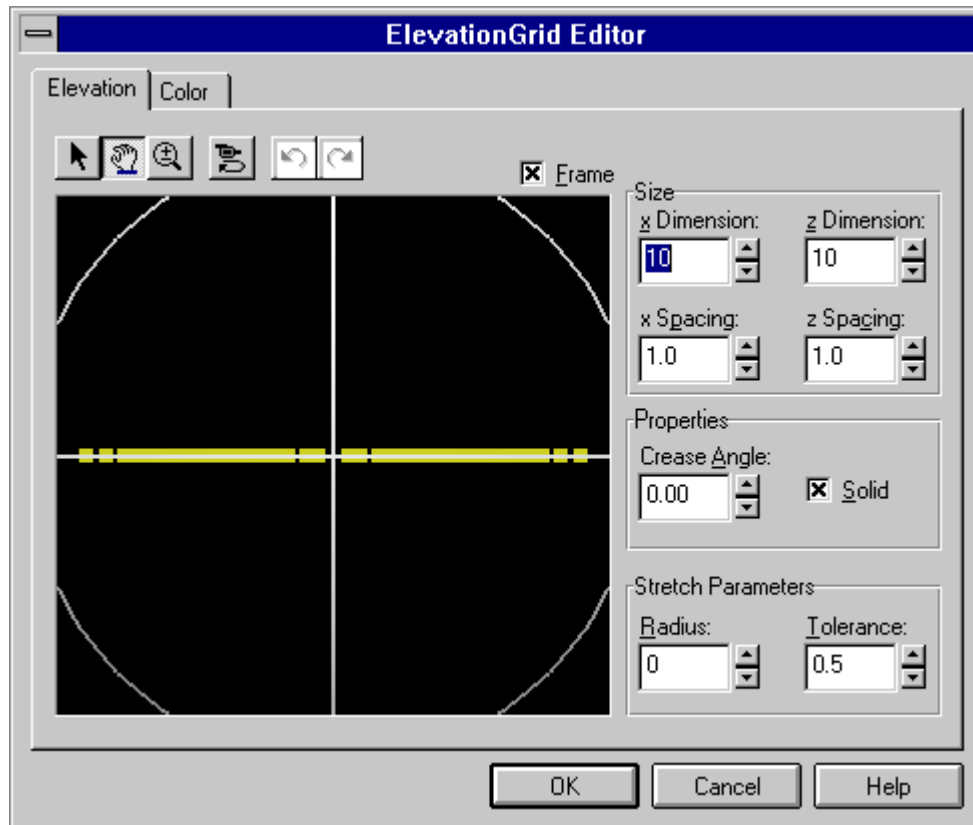


Figure 113: Elevation Grid

Left clicking on the ElevationGrid node in the node tree will open the ElevationGrid Editor dialog box. The main dialog box contains the OK, CANCEL, and HELP buttons as well as two tabs for the **Elevation** and **Grid Color** property sheets. The **Elevation** property sheet is the default.

The grid displayed here is hard to see; you may need to manipulate it by hand. When you do you will see something like this:

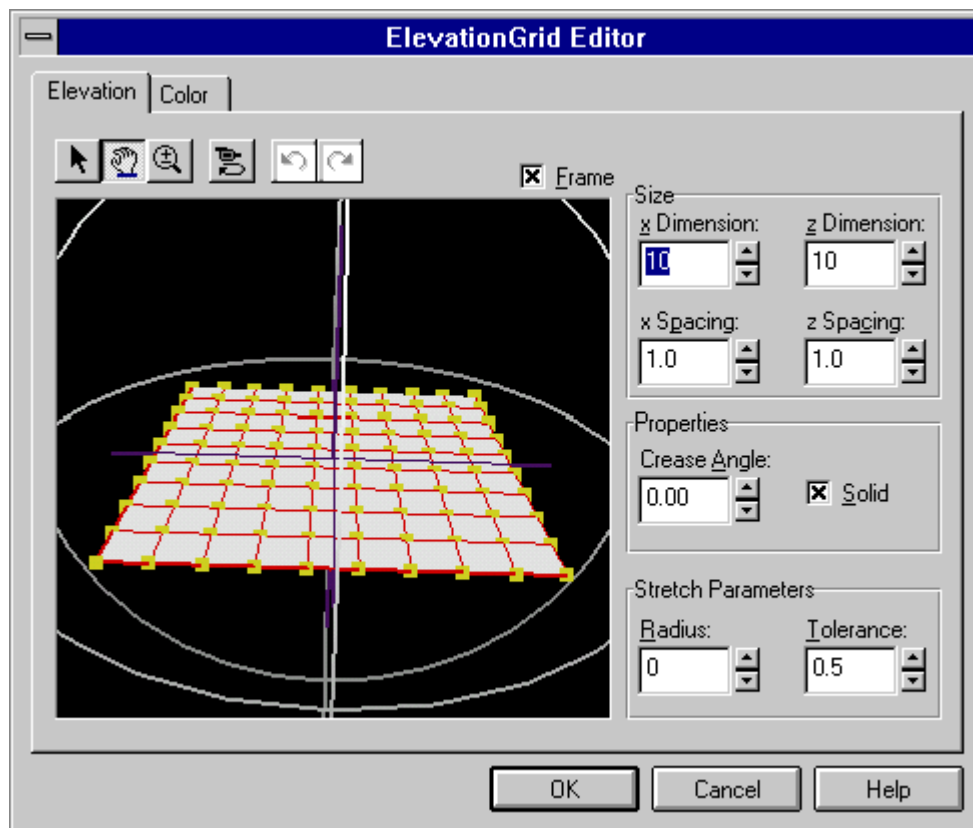


Figure 114: Rotated Elevation Grid

The **Elevation** property sheet allows the user to edit the ElevationGrid's **X** and **Z Dimensions**, **X** and **Z Spacing**, **creaseAngle**, **Solid**, **Radius**, and **Tolerance** Variables, as well as the **Frame** feature, and the ElevationGrid surface. Increasing and decreasing the **X** and **Z Dimensions** will change the size of the ElevationGrid. The **X** and **Z Spacing** controls the amount of space between the vertices that make up the grid. **creaseAngle** is used for creating sharper or more rounded appearing angles throughout the entire grid. **Radius** and **Tolerance** control the amount of influence manipulating a point or series of points has on the surface of the grid. The **Frame** feature refers to the visible net that overlays the ElevationGrid in the Editor view window and allows the user to locate, pick, and manipulate the vertices that make up the grid. **Solid** describes whether the ElevationGrid is solid or not.



Figure 115: Elevation Property Page Buttons

The buttons on the **Elevation** property sheet operate Pick mode, Model mode, Zoom In/Out, Reset Viewer, Undo, and Redo. Pick mode is the default manipulator when the sheet opens and uses a 3D arrow pointing in opposite directions to indicate the last vertex chosen. Clicking and holding the 3D arrow allows the user to raise or lower the surface of the ElevationGrid. Holding down the control key while clicking on a series of vertices allows the user to alter specific sections of the surface. Model mode is used here for rotating the ElevationGrid into better positions for editing and is especially useful when many vertices are manipulated to various heights. The ZOOM IN/OUT feature works by left clicking on a spot for closer inspections, or right clicking for an overall view. Use of the RESET VIEWER button will reset the ElevationGrid to the last accepted or previously saved changes. UNDO and REDO perform the task they describe.

HINT: Keep in mind that polygon count increases along with size. If you want to create an elevation on a notably larger scale than the default 10 X 10, it would probably be best to start providing a series of elevation grids so that when your world is viewed in a browser, the browser could speed rendering by culling things not visible.

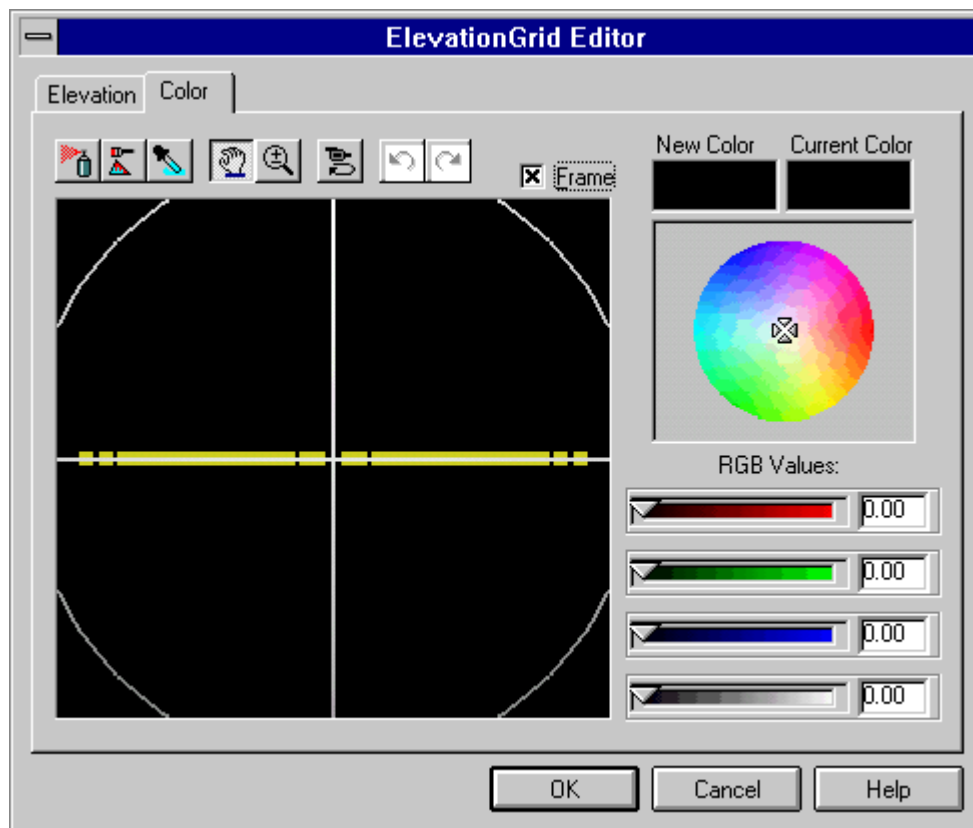


Figure 116: ElevationGrid Color

The **ElevationGrid Color** property sheet allows the user to apply color to the Elevation Grid.





Figure 117: Grid Color Property Page Buttons

Paint mode is used to apply the chosen colors to the grid. Holding down the left mouse button in the Paint mode will paint a continuous line. Model mode again is used to rotate the ElevationGrid for easier editing. Paint vertex will apply paint to the intersections or vertices of the grid and gives a softer and more blended look. Paint Face gives crisper straight lines by applying the paint to faces or areas between vertices. Zoom In/Out offers different levels of viewing inspection. Colors can be chosen by moving the color cursor to the appropriate color on the color wheel, adjusting the intensity of any or all of the three RGB sliders, or by entering an exact color value into the RGB fields. RESET VIEWER, UNDO, and REDO buttons are also available. They perform the task they describe.

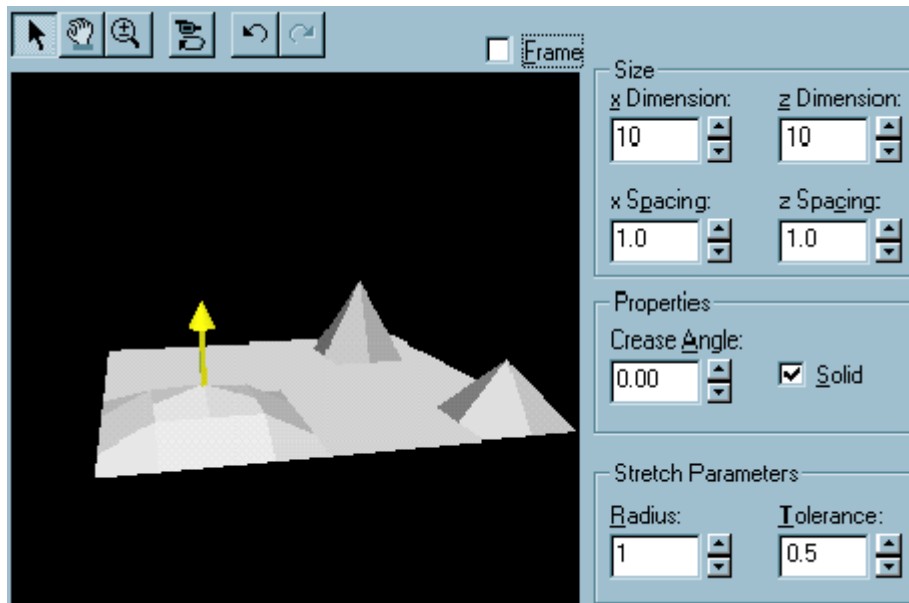
HINT: Applying colors with the grid frame off can increase the speed of applying the colors to the grid.

Exercise 4: Elevation Grid

1. Using the **ElevationGrid Editor**, we'll create a hilly landscape for our temple. *Start a new file.*
2. Click on the **Insert Elevation Grid** . The Node Tree will update, and the View Pane shows a large flat plane selected.
3. Double-click on the *Elevation Grid* icon in the Node Tree, and the ElevationGrid Editor opens. The scene view inside the editor is similar to the Main View Pane, except that you're looking at the grid from straight on. Using the *Model mode* within the editor, "grab" the plane by clicking and dragging down, and tilt it towards you a bit.

4. Switch to Select mode  in the editor, and left-click down on one of the points (known as **vertices**) inside the grid near the far right corner ([see below](#)). Keeping the mouse button down, drag up to raise the vertex off the surface. You may notice that the scene is updated in the View Pane underneath as well. Repeat this step with a vertex closer to your viewing position.

5. Adjust the **Stretch Parameter Radius** setting to 1. Near the front left side of the grid, select a vertex and drag it up. Notice how increasing the radius increases the influence of your adjustment on the grid. Also, note in the figure below that the **Frame** option can be turned off to make viewing the effect easier.



6. Switch to the **Color Tab** within the Editor. In this part of the Editor, you can paint individual faces of the grid, or paint the vertices themselves to produce very unique effects (**NOTE: Changes made to color here override any settings made in the Material Node for the Elevation Grid... if you want only one diffuseColor or a texture for your entire grid, use the choices available within the Appearance node**). In the Color wheel, select a dark green shade and click on the Paint Face icon. Paint the individual faces with this color on the grid. If necessary, you can temporarily switch to Model mode to spin the grid around to make sure you're painting all the faces, or use the Zoom tool to inspect your work up close (left click=zoom in, right click=zoom out).

7. Click on **Paint Vertex**, and you'll notice the landscape changes again. Continue to paint the whole grid green. Then, switch to a whitish shade in the color wheel, and paint the vertices at the caps of the "mountains" you've created.

8. Switch back to the **Elevation Tab**, and switch off the Frame box. Use the dialer for **Crease Angle** and adjust the number to about 2. You'll see that the sharp edges of the grid are now smoothed out. Click on OK to exit the Editor.

9. Now, open the temple.wrl file and copy and paste the grid you just created to the bottom of the Node Tree for temple.wrl. **DEF** the Transform for the landscape as GRID.

10. Rescale the grid about 20 times on all three axes, and position it so that the temple sits on the ground in the middle, between the mountains. Save your file.

You can, of course, create an Elevation Grid within the temple file itself and save a few steps, but the process is easier to illustrate in this fashion. In the future, when you work with the Elevation Grid node as part of a world, note that only the Grid itself will be displayed in the Editor.

Extrusion Editor

The Extrusion Editor takes an input shape (2D) and elongates it in a continuous manner. The cross section of the shape can be selected from the list of primitive shapes that include a triangle, sphere, and square or

alter any of these for a custom shape. The spine determines if this "extruded" 2D shape will be continuous or terminating. A circular "extrusion" will terminate when the two ends meet, but a linear shape can be extruded indefinitely but will extend only to the endpoints of the line segment. The scale determines how fat or thin the shape will be. We can also twist the shape when we desire an irregular extrusion. The Extrusion Editor can create almost any shape imaginable, all that is required is a good mental picture of objects in 3D.

