Contents

Pueblo

Pueblo overview

Over the past several years, people have been building worlds on the Internet.   People have been constructing castles, designing starships, and digging hobbit-holes.   Some of these worlds are serious, simple meeting places where you can speak French or Italian.   Some worlds are much less serious.

Pueblo does two things.   It gives you a list of these worlds, categorized to help make your browsing easy.   The list is broken down by categories such as 'Social' and 'Educational'.   It's also broken down (for the techies) by the type of software the world is using.

The world browser is hypertext, which mean that you can click on things.   If you're looking at a list of categories, you can click on one to see the worlds in that category.   If you're looking at the description of a world that intrigues you, just click on the name of that world and Pueblo will try to connect you.

That's the second thing that Pueblo does.   Pueblo allows you to enter these worlds, using a friendly windowed interface.

We hope you enjoy using Pueblo.   If you have any comments, please send mail to **pueblo-support@chaco.com**.   We **will** listen.

Topics

Connecting to the Pueblo Server

Connecting to the Pueblo Server

To connect to the Pueblo server, and use Pueblo, you must have a TCP/IP connection to the Internet. This may be a direct connection, a SLIP connection, or a PPP connection.

A good way to check whether you have an active Internet connection is to open a DOS window and use the `ping` command. The following command will test your connection:

```
    ping chaco.com
```

If you get a reply from Chaco, or you see the message 'Chaco is alive', then you have a valid and active Internet connection.

If you do have an Internet connection, then Pueblo should work with your machine. To log in to the Pueblo server, simply select **Login** from the **File** menu. You may also press the button in the toolbar. This will initiate a connection to the Pueblo server and the Login dialog box will appear.

Your Pueblo Account

Your Pueblo account allows you to download world lists from us. We intend to keep these world lists up-to-date, eliminating worlds that no longer exist and adding worlds that we discover. If we're missing a world that you would like to let us know about, you can send a letter to **world-list@chaco.com**. Include the world name, address, and a short description of the world. If you wish, you can also suggest a category for the world (i.e., Educational, Social, etc.)

NOTE that your Pueblo account is not associated with your name and password for each world you visit.

What's New?

Pueblo is an evolving product. Here is a list of features that we've included in each version. Remember that if you really want a feature added, you can send us mail at **pueblo-beta@chaco.com**, and we'll see what we can do. You can also use the 'Submit Bug Report' item from the 'Help' menu in Pueblo.

Release 1.00 (our first 'official' release)

Features added:

- New user registration is now done using a Registration Wizard, which leads the new user through a series of questions necessary to register.
- Pueblo now supports the creation of 'shortcut' files in the Start menu (on Windows 95) or in the Program Manager (on Windows NT). Once a shortcut is created, you can select it to immediately log into that world.
- In the personal world list, you can now enter your username and password for the world. If you've entered a username and password, Pueblo will automatically log you into the world (when you connect from your personal world list.)
- Pueblo now supports simple logging of output to a file. Output can be logged in either raw text or HTML format.
- A new 'Notification' preference page allows you to get notified when the application is iconic and new text arrives. You can select to either play a sound, or blink the Pueblo icon, or both. In addition, you can only notify if incoming text matches text you specify.
- Pueblo now supports a 'stop' button on the toolbar. Pressing the stop button aborts connecting to a world and file downloads.
- Users can generate bug reports with a form in Pueblo. You can display this form by selecting 'Submit Bug Report' from the 'Help' menu.
- The world list has been redesigned with a multi-level hierarchy and graphics.
- The current world is now displayed in the caption.
- The status bar now displays better information on download status.
- The help system now supports the Windows 95 look and feel.

HTML (rich text) improvements:

- Pueblo now supports image maps, with the click coordinates being sent to the world server with a author-specified command.
- Pueblo allows web pages to be loaded using normal HTML (<a href=...>load</a>). When a web page is loaded, Pueblo allows you to either use an external browser (such as Netscape Navigator™) or Web Tracker, the internal Pueblo web browser, which displays in a separate window.
- Pueblo now includes full support for HTML forms.
- Pueblo now supports displaying HTML pages (including graphics) in multiple, arbitrary windows, called panes. Using HTML forms, this allows a world programmer to present the user with dialog boxes. Other uses of panes are for displaying graphics (such as maps) or side text (such as when reading a book.)
- When URLs are displayed in the output window, they are automatically translated into clickable anchors. Clicking on the anchor causes that web page to be loaded.
- This help system now includes a full description of all of the HTML supported by Pueblo.
- Image anchors can now specify a border width of 0.
- Pueblo now supports the full HTML 3.0 standard, with the exception of tables.
- You can now select and copy text from the HTML output window.

VRML (3D graphics) improvements:

- Pueblo now uses the faster and better Intel 3DR 2.1 rendering engine.
- Many rendering improvements have been made, resulting in better-looking 3D scenes.
- Pueblo now supports collision detection when moving forward through 3D spaces.

- Anchors in VRML scenes are now more accurate.

Bugs Fixed:

- Some worlds used to have double-spaced output. This has been fixed.
- On high-color displays, display of 2D graphic files would sometimes cause Pueblo to crash. This has been fixed.

Release 0.90 (our first public beta release)

Actually, everything in this release was new, as it is our first public release.

How to...

- Access worlds not on Pueblo's list?
- Log into worlds automatically?
- Connect to the Pueblo server?
- Create a macro?
- Use ANSI color?

- Move through a 3D scene?

- Make a world Pueblo Enhanced?
- Add Pueblo enhancements to a TinyMUSH?
- Add Pueblo enhancements to an LP-MUD?
- Create and use dialogs in a world?
- Use image maps with worlds?

- Send us a feature request or bug report?

Worlds on the Internet

About Worlds

You're not alone.

First there was Adventure.   You entered a cave and explored.   There were other creatures there, but no people.   Adventure was fun, but it was lonely.

Then in 1979 Richard Bartle and Roy Trubshaw wrote MUD1.   MUD1 allowed several people to enter an online environment together, explore, and talk.   It was great.   You could explore the dungeon with your friends.   Since then there are a large number of different types of worlds that have been developed.

On a world, you can talk to people and slay monsters.   (In some worlds, you can slay knights in shining armor.)   You can work together with your friend to solve a puzzle, or speak Italian with a native.   You can smile, or dance, or hide behind yourself.

Unfortunately, the way you do these things on different worlds differ quite a bit, so we can't give you help here on how you actually **do** things on worlds.   Fortunately, most worlds have a tutorial for new players, and other players in the world are usually very friendly about helping out new players.   On most worlds, you can type **help**.

Get a Life (or Remember your Wife)

Worlds can be addictive.   Like a good drug, they allow you to do most anything your heart desires.   But please remember that there is a real world out there, where you need to eat and sleep and interact face to face.   We put a timer at the bottom of the Pueblo window so that you'd always know what time it is and how long you've been playing.

Topics

- Types of Worlds
- Creating a character
- Etiquette
- Dictionary of Terminology

Types of Worlds

There are many types of worlds on the Internet. They can be very roughly divided into combat-oriented and social worlds. This division is a fuzzy one. Some people join combat-oriented worlds and then spend most of their time socializing. Some socially-oriented worlds have combat areas. It's like a big family. Sometimes you talk, and sometimes you fight.

The division between combat and social worlds is influenced by the software that was used to develop the world. (This is called the world server.) Here are some of the names that you may see associated with worlds:

Combat-oriented software:

AberMUD	Circle	DGD
DikuMUD	LPMUD	

Social-oriented software:

TinyMUCK	TinyMUD	TinyMUSH

Hard to categorize software:

LambdaMoo	Moo	UberMUD

Creating a Character

Most worlds require that you create a character to play. The welcome message when you connect to the world should tell you how to get a character. If you are checking out a world for the first time, use the **guest** character before you create you own. This saves space on the world and allows you to see if you want to get more serious.

Many worlds are restricted. This means you have to send a mail message to the wizard and ask for a character. (Worlds end up restricted because the wizard had a bad experience with clueless newbies.)

Etiquette

Be nice. (Unless you're nasty.)

The Lay of the Land

When you first join a world, take the time to learn how people in that world relate to one another. You're a stranger in a strange land, a foreigner, a newbie. Like any stranger, you should watch the natives and learn how they relate to one another.

Different worlds have different customs. Most social worlds frown on abuse, either verbal or physical. Combat-oriented worlds may thrive on abuse. These customs are usually displayed somewhere near the entrance to the world.

Most worlds welcome new players. Most combat worlds don't slay new players, so you'll have some time to look around and get comfortable before someone slices your head off with a laser.

Role Playing

One of the wonderful things about these worlds is role playing. In fact, role playing is the entire point of some worlds. In addition to existing in the world, you can take a role in the world, such as Queen or duchess. (Not many people play peasants.)

Even in worlds where role playing isn't the focus, you can still take on a role. If you're a quiet woman and have always wanted to be a strutting hero, then do it. Speak loudly of your virtues, rescue fair lads from dragons, wield a sword, and hum Wagner.

Men can play women (and experience sexual harassment) and women can play men. Don't lead people on, though. You wouldn't like it if it happened to you, and it's not nice.

Remember that many worlds may have players who are kids. They may not look like kids... they may look like strutting heroes. Keep your bawdiness to a reasonable level, except perhaps on those worlds or portions of worlds where bawdiness is rampant.

Adding Worlds not in Pueblo's lists

If you want to access a world that is not on Pueblo's world list, you can add it to your personal world list. You can do this anytime you are logged into the Pueblo system.

You can access the personal world list dialog by first selecting the **Worlds** menu item from the main menu bar, and then selecting the **Edit Personal List...** item. A dialog will appear allowing you to edit your personal list.

To add a list, fill in the fields at the bottom of the dialog. You must fill in at least the world Name, Host, and Port field. You may also specify the type of world and a description that will appear in your personal world list.

Important: When you are finished entering the information, press the **Add** button to add the information to your list. (If you are changing an existing entry, the Add button will be replaced by a Change button.)

Once the information is added, you can then click on the Connect button to connect to a world selected in the list at the top of the dialog. You can also click on the world in your personal list.

Creating and editing shortcut files

Pueblo supports the ability to create desktop shortcuts for worlds. These appear as icons, either in the Start menu (on Windows 95) or in the Program Manager (on Windows 3.1 and Windows NT.) You can then run these icons, and Pueblo will automatically connect you to the world.

You create a shortcut file through the Shortcut Wizard. You can use the Shortcut Wizard when you're viewing the Pueblo world lists and when you're connected to a world. If you create a shortcut when you're viewing the world lists, then you'll need to know information about the world. The Shortcut Wizard will need to know the world's host name, port number, and server type. For some world types, you'll also need to know whether the world expects you to log in using '*connect username password*' or whether the username and password should be entered on separate lines.

If you're connected to a world and create a shortcut, then you'll have the option of creating a shortcut for the world you're currently using. If you do this, then the process is much simpler. You can optionally enter your username and password, and the process is complete.

When you're finished entering information about the world, the Shortcut Wizard will ask you to select a folder for the shortcut file. You can select either an existing name or enter a new name. The shortcut file will be created in the selected folder. (On Windows 95, the folder will be filed under 'Programs' in the Start menu.)

Once the shortcut file is created, select it from the Start menu (on Windows 95) or double click on the icon (in the Program Manager). The shortcut file will attempt to log you into the world using the following rules:

- Shortcuts are created for the current Pueblo user. If the current Pueblo account has a saved password, the user is automatically logged into Pueblo using that account. Otherwise, the user is asked to enter their Pueblo password.
- The user is connected to the world specified in the shortcut.
- If the shortcut includes a username and password, the user is logged into the world using that username and password. If only a username is specified and the world expects the username and password on separate lines, then the username is sent to the world.