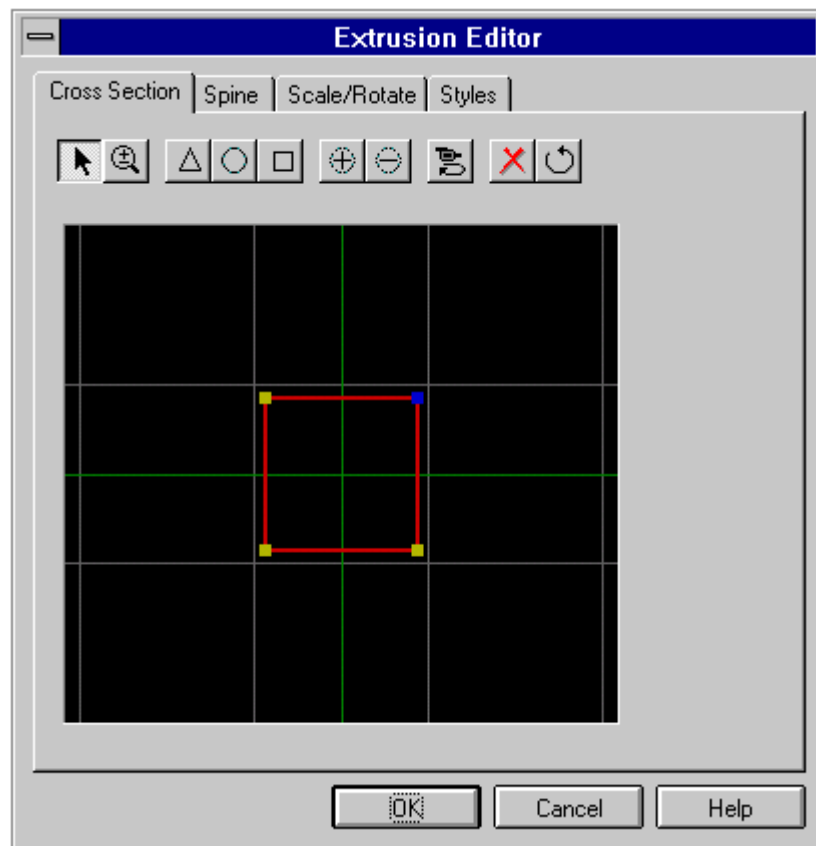


Figure 118: Extrusion Editor Cross Section Property Page

This is the default page displayed when the Extrusion node is selected for edit. There are three other property pages that can be chosen from this main property page. They are Spine, Scale/Rotate, and Styles. The buttons provided for this page are listed below:



Figure 119: Cross Section Button Bar

These buttons allow the user to select a beginning shape for the extrusion and add or delete vertices from the shape. Also there are buttons for RESET VIEWER, REDO, and UNDO.

The Spine Property page is the second of the property pages available in the Extrusion Editor. It is listed below:

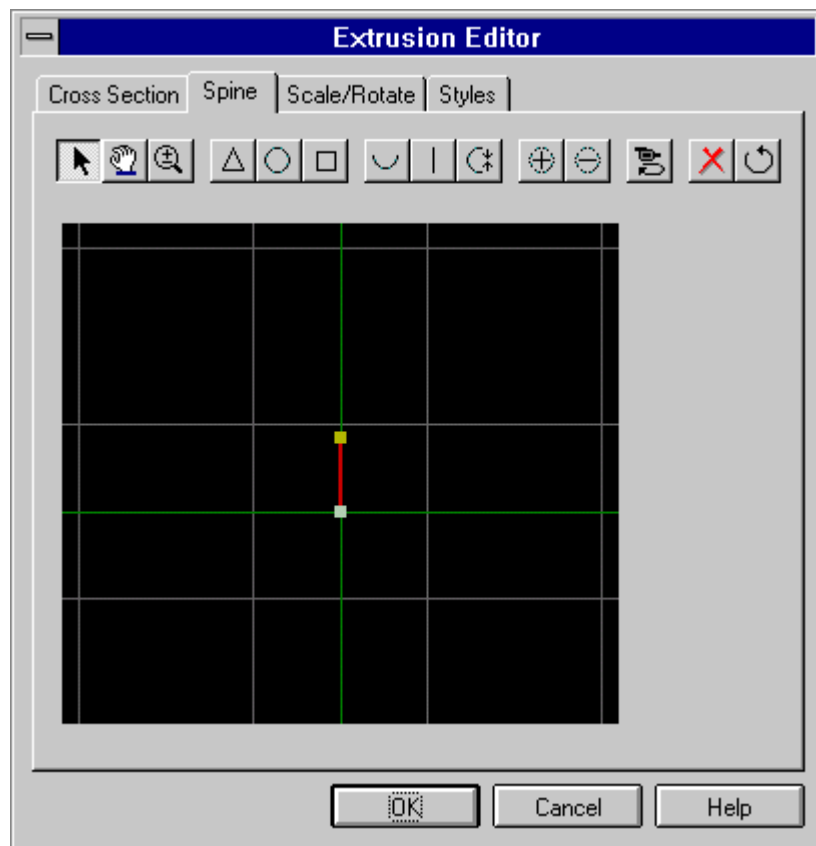


Figure 120: Spine Property Page

This property page gives the user the ability to choose the overall shape of the extrusion. The buttons provided for this page are listed below:



Figure 121: Spine Property Page Button Bar

This button bar allows the user to choose the shape of the extruded shape by choosing the initial shape from one of the primitive shapes or by specifying it vertex by vertex. The ability to add or delete vertices from a shape is also provided. Also there are buttons for RESET VIEWER, REDO, and UNDO.

The Scale/Rotate Property Page is shown below:

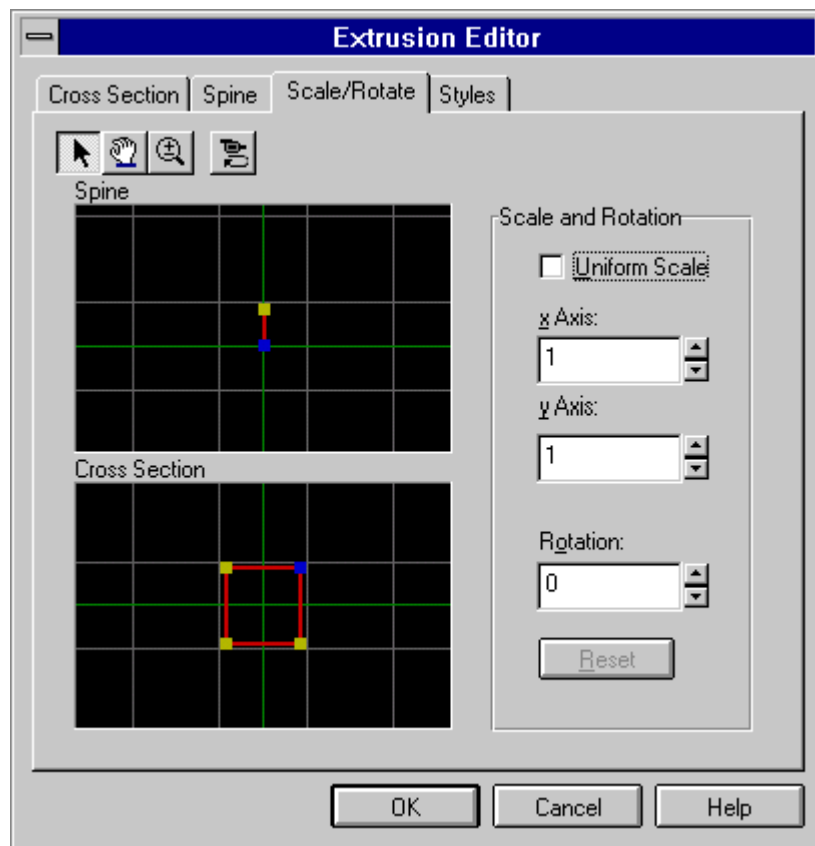


Figure 122: Scale/Rotate Property Page

This property page allows the user to select x, y axis changes, rotation and scale alterations for the extruded shape. There is one button bar available which gives the user the ability to select, manipulate, RESET VIEWER, and zoom in or out. This button bar is shown below:



Figure 123: Scale/Rotate Property Page Button Bar

The Styles Property Page is the last of the available property pages. It gives the user the ability to select the values for **EndCap**, **StartCap**, **Solid**, **Convex**, **CounterClockwise**, and **CreaseAngle**. **Convex**, and **CounterClockwise** cannot be changed; these options are grayed out. These variables control how the shape looks and is rendered. The Styles Property page is shown below:

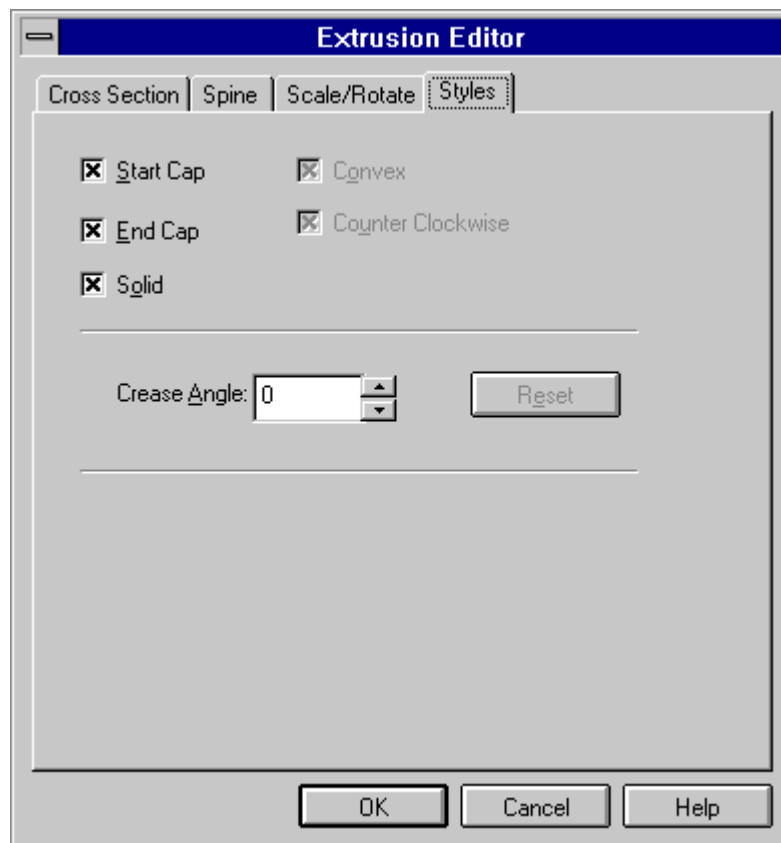



Figure 124: Styles Property Page

Exercise 5: Extrusion Editor

1. Using the **Extrusion Editor**, we'll create a fountain that sits in the center of our temple. This time, we'll do it in the same file. Open `temple.wrl`, and collapse the hierarchy (by clicking on the minus signs next to each node) so that only the highest items show. Select `GRID` so that the cursor is active at the bottom of the Node Tree.

2. Click on the **Insert Extrusion** . The Node Tree will update with a base **Extrusion** at the bottom, and the View Pane shows a small object in the center of the temple. Using Navigation Mode, move yourself into the center of the temple so that you can see what your working on. Switch to Pick Mode, and raise the Extrusion out of the floor.

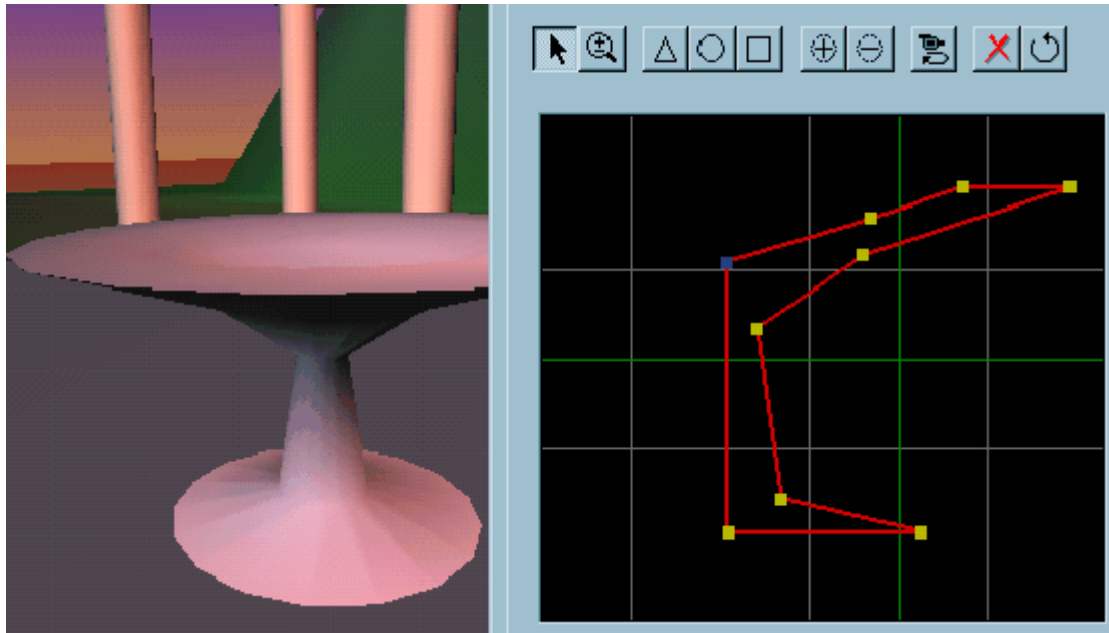
3. Double-click on the Extrusion icon in the Node Tree and the Extrusion Editor opens. The scene view inside the editor shows a red square, representing a top down view of a cross section. Flip through the different tabs to get a better idea of how Extrusion Editor works. You can even make some changes if you like and see the resulting shape reflected in the View Pane underneath.



4. Switch over to the **Styles** tab. Make the shape smooth by changing the **Crease Angle** value to about 2, uncheck **Start** and **End Caps** (they're inside and won't be seen), and check on **Solid** (because we only need render the outside). Uncheck the box for **Counter-Clockwise**.

5. We don't need to change the Scale or Rotation of any points, so switch to the **Spine** tab and click on "Creates a Circle".

6. Switch to the **Cross Section** tab, and click on the "Creates a Square" button. Using your left mouse button, click down on the right side line of the square to create a new vertex. Clicking down and dragging on a point reshapes the cross section of the Extrusion. Compare what you're doing in the editor to what is displayed in the View Pane below, as it is updated in real-time.

7. You can edit the Node Tree even when in the Editor. *Double-click on the rotation field for the Extrusion's Transform, and change the rotation to spin around the X axis -90 degrees* $x=1.0$, $y=0$, $z=0$ rotation = -90). Click OK.



8. Return to the Editor, and finish working on your cross section. Add and maneuver vertices until you have a cross section that resembles the figure above. If you need to Undo  a change or remove a Vertex , both options are available to you in the Editor.

9. When you are finished, click on OK to exit. You may need to do some fine tuning to position your fountain on the temple floor. Save the file.

The Extrusion Editor is a great tool, capable of producing many complex or unusual shapes. Experiment with it and browse through other objects included with the Builder (such as gear.wrl and alien.wrl) to see the kinds of advanced 3D modeling that can be done with extrusions.

Face Set Editor

The Face Set Editor allows the user to create IndexedFaceSets made up by applying a material to each plane made by a set of vertices. The initial geometry can be any of the primitives and vertices added or subtracted, until the desired number of vertices is reached. In this editor, individual vertices or groups of vertices can be moved to create the wanted shape. It may take some trial and error before you get comfortable with the way vertices and line segments are manipulated to obtain particular objects. As with all the editors, the Face Set Editor has the ability to model the shape in 3D giving the user the ability to see the shape from any angle, and the zoom feature allows the user to see the shape from close up or far away. There is also a color editor for the Face Set Editor which allows the user to paint the shape. There are options for painting either by vertex or by face. If the user selects by face, each face can have it's own color. If the user selects by vertex, then each vertex will have it's own color, and the color is interpolated from the color at one vertex to the color at the next vertex. This editor was intended for editing and creating **simple** Indexed Face Sets.

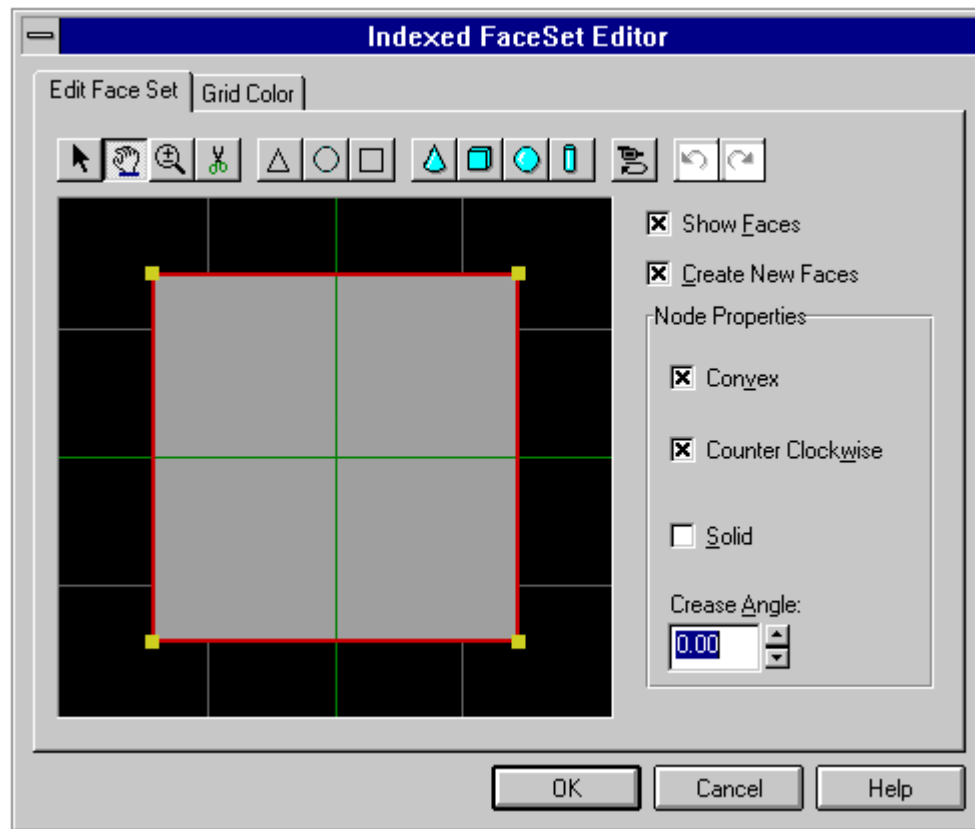


Figure 125: Edit Face Set Dialog Box

The Edit Face Set Dialog box is the default property dialog box to be displayed when the user chooses to edit an IndexedFaceSet node. The fields controlling the display of faces, convex, ccw, solid, and crease angle describe the IndexedFaceSet. The display window gives the user the ability to modify and manipulate the geometry until it takes on the desired shape. OK, CANCEL, and HELP buttons give the user the options to close the property dialog while saving, just exit the property dialog without saving, or get help with the dialog.



Figure 126: Edit Face Set Dialog Toolbar

This toolbar gives the user the ability to create and customize any FaceSet. The tools available are triangle, circle, square, cone, box, sphere, cylinder, select, manipulate, zoom, cut, Reset Viewer, redo, and undo.

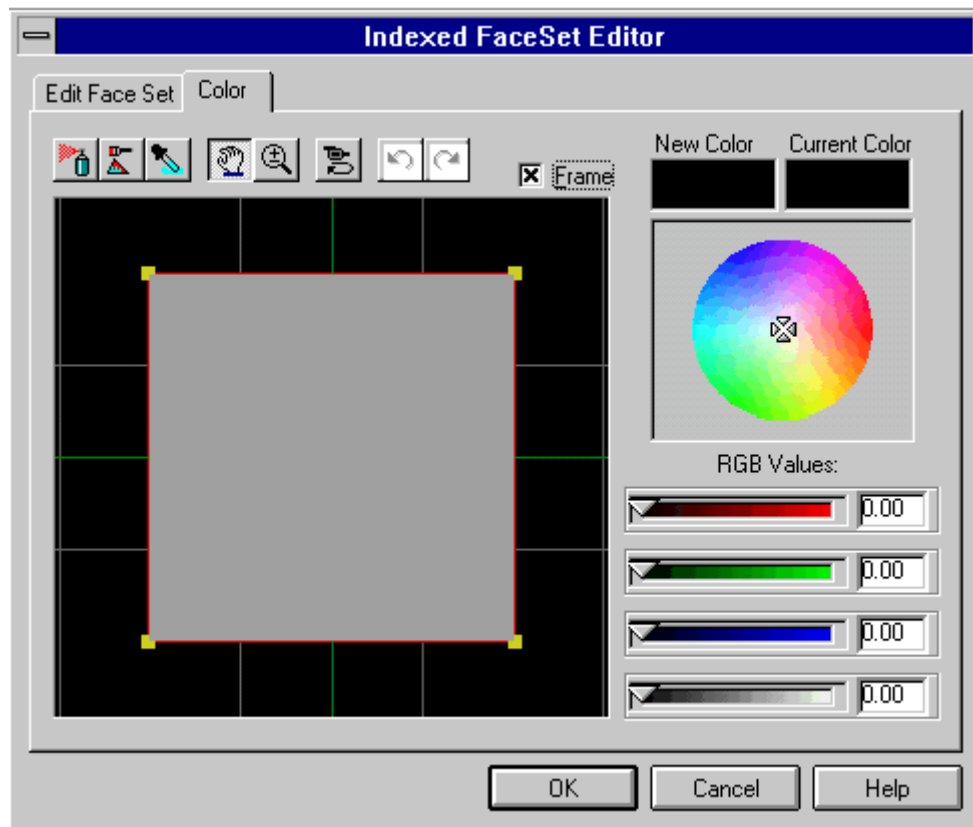


Figure 127: Edit Face Set Color Dialog Box

The Edit Face Set Color Dialog box allows the user to choose and/or set colors for the faces of the IndexedFaceSet shape. The color can be created using the color editor or the color can be selected from the material library. There is also an option for frame which allows the user to display the vertices that frame the object.





Figure 128: Edit Face Set Color Dialog Toolbar


This toolbar gives the user access to the tools that are needed to customize the color and viewpoint of the FaceSet. Tools available are spray painter, paint face, acquire color, manipulate, zoom, Reset Viewer, undo, and redo.

Exercise 6: IndexedFaceSet


1. Using the Indexed FaceSet Editor, we'll create the water for our fountain. *Open temple.wrl, and collapse the hierarchy so that only the highest items show. Select FOUNTAIN so that the cursor is active at the bottom of the Node Tree. Add an empty Group node, and expand it. DEF the Group as WATER and select the children field.*

2. Click on **Insert Indexed Face Set**  on the toolbar. The Node Tree will update with an empty box at the bottom, and the View Pane shows an empty box at the bottom of our extruded fountain. *Using Navigation Mode, move yourself into the center of the temple so that you can see what you are working on (you may want to add a Viewpoint here so that you have easy access to it via the Viewer Position List  in the View Pane). Switch to Pick Mode, and raise the Indexed Face Set (IFS) out of the floor and above the fountain.*





3. Double-click on the Indexed Face Set icon in the Node Tree and **the Indexed FaceSet Editor** opens.

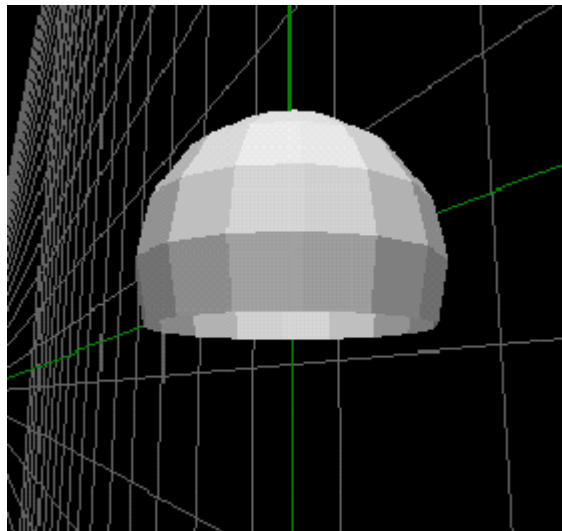
The scene view inside the editor shows a polygon, representing a top down view of a square. *Change the shape to a circle* , and check the Solid selection "on" (this will keep the other side from being rendered). Click on OK to exit. DEF this Transform as POOL.

4. Rotate, translate and scale the circle so that it rests like a pool of water on top of the fountain.

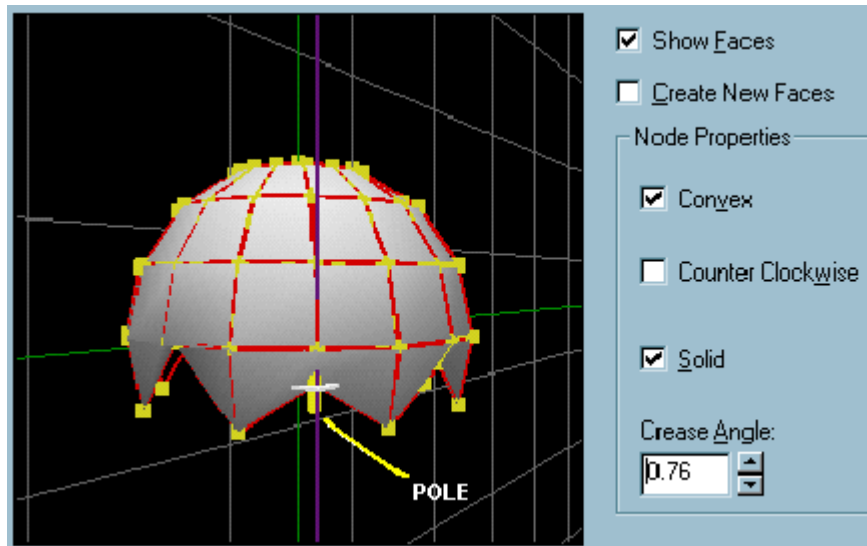
5. We haven't worked with **Color Mode** , a painting tool to set Material, so let's try that. Switch to Color Mode and the **Material Painter** dialog opens. Double-click on Diffuse Color to bring up a color wheel, and set the color to a light blue shade. Click on OK. Check off **Paint** for Mode, and place the cursor directly over the circle. The cursor changes to a paintbrush. Left click once on the circle and the Material is updated.

The Material Painter is very useful, and we'll see how it can acquire color from the View Pane in a moment. You can even add a new color to the Material Library once you've set it here.


6. Click on Close to exit the Material Painter and save your file. Select children for the WATER group, and add another Indexed Face Set. Go into the IFS Editor, and change the base shape to a Sphere . Click on the Cut icon , and hold the cursor over the sphere. Note how faces are highlighted in blue to indicate that they are selected. Using the cursor, begin to cut away the lower three levels of the Sphere by clicking down on the selected face. You'll probably need to switch to Model  mode to occasionally spin the Sphere around so that you can cut away all the faces. If you make a mistake, you can always undo your last step with the Undo or Redo keys . If you need to start over, just click any of the shape icons to begin again. When finished, you should have a shape like this:



7. Now we'll adjust some of the vertices around the bottom of our new shape. You may want to zoom in to the bottom of the sphere. Return to Select mode, and hold the cursor over a vertex along the bottom. Note how it turns blue to indicate that it is ready for selection. Click down on a vertex, and a manipulator handle appears. Click down on the "pole" of the manipulator, drag it up a bit, and release. Continue this process around the perimeter of the sphere, selecting every other vertex. Again, you'll want to use Model mode to occasionally rotate the sphere. Uncheck "Counter Clockwise" and check off "Solid" under Node Properties. Smooth out the object by increasing the Crease Angle. When you are finished, you should have a shape similar to this:



8. Click on OK to exit the Editor. Rescale the shape to make it about 2 or 3 times taller, and raise it above the fountain so that it resembles water shooting out. Now we can paint it as well. Switch to Color Mode, and check off **Acquire** in the Material Painter. Click on the circular pool of water to pick up its Material properties. Switch back to Paint Mode, and click on the spray shape you've just created. They now share the same material properties. **DEF** the new shape as **SPRAY**.

9. Now it's time to texture our scene. Click on Show/Hide Texture Library  on the toolbar and the Select Texture box opens. Scroll down to Water (Wavy) and select it. Left-click down in the display window, and drag and drop the texture directly onto the **SPRAY** shape, and the View Pane is updated. Save your file. You can also drag and drop a texture into the Node Tree on the texture field under Appearance (this is useful in crowded scenes).

Now that you have textures easily accessible, you can texture your entire scene for added realism. There are a couple of things worth noting. You don't have to texture anything you **USE**d... drop a texture onto the originally **DEF**'d object (like the pillar), and all of the **USE**d objects update automatically! Another thing to take into consideration is that you may want to add Texture Transform to Appearance to rescale or slide a texture, especially on very large objects. Finally, once you've textured a scene, you'll probably find it best to re-adjust the color and intensity of your lights. Like each of our previous exercises, an example of the temple up to this point is available in the Tutorials folder. Open it up to see how textures were used, and pay special attention to how the Texture Transform was used on the temple roof and floor.

[9] Moving Worlds

Interpolators

Interpolators perform a smooth transition between two values. These values may represent 3D coordinate points, color, time, or any kind of data that needs to be smoothed over a period. Listed below are the interpolation nodes and their function.

ColorInterpolator



Figure 129: Color Interpolator Node Icon

This node interpolates between a set of color key values, to produce a color (RGB) **value_changed** event. The number of colors in the **keyValue** field must be equal to the number of keyframes in the **key** field. The **keyValue** field and **value_changed** events are defined in **RGB** color space. A linear interpolation, using the value of **set_fraction** as input, is performed in **HSV** space.

CoordinateInterpolator



Figure 130: Coordinate Interpolator Node Icon

This node linearly interpolates among a set of floating point values. This would be appropriate for interpolating Coordinate positions for a geometric morph. The number of coordinates in the **keyValue** field must be an integer multiple of the number of keyframes in the **key** field; that integer multiple defines how many coordinates will be contained in the **value_changed** events.

NormalInterpolator



Figure 131: Normal Interpolator Node Icon

This node interpolates among a set of floating point values, suitable for transforming normal vectors. All output vectors will have been normalized by the interpolator.

OrientationInterpolator



Figure 132: Orientation Interpolator Node Icon

This node interpolates among a set of rotation values. The rotations are absolute in object space and are, therefore, not cumulative.

PositionInterpolator



Figure 133: Position Interpolator Node Icon

This node linearly interpolates among a set of floating point values. This is appropriate for interpolating a translation. The vectors are interpreted as absolute positions in object space. The **keyValue** field must contain exactly as many values as in the **key** field.

ScalarInterpolator



Figure 134: Scalar Interpolator Node Icon

This node linearly interpolates among a set of floating point values. This interpolator is appropriate for any parameter defined using a single floating point value, e.g., width, radius, intensity, etc. The **keyValue** field must contain exactly as many numbers as there are keyframes in the **key** field.