Editing Shortcut files

Shortcut files are simply text files that may be edited using a standard text editor like Notepad. Normally you don't need to edit shortcut files… Pueblo will create them for you automatically. If you wish to change a file, however, you need to edit it 'by hand'.

Shortcut files are stored in a sub-directory called 'Shortcut' under the directory where Pueblo is installed. The file name is derived from the name you gave the shortcut, and ends in the extension 'pbl'.

The file consists of label/value pairs, with one pair per line. Each line is of the following form:

```
label = value
```

The label may be any of the following strings:

label:	Example value:	Description:
username	Coyote	This is the username to use when logging into the *Pueblo* server.
password	mypassword	This is the password to use when logging into the *Pueblo* server. This password should match the specified user name. This field is **not** automatically filled in when creating a shortcut file.
worldname	PuebloMUSH II	This is the name of the world.
worldserver	zuni.chaco.com	This is the host name for the world server.
worldport	4203	This is the port for the world server.

worldtype	tinymush	This is the server type for the world. Current legal values are *Aber*, *Circle*, *DGD*, *Diku*, *LPMud*, *Merc*, *Moo*, *Muck*, *Muse*, *Mux*, *Talker*, *TinyMush*, and *Other*. Types are case-insensitive. If you don't know the server type, use 'Other', and specify the 'logintype' label if the world uses a connect-style login.
worldusername	Trickster	This is the username you use to log into the world. This field is only filled in automatically if you entered this information in the Shortcut Wizard.
worldpassword	boyfriendsname	This is the password you use to log into the world. This field is only filled in automatically if you entered this information in the Shortcut Wizard.
logintype	connect	If this field is specified, it may only contain the value 'connect'. This field is used for worlds types that use both a connect-style and username/password-style login. For example, LPMud worlds always require the user to enter the username, press return, and then enter the password. TinyMush servers always require that the user type 'connect username password'. But Moo servers can be set up to allow login to work either way. For Moos that expect the connect-style login, this label must be specified for automatic login to work correctly.

Note: Passwords are encrypted if the shortcut file is created by the Shortcut Wizard. There is currently no way to encrypt passwords when editing the file directly. Pueblo will automatically recognize if a password is encrypted.

Sending us a feature request or a bug report

If you think of a neat feature you'd like to see in Pueblo, or find a bug, use the 'Submit Bug Report' entry under the 'Help' menu to send us a note about it!

You can also send us email to **pueblo-support@chaco.com**.

Menus

File menu commands

The File menu offers the following commands:

Login	Logs you into the Pueblo server.
Logout	Disconnects you from the Pueblo server.
Log to a file	Starts and stops saving world output to a file.
Exit	Exits Pueblo.

Edit menu commands

The Edit menu offers the following commands:

Undo	Reverse previous editing operation.
Cut	Deletes data from the document and moves it to the clipboard.
Copy	Copies data from the document to the clipboard.
Paste	Pastes data from the clipboard into the document.
Preferences...	Allows you to customize Pueblo features.

View menu commands

The View menu offers the following commands:

Previous	When browsing the world lists, flips you to the previously shown page.
Stop Loading	This menu item will terminate a load in progress.
Toggle pane orientation	This will toggle the two window panes between horizontal and vertical orientation. You can only do this when there is more than one pane visible.
Swap panes	This will swap the positions of the two panes when two panes are visible.
Toolbar	Shows or hides the toolbar.
Status Bar	Shows or hides the status bar.
Headlight	Turns your head-mounted light on and off for 3D scenes.

Worlds menu commands

The Worlds menu offers the following commands:

Edit Personal List...	Allows you to edit your personal list of favorite worlds.
Add to personal list	When you are connected to a world, this menu item adds that world to your personal list of worlds.
Create shortcut...	Allows you to create a desktop shortcut to a world using the Shortcut Wizard.
Disconnect	Disconnects you from a world.

Note: This menu will not be visible until you are logged into the Pueblo server.

Help menu commands

The Help menu offers the following commands, which provide you assistance with this application:

Index	Offers you an index to topics on which you can get help.
Using Help	Provides general instructions on using help.
Submit Bug Report	Allows you to send us a bug report or change request for Pueblo.
About Chaco	Displays information on Chaco Communications, Inc., the company that wrote Pueblo.
About VR Scout	Displays information on VR Scout, Chaco’s viewer for Virtual Reality Modeling Language files.
About Pueblo...	Displays information on the Pueblo client and Chaco Communications.

Exit (File menu)

Use this command to end your Pueblo session.   You can also use the Close command on the application Control menu.   If you're currently connected to the Pueblo server, you will be disconnected.

Shortcuts

Keys: ALT+F4

Login (File menu)

This command will start a connection to the Pueblo server. If your connection to Pueblo is successful, then you will be asked to supply a user name and password in the Login dialog box. (If you do not have a user name and password, you can always create a new account.)

Shortcuts

Toolbar:

Logout (File menu)

If you are connected to the Pueblo server, this command will terminate the connection.   You'll be prompted to make sure this is what you really want to do.

Shortcuts

Toolbar:

➪ *Log to a file* (File menu)

This command allows you to save incoming text to a specified file. You can choose whether text should be stored in plain text or raw HTML formats. This menu item is only available when you are connected to a world.

When this menu item is unchecked, selecting it will cause a file selection dialog to appear. Browse to the directory where you wish to store the file, and then type in the file name that you wish to use. You may check the box at the bottom of the dialog to include all text that is currently in the output window.

You may also select the type of data to save, according to the extension you specify. If you create a file with the extension 'htm' or 'html', then that log file will include all of the HTML directives. Otherwise the file will consist of plain text.

When this menu item is checked, then selecting the item will stop file logging.

Note: This menu item will not be visible until you are logged into the Pueblo server.

➪ *Undo/Can't Undo* (Edit menu)

Use this command to reverse the last editing action, if possible. The name of the command changes, depending on what the last action was. The Undo command changes to Can't Undo on the menu if you cannot reverse your last action.

Shortcuts

Keys:	CTRL+Z or ALT-BACKSPACE

Cut (Edit menu)

Use this command to remove the currently selected data from the document and put it on the clipboard. This command is unavailable if there is no data currently selected.

Cutting data to the clipboard replaces the contents previously stored there.

Shortcuts

Toolbar:

Keys: CTRL+X

Copy (Edit menu)

Use this command to copy selected data onto the clipboard.   This command is unavailable if there is no data currently selected.

Copying data to the clipboard replaces the contents previously stored there.

Shortcuts

Toolbar:

Keys: CTRL+C

Paste (Edit menu)

Use this command to insert a copy of the clipboard contents at the insertion point.   This command is unavailable if the clipboard is empty.

Shortcuts

Toolbar:

Keys: CTRL+V

Preferences (Edit menu)

This command will bring up the preferences dialog box.   You can use this dialog to customize Pueblo's behavior.

Previous (View menu)

When you're looking at the Chaco world lists, this item will flip you back to the last list you were looking at before the current list.   If there was no previous list, this item will be disabled.

Shortcuts

Toolbar: ⇦

Stop Loading (View menu)

This menu item will terminate a load in progress.   You should use this command if it appears that Pueblo has hung while loading a page, or if the load takes an incredibly long amount of time.

Toggle pane orientation (View menu)

This item will toggle the orientation of the two panes (usually text and graphics) from horizontal to vertical.

Shortcuts

Toolbar:

Swap panes (View menu)

This item will swap the positions of the two panes.

Shortcuts

Toolbar:

Toolbar **(View menu)**

Use this command to display and hide the Toolbar, which includes buttons for some of the most common commands in Pueblo, such as File Open.   A check mark appears next to the menu item when the Toolbar is displayed.

Also see

Toolbar   -- help on using the toolbar.

Status Bar (View menu)

Use this command to display and hide the Status Bar, which describes the action to be executed by the selected menu item or depressed toolbar button, and keyboard latch state. A check mark appears next to the menu item when the Status Bar is displayed.

Also see

Status Bar -- help on using the status bar.

Headlight (View menu)

In VRML scenes, there is a headlight strapped to your virtual forehead. You can use this headlight to illuminate scenes that are too dark to see. Selecting this menu item toggles the headlight between ‘on’ and ‘off’. Using the Preferences menu item in the Edit menu, you can also change the brightness of the headlight.

Note: This menu item will not be visible until you are logged into the Pueblo server.

Edit personal list... (Worlds menu)

This command will allow you to add, edit, and delete items in your personal world list. A dialog will appear listing the worlds in your personal list. (The list of worlds may be empty.) To add a world, select the ‘Add’ button. To edit an entry in your personal world list, select that world and click on the ‘Edit’ button.

To remove an entry from your personal world list, select the world and click on the ‘Delete’ button.

The ‘Connect’ button will connect you to the selected world. If you are already connected to a world, then the ‘Connect’ button will be disabled.

Press the ‘Okay’ button to save your changes. If you press the ‘Cancel’ button, then none of the changes you’ve made to your personal world list will be saved.

Note: This menu item will not be visible until you are logged into the Pueblo server.

Add to personal list (Worlds menu)

This command will add an entry for the currently connected world to your personal worlds list. You may then edit your personal list to enter a description for the world, or enter a user name and password for automatic login.

Note: This menu item will not be visible until you are logged into the Pueblo server.

Create shortcut… (Worlds menu)

This command will bring up the Shortcut Wizard.   This wizard will take you step-by-step through creating a shortcut to a world.

Note: This menu item will not be visible until you are logged into the Pueblo server.

Also see

Creating and editing shortcut files

➪ *Disconnect* (Worlds menu)

This menu item will immediately disconnect you from the current world. This command simply severs the connection, so the world may think that you had connection problems and may try to preserve your character for a while. This is the equivalent of hanging up the telephone without saying ‘goodbye.’

A better way to leave a world is through the worlds’ **quit**, **QUIT**, or **@quit** command. (The commands vary depending on the type of the world, but one of the preceding three commands usually works.)

➪ *Index* (Help menu)

Use this command to display the opening screen of Help. From the opening screen, you can jump to step-by-step instructions for using Pueblo and various types of reference information.

Once you open Help, you can click the Contents button whenever you want to return to the opening screen.

Using Help (Help menu)

Use this command for instructions about using Help.

⇨ *Submit Bug Report* (Help menu)

This command allows you to send a bug report to the Pueblo team. This downloads a form to your computer and then displays the form in either Netscape or the internal Pueblo web browser.

Fill out the form with your bug or change request, and then click on the ‘Send Bug Report’ button at the bottom of the page. The information you supply will be sent to us, and you can then close the window.

About Pueblo (Help menu)

This command shows a dialog box which contains information about Pueblo and Chaco Communications, Inc.   Among other information, this displays the version number and copyright notice for Pueblo.

➪ *About VR Scout* (Help menu)

This command will bring up a Web page describing VR Scout, Chaco Communications' way-cool viewer for Virtual Reality Modeling Language (VRML) files. VR Scout is available as a plugin for NetManage Chameleon and Netscape Navigator, as well as a separate viewer application that will work with most other Web applications. Pueblo incorporates VR Scout technology.

If Pueblo finds Netscape Navigator on your system, the web page should appear in Navigator. Otherwise the page will be displayed in Pueblo's internal web browser.

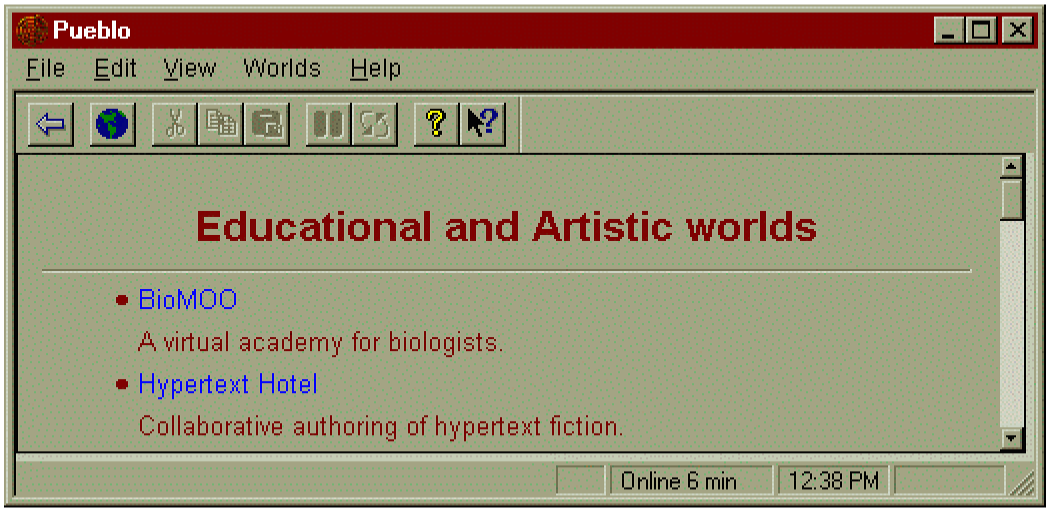
➪ *About Chaco* (Help menu)

This command will bring up a World-Wide Web page describing Chaco Communications, Inc. If Pueblo finds Netscape Navigator on your system, the web page should appear in Navigator. Otherwise the page will be displayed in Pueblo’s internal web browser.

User Interface

The Pueblo user interface (world list)

(Browsing the world list)



Educational and Artistic worlds

- BioMOO

  A virtual academy for biologists.
- Hypertext Hotel

  Collaborative authoring of hypertext fiction.

(You may click on this picture to get help on parts of the user interface.)

Pueblo allows you to connect to the many worlds people have built on the internet.

When you first connect to Pueblo, you're presented with a world list browser. You may select a world category to see the worlds in that category. When looking at a list of worlds, click on a world to attempt connecting to that world.

Help is also available for the user interface when you're connected to a world.

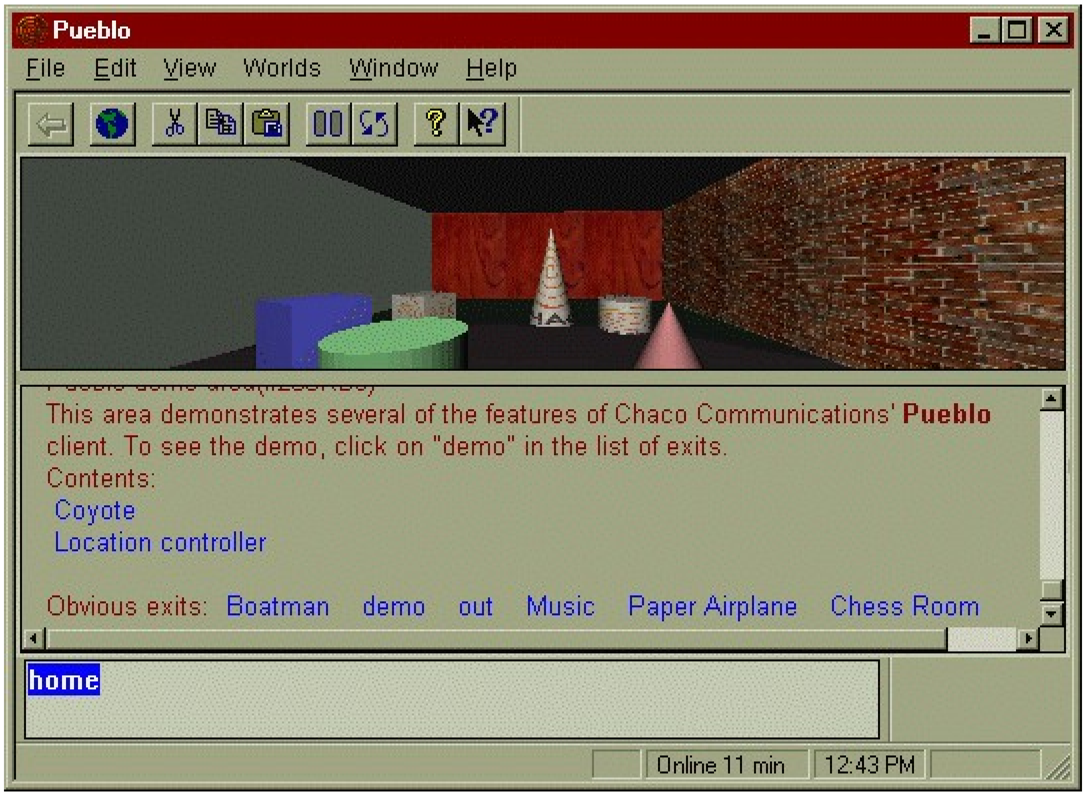
Also see

Creating and using macros

Creating shortcut files

The Pueblo user interface (connected)

(Connected to a world)



This area demonstrates several of the features of Chaco Communications' **Pueblo** client. To see the demo, click on "demo" in the list of exits.
Contents:
Coyote
Location controller

Obvious exits: Boatman demo out Music Paper Airplane Chess Room

home

(You may click on this picture to get help on parts of the user interface.)

When you are connected to a normal world, only the text pane is visible, and it occupies the entire content area of the window. When you are connected to a Pueblo Enhanced world, however, a graphics pane may appear, allowing you to see images associated with the world.

The image above shows a connection to a Pueblo Enhanced world, and the top pane of the window shows a 3D image of a room.

Also see

Creating and using macros

Creating shortcut files

The Input window

The input window allows you to send text to the world. When you press the Enter key, Pueblo sends whatever is in the input window.   (If the input window is empty, then you send a blank line.)

The window also keeps track of what you've entered in the past (this is called a *command history*.)   Holding the control key while pressing the up arrow moves you back through the history, while pressing control and the down arrow moves you forward.   As you move through the history, the text in the input window is replaced.   The command isn't sent until you press the Enter key.

A short cut to using the history is to click the right mouse button while the pointer is in the window. A list of your commands will pop up, and you may choose one.   Long items in the menu show up cut short with ellipses, but they will be sent full-length.   When you select something from the pop-up menu, it is sent immediately.   You don't need to press the Enter key.

Also see

Creating and using macros

The Graphics pane

This pane is used to display graphical information, including 2d images and 3d VRML rendered scenes (like that shown above, only better.)

If the scene is a VRML images, you can move through the image using the keyboard or the mouse.

Also see

What is VRML?

How to move through a VRML scene

Creating macros

Pueblo currently supports the ability to use very simple macros. (We know that you want more. So do we. Next release…) They're so simple, the term 'macro' is probably incorrect, but it reflects our goals.

In order to use or edit macros, right-click on the input window. A menu will appear. This menu contains the last 5 commands that you sent to the world. You can click on a command to send that command again.

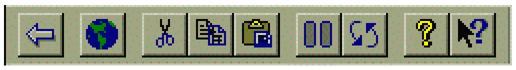
The first item in the menu is the 'Edit macros…' item. Selecting this item causes the *Personal Macro List* dialog to appear. This dialog lists all of the macros that you've currently defined, and allows you to add, edit, and delete macros.

Selecting the Add or Edit buttons brings up the *Macro Information* dialog. You can use this dialog to enter or change a macro. For now, macros only consist of a label (for identification) and the text that is sent. You may also select to have the macro displayed in the macro pop-up menu.

Macros may be associated with a specific type of world or a specific world. You can use the *Macro Information* dialog to select a world type, and the macro will only be active for worlds of that type. You can also type in a world name, and the macro will only appear for that world.

Once a macro is entered, it will appear on the macro pop-up menu. You can right-click on the text input window, and the menu will appear. Macros will be shown right after the 'Edit macros' item, and before the recent history items. Selecting the macro label will cause the associated text to be sent to the world.

The Toolbar

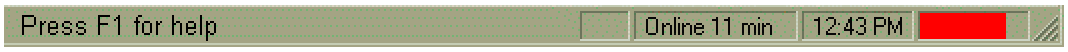


(You may click on this picture to get help on parts of the toolbar.)

The toolbar is displayed across the top of the application window, below the menu bar.   The toolbar provides quick mouse access to many tools used in Pueblo,

To hide or display the Toolbar, choose Toolbar from the View menu (ALT, V, T).

The Status Bar



(You may click on this picture to get help on parts of the status bar.)

The status bar is displayed at the bottom of the Pueblo window. To display or hide the status bar, use the Status Bar command in the View menu.

The left area of the status bar describes actions of menu items as you use the arrow keys to navigate through menus. This area similarly shows messages that describe the actions of toolbar buttons as you depress them, before releasing them. If after viewing the description of the toolbar button command you wish not to execute the command, then release the mouse button while the pointer is off the toolbar button.

Title Bar

The title bar is located along the top of a window.   It contains the name of the application and document.

To move the window, drag the title bar.   Note: You can also move dialog boxes by dragging their title bars.

The title bar contains the following elements:

- Application Control-menu button
- Maximize button
- Minimize button
- Name of the application
- Restore button

Context Help command

Use the Context Help command to obtain help on some portion of Pueblo. When you choose the Toolbar's Context Help button, the mouse pointer will change to . Then click somewhere in the Pueblo window, such as another Toolbar button. The Help topic will be shown for the item you clicked.

Shortcut

Keys:	SHIFT+F1

Scroll bars

Displayed at the right and bottom edges of the document window.   The scroll boxes inside the scroll bars indicate your vertical and horizontal location in the document.   You can use the mouse to scroll to other parts of the document.

Size (System menu)

Use this command to display a four-headed arrow so you can size the active window with the arrow keys.

After the pointer changes to the four-headed arrow:

1. Press one of the DIRECTION keys (left, right, up, or down arrow key) to move the pointer to the border you want to move.

2. Press a DIRECTION key to move the border.

3. Press ENTER when the window is the size you want.

Note: This command is unavailable if you maximize the window.

Shortcut

Mouse: Drag the size bars at the corners or edges of the window.

Move (Control menu)

Use this command to display a four-headed arrow so you can move the active window or dialog box with the arrow keys.

Note: This command is unavailable if you maximize the window.

Shortcut

Keys:	CTRL+F7

Minimize (application Control menu)

Use this command to reduce the Pueblo window to an icon.

Shortcut

Mouse: Click the minimize icon on the title bar.

Keys: ALT+F9

Maximize (System menu)

Use this command to enlarge the active window to fill the available space.

Shortcut

Mouse: Click the maximize icon on the title bar; or double-click the title bar.

Keys: CTRL+F10 enlarges a document window.

Next Window (document Control menu)

Use this command to switch to the next open document window. Pueblo determines which window is next according to the order in which you opened the windows.

Shortcut

Keys:	CTRL+F6

Previous Window (document Control menu)

Use this command to switch to the previous open document window. Pueblo determines which window is previous according to the order in which you opened the windows.

Shortcut

Keys:	SHIFT+CTRL+F6

Close (Control menus)

Use this command to close the active window or dialog box. Double-clicking a Control-menu box is the same as choosing the Close command.

Shortcuts

Keys:	ALT+F4 closes the Pueblo application.

Restore (Control menu)

Use this command to return the active window to its size and position before you chose the Maximize or Minimize command.

Switch to (application Control menu)

Use this command to display a list of all open applications.   Use this 'Task List' to switch to or close an application on the list.