KeyFrame Animation

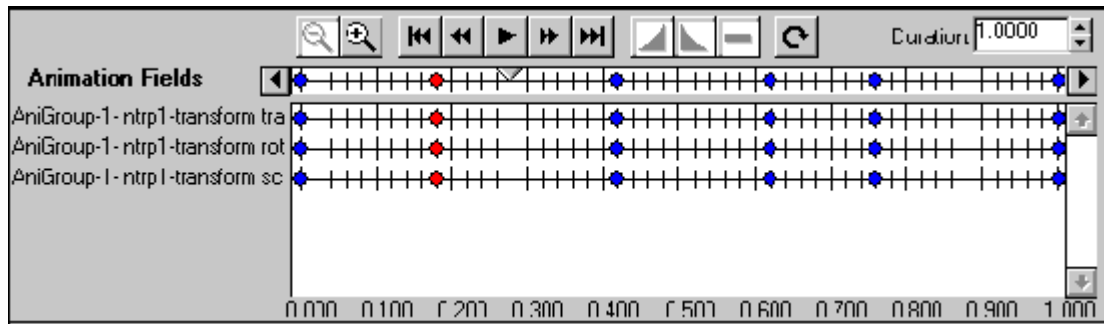


Figure 135: KeyFrame Animator

The **KeyFrame Animator** mode is provided to give the user access to powerful animation tools and allow the user to incorporate actions and reactions in the worlds he/she builds. It is important to understand the basic idea behind the **KeyFrame Animator**. The **KeyFrame Animator** allows the user to add fields to the animator, add frames to the animator, and modify fields from frame to frame. An animation can be added to the current scene in one of two ways: the first is to press the **KeyFrame Animator Icon** which will start the **KeyFrame Animator**, which will search the scene graph and add a new **TIMER** group to the scene if one is not present. The second is to press the **Time Sensor Icon** to add a **TIMER** group to the scene manually. The **Time Sensor Icon** can be used to add a new **TIMER** group to the scene only if you are editing the scene. This cannot be done while the **KeyFrame Animator** is activated. When the **KeyFrame Animator** is activated it will go to the **first** **TIMER** group and load that timer's animation information. If there are other Animations in the scene that you wish to edit, you will need to find the appropriate **TIMER** group in the **Tree View Window** and select it. This action will load the newly selected **TIMER** group's information into the **KeyFrame Animator**.

KeyFrame Animator marks the active frame with a red marker.

All other frames are marked with a blue marker. When a frame is selected, the active fields below it will be selected also. If however, the frame is activated before the fields are added, the fields may not be activated as they are added. In this case you will need to add the fields individually to the frame in which you want to make changes.

Fields can be added to the **KeyFrame Animator** by selecting the fields in the **Tree View Window** individually, or they can be added automatically by selecting one of the manipulators available to you. In either case, once the desired objects' fields have been added to the animator the animation can begin.

Frames are added to the animation by clicking on a new spot along the time line to perform animation changes. A frame must be activated before the animation can be added to the frame. Also note that there is a time index called **duration** that can be changed to allow for a shorter or longer animation, a few seconds or a few minutes or more. This field is saved as units of seconds. There are some special keystrokes that perform useful functions in the **KeyFrame Animator**. If you select a field in the **KeyFrame Animator** and right click on the field, you will be given the option to delete the field from the animation. If you have selected a frame and press the **CTRL + X** keys on the keyboard, the selected item will be cut from the animator.

When manipulating an object in a scene to create animation, remember to use the **Centerball Manipulator** for any manipulation referring to the use of a center point. This would specifically pertain to moving an object in a circular pattern or if you want to move an object as though a hinge were attached to it. These types of movements may seem to be the same as a modeled object with the **Universal Manipulator**, but you will not get the expected results.



Figure 136: KeyFrame Animator Icon

The **KeyFrame** Mode is activated when the **KeyFrame** button is clicked. This action will automatically add a **AniGroup-#-TIMER** to the scene if the transform for the object is not **DEF'd**. This is a default variable name that can be given any name by the user to specify a particular animation sequence. If a **DEF** name is given to all transforms in the scene, the **DEF** name will be used to describe the animation fields in the **KeyFrame Animator**.

The buttons and manipulators available on the **KeyFrame Animator** are the tools used to move objects around in the scene and to test the results to see if they match your needs. The **Universal Manipulator** and

the Centerball Manipulator are discussed in **Appendix C**. These manipulators can be used to manipulate any geometry in the scene to achieve a desired result. If you do not wish to alter geometry rotation, center, scale, translation or other fields using the manipulators, you are allowed to make these changes manually by editing the field values directly. The editing of these fields is done through dialog boxes for each field type. See the section of this manual that discusses Dialog Boxes and the editing of VRML 2.0 fields.



Figure 137: KeyFrame Animator Toolbar

The toolbar above has many buttons used to test the results of your animation: the Zoom in, Zoom out, First Frame, Previous Frame, Play/Stop, Next Frame, Last Frame, Ramp up, Ramp down, Ramp even, and Continuous play. These buttons all act on the animation itself. The scroll buttons to the left and right and top and bottom (right side) of the animation area are for displaying different sections of lines in the animation area for display or modification. When using the ramp buttons remember that the actual speed of play will change as the clustering of keyframes is changed. The ramp up will start with the keyframes being further apart and end with them being clustered very close together. The ramp down will start with the keyframes being clustered close together and end with them being placed further apart. The ramp even will evenly spread out the keyframes over the designated time period. To select a range of keyframes for modification with the ramp functions, left-click on the first keyframe and hold down the shift key while you left-click on the last keyframe. The continuous play button sets the **loop** variable of the sensor to true, so that the animation plays in a continuous fashion. The use of the Zoom in button to "enhance" or "blow up" an area of the animation for study and modification is a very powerful tool. In reverse, the zoom out button causes you to lose the detail but makes it possible to see the entire animation (though in some cases it may be a blur if there are a lot of frames being modified over a long period of time).



Figure 138: KeyFrame Animator Selection Toolbar

The above toolbar gives the user access to the Toggle Edit Window, KeyFrame Options, and Trigger functions. The Toggle Edit Window button allows the user to display the last dialog box that was opened for edit or open the current dialog box for editing a selected field value. The KeyFrame Options button opens the KeyFrame Options Dialog box, which allows the user to set the desired keyframe settings. The Trigger function allows the user to choose the type of trigger mechanism to be used to animate the scene. The Event Sensors of Visibility, Proximity, or Touch are used with the Time sensor to animate the scene and can be used to set parameters or trigger events. Other sensors must be activated manually through the use of the Route Dialog box.

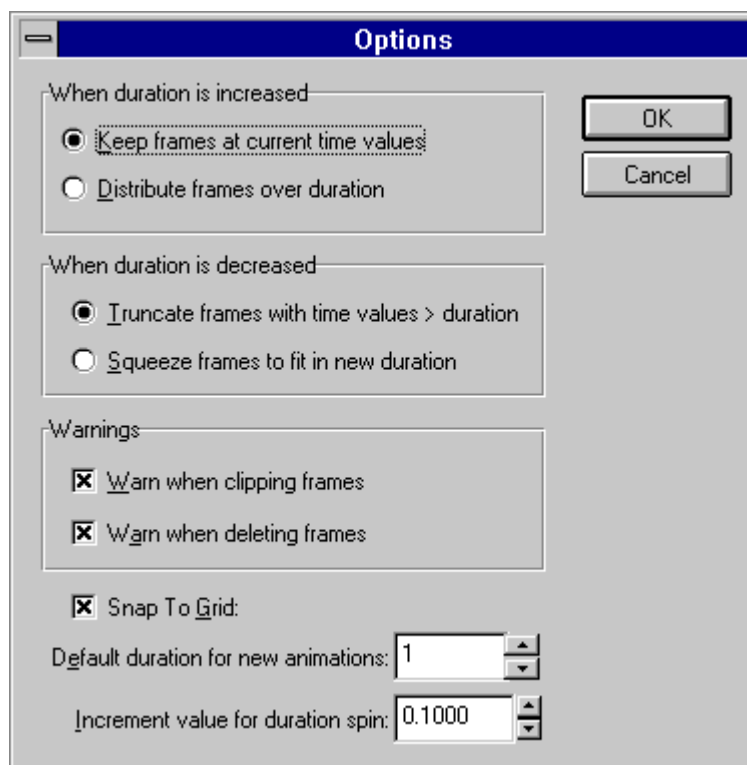
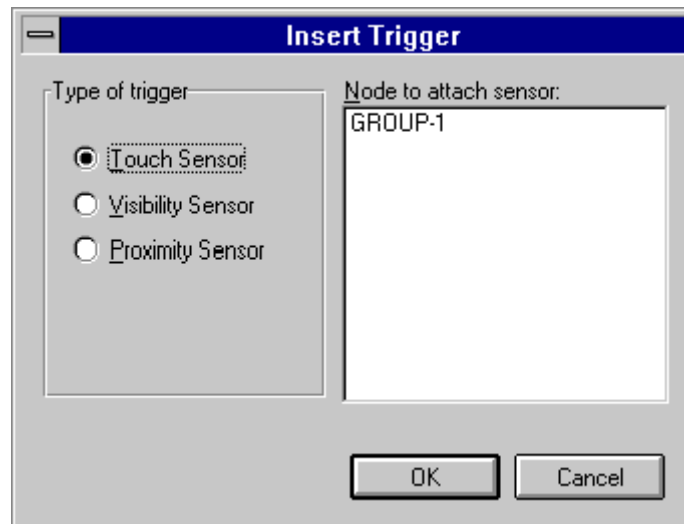


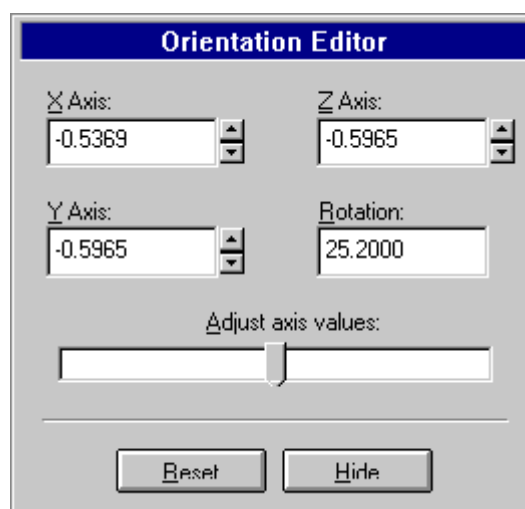
Figure 139: KeyFrame Animator Options Dialog Box

The KeyFrame Options dialog box allows the user to set the options for the KeyFramer. The first option is what to do with frames in the animation when the duration of the animation is increased. Should the animator keep frames at their current time value or distribute frames over entire duration? The next option is what to do with frames in the animation when the duration is decreased. Should the animator truncate frames with time values greater than the duration or squeeze frames to fit duration? The next option asks the user if he/she wishes to be warned when frames are clipped or deleted. The last option asks the user if he/she wishes to snap the animation to grid or not. There is also a field for the default duration for new animations. The default setting will be one unless it is changed. There is also an Increment Value for duration value. The OK and CANCEL buttons either exit while accepting the changes made or exit while not accepting the changes made.

**Figure 140: KeyFrame Animator Trigger Dialog Box**

The KeyFrame Trigger Dialog box is opened when the trigger button is pressed. This dialog box allows the user to set the trigger type and which sensor the trigger will be attached to. The dialog box lists three types of trigger sensors: Touch, Visibility, and Proximity. The list of nodes in the box to the right consists of nodes that contain a Time sensor. Together these can be used to create animation. The OK and CANCEL buttons either exit while accepting the input information or exit without accepting the input information.

Another dialog box that is connected with the KeyFrame Editor is the Interpolator Editor Dialog box. This dialog box can be opened by selecting View - Interpolator Editor while in KeyFrame Editor Mode. The dialog box that will be opened will look similar to this:

**Figure 141: Interpolator Editor Dialog Box**

This dialog box allows the user to fine tune the values of the variables along the KeyFrame Animation. The position and rotation values can be changed, as you can see from the dialog box. The slider bar is for editing

of these fields either one at a time or all at once. The RESET button will reset the values to what they were when the dialog box was opened. The HIDE button will hide the dialog box so that it is no longer

[10] Advanced Worldbuilding

Getting Started

There are two different kinds of sensor nodes: User Action and Automatic. User Action sensors like CylinderSensor, PlaneSensor, SphereSensor, and TouchSensor require the user to activate the action of the sensor, generally with the mouse. All sensors are attached to geometry when they are added to the scene. When a sensor is added the user will either choose the actual geometry node or something tied to it like a shape or appearance node. The actual placement of the sensor node is taken care of automatically by the node tree. It places the sensor in the "optimal" position in reference to the geometry for the sensor to be used by any routes that may be attached later. When the phrase "maps pointing device" is used, it refers to the action of translating click and drag motions of the mouse or other device directly to the sensor's shape and action.

Automatic sensors like ProximitySensor, TimeSensor, and VisibilitySensor are activated not by direct action, but by the indirect actions of the user, such as moving into view, a specific time being reached, or just moving close to an object.

CylinderSensor



Figure 142: Cylinder Sensor Node Icon

The CylinderSensor maps the pointing device (e.g. mouse or wand) motion into a rotation on an invisible cylinder that is aligned with the Y axis of its local space. Therefore, an object attached to a cylinder sensor can be rotated around the Y axis by clicking and dragging on the object.

PlaneSensor



Figure 143: Plane Sensor Node Icon

The PlaneSensor maps the pointing device (e.g. mouse or wand) motion into translation in two dimensions, in the XY plane of its local space. The PlaneSensor uses the descendant geometry of its parent node to determine if a hit occurs.

ProximitySensor



Figure 144: Proximity Sensor Node Icon

The ProximitySensor generates events when the user enters, exits, and moves within a region in space (defined by a box). A ProximitySensor can be enabled or disabled by sending it an **enabled** event with a value of TRUE or FALSE - a disabled sensor does not send output events. A ProximitySensor with a (0,0,0) **size** field cannot generate events - this is equivalent to setting the enabled field to FALSE.

SphereSensor



Figure 145: Sphere Sensor Node Icon

The SphereSensor maps the pointing device (e.g. mouse or wand) motion into spherical rotation about the

center of its local space. The SphereSensor uses the descendant geometry of its parent node to determine if a hit occurs. The feel of the rotation is as if you were rolling a ball.

TimeSensor



Figure 146: Time Sensor Node Icon

TimeSensors generate events as time passes. TimeSensors can be used to drive continuous simulations and animations, periodic activities, and/or single occurrence events such as an alarm clock.

TouchSensor



Figure 147: Touch Sensor Node Icon

A TouchSensor tracks the location and state of the pointing device and detects when the user points at geometry contained by the TouchSensor's parent group. If the TouchSensor is disabled, it does not track user input or send output events.

VisibilitySensor



Figure 148: Visibility Sensor Node Icon

The VisibilitySensor detects visibility changes of a rectangular box as the user navigates in the world. The VisibilitySensor is typically used to detect when the user can see a specific object or region in the scene and to activate or deactivate some behavior or animation in order to attract the user or improve performance.

Time Dependent Nodes:

AudioClip, MovieTexture, and TimeSensor are time dependent nodes. They each have exposedFields of **startTime**, **stopTime**, and **loop**, and the eventOut: **isActive**. Time dependent nodes start their execution when **startTime** is reached and will stop when **stopTime** = now > **startTime**. The **loop** field controls whether the time frame for execution is continuous or not.

AudioClip



Figure 149: Audio Clip Node Icon

An AudioClip node specifies audio data that can be referenced by other nodes that require an audio source.

MovieTexture



Figure 150: Movie Texture Node Icon

The MovieTexture node defines a time dependent texture map (contained in a movie file) and parameters for controlling the movie and the texture mapping. This node will accept AVI and MPEG files, but the V-Realm Builder does not support compressed audio tracks in AVI files.

The Sound Node

The Sound node describes a sound at an exact point in space. As you move around in the scene, the sound changes based on the distance and relative position from the point of the sound.

Sound



Figure 151: Sound Node Icon

The Sound node describes the positioning and spatial presentation of a sound in a VRML scene. The sound may be located at a point and emit sound in a spherical or ellipsoid pattern in the local coordinate system.

Manually Connecting Routes



Figure 152: Route Node Icon

When the Route Node Icon is pressed the Route Dialog Box is displayed.

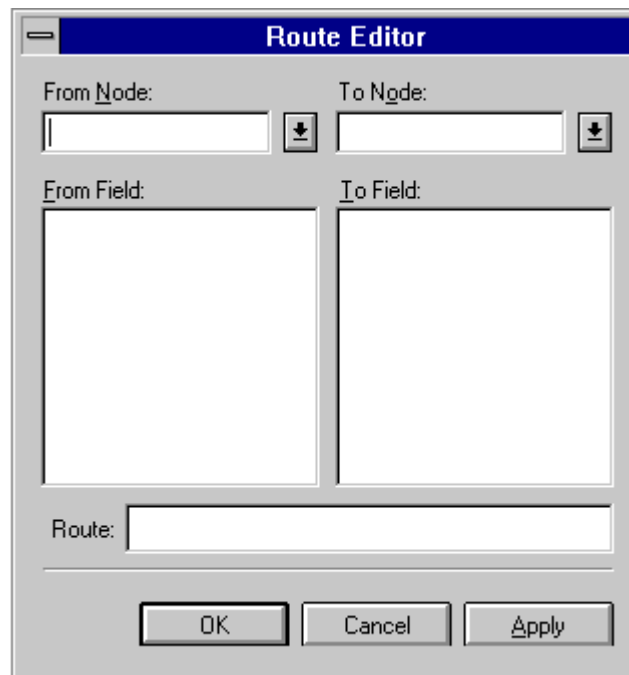


Figure 153: Route Dialog Box

The Route Dialog box allows the user to route messages from one node to another. The only restriction here is that the message must be routed from fields of the exact same type. Messages are routed from nodes that have output events to nodes that have input events. Route messages are links from one node to another. For example, if I attach a route from a SphereSensor to a Sphere with an earth texture, I can cause the Sphere to rotate if I move the mouse over the SphereSensor, in a circular motion. This single action of the mouse movement can be transferred to more than one action. According to how many nodes are attached to the SphereSensor there can be that many other objects taking advantage of the mouse movements. Manipulator sensors like Plane, Sphere, and Cylinder sensors require manual connection where other sensors can be easily connected in the **KeyFrame Animator**. Manual routing of messages is rarely done, but if you connect nodes with routes and afterwards decide to use the **KeyFrame Animator** to add additional animation, your manual route connections will automatically be displayed in the Node Tree View of the scene.

When the routing node is selected for edit, it will initially display the dialog box above with no node or field names.

First, use the down arrow buttons beside the field **From Node** to display a list of nodes to select from. Only nodes that have been **DEF'd** will be displayed. Select the desired node.

Secondly, when the node has been selected, a list of field variables will be listed. Only field variables for the selected node will be displayed. Only **exposedField** and **eventOut** field names will be displayed since the **From Node** selection must have an output variable selection. Select the desired field variable.

Thirdly, use the down arrow buttons beside the field **To Node** to display a list of nodes to select from. Only the nodes that have been **DEF'd** *and* have fields of the same type as the field variable in the second step above will be displayed. Select the desired node.

Lastly, when the node has been selected, a list of field variables will be displayed. Only field variables that are of the same type as the field variable selected in the second step above will be displayed for selection. Only **exposedField** and **eventIn** field names will be displayed since the **To Field** selection must have an input variable selection. Select the desired field variable.

↓↓ From Node ↓↓		To Node	
		exposedField ↓	eventIn ↓
exposedField →	YES		YES
eventOut →	YES		YES


The above table shows the node field types that can be routed from within a node to the node field types that can be routed to within a node. Only these field types will be displayed in the Route Dialog Box. If you have questions about what the exact field names each node has, please refer to the VRML 2.0 Specification.

When the route has been set you will see the syntax of the route command in the **Route** field in the dialog box. This type of manual editing of route nodes is rare and is usually done automatically by the V-Realm Builder's **KeyFrame Animator**. But for that instance where you would like to manually set a route and attach sensors, this dialog will make the job easier.

Exercise 7: KeyFrame Animation Editor

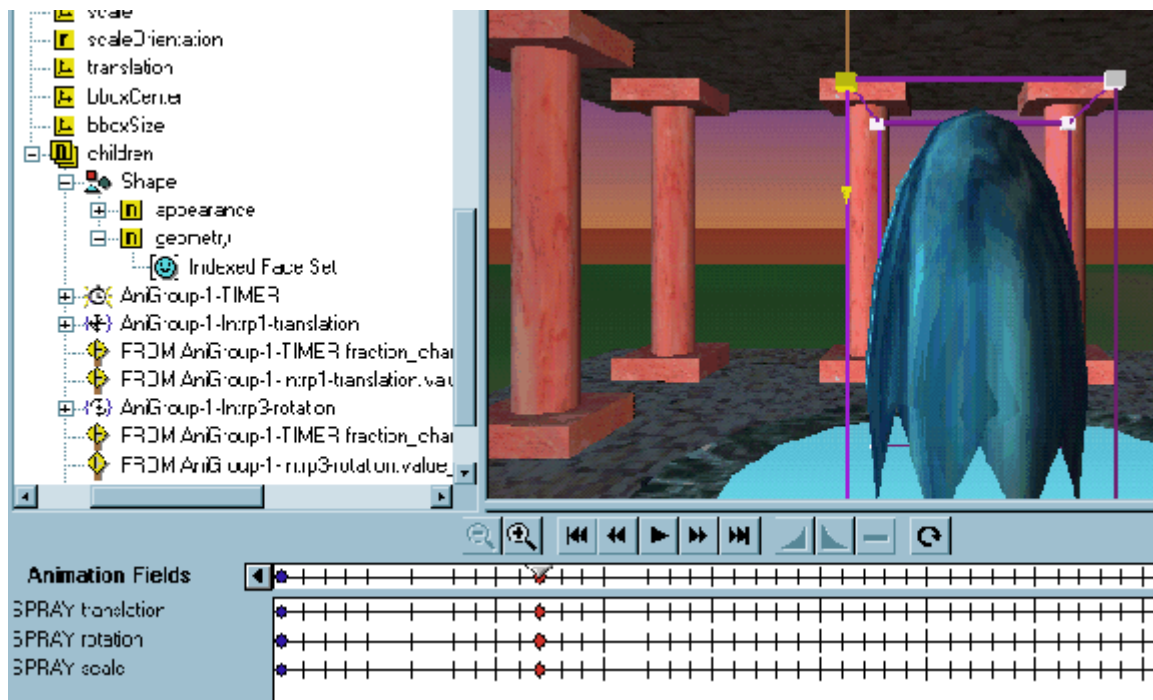
Up to this point, we've concentrated on constructing our worlds. Now it's time to add interaction, motion, and sound to the temple scene. In this exercise, we'll animate the water fountain and attach the sound of running water. We'll set the whole thing up to be triggered by some action a visitor to the world takes.

1. Open `temple.wrl`, and collapse the hierarchy so that only the highest items show, with the exception of the **WATER** group which we'll be changing. Move into the temple so that the fountain is in front of you. Using the **Pick** tool, rescale the **SPRAY** object so that it is nearly flat. Use the **CTRL+SHIFT** keyboard combination, then click down on one of the top manipulator handles and drag down. Now move **SPRAY** down so that it just peeks out above the water pool. You may want to fine tune these adjustments inside the Node Tree. Save your file.

2. Switch to **KeyFrame Mode** . The interface changes to display a timeline.


3. Reselect the **SPRAY** object by clicking once on it. Note that **Animation Fields** for translation, rotation, and scale drop in automatically. If the duration box is not currently set to 5, set it with the dialer to 5 now. The white timeline across the top is the **Parent timeline**. The individual lines for each of the fields are **Children timelines**.



4. With your left mouse button, click into the Parent timeline at about 1.25 seconds. Tickers are dropped into place for each of the Children timelines beneath. Using your **Pick** tool and the **CTRL+SHIFT** keyboard combination, click down on one of the top manipulator handles and drag up to make the water taller. You may want to drag it's position up a bit as well.





5. Click at the 2.50 seconds mark, and scale the **SPRAY** back down a bit, as if the water is pulsing up and down over time.


6. Click at the 3.75 seconds mark, and scale the water back up one more time. If we wanted, we could make one last adjustment at the 5 second mark. But if you click on the **Next Frame** button, you'll see that it already has the same settings as 0.00 seconds, which works for this example.

7. Click on the **First Frame** button , then click **Play**  to see your fountain in action. Note how the Builder has interpolated all the frames in-between to make the animation smooth. This is how all animation in the Builder works. At any point you can also toggle to show the **Edit Window**  which, like the Node Tree, gives you the ability to make precision adjustments to your animation.

8. For this particular animation, we're not going to turn **Loop**  on. We need only set an event that will start the animation. Click on **Insert Animation Trigger**  and a dialog pops up. Leave it set on **Touch Sensor**, and in the sidebar scroll down and select **WATER**. By doing so, we've just set up a routing of events. That is, when a visitor clicks (touches) anything in the **WATER** group (either the **POOL** or the **SPRAY**), it will trigger the animation.


9. Click on **KeyFrame Mode** to return to the normal interface and save the file. Note how the Node Tree has been updated to reflect everything we did in KeyFrame Mode. Any of these fields can be edited by hand if need be. For instance, suppose you want to extend the time of the animation. You don't need to go back into KeyFrame Mode to do so. In the Node Tree, expand **Anigroup-1-TIMER** , and double-click on **cycleInterval**. This is equivalent to the Duration setting in the KeyFramer. Change it to 10.

10. Now to test our animation. Switch to **Test Mode** , and click on either the **POOL** or the **SPRAY** in the View Pane and the animation will run. Test Mode saves you the trouble of having to exit out to a VRML browser to check your work.

11. One more element we wanted to add is Sound. As a child to the **WATER** group, add a **Sound**  node. Expand the node, this time by right-clicking on the node and selecting "Expand All Children". The default values for spatialization here will work fine. All we need to do is set the source.

12. Select the source field in the Node Tree, and click on the **Audio**  node. **DEF** the Audio node as

`WATER_SOUND`, and expand it. Expand the `URL` field, double click on the string icon, and use the dialog to browse to the `.WAV` file `Waterrun.wav`. It is located in the `Program/Sounds` directory. Click on `OK` to close, and the Node Tree is updated.

13. In order to get the sound to play, we'll need to create a **ROUTE** by hand. Select the Sound icon in the Node Tree, and click on **Insert Route**  from the toolbar. A blank ROUTE appears in the Node Tree.

14. Double-click on the ROUTE to bring up the **ROUTE Editor**. We want the same Touch event that sets off the animation to set off the sound. Using the drop-down From Node box, select `WATER_TouchSensor`. A list of possible events appears. Select `touchTime`. Now with the To Node box, select `WATER_SOUND`, and for event choose `set_startTime`. Although this looks complex, we've basically written our ROUTE to say, "At the moment the cursor clicks on this group, start the sound." Save your file.

15. Switch over to Test Mode one last time, and click anywhere on the water. You should see the animated water rise out of the fountain, accompanied by the sound of running water.

There are an infinite variety of events which can trigger each other. For instance, we could have set both the animation and sound to loop endlessly, but triggered by a **Proximity Sensor** based on our distance from the fountain. That way, the fountain would appear to always be running whenever we were close enough to see and hear it. It's always a good idea to specify these processor-intensive type of events to run only under certain conditions. Or we could have also animated the color of the water over time by dropping the `diffuseColor` field into the KeyFrame Editor. As mentioned earlier in the manual, you can even animate an object with a viewpoint attached to it, so that the visitor moves automatically through the world (see the file `park.wrl` for a good example of this).

VRML is all about interaction, behaviors and events, and multiple forms of media. The best thing to do is to experiment with these features, and always construct the geometry of your world with these features in mind.

Other Advanced Nodes

These advanced worldbuilding nodes allow the user to control very complex functions related to geometry and scene appearance. These nodes give the user access to world information, texture coordinate mapping, and modification of normal vectors.

PixelTexture



Figure 154: Pixel Texture Node Icon

The `PixelTexture` node defines a 2D image-based texture map as an explicit array of pixel values and parameters controlling tiling and repetition of the texture onto geometry.

TextureTransform



Figure 155: Texture Transform Node Icon

The `TextureTransform` node defines a 2D transformation that is applied to texture coordinates. This node affects the way textures are applied to the surface of geometry.

Note: `TextureTransforms` cannot combine or accumulate.

WorldInfo



Figure 156: World Info Node Icon

The WorldInfo node contains information about the world. This node has no effect on the visual appearance or behavior of the world - it is strictly for documentation purposes. The **title** field is intended to store the name or title of the world so that browsers can present this to the user - for instance, in their window border. Any other information about the world can be stored in the **info** field - for instance, the scene author, copyright information, and public domain information.

Scripting

Although the VRML language itself provides methods for creating moving and interactive worlds, it cannot do everything. VRML is more than a 3D language, but less than a programming language. For instance, VRML can animate an opening door triggered by a TouchSensor. However, clicking on the door a second time will not close the door... it simply starts the same animation again. In order to perform the logic required for such a level of interaction, VRML makes provisions for implementing *scripting* via a separate programming language.

A script allows the user to define their own nodes and fields, and then define instructions to perform a series of actions. Scripts can be used to control animation, interact with a VRML browser, dynamically change a running VRML file, or communicate across the internet with another server. As per the VRML specification, V-Realm Builder supports scripting with VRMLScript, which is a subset of JavaScript (also supported). V-Realm Builder will also run scripted worlds in Test Mode.

A full description of scripting is beyond the scope of this manual, but there are a number of books on VRML and JavaScript available which go into great detail on the subject. If you wish to include scripting as a function of worlds that you are building, you'll want to add some of these books to your library (see the tips section below). There are also some example worlds such as `Dorscript.wrl`, `Bldscript.wrl`, and `Wave.wrl` included in the "Worlds" sub-directory to help get you started.

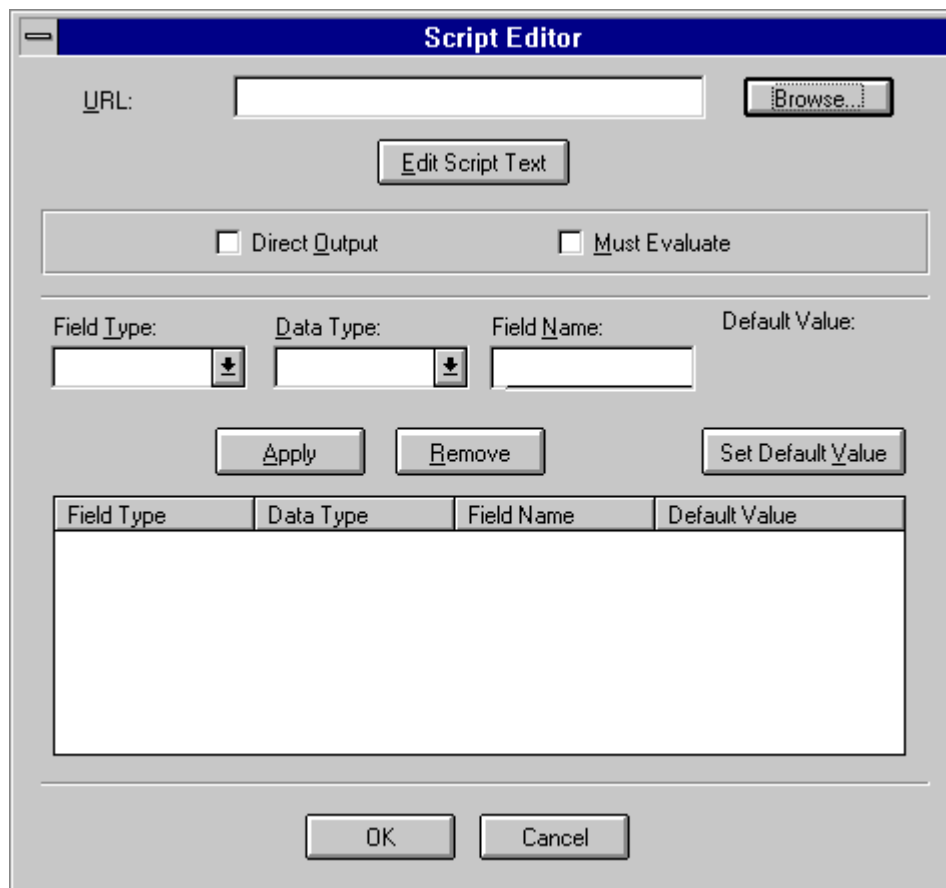
Scripts are either entered directly into the VRML file, or referenced as external files, in the **Script** node.