Shortcut

Keys: CTRL+ESC

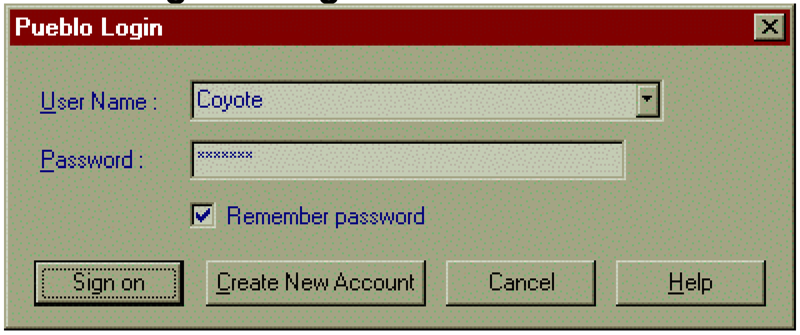
Dialog Box Options

When you choose the Switch To command, you will be presented with a dialog box with the following options:

Task List	Select the application you want to switch to or close.
Switch To	Makes the selected application active.
End Task	Closes the selected application.
Cancel	Closes the Task List box.
Cascade	Arranges open applications so they overlap and you can see each title bar. This option does not affect applications reduced to icons.
Tile	Arranges open applications into windows that do not overlap. This option does not affect applications reduced to icons.
Arrange Icons	Arranges the icons of all minimized applications across the bottom of the screen.

Dialog boxes

The Login dialog box


Pueblo Login

User Name : Coyote

Password : *******

✓ Remember password

(You may click on this picture to get help on parts of the dialog.)

This dialog asks you to supply a user name and password to connect to the Pueblo server. If you've never connected to Pueblo before, enter the user name you'd like to use and the password, and press the **Create New Account** button. You will be prompted to supply some new account information.

If you have logged into Pueblo before, then select the user name you wish to use, enter your password, and press the **Sign on** button.

If you're not as concerned about the security of your account, you can enter your password and then click on the **Remember Password** checkbox before logging in. The next time you start Pueblo, your password will be filled in automatically.

The About box

This dialog has several pages of information on Pueblo.   You can click your mouse on one of the tabs at the top of the window to go to that page.

Please read the **Privacy** page.   Chaco Communications is interested in which worlds people enjoy visiting, and we're collecting some usage information.   This page exists to let you know that we're very serious about protecting your privacy.

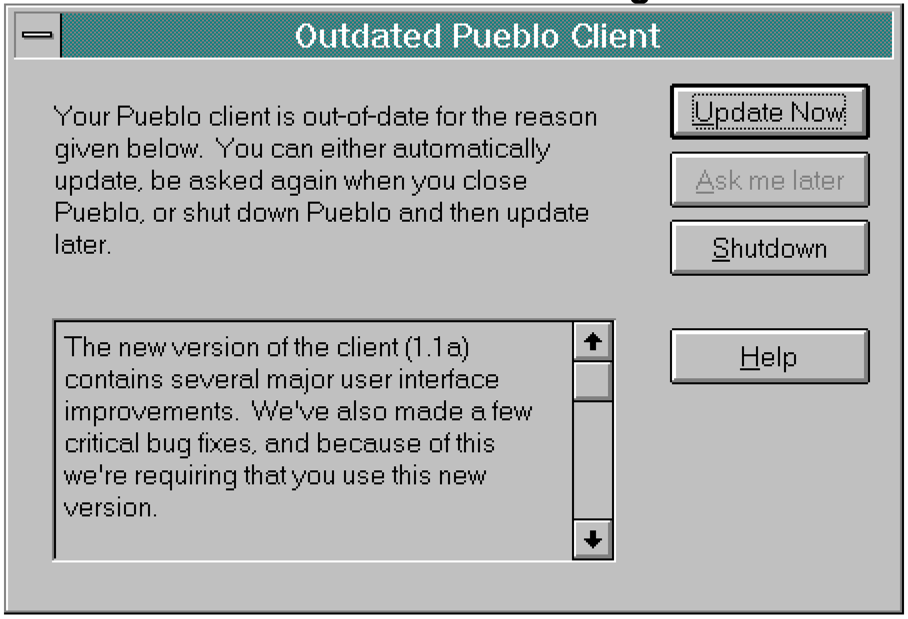
The Password Verification dialog

This dialog is displayed when you are creating a new account. The password you entered in the login dialog box is in the first edit field. Enter the password again in the second edit field and click the Okay button to create the account. If the two passwords you entered don't match, you'll be asked to enter them again. Note that the match is case-sensitive (**`Pickle`** doesn't match **`pickle`** or **`PICKLE`**.)

Your password must not contain your username, nor may your username contain your password. This makes it harder for anyone to guess your password. For security, you should also use a combination of letters and numbers.

And of course, don't forget your password. (But if you do, you can send mail to **pueblo-support@chaco.com**.)

The Outdated Pueblo Client dialog



Your Pueblo client is out-of-date for the reason given below. You can either automatically update, be asked again when you close Pueblo, or shut down Pueblo and then update later.

The new version of the client (1.1a) contains several major user interface improvements. We've also made a few critical bug fixes, and because of this we're requiring that you use this new version.

(You may click on this picture to get help on parts of the dialog.)

This dialog appears when the server has a newer version of Pueblo that you should be using. When you tell Pueblo to update itself, it will spend some time downloading a new version. It will then disconnect from the server and run the unpack and install program for that version.

If the new Pueblo client isn't critical, then you can elect to download the new client later. If you do this, Pueblo will ask you when it is closing if you want to download a new client.

If you close this dialog box using the system menu, Pueblo will shut down.

Dialog box pop-ups

This button will cancel the dialog without making any changes.

This button will display help for the dialog (this help page.)

You should type in the user name you want to use or select the user name from the pop-up list. (The pop-up list contains all user names that you've used to log in to Pueblo. If you haven't logged in yet, this field is empty.)

If you are logging in to an existing Pueblo account, enter the password associated with the user name. If you are creating a new Pueblo account, enter the password you want to use in the future to access this account. Don't forget the password you choose!

If the user name and password you entered are for an existing Pueblo account, pressing this button will log you in to that account.   This button will be disabled unless you've typed in a user name and a password.

This button will create a new account for the user name and password you've specified. If someone has already chosen the same user name you want to use you'll be asked to select another. This button will be disabled unless you've typed in a user name and a password.

When browsing world lists, this button will flip you to the previously-viewed list.   This button will be disabled when you're looking at the main world list.

When there are more than one pane in the Pueblo window, this button will toggle the orientation of the panes between horizontal and vertical.

When there are more than one pane in the Pueblo window, this button will swap the positions of the two panes.

Logs you out of the Pueblo server (and returns you to the real world.)   When you're logged out, the button appearance changes to ➡, and logs you *in* to the server.

Removed the selected text and puts it onto the Clipboard.   This command is available only when you select text.

Copies the selected text to the Clipboard.   This command is available only when you select text.

Inserts a copy of the Clipboard contents at the insertion point, replacing the selection (if any) with the text on the Clipboard. This command is not available if the Clipboard is empty or if the selected text cannot be replaced. (You cannot paste into the output window, for example.)

Displays information about Pueblo.

Allows you to get context sensitive help on the application. The cursor changes to and Help will be displayed for the next thing you select.

This field displays information and messages that help you use Pueblo.   When the mouse is over a menu item help will be displayed on that menu.

This field indicates whether or not the Caps Lock key is pressed.

This field displays whether you are online or offline.   If you are online, the field contains information on how long you've been connected.

This field displays the current time-of-day.

This field contains a display of the progress of any network transfers. As a transfer approaches completion, the field will fill with color. (The example shows a transfer about 3/4 done.) When no transfers are underway, the field is empty (and in the background color.)

This field contains an explanation of why you should be using the new client.

This button results in the new client being downloaded immediately. When the download is complete, Pueblo shuts down and runs the setup program for the new client code.

If this button is enabled, it indicates that you can continue using the Pueblo code that you currently have.   If you press this button, Pueblo will continue normally.   Later, when Pueblo is closing, it will ask you again if you wish to perform the download of the new client.

Press this button if you do not want to immediately download the new client or use Pueblo. Pueblo will shut itself down.

Click on this button to have Pueblo remember your password and automatically fill it in the next time you start Pueblo. Don't do this if you're worried about someone else using your Pueblo account.

You can click and drag in the space between the windows to change the amount of space allocated to each window.

When you connect to a world, this area displays the output coming from that world.   It may also echo the commands you send to that world using the input window.

This area contains a list of worlds that Pueblo knows about. You may click on a hilighted item to jump to that item. In this list, if you clicked on **BioMoo** or **Hypertext Hotel**, you would be connected to those worlds.

World type definitions

World stands for Multi-User Dimension or Multi-User Dungeon.

MUSH stands for Multi-User Shared Hallucination.

AberMUD is one of the first combat-oriented worlds.   It's named after the university where it was developed, Aberystwyth in England.

Circle is a derivative of DikuMUD developed by Jeremy Elson (jelson@cs.jhu.edu).   Circle is very similar to LPMUD, though it hasn't been around as long.

DGD is a leaner, meaner version of LPMUD. The name stands for Dworkin's Generic Driver.

DikuMUD is very similar to LPMUD, though it hasn't been around as long. It's named after the university at which it was written, Datalogisk Institut Koebenhavns Universitet (Dept. of Datalogy, University of Copenhagen).

LambdaMOO is based on the original MOO, rewritten to add functionality and features and to trim the code. Like MOO, LambdaMOO contains an internal object-oriented programming language. Lambda MOO is still being developed by Pavel Curtis at Xerox PARC.

LPMUD is a Multi-User Dimension written by a guy named Lars Pensj|. LPMUD is one of the most popular combat-oriented worlds.

MOO stands for *MUD, Object-Oriented*. MOOs are written using an internal C-like programming language.

TinyMUCK is derived from TinyMUD, and added some new programming constructs.   The name is derived from MUD, and means nothing in particular.

TinyMUD is the first social world.   Players on many Tiny MUDs are allowed to build their own areas and objects.

TinyMUSH is derived from TinyMUD, and added a simple programming language.   MUSH stands for Multi-User Shared Hallucination.

UberMud contains an internal programming language that is used to write most of the rules that guide objects in the world. Because of this, UberMuds may vary quite a bit from world to world.

ANSI is the American National Standards Institute.   Among other things, they set standards for codes that may be sent to terminals to change text attributes.

Miscelaneous

No Help Available

No help is available for this area of the window.

No Help Available

No help is available for this message box.

Terminology

Worlds have their own terminology, and it helps to know some of the terms when you connect. Here are some that you may find useful.