Script



Figure 157: Script Node Icon

Script nodes typically receive events that signify a change or user action, contain a program module that performs some computation, or effect change somewhere else in the scene by sending output events. Each Script node has associated programming language code, referenced by the URL field, that is executed to carry out the Script node's function. The script can be typed directly in the Builder's Edit Script Dialog, or referenced by URL as an external text file. When the Script Node Icon is pressed, a Script node is added to the scene. When this node is double-clicked in the Node Tree, the following dialog box is displayed:



The **Script Editor** dialog box is used for configuring a script. It features a title bar with the text "Script Editor". Below the title bar, there is a "URL:" label followed by a text input field and a "Browse..." button. A central "Edit Script Text" button is positioned below the URL field. A horizontal section contains two checkboxes: "Direct Output" and "Must Evaluate". Below this, there are four labels: "Field Type:", "Data Type:", "Field Name:", and "Default Value:". Each label is followed by a text input field; the "Field Type" and "Data Type" fields include a downward arrow icon. Below these input fields are three buttons: "Apply", "Remove", and "Set Default Value". At the bottom of the dialog is a table with four columns: "Field Type", "Data Type", "Field Name", and "Default Value". The table body is currently empty. At the very bottom are "OK" and "Cancel" buttons.

Figure 158: Script Node Edit Dialog Box

From this window a script can be setup with all of its fields, data types, and default values. At the top of the dialog box you will notice the **Field Type** and **Data Type** drop down boxes that list the options that can be chosen. Input a name for **Field Name** and press the APPLY button to add the field to the Script. The script can also be saved to a URL or file location. The BROWSE button is used to search for a location or particular file. There are check boxes here for the Boolean variables of MUST EVALUATE and DIRECT OUTPUT. There are also APPLY and REMOVE buttons to add input data to the script or remove selected data from the script. The SET DEFAULT VALUE button is used to set the initial values of some variables. The OK and CANCEL buttons are for accepting changes and exiting or ignoring changes and exiting.

When the EDIT SCRIPT TEXT button is pressed on this dialog box the following dialog box will be displayed:

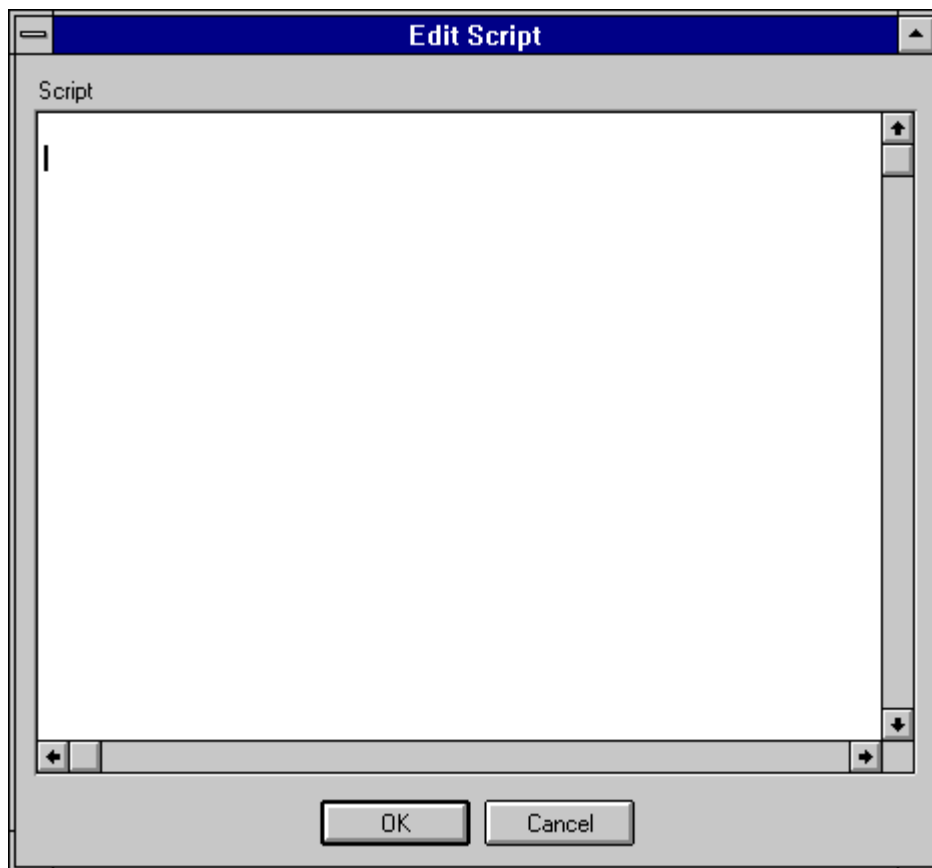


Figure 159: Edit Script Dialog Box

This dialog box allows the user to actually type in or change the text of a script. The scroll bars are used to scroll within the script and the OK and CANCEL buttons are for exiting while saving changes and exiting without saving changes.

Teach Yourself VRML in 21 Days by Chris Marrin, [The VRML 2.0 Sourcebook](#) by Ames, Moreland and Nadeau, and [The JavaScript Handbook](#) by Danny Goodman are a few of the great books out there that can help you get started with scripting in VRML. Also, you can check the Ligos website at <http://www.ligos.com> for tutorials and links to online web resources on the subject. There are also examples of scripts in the VRML 2.0 Specification (included on the CD-ROM) and in the V-Realm Builder sub-directory, "Worlds".

PROTO and External PROTO nodes

External PROTOs are not supported in this release of the V-Realm Builder. Their inclusion here in the documentation is given as a convenience since the explanation of PROTO and External PROTOs is so similar.

PROTO nodes define new classes of variables by extending the capabilities of existing VRML nodes. A PROTO is how we create custom nodes with custom fields so that we can extend the VRML standard beyond the nodes included in the specification. PROTOs define default values within their definition just as all VRML nodes define default values, but the default values can be overridden when an instance of the PROTO is called. The use of the **IS** keyword is used to overwrite a default value with another value. PROTOs use the inline method to input data to the current scene. When variables are attached using the **IS** keyword the variables now represent the same thing, but this is a **one-way** relationship from the prototype field to the field contained in the prototype body. Attach fields with the **IS** keyword by right-clicking on the field and choosing the **IS** option from the pop-up menu. The **IS** option will be displayed only for fields for which it is valid.

External PROTOs define the same information as a PROTO except the information is external to the file and usually exists somewhere on the Web or distributed network. They do not define the default values of a new node type but refer to a file that contains the prototype implementation and default values. Because External PROTOs rely on obtaining data from the Web, they use the URN or URL methods to function.

Once a PROTO (external or internal) node has been created, the user must create an instance of the newly defined PROTO before the new node is actually added to the scene. It is during this phase that new values can be given to the fields of the new node if the default values are not wanted.

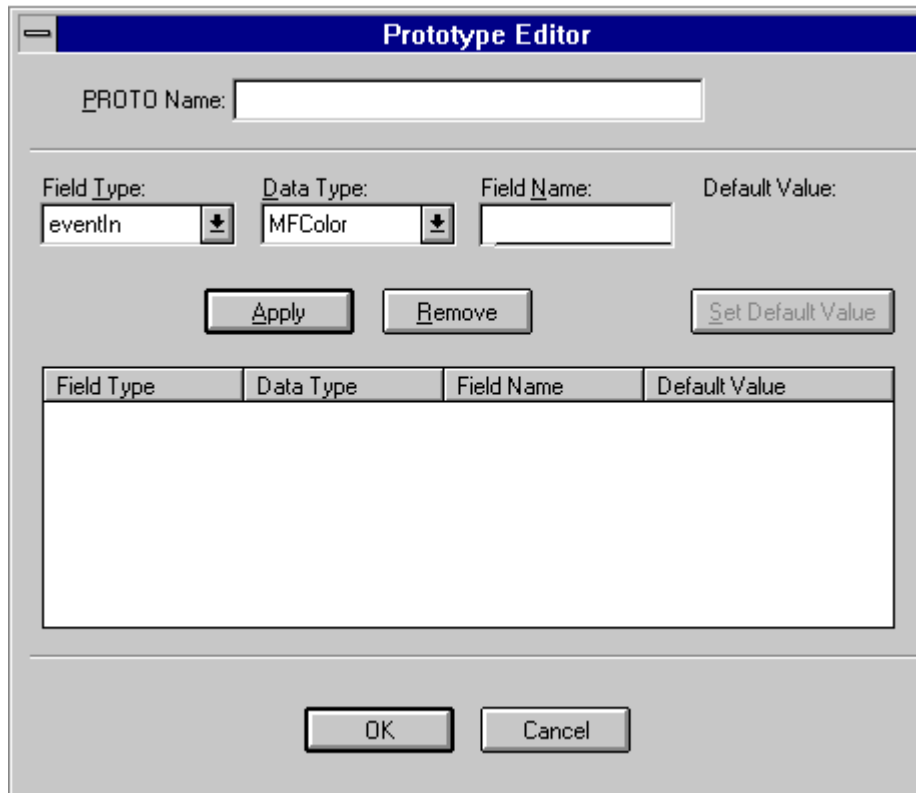


Figure 160: Prototype Editor Dialog Box

The Prototype Editor Dialog box allows the user to add PROTO node definitions to a world. At the top of the dialog you will notice the **Field Type** and **Data Type** drop down boxes that list the options that can be chosen. Input a name for **Field Name** and press the APPLY button to add the field to the Prototype. If you wish to delete an already defined field, just select the field in the list of Prototype fields and press the REMOVE button. The PROTO Name field allows the user to give the prototype a name. The SET DEFAULT VALUE button is for setting an initial value of a field. OK and CANCEL perform the normal function of exiting with saving and exiting without saving changes.

Recursive PROTOs or External PROTOs are illegal.

IS statements must refer to fields or events defined in a PROTO or External PROTO.

Instancing with Prototypes

Instancing with prototypes is as easy as using instancing using **DEF** and **USE** discussed earlier. The one major difference is that when we are instancing with prototypes we are making reference to a new VRML class that has been created using PROTO or External PROTO nodes. When these prototype nodes are created they are defined in every way, and usually default values are given to the fields (External PROTOs do not have default values).

We instance a previously defined node by the use of the **USE** keyword just as before, but this time we also can define new values for the node. Let us assume for a minute that the VRML spec did not have a definition for Box. We decide to define the box as it is currently defined in the VRML 2.0 Specification. Therefore, we have a single field called **size** that has three components for the size in the X, Y, and Z directions. By default this box will be 2 by 2 by 2 if we just instance it.

```
PROTO Boxit [ boxSize 2 2 2 ]
{ boxSize IS size}
```

But if we want a bigger box we could

```
Boxit { boxSize 4 5 6}
```

which would create a box that would be 4 by 5 by 6. Rather than call a variable by direct call, we have used another variable that is local to the instance to load the variable in the prototype.

Exercise 8: PROTO

In the last of our lessons, we'll take a look at the **PROTO** node. PROTO is powerful simply because it allows us to extend VRML beyond what is explicitly written in the specification. For instance, using PROTO, we could take an animated piece of geometry, declare it to be a PROTO, and specify just a few custom fields so that it can easily be used over and over. It's similar to instancing (**DEF/USE**), except PROTO gives you the ability to change the fields you specify. For all intents and purposes, you are creating a new VRML primitive.

In this example, we'll prototype the tree (`tree.wrl`) we created and added to the Object Library earlier and give it one field... `size`.

1. Create a new file in the Builder. Insert a **PROTO node** .

2. Double click in the Node Tree to open the **PROTO Editor**. This is where we'll specify the only field to our node, `size`.

3. In the editor, use the dropdown boxes to specify that:


- Field type will be a simple `field`;
- Data type is `SFVec3f` (like a scale field... 3 numbers representing X, Y, and Z);
- The new Field Name is `size`; - Default Value is `1 1 1` (no need to change this);

Click "Apply", and the field is dropped into place. For **PROTO Name**, enter `TREE_PRO`. Click on OK to exit.

4. Now that we've declared a PROTO, we need to specify what makes it up. Fortunately, we've already done the work. Open the Object Library and, under the category "Landscapes", choose the Tree you added in Lesson 3. Drag and drop the tree into the Node Tree, onto **PROTO Body**. The Node Tree will update. The View Pane, however, remains blank. We haven't actually inserted a tree into the scene, we're just defining one.

5. In the Node Tree, locate the `TREEHIXFORM` group and its `scale` field. Scale will be the basis for our `size` field we set up in the PROTO Editor. Right-click on the `scale` field, and from the pop-up choose **is**. The only available field is the one we specified, `size` (field `SFVec3f size`). Select and click OK. We're now saying that anytime we use the `size` field of our PROTO, it should be treated as `scale`.

6. In the Node Tree, locate the `TREELOXFORM` group and repeat the above step. Remember that our tree was actually made up of two trees... hi-rez and lo-rez. We want our PROTO'd tree to work the same no matter how far away it might be.

7. That's it! You've defined your PROTO. Collapse the PROTO in the Node Tree. Select New World. Now, if you want to use this new PROTO, you simply click on **Insert PROTO Instance** . **Legal choices for PROTOs are listed in the box that comes up. In this case, we've only defined** `TREE_PRO`, so select that. `TREE_PRO` appears as a node in the Node Tree, and the Tree appears in the View Pane.

8. To adjust the size of the tree in any direction, adjust its `size` field in the Node Tree. Save your file.

Now you could create a whole forest of these highly optimized trees. Unlike a tree that was **DEF'd** then **USE'd**, however, each tree can now be customized to your liking. Not only is this ultimately easier, each new tree will have little impact on file size.

Just like VRML behaviors and events, the best thing to do with PROTOs is to experiment with them. There are situations where PROTO can be invaluable (like building a whole low-cost forest), and it also shows

how extensible VRML is. Just because a feature isn't currently in the VRML specification doesn't mean you can't construct it then prototype it. In fact, as VRML becomes more popular, you'll find entire libraries of PROTOs out on the web, much like object class libraries that are available to programmers.

Appendix A - Keyboard Shortcuts

These Quick-Key definitions listed here are provided to assist the user in quickly navigating the screens and in executing commands.

CTRL+C Copy a file or selected part of a file.
CTRL+O Open a pre-existing file.
CTRL+S Saves a file.
CTRL+V Pastes selected information into a file.
CTRL+X Cuts selected information from a file.
CTRL+Z Undo the last action.
CTRL+Insert Copy selected text to clipboard.
Alt+Backspace Undo last edit changes.
Shift+Insert Paste selected text.
Shift+Delete Cuts selected text to clipboard.
F1 Opens help screen.
F3 Increase movement speed.
F4 Decrease movement speed.
Return OK.

The function these keys perform can be obtained by use of the pull down menus in the main toolbar, or by using the right click menus. They are listed here for those who prefer to use the keyboard whenever possible.

There is a second class of function keystrokes that can assist the user in speeding up his/her viewing and editing process. These keystrokes are specifically tied to the special editors and allow for more complex actions to be performed quickly. Below is a list of the editors that have such keystrokes and what they accomplish. Function keystrokes that are listed in menus are not displayed here.

IndexedFaceSet

SHIFT If the SHIFT key is held down before activating the Point Dragger in the IndexedFaceSet Editor, the movement of the point will be limited to one plane instead of two.
CONTROL Holding down the CONTROL key while selecting vertices will allow the user to "collect" vertices for manipulation.

Extrusion

NONE LISTED

ElevationGrid

CONTROL Select point on ElevationGrid and hold down CONTROL key to select other points to manipulate.

KeyFrame Animator

SHIFT Select keyframe pointer and then hold down SHIFT key and select end- of-range keyframe pointer to select a range of keyframes.
CONTROL Select keyframe pointer and then hold down CONTROL key to select individual keyframes to group non-continuous keyframes together.
SHIFT Select keyframe drag pointer to move the keyframe and all keyframes in the direction of movement, forward or backward.
CONTROL Select keyframe drag pointer to move the keyframe forward or backward.