Abbreviations used:

adj.	adjective
adv.	adverb
interj.	interjection
n.	noun
v.	verb

ban, *v.*	To bar from entering. Most often used to describe the locking of a game from entry by users from a particular site. See also: banish
banish, *v.*	To bar from entering. This is what administrators do when they decide they no longer wish for a particular player or users from a particular site to use the game. Most often it means that a character has been nuked and no user may create a new player of the same name. See also: ban
bot, *n.*	1. A computer program which can direct a PC on some MUD without human intervention. 2. Any NPC of above average intelligence, esp. those with descriptions similar to robots.
BRB	(BRB is a shortened term for *be right back.*) See you again very soon.
character	The personality and appearance that a player takes when in a world. Also known as an avatar, persona, or incarnation.
clueless newbie	Particularly bad newbies, usually new users who are annoying people.
dino	Short for dinosaur, this is someone who has been around a long time.
emoting	The use of the `emote` command, ':', or equivalent to simulate an action. For example if your name is Tinkerbell, entering the command `:moons you.` would result in people seeing `Tinkerbell moons you.`
furry	An anthropomorphic intelligent animal. Any character which has fur and is cute could be called a furry. Furries tend to hang out on FurryMUCK.
haven	A room where one character cannot kill another character. The term originates from the Tiny worlds, where the HAVEN flag can be set that has this effect.
hit point, *n.*	See HP.
host, *n.*	The machine from which a game operates.
HP, *n.*	(HP is a shortened term for *hit point*.) A measure of how many more times an object may be struck by a force causing one damage point before hit points are less than or equal to 0. Usually when hit points fall below 1, death or destruction of the object will occur.
IC	(IC is a shortened term for *in character*.) On role-playing worlds, players are expected to play the part of a character in the world, adopting behavior and

	mannerisms appropriate to the world. This is called being *in character*.
lag	Between your computer and the world you're connecting to, there may be up to 30 links consisting of serial lines, high-speed modems, leased lines, satellite uplinks, etc. If one of these gateways or lines crashes, is suddenly overloaded, or gets routing confused, you may notice a large amount of lag time between your input and the world's reception of that input. Computers which are nearer to the computer running the world are less susceptible to lag. Another source of lag is if the computer which hosts the world is overloaded. When lag happens, it is best to just patiently wait for it to pass.
log	A log is the record of what happens in a world.
MUD	Multi-User Dimension or Multi-User Dimension. This is a class of world.
MUSH	Multi-User Shared Hallucination. This is a class of world.
newbie	Someone new to the Internet, to a specific world, or both.
NPC, *n.*	(NPC is a shortened term for *non-player character*.) All of the data and characteristics of any object used by a game to represent an actor which is meant to operate without much human intervention.
PC, *n.*	(For definitions 1 and 2, PC is a shortened term for *player character*.) 1. All of the data and characteristics of an object which a game uses to represent an actor whose actions and description are most often under the control of a human user. 2. Any actor within a game, human controlled or automatic. 3. A Personal Computer. Eg: An IBM clone.
PK	Short for 'player killing.' This refers to one player killing another. On many worlds this is forbidden, while on others it is the whole point of the world.
posing	See emoting.
race, *n.*	In MUDs the term nearly always refers to the species a player's character belongs to. Eg: His race is human.
reboot, *v.*, *n.*	The events which cause a game to start performing its normal operating jobs. This can only occur after a game is down (off.) (Some games can reboot without disconnecting users.)
RL	(RL is a shortened term for *real life*.) Usage: *Bob farts (in RL).*
room, *n.*	A location to which players may move (mostly.) It generally consists of a title and description. A common error of newbie players is to visualize rooms as cells of equal size. This is usually wrong. The only assumptions you should make about a room are those contained in the room’s description.
RW	Short for 'real world.'
spam	Spam Spam Spam Spam! Spamming is flooding the world with inappropriate information, such as saying the same thing over and over. This is generally considered bad form, and it can get you barred from a world.
tinysex	This is when you have consentual sex in the world with another character. It challenges your creativity. You need to type quickly to keep your partner intrigued, and this may be difficult with one hand.
wizard	A person who has special privileges on a world. Often the wizard is running the world on their particular machine, so you already owe them one. Treat these people well, for they are subtle and quick to anger.

Pueblo enhancements to HTML

Pueblo HTML Support

Pueblo uses a HTML engine called WebTracker for text output. This section lists all of the standard HTML supported by WebTracker. In the case where there is industry differentiation on how HTML should be interpreted, WebTracker attempts to look as much like the Netscape standard as possible.

With the goal of giving our users as much flexibility as possible, Pueblo has added many of the Netscape extensions to HTML. The following list of HTML keywords indicates what is supported by WebTracker. If you don't see an extension listed in the following tables, then WebTracker probably doesn't support the extension.

Document Structure Elements

<!-- Comments -->   <body>   <head>   <html>   <title>

Anchor Element

<a>

Image Element

<img>

Block Formatting Elements

<address>	<basefont>	<blockquote>	<br>	<center>	<h1>
<h2>	<h3>	<h4>	<h5>	<h6>	<hr>
<listing>	<p>	<plaintext>	<pre>	<samp>	<xmp>

Character Formatting Elements

<b>	<cite>	<code>	<em>	<font>	<i>
<strike>	<strong>	<tt>	<u>		

List Elements

<dl>   <dir>   <li>   <menu>   <ol>   <ul>

Form Elements

<form>   <input>   <option>   <select>   <textarea>

Unsupported HTML elements

<base>   <isindex>   <kbd>   <link>   <menu>   <meta>   <nextid>   <var>

Also see

Pueblo HTML Extensions
Pueblo Enhancer's Guide

Pueblo HTML Extensions

Most of Pueblo's HTML extensions are prefixed with xch_ ('x' for eXtension, 'ch' for Chaco), to avoid conflicts with standard HTML and with other extensions. Our intent is to try to get some of these features included in future versions of the HTML standard, and these strings will most likely be renamed when that happens.

Extensions to HTML

This section describes some general HTML extensions Chaco has made with the intent of improving usability of HTML documents, streams, and clients.

Basic extensions to HTML:

This section lists some general HTML extensions Chaco has made with the intent of improving usability of HTML documents, streams, and clients.

xch_hint <xch_page> xch_pane <xch_prefetch> xch_prob

Extensions for interactivity:

This section lists the HTML extensions Chaco has made to allow interactive HTML sessions.

xch_cmd xch_mode xch_mudtext

Extensions for graphics:

This section describes the HTML extensions Chaco has made to allow an HTML session to control a VRML scene.

xch_graph xch_graph_node

Extensions for sound:

This section describes the HTML extensions Chaco has made to allow an HTML session to control sound.

xch_sound xch_volume xch_alert xch_device

Also see

Pueblo HTML Support
Pueblo Enhancer's Guide

ANSI vs. HTML color

Pueblo supports ANSI color sequences for foreground colors, so that existing ANSI-enabled Worlds should work very well with Pueblo.

If you'd rather not deal with obscure ANSI sequences, Pueblo offers a HTML option to color text through the <font> tag. If you're developing a HTML MUD, we recommend that you use this method to set text colors. The <font> tag gives you control over both foreground and background color.

For overriding the default text color for a large block of text, the <xch_page> element may be more satisfactory. Also, the <body> tag may be used to set both text and hotlink colors.

Also see

<body>
<font>
<xch_page>

Pueblo Enhancer's Guide

This document describes the protocol by which a World (MUD, MUSH, MOO, etc.) can determine whether or not a user's client software supports the Pueblo enhancements, and then interact with Pueblo clients to take advantage of all of Pueblo's special features.

First off, you should check the Chaco ftp site to see if there's a patch for your mud server already. Look in pub/pueblo/enhanced on either ftp2.chaco.com or ftp.chaco.com.

The Pueblo client watches for lines of this form:

```
This world is Pueblo 1.0 Enhanced.
```

When it sees a line of that form, it sends the World this command:

```
PUEBLOCLIENT 0.92
```

(Where the '0.92' is the version of the Pueblo client sending the command) Upon receiving this `PUEBLOCLIENT` command, the World may send the following sequence, to tell the Pueblo client to begin interpreting the World's output as HTML:

```
</xch_mudtext><img xch_mode=html>
```

You may want to send the "This world is Pueblo 1.0 Enhanced" line and accept the `PUEBLOCLIENT` command before the user logs in, so that you can send HTML from the outset. You may also want to provide special commands for sending HTML-formatted text and convert other text to escaped HTML (converting < to &lt, etc.).

This is the sequence of changes we typically make when we're Pueblo Enhancing a MUD:

1. Create a new "flag" which indicates that a user is using a Pueblo compatible client. This flag should be removed from the player when they log out, since they may log in with both Pueblo and a text client.

   On TinyMUSH, this was a literal flag and an object in the Master Room which had this code:

   ```
   @adisconnect: @set %#=!html
   ```

   On LP-MUD, this was a static int in player.c, so it got reset when they logged out automatically. We also added a new function to player.c: query_is_html(), which allows rooms and objects to query a player's object about whether it supports HTML.

2. Display the string "`This world is Pueblo 1.0 Enhanced`" and watch for the `PUEBLOCLIENT 0.92` command coming back from the client. When you get PUEBLOCLIENT, set the flag created in step 1 above.

   For TinyMUSH, we display this string as part of connect.txt and added an "external" command (like WHO and QUIT) called PUEBLOCLIENT. The only trick is that on TinyMUSH, the player object doesn't exist until they log in, so we had to set an HTML flag on the network descriptor, then copy it over when they log in.

   For LP-MUD, we added this in player.c's logon2(). We also did an 'add_action("puebloclient", "PUEBLOCLIENT");' and the corresponding puebloclient function so that people could do a PUEBLOCLIENT after they log in, but this was optional.

   You may also want to add support for 'PUEBLOCLIENT off', which turns off the HTML output, for testing.

3. Add HTML escaping. HTML has a few special characters that require escaping. You should convert these characters to the corresponding HTML sequence, as described in the following table. (Note: those trailing semicolons inside the strings are important.)

Character:	**HTML sequence:**

<	&lt;
>	&gt;
&	&amp;
"	&quot;

In TinyMUSH, we did this in game.c's notify_check().

In LP-MUD, we did this in dgd/lib/player.c's catch_tell().

4. Now you won't be able to output any HTML anchors, since all your output will be escaped, so you need a second output path that gets around the HTML escaping you did in step 3.

In TinyMUSH, we added @emit/noreturn and @pemit/noreturn to allow MUSH coders to output non-escaped (and non-newlined) HTML. We added a flag to notify_check() which tells it not to do the escaping it normally does, and several "_noreturn" versions of the macros in externs.h (i.e. notify_noreturn() which included this flag to notify_check()).

In LP-MUD, we added write_html(), which was just like write() but skipped the HTML escaping. This was in dgd/lib/interactive.c. We also had to do tell_object_html(), catch_tell_html(), etc.

You basically have to duplicate the output path from the normal input points (write(), catch_tell(), etc.) down to the function where you do the escaping, either calling different functions or passing some flag that indicates that escaping should be skipped.

5. Add HTML anchors for exits. When you list the obvious exits in a room, you might want to have each exit be an anchor, like this:

```
<a xch_cmd="east" xch_hint="Go east">east</a>
```

In TinyMUSH, this was in look.c's look_exits().

In LP-MUD, this was in room/room.c and looked like:

```
while (i < sizeof(dest_dir))
{
    if (this_player()->query_is_html())
    {
        /* Write the exit as an anchor */

        write_html( " " + "<a xch_cmd=" );
        write( dest_dir[i] );
        write_html( " xch_hint=\"Go " );
        write( dest_dir[i] );
        write_html( "\">" );
        write( dest_dir[i] );
        write_html( "</a>" );
    }
    else
    {
        write( " " + dest_dir[i] );
    }
}
```

Note that we mix the use of write() and write_html(). If the exit name has one of <>&" in it, you'll need those characters to be escaped, but you don't want the "<a", ">", or "</a>" to be escaped.

6. Add VRML URL support to rooms. When a user walks into a room, if that room has the URL of a VRML file associated with it, you want to send them a <img xch_graph=load

href="<the URL>">.

For TinyMUSH, we added a new attribute: @vrml_url. When the user enters a room, we check to see if the user has HTML enabled, then check to see if the attribute exists on the room, all in move.c.

For LP-MUD, this was in room/room.c. We added a static string vrml_url;. In init(), we send the load if appropriate.

7. Do the same, for MIDI background sounds. This is just like the VRML URL support in #6, only for a sound URL.
8. If no VRML_URL is set on the room, send <img xch_graph=hide>
9. Build away!

If you're building VRML files, and want to put them on your own Web server, you'll need to tell your Web server what MIME type a VRML file is. If you're using NCSA or Apache HTTPD, add this line to config/mime.types:

```
x-world/x-vrml          wrl vrml
```

If you Pueblo Enhance a MUD, send us your patches, so others can follow in your footsteps! Send them to info@chaco.com. If you run into problems enhancing your mud, send us mail at info@chaco.com.

You might want to join the pueblo-enhancers mailing list. To do so, send mail to pueblo-enhancers-request@chaco.com.

The Pueblo home page is http://www.chaco.com/pueblo. It has the Pueblo FAQ, instructions on downloading the latest version, etc.

Also see

Building in LP-MUD
TinyMUSH Extensions

<xch_mode>
<xch_mudtext>

TinyMUSH extensions

This section documents the extensions Chaco has made to TinyMUSH 2.x.

@emit/noreturn

@pemit/noreturn

This new options changes @emit and @pemit to not a newline after the text specified on the command-line.   This is because Pueblo automatically converts newlines to HTML <br> tags, which typically isn't desired when special <img> tags are being sent without any surrounding text.

HTML

This new flag indicates that a player's current client supports HTML.   If they have the HTML flag, all text sent to them except via @*emit/noreturn is HTML escaped ('<' becomes '&lt;', etc.).

Also see

Building TinyMUSH sound generators

Building TinyMUSH VRML animations

Building in LP-MUD

In the LP-MUD reference implementation of Pueblo's HTML extensions, we extended the 2.4.5 mudlib's room.c to automatically output exits as anchors for Pueblo users, so nothing special needs to be done when rooms are created.

Descriptions of 2.4.5 rooms come from these variables:

```
static string short_desc;
static string long_desc;
```

Which can be set in your room's reset() function:

```
                                /* vill_green.c */
inherit "room/room";
reset(arg)
{
   if (arg) return;
   set_light(1);
   short_desc = "Village green";
   no_castle_flag = 1;
   long_desc = "You are at an open green place south of the " +
               "village church.\n" +
               "You can see a road further to the east.\n";
   dest_dir = ({"room/church", "north",
               "room/hump", "west",
               "room/vill_track", "east"});
}
```

To support HTML descriptions, we added two new variables to room.c:

```
                                /* Short HTML description of the room */
static string short_html_desc;
                                /* Long HTML description of the room */
static string long_html_desc;
```

These are set just like short_desc and long_desc:

```
reset(arg)
{
   ...
   short_desc = "Village green";
   short_html_desc = "Village <em>green</em>";
   long_desc = "You are at an open green place south of the " +
               "village church.\n" +
               "You can see a road further to the east.\n";
   long_desc = "You are at an open green place south of the " +
               "village church.\n" +
               "You can see a road further to the east.\n";
   long_html_desc =
          "You are at an open green place south of the " +
          "<em>village church<em>.  " +
          "You can see a road further to the " +
          "<a xch_cmd=\"go east\" xch_hint=\"Go east\">east</a>.\n ";
   ...
}
```

Note that it is generally a good idea not to embed newlines in HTML strings like long_html_desc (except at the very end), because HTML clients like Pueblo will automatically format the text they receive.

If you want a VRML document to be displayed when users enter the room, you can set this new variable, also from room.c:

```
                                /* VRML scene URL */
```

```
static string vrml_url;
```

Like this (replacing the URL with the URL of the VRML file you want displayed when people enter the room):

```
reset(arg)
{
    ...
    short_desc = "Village green";
    short_html_desc = "Village <em>green</em>";
    long_desc = "You are at an open green place south of the " +
                "village church.\n" +
                "You can see a road further to the east.\n";
    long_desc = "You are at an open green place south of the " +
                "village church.\n" +
                "You can see a road further to the east.\n";
    long_html_desc =
            "You are at an open green place south of the " +
            "<em>village church<em>.  " +
            "You can see a road further to the " +
            "<a xch_cmd=\"go east\" xch_hint=\"Go east\">east</a>.\n ";
    vrml_url =
            "http://www.chaco.com/pueblo/0.6/ChAnim/msw32/pueblo.wrl";
    ...
}
```

To send HTML text to users, use **write_html()** instead of **write()**. (The write() function escapes some characters when they are sent to Pueblo users so that they can edit programs which include URLs, etc. The escaping turns '<' into '&lt;', etc.)

To determine whether a client is Pueblo-enhanced, use the **query_is_html()** function. For example, if you wanted to output pre-formatted HTML text for HTML-enabled users, you might do this:

```
if (this_player()->query_is_html())
{
    write_html("<pre>");
}
write("This text is preformatted.\n");
if (this_player()->query_is_html())
{
    write_html("</pre>");
}
```

Building TinyMUSH sound generators

First, let's create a simple CD object which causes people who look at it to hear a MIDI file:

```
@create Jaws CD
@lock Jaws CD==me
@Adesc Jaws CD=@pemit/noreturn %#=The soundtrack of the film 'Jaws', more or
       less.<img xch_sound=play
       href="http://www.chaco.com/pueblo/midi/samples/jaws22.midi">%r
```

When this object is looked at, it will emit an <img xch_sound...> HTML tag to the player who looked at it, and their browser will begin playing the MIDI file at the specified URL.

To add a bit more realism to the situation, let's make a CD player to go with the CD, and generalize the CDs so we can expand our collection easily:

```
@create CD Player
@lock CD Player==me
@Desc CD Player=A standard cd player.  There is a play, stop, and loop
       button.
&PLAY CD Player=$play *:@emit/noreturn %n inserts '%0' into the player and
       press the 'play' button.<img xch_sound=play
       href="[get(%0/sound_url)]">%r
```

The 'play' command uses an HTML tag like the Jaws CD above did, but gets the URL of the MIDI file out of the CD object the user specifies in their 'play' command.   Here's a sample CD:

```
@create 'Jazz for Airy Nights' album
@lock 'Jazz for Airy Nights' album==me
@desc 'Jazz for Airy Nights' album=This is an album of light jazz for those
       mellow, romantic moments that everyone experiences in elevators.%r
&SOUND_URL 'Jazz for Airy Nights'
       album=http://www.chaco.com/pueblo/midi/samples/lite_jazz.midi
```

When a player types 'play jazz for airy nights', the **SOUND_URL** attribute on that object is read by the CD player, and the CD player emits the appropriate <img xch_sound....> HTML to everyone in the room.

Next, we can add a 'stop' command to the CD player:

```
&STOP CD Player=$stop:@emit/noreturn %n presses the stop button on the cd
       player.<img xch_sound=stop>%r
```

Also see

TinyMUSH extensions

Building TinyMUSH VRML animations

Building TinyMUSH VRML animations

First, create a VRML room (See the Building In GozenMUSH Section for how to do that).

Next, build a VRML scene. For this tutorial, we'll use this simple VRML scene:

```
#VRML V1.0 ascii
Separator
{
   DEF Cube_transform Transform { rotation 0 1 0 0 }
   Cube { }
}
```

This scene consists of one cube and a 'Transform' node. The Transform node as it exists in the VRML file does nothing (because the fourth parameter after 'rotation' is 0), but using Pueblo's xch_graph_node <img> field, we can replace this transform node with one that contains an actual rotation.

To see this effect, save the VRML file, put it on a World-Wide Web server, and configure your room to point to the file's URL. (In GozenMUSH, you would do: '&vrml_url here=http://www.chaco.com/~vrmldemo/demo1.wrl').

Then, leave your demo room and re-enter it. Be sure the cube is visible. To rotate the cube, you can do this:

```
say <img xch_graph_node="DEF Cube_transform Transform { rotation 0 1
      0 .1 } }">
```

This should rotate the cube slightly.

To make the cube rotate on its own, you could use this TinyMUSH code:

```
@create Cube Rotator
@lock cube rotator==me
drop cube rotator
&spin cube rotator=$spin:@dolist [lnum(32)]=@wait ##=@emit/noreturn <img
      xch_graph_node="DEF Cube_transform Transform { rotation 0 1 0
      [mul(0.1,##)] }">
```

Then type 'spin' to cause the cube to rotate by .1 radians per second for 32 seconds.

One drawback of this method of animation is that there is no synchronization between the World and the client. Here is another version of the Cube Rotator that synchronizes using <img xch_cmd> to force the client command to execute a command which causes the next <img xch_graph_node> rotation to be sent:

```
@create Rotation
@lock Rotation==me
&ROTATE Rotation=&angle_%0
      me=[add(v(angle_%0),v(angle_inc))];@pemit/noreturn %0=<img
      xch_graph_node="DEF Cube_rotation Transform \{ rotation 0 1 0
      [v(angle_%0)]\}"><img xch_cmd="rotate">
&RO Rotation=$rotate:@trig me/rotate=%#
&ANGLE_INC Rotation=-.1
```

The 'rotate' command starts this object's execution. The output of this command looks like this:

```
<img xch_graph_node="DEF Cube_rotation Transform { rotation 0 1 0 0.1}"><img
      xch_cmd="rotate">
```

The first <img> changes the cube's rotation transform. The second one causes your Pueblo client to send another 'rotate' command on your behalf. Because the next 'rotate' command is sent after the new cube rotation is displayed, the World server stays in synch with your Pueblo client.

Also see

TinyMUSH extensions

Building TinyMUSH sound generators

Creating and using HTML dialogs

Pueblo gives you the ability to use dialog boxes in your virtual worlds, through simple (and mostly standard) HTML. The dialog is displayed to the user, and when the user submits the information, a command of your choice is called with the results of the dialog.

Opening the dialog

You should be familiar with the use of panes. A dialog is displayed by outputting the following HTML sequence:

```
<img xch_pane=open name="my dialog" xch_pane_title="Registration Form"
      href="http://www.chaco.com/reg_dialog.html" width=640
      xch_pane_options="floating closeable fit">
```

Note a few things about this HTML sequence:

- The **name=** tag specifies the name of the dialog pane. This can be any name you choose, and should be unique for the dialog. In this example, I'm using the name "my dialog". The name is important when you wish to close the dialog.
- The **width=** tag here specifies an optimal width for the dialog. I'm using 640 pixels since that's the smallest resolution folks use these days.
- The **xch_pane_options=** tag includes the **fit** option. This ensures that the dialog will be sized to properly fit the fields you've specified. Use this. It looks good.
- The **xch_pane_options=** tag also includes the **closeable** option. This means that the user can close the dialog. See the discussion of this below.

The **href=** attribute points to a file which contains the dialog description. This is a file stored on your web server, and is just a web page containing a form. If you're not familiar with forms, there are several very good tutorials available on the web. The best I've seen is Carlos' Forms Tutorial. You'll find it at **http://robot0.ge.uiuc.edu/~carlosp/cs317/cft.html**.

All of the forms elements Pueblo supports are documented in this help file. They're listed in the 'Also see' section at the bottom of this page.

Processing the data

HTML forms normally send their results to the HTTP server. Pueblo allows the result of the form to go to the world server by specifying **method=xch_cmd** in the form statement. When this method is used, the **action=** statement specifies the command that is sent to the world when the form is submitted. The results of the form are immediately appended to the command. No spacing is added. Normally, the action you specify should include a trailing space so that the command name is separated from the results. Here is an example:

```
<form method=post action="register_form ">
…
</form>
```

When this form is submitted, the command "register_form *form_data*" will be sent to the world, where *form_data* is the data from the form. Form data consists of a question mark ('?') followed by a stream of name=value pairs separated by the & character. Each name=value pair is URL encoded

The 'value' portion of the form data is encoded according to a few simple rules:

- Spaces are changed to plusses ('+')
- Non-alphanumeric characters are encoded into hexadecimal notation. These characters are encoded by a character triplet consisting of the character "%" followed by the two hexadecimal digits (from "0123456789ABCDEF") which forming the hexadecimal ASCII value of the character. (The characters "abcdef" may also be used in encodings.)
- Alphanumeric characters (a-z, A-Z, 0-9) are **not** encoded.

For example, let’s consider the following form:

```
<FORM METHOD=xch_cmd ACTION="user_info ">
What's your first name? <INPUT TYPE="text" NAME="name"><BR>
<P>
What's your favorite color? <SELECT NAME="color">
<OPTION>black
<OPTION>burnt umber
<OPTION>forest green
<OPTION>indigo
</SELECT>
<P>
<INPUT TYPE="submit" VALUE="Okay!">
</FORM>
```

If the user enters “Coyote & friends” for their name and selects ‘indigo’ for their favorite color, and then presses the ‘Okay!’ button, the following command will be sent to the world server:

```
user_info ?name=Coyote+%26+friends&color=indigo
```

Note that the spaces in ‘Coyote & friends’ were changed into plusses, while the ampersand in the string was escaped into hexadecimal (the ASCII value of ampersand is 0x26.)