Appendix B - Hints and Tips

Beyond the qualities of imaginative and artistic content, a VRML world is often judged in terms of two other factors: file size and performance. Maintaining a balance between the two is often a challenge, but the Builder gives you all the tools and control necessary to do so. Here are some tips for producing highly optimized worlds (*examples referred to here can be found in the "Worlds" sub-directory of V-Realm Builder*):

- Use Primitive geometry (Sphere, Box, Cylinder, Cone) to keep file size down, but use them with caution. The number of polygons generated by a primitive may be more than you actually require. *Lower polygons = faster rendering = better frame-rate*. For instance, combining a cylinder and a sphere to produce a domed, silo-like shape is not efficient. A VRML browser has to render the lower half of the sphere whether or not it's visible. Use an Indexed Face Set instead (for instance, a sphere IFS with the bottom cut off).
- If parts of a shape will never be seen by a viewer, cut them out (either using the FALSE option on primitives, or the cut tool in the IFS Editor);
- If the inside or backside of an object will never be seen, set the solid field to TRUE so that that side is not rendered;
- Extrusion can be a low-bandwidth, low polygon substitute for many objects that are normally Indexed Face Sets (see the file `human.wrl`);
- Large area polygons are slow to render... break large objects (such as very large planes or elevation grids) into several smaller objects to speed the frame-rate;
- Lights and textures can be expensive items in terms of processing... use them efficiently and sparingly;
- Background textures have to be downloaded... use Background colors instead whenever possible;
- Utilize Level of Detail (LOD) whenever possible for large scale scenes and far off objects;
- Utilize the Billboard node for distant objects that are low in detail... when used with LOD, Billboard can be very effective;
- Use the transparency feature of some GIF textures to simulate complex geometry;
- Use Inline for objects that are distant from the entry viewpoint. That way, the visitor can begin to navigate the world without waiting for the whole scene to load and render;
- DEF all major groups and anything you're going to animate or reference... the Builder will do it for you when you save your file, but the name you give it will probably mean more to you than something like "Anigroup-31-Node";
- Utilize instancing with DEF/USE and PROTO as much as possible to keep files compact;
- Custom textures can very effectively simulate the effects of lighting and shadow (see the file `lamp.wrl`);
- The Text Node produces objects that are very heavy with polygons... use it sparingly. A combination of VRML and HTML with frames is often a better solution for communicating a lot of textual information;
- Take the time to specify bounding boxes for Inlines, first level Transforms, Anchors, LOD's, etc... some browsers can take advantage of this and not render objects that are out of sight;
- Arrange and group your scene spatially in the Node Tree, as though you are walking through the world... group interior furnishings with the architecture that surrounds them;
- Animations and sound are expensive in terms of processor resources... you should rarely have them running all the time. Instead, trigger animations and sounds with Proximity or Visibility Sensors so that they only run when needed (see the file `temple.wrl` in the "Worlds" directory);
- Multiple TimeSensors are expensive... if appropriate, set many events up so that they share a single TimeSensor;
- Use DEF'd, multiple Viewpoints to ease navigation through the world... most browsers support some form of moving from Viewpoint to Viewpoint with a single keyboard command;
- Animated Viewpoints can be easily created by making a Viewpoint a child to a moving Transform (see the file `park.wrl`);
- MIDI files are much smaller than WAV files for music, and easily found on the web... try to limit WAV files to short noises, and loop them whenever appropriate;
- Scripting is an expensive function in a browser... don't use it for simple animations that are best left to Interpolators;
- When you are finished with your world and ready to post it to the web, use "Save As Gzip"... your VRML file will be *much* smaller as a result, and the Builder can always re-open it if need be;

Appendix C - Manipulator Reference

Universal Manipulator

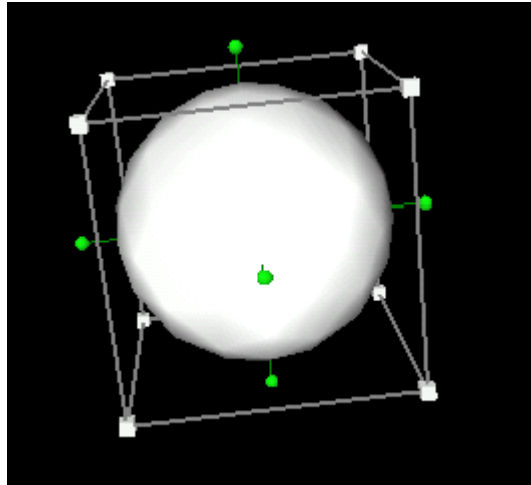


Figure 161: Universal Manipulator

Universal Manipulator Reference

Mouse and/or Key Action	Description Of Manipulation
Left mouse click on Object	Translate2 Manipulator (Ability to move object anywhere within the plane selected.)
Left mouse click on a green ball	Centerball Manipulator (Ability to move object around its' center in the direction the mouse is "pulled".)
Left mouse click on a white corner	Uniform Scale Manipulator (Ability to scale object in all directions no matter which corner is selected.)
Left mouse click on line segment	Translate2 Manipulator (Ability to move object anywhere within the plane selected.)
SHIFT + Left mouse click on Object	Translate1 Manipulator (Ability to move object in one plane motion based on what direction the mouse is "pulled" first.)
SHIFT + Left mouse click on a green ball	Trackball Manipulator (Ability to rotate object in any spherical direction using center of object as rotor point.)
SHIFT + Left mouse click on white corner	Scale1 Manipulator (Ability to scale an object uniformly in only one direction.)
SHIFT + Left mouse click on line segment	Translate1 Manipulator (Ability to move object in one plane motion based on what direction the mouse is "pulled" first.)
CONTROL + Left mouse click on Object	TabPlane Manipulator (Ability to move object in one plane based on which plane is selected and then translating perpendicular to that plane.)
CONTROL + Left mouse click on a green ball	Centerball Manipulator (Ability to rotate object around center point, which is calculated as point opposite the point selected.)
CONTROL + Left mouse click on white corner	Scale Manipulator (Ability to scale object in all directions based on which point selected, while maintaining the position of the point opposite the selected point.)
	TabPlane Manipulator (Ability to move object in

CONTROL + Left mouse click on line segment	one plane based on which plane is selected and then translating perpendicular to that plane.)
CONTROL + SHIFT + Left mouse click on Object	TabPlane Manipulator (Ability to move object in one plane based on which plane is selected and then translating perpendicular to that plane.)
CONTROL + SHIFT + Left mouse click on a green ball	Trackball Manipulator (Ability to rotate an object in any spherical direction around a center point calculated as being opposite the point selected.)
CONTROL + SHIFT + Left mouse click on white corner	Scale1 Manipulator (Ability to perform uniform scaling of the selected object in the direction the corner is "pulled" while maintaining a stationary point.)
CONTROL + SHIFT + Left mouse click on line segment	TabPlane Manipulator (Ability to move object in one plane based on which plane is selected and then translating perpendicular to that plane.)

Centerball Manipulator

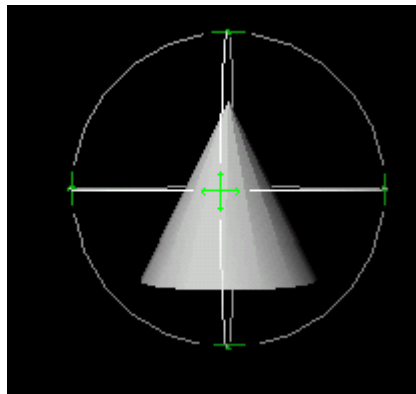


Figure 162: Centerball Manipulator

The Centerball Manipulator allows the user to change the position of the center point of an object. If we were to apply a cylinder sensor to a box and use that sensor to rotate the box, the box would rotate around the calculated center of the box as if a rod were inserted through its center. If, however, we move the center of the box to an outside corner and apply the cylinder sensor as before, we would have the box rotating around the rod as though the rod were attached to the corner of the box. This works well for simulating the action of a hinged door.

PointLight Manipulator

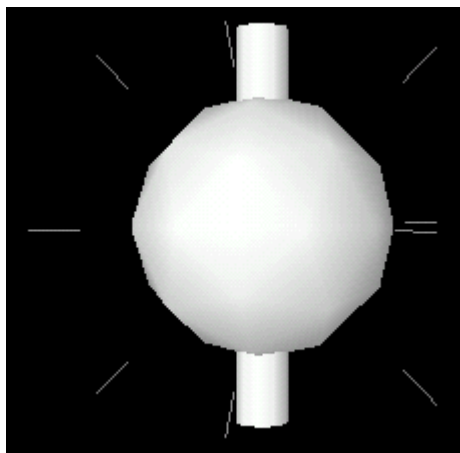


Figure 163: Point Light Manipulator

The PointLight Manipulator allows the user to alter the ambient intensity, attenuation, color, intensity, and radius of a PointLight source. The light can be set to ON or OFF. The manipulator itself can change only the location of the light source while the other fields can be changed in the Tree View Window by altering the

field values. To change to position of the light source, select the manipulator in the scene window and drag the manipulator to the desired position.

SpotLight Manipulator

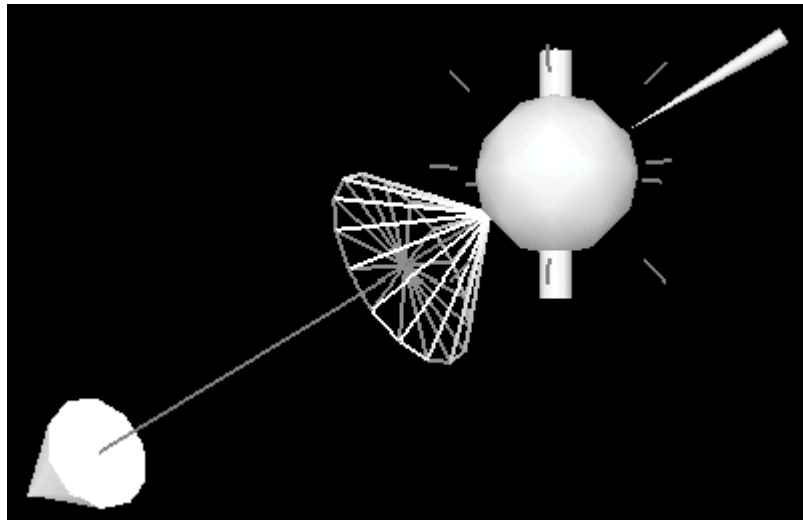


Figure 164: Spot Light Manipulator

The SpotLight Manipulator has the appearance in the scene of a sphere with an arrow through it. The tip of the arrow is a large cone and the opposite end is a small cone pointing in the same direction. Just past the Sphere is another cone pointing at the sphere. The sphere has a small cylinder passing through it like a spinning top.

The direction of the SpotLight can be changed by grabbing either of the cones at either end of the arrow. The location of the light source in the XZ plane can be changed by selecting the sphere and dragging it. The location in the Y plane can be changed by selecting the vertical cylinder through the sphere and pulling it up or down. The **beamWidth** of the SpotLight can be changed by selecting the cone pointing backwards at the sphere and pulling it in/outwards to flare the cone to the desired width. All other fields of the SpotLight are changed by altering the field values in the Tree View Window these fields include **ambient intensity**, **attenuation**, **color**, **cutOffAngle**, **intensity**, and **radius**. The light can be set to ON or OFF.

DirectionalLight Manipulator

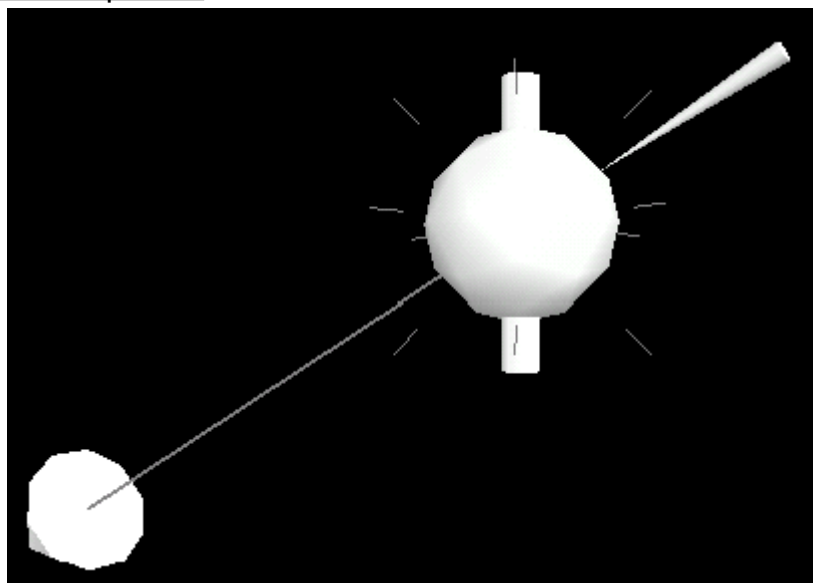


Figure 165: Directional Light Manipulator

The DirectionalLight Manipulator has the appearance in the scene of a sphere with an arrow through it. The tip of the arrow is a large cone and the opposite end is a small cone pointing in the same direction. The sphere has a small cylinder passing through it like a spinning top.

The direction of the DirectionalLight can be changed by grabbing either of the cones at either end of the arrow. The location of the light source in the XZ plane can be changed by selecting the sphere and dragging it. The location in the Y plane can be changed by selecting the vertical cylinder through the sphere and pulling it up or down. The DirectionalLight will only light geometry found under its parent transform, no matter where it is located. All other fields of the DirectionalLight are changed by altering the field values in the Tree View Window. These fields include ambient intensity, color, and intensity. The light can be set to ON or OFF.

Appendix D - Glossary

anchor

A type of VRML node used to link 3D scenes.

appearance node

A node of type Appearance, FontStyle, ImageTexture, Material, MovieTexture, PixelTexture, or TextureTransform. Appearance nodes control the rendered appearance of the geometry nodes with which they are associated.

apply

Apply is a term used when attaching a shell to an object. For example, applying a coat of paint to a house. This shell can refer to textures, materials, colors, or transforms.

ASCII

American Standard Code for Information Interchange. This standard defines and codifies the standard character set for computer communications.

avatar

A 3D incarnation assumed by a user in a 3D multi-user world.

backface removal

Elimination of portions of a rendered object turned away from the viewer.

bindable leaf node

A node of type Background, Fog, NavigationInfo, or Viewpoint. These nodes may have many instances in a scene graph, but only one instance can be active at any instant of time.

bitmap

A bitmap is a pixel representation of a picture. Pixels have varied value depending on the color map used. Most of the bitmaps used in the V-Realm Builder are high color (65,535) or true color (16.7 million) bitmap images. Bitmaps are used in VRML to create textures and materials. These **texture maps** are applied to objects and surfaces.

bounding box

Refers to a geometry selection mode where the object selected is wrapped with a shape (usually a wireframe cube) to outline the extent of the object in 3D space.

browser

Software that provides a means of viewing and navigating a 3D world.

camera

A camera is the viewpoint from which scenes are viewed in the V-Realm Browser. The perspective from which you see images is that of the viewport of a camera. It is as if you walked around looking through the lens.

camera height

Camera height refers to the height of the camera from the ground, or the zero X-Z axis plane from which the camera is elevated. A value of 10 specifies a height of 10 camera units from the zero X-Z plane.

camera position

Camera position refers to a point where a perspective camera is placed. In the V-Realm Builder these positions can be created anywhere in the scene. But for clarity they should be defined first in the scene so

that all the camera positions can be loaded before the rest of the world. In the V-Realm Browser the camera positions can be created by the creator of the world (pre-defined positions) or by the user (user defined positions).

children

Children are nodes that are subordinate to or are part of a larger group. For example, a building node can have child nodes of doors, windows, offices, and walls. If a modification to the building occurs, such as moving the building from one position to another, the modification will affect the windows, doors, offices, and walls as well.

children nodes

Nodes which are parented by grouping nodes and thus are affected by the transformations of all ancestors. Anchor, Background, Billboard, Collision, ColorInterpolator, CoordinateInterpolator, CylinderSensor, DirectionalLight, Fog, Group, Inline, LOD, NavigationInfo, NormalInterpolator, OrientationInterpolator, PlaneSensor, PointLight, PositionInterpolator, ProximitySensor, ScalarInterpolator, Script, Shape, Sound, SpotLight, SphereSensor, Switch, TimeSensor, TouchSensor, Transform, Viewpoint, VisibilitySensor, WorldInfo, and any PROTO'd child nodes are all valid children nodes.

collision detection

The process of determining whether two objects, or the viewer and an object, have made contact in a virtual environment. This is used as a feature in 3D Web browsers to prohibit viewers from walking through objects.

color model

Characterization of a color space in terms of explicit parameters. VRML allows colors to be defined only with the RGB color model.

continuous movement

Continuous movement is initiated when the user begins movement inside the current scene by double-left clicking with the mouse. As the user moves the mouse, movement in the scene will track with the movement of the mouse. This makes it easier to navigate a scene because constant directional clicking is no longer necessary. Continuous movement is stopped by left clicking the mouse button, on the scene or movement pad.

control point

Control point defines a set of coordinates where a knot vector is attached to a **nurbs** surface or plane.

coordinates

A coordinate is the numerical value of the X, Y, and Z values that define a point in space. This point can be 2D or 3D and so on. Coordinates are essential to building complex shapes and complex worlds.

culling Removing portions of a scene that are not visible to the viewer as a method of increasing performance.

display device

A graphics device on which VRML scenes can be represented.

directional light

Directional light has a surface light source and a direction. An example of this could be the Halogen lights used in ball parks. These lights are affixed to poles and are directed onto the field. The light does not shine all around the poles but only in the direction pointed.

dragger

A dragger is a set of tools that allows the user to turn, pull, push, and rotate objects in space. Manipulators use draggers to manipulate objects.

drag sensor

Drag sensors (CylinderSensor, PlaneSensor, SphereSensor) cause events to be generated in response to pointer motions which are sensor-dependent. For example, the SphereSensor generates spherical rotation events.

event

Messages sent from one node to another as defined by a route. Events signal changes to field values, external stimuli, interactions between nodes, etc.

exposed field

A field which can receive events to change its value(s) and generate events when its value(s) changes.

execution model

The characterization of the way in which scripts execute within the context of VRML.

external prototype

Prototypes defined in external files and referenced by a URL.