Chaco may already have code available to decode this information for your world server. Surf to **http://www.chaco.com/products/servers/** for more information.

Closing the dialog

Panes are not automatically closed when their contents are submitted. You can cause the pane to be closed by outputting the following HTML string:

```
<img xch_pane=close name=”my dialog”>
```

Note that the **name=** field should contain the same string that you specified when you opened the pane.

Also see

Panes

Form Elements: <form> <input> <option> <select> <textarea>

Using image maps with worlds

Pueblo allows you to use image maps with your existing worlds. It’s a fairly simple process. Simply specify the image with the ISMAP attribute within an anchor statement. Here is an example:

```
<a xch_cmd=”do_map ”>
<img src=”http://…/world_map.gif” ismap>
</a>
```

Note that the anchor statement is using the xch_cmd attribute rather than the standard ‘href’. The use of this attribute is explained on the anchor element page. Basically, xch_cmd causes the specified command to be sent to the world from the client. Thus, when the anchor is clicked, the string “do_map “ should be sent to the world. This isn’t completely right, however. Read on…

Note that the <img> element contains the ‘ismap’ attribute. This attribute indicates that the image is a *map*. This means that when you click on this image in an anchor, the coordinates of your click are appended to the anchor, in the form **?x,y**. In Pueblo, the coordinates are appended to the xch_cmd value from the <a> element.   Thus, if you click in the above image map anchor, the following string could be sent to the world:

```
do_map ?100,50
```

This means that the image was clicked on the pixel that is 100 over from the left and 50 down. (The question mark and comma are always sent as delimiters.)

Note that the space is only sent between the command and the question mark if you specify it in the xch_cmd attribute.   For example, if the image anchor were specified like this:

```
<a xch_cmd="map_click">
<img src="http://…/world_map.gif" ismap>
</a>
```

Then the click would be sent to the world in the following form:

```
map_click?100,50
```

As a world programmer, you're responsible for processing the command and interpreting the coordinates that you receive.

Also see

<a>

<img>

xch_cmd

xch_hint

<!-- Comments -->

Comments begin with the `<!--` sequence.   After the comment start sequence, all text up to the next occurrence of `-->` is ignored.

<a>...</a>

An Anchor element is a marked text that is the start and/or destination of a hypertext link. Anchor elements are defined by the <a> element.

Attributes

HREF

If the HREF attribute is present, the text between the opening and closing anchor elements becomes hypertext. If this hypertext is selected by readers, they are moved to another document, or to a different location in the current document, whose network address is defined by the value of the HREF attribute.

NAME

If present, the NAME attribute allows the anchor to be the target of a link. The value of the NAME attribute is an identifier for the anchor. Identifiers are arbitrary strings but must be unique within the HTML document.

For example, a document can define a target as follows:

```
<a name="LPMud">LPMud</a>
```

An anchor can then jump to this target using an anchor as follows:

```
Another type of World is Lars Pensj|'s <a href="#LPMud">LPMud</a>.
```

An anchor can also jump to a different document appending the target to the document URL, as follows:

```
Another type of World is Lars Pensj|'s <a
        href="mudtypes.html#LPMud">LPMud</a>.
```

TITLE
REL
REV
URN
METHODS

These attributes are not currently supported by WebTracker.

Extended WebTracker Attributes

XCH_CMD

This is an anchor field which indicates that the specified command should be sent to the World when the anchor is selected. For example, when the following anchor is selected, the text "look communicator" will be sent to the World:

```
In the corner of the office is a small <a
        xch_cmd="look communicator">personal communicator</a>.
```

XCH_HINT

The XCH_HINT attribute can be used to give users more information about what will happen if they select the anchor their cursor is over. The information is displayed in the Pueblo status bar.

Also see

<img>
xch_cmd
xch_hint

Using image maps with worlds

<address>...</address>

The Address element specifies such information as address, signature and authorship, often at the top or bottom of a document.

An address is rendered in an italic typeface.   The <address> element implies a paragraph break before and after.

Also see

<blockquote>

<b>...</b>

The Bold element specifies that the text should be rendered in boldface.

Also see

- <em> (emphasis)
- <i> (italic)
- <strike> (strikethrough)
- <strong>
- <u> (underline)

<basefont> element

This changes the size of the base font that all relative <font size=...> changes are based on. The value defaults to 3, and has a valid range of 1-7.

Attributes

SIZE

The SIZE attribute may have a value in the range 1-7. If this attribute is omitted, the default value is 3.

Examples

The following statement gives the entire document slightly smaller-than-normal fonts:

```
<basefont size=2>
```

Also see

<font>

<blockquote>...</blockquote>

The Blockquote element is used to contain text quoted from another source.   The element is rendered with a slight extra left indent.   Space is left above and below the quote.

Also see

<address>

<body>...</body>

Note: This tag is not necessary for World output to Pueblo. WebTracker supports this element so that normal web pages may be displayed. For World output, the xch_page element should be used.

The body of a HTML document contains all the text and images that make up the page, together with all the HTML elements that provide the control & formatting of the page.

Attributes

BACKGROUND

WebTracker supports the BACKGROUND attribute to the <body> element. The purpose of this attribute is to specify a URL pointing to an image that is to be used as a background for the document. In WebTracker, this background image is used to tile the full background of the document-viewing area.

For example, the following statement creates a document where the entire document background is tiled with the lt_gray_rock.jpg texture:

```
<body background="http://www.chaco.com/textures/lt_gray_rock.jpg">
The contents of the document go here…
</body>
```

TEXT

The TEXT attribute is used to specify the text color for the document (i.e., text that is not colored to indicate a hotlink.) The statement takes the following form:

```
<body text="#rrggbb">
```

Where '#rrggbb' is a red-green-blue triplet used to specify the color. Note that these values are in hexadecimal.

BGCOLOR

The BGCOLOR attribute changes the color of the background for the document. The format for this attribute is the same as that for TEXT.

LINK
ALINK
PLINK
VLINK

These attributes let you control the coloring of link text. LINK is the color of normal, non-visited links. VLINK stands for 'visited link', PLINK stands for 'pre-fetched link', and ALINK stands for 'active link'. The format for these attributes is the same as that for TEXT.

Extended Web Tracker Attributes

ALIGN

The ALIGN attribute may only be assigned the value **middle**. If `<body align=middle>` is included in a document, then the document is vertically centered between the top and bottom of the page. If the document is larger than the page, then the document is aligned normally (i.e., top-aligned.)

FGCOLOR

The FGCOLOR attribute is another (and more consistent) name for the TEXT attribute.

PLINK

This attribute lets you control the coloring of pre-fetched link text. The format for this attribute is the same as that for TEXT.

Examples

HTML documents have the following format:

```
<html>
<head>
<title>Coyote's Den</title>
</head>
<body>
The majority of the document goes here.
</body>
</html>
```

Also see

<head>

<html>

<xch_page>

<br> element

The Line Break element specifies that a new line must be started at the given point. The new line indents the same as that of line-wrapped text.

Attributes

CLEAR

The CLEAR attribute may be used on the <br> tag to break the line and move down past any floating images.

clear=left	Breaks the line, and move vertically down until you have a clear left margin (no floating images).
clear=right	Does the same for the right margin.
clear=all	Moves down until both margins are clear of images.

<center>...</center>

All lines of text, graphics, and other objects between the begin and end of the <center> element are centered between the current left and right margins.

Also see

<p align=...>

<cite>...</cite>

The Citation element specifies a citation; which is rendered by Web Tracker as italics.

Examples

The following sequence

```
A great book to read is <cite>"Wicked: The Life and Times of the Wicked
       Witch of the West", Gregory Macguire, ReganBooks, 1995.</cite>  I
       heartily recommend it.
```

Will be formatted as follows:

A great book to read is*"Wicked: The Life and Times of the Wicked Witch of the West", Gregory Macguire, ReganBooks, 1995.*   I heartily recommend it.

Also see

<strong>

<samp>

<code>...</code>

The Code element indicates an example of code; and is rendered as monospaced text.   Don’t confuse this element with the <pre> element.

Examples

```
Dave, here is some code I’m working with:
<code>
#include <stdio.h>
                        /*MLSPXPMPVPOQN
                     POPJLVMMQNQNPSPYPXLRP
                   XMTJMLSPXPMPVPOQNPOPJLVMN
                 QSPWPOPXLRPXMTJMLNPOPPPSPXPOP
                JLYPRLTPVLUPVLWPVLXPVLVPSLJLUQP
              PJQLQSPXPNQPPRLTPVL*/main(){int K/*
             PUOUPWOV*/,L/*WPVLXPVLKPUOVPWOSLU*/,M
            /*PPPPVPOQMQRPR*/=0;FILE/*TPSLUMWQT*/ *
           J/*MPRPKPLQ*/;     J=      fopen(__FILE__
          /*T*/,"r");for      (;(       L/*KPUOWOWMUQ
         LLOLMPULWLWL         WLW         LWLWLWLWLWLW
         LWLWLULVOXP          LLV          LLLOLMPULOL
        MMNP*/=getc           (J)            )!= -1;L/*
       OLWLRMMQUL             VOX             PLLVLLLMQO
       PMPYPXPNP              MQL              LVLLLWPSP
      XPOQNQOPMQ              LLV              LTj*/>='J'
     &&L/*LRPYP               OQL               QMQLLVLLL
     OLMQVOXPL                LVL                LLVOe*/<=
     'J'+15&&(                (M=                !M/*MMMQMV
     OJMMMMMU                 OOL                 NPU MPMOM
    UMRNOLMQV                 OJM                 MMMMRMLLW
    QUMTJSPX                  PNQ                  JLLPUOWO
    WMUQJMVL                  JMV                  L*/)?K/*
    MQMV*/=L:               putchar                (K/*deMN
    MaVbcLKM                  KMV                  Le*/-'J'
   /*MWQUMTJ                  WPK                  PSPXPR*/
    |(L/*PQV                  LMPSLSPXPNQ*/-       'J'<<4))
    ));exit(                  0);}/*LPQUMMPR       PKPLQJLT
    LTLJLMPU                   MTJUQMPRPKPLQ       JLNPUONM
    JMWOVLTL                                       OPVLPPWM
    PMUMTJNQ                                       SPWPOPYO
    NQJLQPUM                                       TJPNSNVN
    ONJLTLRP                                       VLTLSPUM
    TJQQRPSP                                       VPOPJLRL
    WLWLPPJLP                                     LPLJLKLJQ
     SPJQOPRL                                     NPSLJLPL
     PLJLKLPPY                                   PLQUPRLSL
     SLUMTJRPW                                   MPPNPYPJQ
      OPXPRLPMW                                 LPPVMVMKM
      VLLLKPLLSL                               UMSPWMPPNP
       YPJQOPXPR                               LRLPMWLPP
       VMVMKMSLUL                             KMVLLLLQLL
        SLUMTJPPYP                           LQRLUMUMSL
         UQSPPPRLPPS                       LUQPPQPOPNQ
         MQRLNPVLNMJM                     VLSPSLUMNPUO
          MQNQLQVPOPXPR                 LNPSLWLKMWOWM
           QLVOJMQLUMTJSP             PPRLTLNPWMWMQL
            JMQLULPPSLUQMQJQLQSPXPNQPPRLTLMPVLLLOLM
             QVOXPLLVLNPULKMSLUMYPRLMQNQNPYPOQNQVL
              PMVLPMWLPPVLNPULKMVLJMSLWQTJSPPPRLT
                LNPXMQLJMQLULPPSLYPRLRPVLOMVLNP
                 VLJMVLJMSLWQOPVPMQOPUQNQSPWPO
```

```
                              PRLPLQPSLUMOPWMKPMQMPNQSP
                                WPOPRLVPYPMPKPVPNQSPW
                                   POPRLPLQPSLSLUM

TJPPYPLQRLPQWMKMUMPQVMPMUMPQULULSLYPRLRPVLPQKLWMKMPLPLPQKLWMPMWLKMVLPQ
ULQLJMQLVLKPNQYPSPRLOPULLPUOPQWOSLVLPQSLMQVPOPOPJQRLKMSLUMWQWQWQTJTJ*/

</code>
Can you add hex output as an option for me?
```

Note: This is actual code from *The International Obfuscated C Code Contest* (**http://reality.sgi.com/csp/ioccc/index.html**). For more information on this program, see **http://reality.sgi.com/csp/ioccc/1994/shapiro.hint**.