Fan-in and Fan-out

Fan-in occurs when multiple events are routed to the same eventIn. If two events with different values but the same timestamp are received at an eventIn, the results are not defined and thus should be avoided. The case where multiple eventOuts are created from a single eventIn is perfectly legal and results in multiple events being created from a single action, which is called fan-out.

fields

Allow you to define specific attributes for nodes, such as the exact size, color, position, and scale.

flat shading rendering

A fast rendering style that does no color enhancement calculations for colors applied to faces of geometry. This yields objects that are textured blandly but with striking contrast between the different colors applied to the faces. Where shading is enhanced by light sources, shadows or other factors, the flat color is displayed with no enhancements.

fly mode

Fly mode is a mode of the V-Realm Browser that resembles the action of flying a airplane. The controls are similar to those of an airplane.

geometry node

Nodes of type Box, Cone, Cylinder, ElevationGrid, Extrusion, IndexedFaceSet, IndexedLineSet, PointSet, Sphere, and Text which contain mathematical descriptions of three-dimensional points, lines, surfaces, text strings, and solid objects.

geometric sensor node

A node of type ProximitySensor, VisibilitySensor, TouchSensor, CylinderSensor, PlaneSensor, or SphereSensor. These nodes generate events based on user actions, such as a mouse click or navigating close to a particular object.

Gouraud shading rendering

See *smooth shading rendering*.

goto

Goto is the action of moving from one position in a VRML world to another in direct fashion. This would have the effect of beaming from one camera location to the next instantly.

gravity A feature in 3D Web browsers that allows viewers to travel up stairs and on top of obstacles rather than through them.

grouping node

A node of type Anchor, Billboard, Collision, Group, or Transform. These nodes group child nodes and other grouping nodes together and cause the group to exhibit special behavior which is dependent on the node type.

group node

A set of objects bound together by a common parent. Although it is possible to build a virtual apple tree of individual apples, a tree, and individual leaves, it is more logical to build a tree as a group node with apples being a subset of the group node and the leaves being another subset of the group node.

GUI

Graphical User Interface is the means by which a user interacts with a program.

headlight

A type of light used to highlight scenes. If a world does not normally have a light source, this option acts like a flashlight to light your path as you explore a new world. In worlds that have light, it will only alter the light amount making some objects brighter when you are facing them and darker when you turn away.

hidden line rendering

Hidden line rendering yields a flat shading view of a scene but does not fill in the texture or geometry that is hidden from view. Here, rendering time is decreased by giving full rendering of "front" objects, not wasting time rendering areas that are "out of sight". Shading, textures, and light sources are not displayed.

HTML

Hypertext Markup Language is the text format used to describe documents on the World Wide Web. HTML is the language used to "mark up" a document; it allows the author to define elements such as headers, paragraph boundaries, and text formatting.

HTTP

Hypertext Transfer Protocol is the communications protocol used by World Wide Web servers.

icon

An icon is a button on the screen that can be clicked on by the mouse to execute a desired action.

image

An image is any scene being viewed on the screen.

index

The term index is used by many of the nodes you encounter building worlds. The index variable refers to the start index required by many nodes to apply a material or texture to an object.

Indexed Face Set

An Indexed Face Set Node is a VRML node that allows the user to build complex shapes from a series of indexed faces.

Indexed Line Set

An Indexed Line Set Node is a VRML node that allows the user to build complex shapes from a series of indexed lines.

Inline node

A VRML node that allows another world file to be loaded into the current world. It is often used in conjunction with the LOD node to allow only the linked world file to be loaded.

instance

An instancing of a previously defined node created by the **USE** syntax.

Instancing

Instancing is accomplished by the use of the **DEF** and **USE** keywords. The **DEF** keyword defines a node's name and creates a node of that type. The **USE** keyword indicates that a reference to a previously named node should be inserted into the scene graph. This has the effect of sharing a single node in more than one location in the scene graph. If the node is modified then all references to the node are modified. If multiple nodes are given the same **DEF** name, the last **DEF** encountered during parsing will be used for the **USE** definition.

interpolator node

A node of type ColorInterpolator, CoordinateInterpolator, NormalInterpolator, OrientationInterpolator, PositionInterpolator, or ScalarInterpolator. These nodes define a piece-wise linear interpolation of a particular type of value at specified times.

IS Statement

The IS keyword associates fields of a prototype with eventIns or exposedFields defined in a prototype. The IS keyword acts like a route statement.

JPEG

Joint Photographic Experts Group. JPEG is a standard file format for 2D pixel image representation of an image.

jump

Jump generally refers to the ability to go directly to a world or position in a world by clicking the mouse. In documentation this refers to going to a new URL position.

jump to

This generally means that the desired position should be achieved by jumping to it from the current position. In most cases, this is performed by using the camera position icon to recall a list of positions to be reached.

library

A collection of objects or nodes that represent some function or geometry. The items can be selected and added to the scene graph at any point. Libraries provided by the V-Realm Builder store objects, textures, and materials to speed the construction of worlds.

lights

A light is something that illuminates the darkness. It can have direction, intensity, and color attributes.

LOD node

The Level of Detail node is a VRML node that allows the level of detail of an object to change as the object is approached.

Loops

Loops can be contained in event cascades but the loop will be broken when an eventOut from a previous node attempts to initiate an event which has the same timestamp that a previous event has from the exact same node.

manipulator

A manipulator is a tool that is used to change the orientation or position of objects in the scene graph. The manipulator employs draggers to perform this function.

material

A material is defined as a color and transparency value for each of the color properties: ambient, emissive, specular, and diffuse.

matrix

A matrix is a set of numbers that hold information about a scene. The values in a matrix can be used to store rotation, position, translation, transformation, and other information necessary to maintain the realism of movement within the scene.

message

A message is defined as one of the following: a manually typed phrase that is relayed to another location, a sound file that is sent to relay information, a live audio chat as if the user were calling on a telephone, a full multi-media file that is sent to relay information, or any combination of the above with the purpose of conveying information from one client to another.

MIDI

Musical Instrument Digital Interface - a standard for digital music representation.

MIME

Multipurpose Internet Mail Extension is a system that defines the means to transfer various types of data files over the internet. It was originally developed for E-Mail, but MIME types are now an important element of the Web. Each type of data file has their own MIME type, which tells both the Web server and the client (browser) what to do with the data file. It is through this system that a Web browser knows what type of helper application to use when viewing different file types.

mode

Mode refers to the three ways to maneuver in the V-Realm Browser: **Fly**, **Walk**, and **Model**.

model mode

This is the movement mode that allows the user to "inspect" objects.

motion sensitivity

Motion sensitivity refers to the distance and speed of each step or individual movement in a world. The larger the number, the faster movement will be. The smaller the number, the slower the movement.

mouse double left click

The double-left click is used for continuous movement in walk and fly modes. Once movement has started, it will continue in the direction of the mouse until discontinued.

mouse left click

Clicking on the left mouse button selects the option pointed to by the mouse. This use of the left mouse button will initiate an action by the browser.

mouse movement

Mouse movement is achieved in several different ways. The first is to use the mouse in the viewport window to identify the direction you wish to move in. This is achieved by depressing the left mouse button and pulling it in the direction desired. Simply release the button to stop movement. The second way is to press the left mouse button in the movement pad in the general direction you want to move. The same movement as above will be initiated, except this will be one step at a time for each click of the mouse button. The third way is to start continuous movement by double-left clicking on the movement pad and allowing the constant movement of the scene to track with the movement direction of the mouse. Turn this option off before initiating other types of movement or commands. Turn this movement style off by double-left clicking on the mouse.

mouse right click

The right-click of the mouse in the view window will open the Pop-Up menu.

multimedia

Multimedia is a term used to describe the peripheral devices needed to play audio and video files that have now become a major part of internet information.

Navigation Pad

The navigation pad allows the user to move in any direction within a world.

node

The fundamental component of a scene graph in VRML. Nodes are abstractions of various real-world objects and concepts. Examples include spheres, lights, and material descriptions. Nodes contain fields and events. Messages are sent between nodes via routes.

node type

A required parameter for each node that describes, in general, its particular semantics. For example, Box, Group, Sound, and SpotLight.

object

An object is 3D geometry displayed on the screen. An object can be seen and modified by manipulators. An object is defined by a geometry, but can be characterized by anything that can be seen. A light is an object because it can be seen and manipulated.

OpenGL

A 3D graphics library developed by Silicon Graphics and a part of Windows NT and an add-on for Windows 95.

Open Inventor

An early scene description language developed by Silicon Graphics, from which VRML evolved.

Orthographic camera

An orthographic camera produces a parallel projection of a scene. This camera does not produce distortion. It views objects from several perspectives, front, back, top, back, bottom, etc.

Orthographic view

The orthographic view provides front, top, side, rear and bottom views from an infinite distance, all of which are perpendicular.

Perspective camera

Perspective camera is used to view .wrl files. It views the world from a camera lens perspective. Distortion occurs when the camera is rotated.

Perspective view

The perspective view is a view of a world from a specified distance and angle. This view simulates a camera and gives the user a feel of depth and width. In spite of image distortion, it gives the user a realistic impression of the relationships among objects in a world space.

plane

A surface defined by the intersection of at least three points in space.

point

An exact position in space represented by the values given to x, y, and usually z. When plotted, it yields exactly one definable spot in space.

point light

A point light is a beam of light projected from a single point. An example is a light bulb in a lamp.

Point Set

A Point Set Node is a VRML node that allows the user to build complex shapes from a series of indexed points.

polygon

A polygon is any 3D geometric shape displayed in a VRML world.

position

A position is a pre-defined or user-defined location that can be selected and displayed. The user can create new positions as needed. Position can also refer to the exact coordinates in the world-space of an object.

pre-defined positions

Pre-defined positions are positions that were created by the author of a world. These positions can be deleted but cannot be added to by the user. Any positions added to this world by the user will be added to the user-defined list of positions.

prototype

The definition of a new node type in terms of the nodes defined in this standard.

Prototypes

Recursive prototypes are illegal. The external prototype does not contain an inline implementation of the node type, instead the prototype implementation is fetched from a URL or URN. There are many possible uses of prototypes and external prototypes. Be familiar with the pros and cons of their use before using them.

reload

Reload opens a connection to a server to re-download the information requested.

rendering

Refers to the style of output delivered to the screen during the viewing of a 3D world. Some rendering styles are chosen for speed, others for realism. More complex rendering methods require more time to complete rendering. One feature available in the V-Realm Builder is the ability to render on-the-move. It provides a lesser rendering quality, and when movement stops, the higher rendering quality is restored.

rendering quality

Rendering quality refers to the "output realism" of the viewed scene.

restore scene

Restore scene will reset an initial image that was seen when the world was opened.

RGB

The VRML color model. Each color is represented as a combination of the three primary colors: red, green, and blue.

rotation

The action of turning an object about a single axis. An example of this is a revolving door.

route

The connection between a node generating an event and a node receiving the event.

Routes

Routes connect eventIns to eventOuts for different nodes in the scene. It is important to remember that the type of data from an eventIn must exactly match the eventOut of a node that has been connected via a route.

scale

Refers to the overall size of the world relative to the perspective camera. In VRML, scale is measured in camera units. If a world has a scale of 10, then it has 10 camera units in the world. If the scale is 1 then there is only one camera unit in the world.

scene

In VRML, the name given to a 3D world.

scene graph

In VRML, the list of VRML nodes in a .wrl file that define objects and properties comprising the 3D world.

Sensors

Sensor nodes generate events. They are activated by actions initiated by the user via the mouse or buttons. Once an event has been generated it can perform one task or, more commonly, it will begin an event cascade whereby the initial event sets off other events until its logical conclusion.

sensor node

A node of type Anchor, CylinderSensor, PlaneSensor, ProximitySensor, SphereSensor, TimeSensor, TouchSensor, or VisibilitySensor. These nodes detect changes and generate events. Geometric sensor nodes generate events based on user actions, such as a mouse click or navigating close to a particular object. TimeSensor nodes generate events at regular intervals in time.

smooth shading rendering

A rendering method using normals to calculate light color and intensity in such a way as to achieve a smooth realistic color across the entire object.

special group node

A node of type LOD, Inline, or Switch. These nodes are grouping nodes which exhibit special behavior, such as selecting one of many children to be rendered based on a dynamically changing parameter value, or dynamically loading its children from an external file.

spot light

A spot light is a light source that concentrates light on an area in a conical pattern.

surface

A surface is any shape having geometry.

texture

A texture is a 2D object that is applied to other 2D and 3D objects. It can be thought of as "wall paper" to be used in building VRML worlds.

texture coordinates

The set of 2D coordinates used by vertex-based geometry nodes (e.g. IndexedFaceSet and ElevationGrid) and specified in the TextureCoordinate node to map textures to the vertices of some geometry nodes. Texture coordinates range from 0 to 1 across the texture image.

texture transform

A node which defines a 2D transformation that is applied to texture coordinates.

toolbox

The toolbox is used to set many of the standard viewer settings.

transfer

Transfer refers to the act of downloading information from one computer to another.

transform

A transform defines a geometric transformation consisting of a translation, rotation, scale, scale orientation, and center point.

transformation

Alteration of the geometry or physical location of an object in a 3D scene.

translation

A set of coordinates used to change an object. Translation usually takes the form of movement in coordinates. For example, the translation (1,0,0) would mean to move a specified object one unit in the positive x direction.

traverse

The action of stepping through the nodes of a scene graph to ensure that all appropriate actions are taken as specified by the nodes in the scene graph. A traversal is required after any change to the scene graph is made.

URL

Uniform Resource Locator is the address used to refer to particular documents on the World Wide Web. An example is *http://www.ligos.com*, which points to the home page for Ligos.

user-defined positions

A user-defined position is a position that is saved by the user in a particular world.

view area

The view area is defined as that which is displayed on the screen. This is the area of the world currently in view.

view mode

View mode refers to the chosen style of viewing. It encompasses both the movement mode and rendering quality.

Viewpoint Nodes

Viewpoint Nodes are like camera positions that can be browsed or jumped to. They can also act as a transporter location to which people browsing a world can beam quickly.

VRML

Virtual Reality Modeling Language is a portable scene description language developed to create 3D objects for the World Wide Web.

VRML file

A file containing information encoded according to the VRML standard.

VRML file header A VRML file should begin with the line `#VRML V2.0 utf8`.

walk mode

The movement mode in the V-Realm Browser that simulates the human action of walking.

web

Refers to the network of computers and communications devices that connect them.

wire frame rendering

A rendering style that shows a line drawing around all objects in the scene and how they relate to each other in size and shape. Shading, texture mapping, and light sources are not displayed.

world

Refers to a 3D scene graph saved in `.wrl` file format that can be displayed by a VRML browser. A world can contain other objects, worlds, or files.

wrl

This is the file extension for VRML world files.