Also see

<pre> (preformatted text)

<xch_mudtext>

<dl>...</dl>

A Definition List is a list of terms and corresponding definitions. Definition lists are typically formatted with the term flush-left and the definition, formatted paragraph style, indented after the term.

Examples

The following example defines two terms:

```
<dl>
<dt>bonk
<dd>In the MUD community, it has become traditional to express pique or
       censure by bonking the offending person.  There is a convention that
       one should acknowledge a bonk by saying 'oif!' and a myth to the
       effect that failing to do so upsets the cosmic bonk/oif balance,
       causing much trouble in the universe.
<dt>mudhead
<dd>Commonly used to refer to a MUD player who sleeps, breathes, and eats
       MUD.  Mudheads have been known to fail their degrees, drop out, etc.,
       with the consolation, however, that they made wizard level.
</dl>
```

Attributes

COMPACT

The definition list type can take the COMPACT attribute, which suggests that a compact rendering be used, because the list items are small and/or the entire list is large.

Unless you provide the COMPACT attribute, Web Tracker will leave white space between successive <dt>, <dd> pairs.

If using the COMPACT attribute, the opening list element must be <dl compact>, which must be immediately followed by the first <dt> element:

<dir>...</dir>

A Directory List element is used to present a list of items containing up to 20 characters each. Items in a directory list may be arranged in columns, typically 24 characters wide. If the HTML user agent can optimize the column width as function of the widths of individual elements, so much the better.

A directory list must begin with the <dir> element which is immediately followed by one or more <li> (list item) elements.

Examples

The following defines a directory of four items:

```
<dir>
<li>Item One
<li>Item Two
<li>Item Three
<li>Item Four
</dir>
```

Also see

<li>

<em>...</em>

The Emphasis element indicates typographic emphasis, and is rendered as italics.

Examples

```
I would like to stress that I am <em>deleriously happy</em>.
```

Also see

<b> (bold)
<i> (italic)
<strike> (strikethrough)
<strong>
<u> (underline)

<font>...</font>

The Font element changes properties of a block of text.

Attributes

SIZE

Valid values range from 1 through 7, and the default size is 3. The value given to size can optionally have a '+' or '-' character in front of it to specify that it is relative the the document base font size. The default base font size is 3, and can be changed with the <basefont> element.

Extended Web Tracker Attributes

FGCOLOR
TEXT

The FGCOLOR attribute (which may also be specified as TEXT) is used to specify the text color. The statement takes the following form:

```
<font fgcolor="#rrggbb">
```

Where '#rrggbb' is a red-green-blue triplet used to specify the text color. Note that these values are in hexadecimal.

BGCOLOR

The BGCOLOR attribute is used to specify the background color for a block of text. The statement takes the following form:

```
<font bgcolor="#rrggbb">
```

Where '#rrggbb' is a red-green-blue triplet used to specify the text color. Note that these values are in hexadecimal.

Examples

The following text shows changing the font size:

```
<font size=6> changes the font size to 6.</font>
<font size=+2> changes the font size to the basefont size + 2.</font>
```

The following text shows changing the font foreground color to dark blue and the background to white:

```
This text is <font fgcolor=#000080 bgcolor=#ffffff>blue</font>.
```

The following text shows changing the font foreground color to dark green while making the text slightly smaller:

```
This text is <font size=-1 fgcolor=#008000>green</font>.
```

Also see

<basefont>

<xch_page>

<form>...</form>

The Form element is used to delimit a data input form. There can be several forms in a single document, but the Form element can't be nested.

The fields of the form are defined by one or more <input> elements. Each <input> element defines a single field which may be edited by the user.

The submitted contents of the form logically consist of name/value pairs. The names are usually equal to the NAME attributes of the various interactive elements in the form.

Note: The names are not guaranteed to be unique keys, nor are the names of form elements required to be distinct. The values encode the user's input to the corresponding interactive elements. Elements capable of displaying a textual or numerical value will return a name/value pair even when they receive no explicit user input.

Attributes

ACTION

The ACTION attribute is a URL specifying the location to which the contents of the form is submitted to elicit a response. If the ACTION attribute is missing, the URL of the document itself is assumed. The way data is submitted varies with the access protocol of the URL, and with the values of the METHOD and ENCTYPE attributes.

METHOD

This attribute selects variations of the protocol. The value may be **get** or **post**. By default, METHOD=get is used, though for many applications, the post method may be preferred.

ENCTYPE

When METHOD=post, the ENCTYPE attribute is a MIME type specifying the format of the posted data; by default, the ENCTYPE is application/x-www-form-urlencoded.

Examples

The following example defines a form:

```
<form method=post action="http://www.chaco.com/htbin-post/myscript">
<p>Please answer the following questions:
<ol>
<li>Name: <input type=text size=40 name=Name>
<li>How did you hear about our product? (check all that apply):<br>
<input type=checkbox name=Source value=Newspaper>Newspaper
<input type=checkbox name=Source value=Radio>Radio
<input type=checkbox name=Source value=TV>TV
<input type=checkbox name=Source value=Person>Friend/colleague
<li>How many would you like to order? <input type="number" name="Quantity"
       size=4 value=1>
<input type=radio name=CC value=Visa checked>Visa
<input type=radio name=CC value=Mastercard>Mastercard
<li>Number: <input type=number name=CCNum size=16>
and expiration date (mm/yy): <input type=text name=Expiration size=5>
</ol>
<input type=submit value=" Send this form ">
<input type=reset value=" Reset all values and start over ">
</p>
</form>
```

Also see

<input>
<option>
<select>
<textarea>

<h1>...</h1>

HTML defines six levels of heading. A Heading element implies all the font changes, paragraph breaks before and after, and white space necessary to render the heading.

The highest level of headings is <h1>, followed by <h2>, <h3>, etc. to <h6>. <h1> is rendered using the largest font, with the subsequent heading levels being continuously smaller.

Attributes

ALIGN

The ALIGN attribute may be used to align headings. The value of this attribute may be **left**, **right**, or **center**.

<head>...</head>

Note: This tag is not necessary for World output to Pueblo. Web Tracker supports this element so that normal web pages may be displayed.

The head of an HTML document is an unordered collection of information about the document. Currently, Web Tracker only recognizes the <title> element.

The following <head>-related elements are currently ignored:

<base>	Allows base address of HTML document to be specified.
<isindex>	Allows keyword searching of the document.
<link>	Indicate relationships between documents.
<nextid>	Creates unique document identifiers.
<meta>	Specifies document information useable by server/clients.

Examples

The following code defines the title of a web page:

```
<head>
<title>Coyote's Den</title>
</head>
```

Also see

<body>
<html>

<hr> element

The <hr> element specifies that a horizontal rule of some sort (The default being a shaded engraved line) be drawn across the page.

Attributes

ALIGN

The ALIGN attribute accepts the values **left, right,** or **center**. Since horizontal rules do not have to be the width of the page the author may specify whether they should be pushed up against the left margin, the right margin, or centered in the page.

NOSHADE

For those times when a solid bar is required, the NOSHADE attribute lets the author specify that the horizontal rule should not be shaded at all. (Normally, the horizontal rule is given a 3d effect.)

SIZE

The SIZE attributes lets the author give an indication of how thick they wish the horizontal rule to be.

WIDTH

The default horizontal rule is always as wide as the page. With the WIDTH attribute, the author can specify an exact width in pixels, or a relative width measured in percent of document width (i.e., width=50%).

<html>...</html>

Note: This tag is not necessary for World output to Pueblo. Web Tracker supports this element so that normal web pages may be displayed.

This element identifies the document as containing HTML elements. It should immediately follow the prologue document identifier and serves to surround all of the remaining text, including all other elements.

Examples

HTML documents have the following format:

```
<html>
<head>
<title>Coyote's Den</title>
</head>
<body>
The majority of the document goes here.
</body>
</html>
```

Also see

<head>

<i>...</i>

The Italic element specifies that the text should be rendered in italic.

Also see

- <b> (bold)
- <em> (emphasis)
- <strike> (strikethrough)
- <strong>
- <u> (underline)

<img> element

The Image element is used to incorporate in-line graphics (typically icons or small graphics) into an HTML document.   This element cannot be used for embedding other HTML text.

Web Tracker supports GIF, BMP, and JPG image files.

Attributes

ALIGN

The ALIGN attribute can have one of the following values: **left**, **right**, **top**, **texttop**, **middle**, **absmiddle**, **baseline**, **bottom**, and **absbottom**.   The values **left** and **right** are somewhat different from the others in that these two values create a floating images that the text then wraps around.

align=left	will float the image down and over to the left margin (into the next available space there), and subsequent text will wrap around the right hand side of that image.
align=right	will align the image aligns with the right margin, and the text wraps around the left.
align=top	aligns itself with the top of the tallest item in the line.
align=texttop	aligns itself with the top of the tallest text in the line (this is usually but not always the same as align=top).
align=middle	aligns the baseline of the current line with the middle of the image.
align=absmiddle	aligns the middle of the current line with the middle of the image.
align=baseline	aligns the bottom of the image with the baseline of the current line.
align=bottom	aligns the bottom of the image with the baseline of the current line.
align=absbottom	aligns the bottom of the image with the bottom of the current line.

ISMAP

The ISMAP (is map) attribute identifies an image as an image map.   Image maps are graphics in which certain regions are mapped to URLs.   By clicking on different regions, different resources can be accessed from the same graphic.

SRC
HREF

The value of the SRC attribute is the URL of the document to be embedded; only images can be embedded, not HTML text.   (Web Tracker allows the HREF attribute as a synonym for the SRC attribute.)   Its syntax is the same as that of the HREF attribute of the <A> element.   Image elements are allowed within anchors.

BORDER

This lets the document author control the thickness of the border around an image displayed.

Warning: Setting ‘border=0’ on images that are also part of anchors may confuse your users as they are used to a colored border indicating an image is an anchor.

The default border thickness is 2.

WIDTH
HEIGHT

The WIDTH and HEIGHT attributes were added to <img> mainly to speed up display of the document.

If the author specifies these, the viewer of their document will not have to wait for the image to be loaded over the network and its size calculated.

VSPACE
HSPACE

For the floating images it is likely that the author does not want them pressing up against the text wrapped around the image. VSPACE controls the vertical space above and below the image, while HSPACE controls the horizontal space to the left and right of the image.

ALT
LOWSRC

These attributes are not currently supported by Web Tracker.

Also see

<a> (anchor)

<br clear=...>

Using image maps with worlds

<input> element

The Input element represents a field whose contents may be edited by the user. This tag takes a variety of attributes which define the name of the field, what type of input it is, the maximum length (in the case of text) or a restricted range of values (in the case of radio or checkbox buttons).

Attributes

CHECKED

Indicates that a checkbox or radio button is selected. Unselected checkboxes and radio buttons do not return name/value pairs when the form is submitted.

MAXLENGTH

Indicates the maximum number of characters that can be entered into a text field. This can be greater than specified by the SIZE attribute, in which case the field will scroll appropriately. The default number of characters is unlimited.

NAME

Symbolic name used when transferring the form's contents. The NAME attribute is required for most input types and is normally used to provide a unique identifier for a field, or for a logically related group of fields.

SIZE

Specifies the size or precision of the field according to its type. For example, to specify a field with a visible width of 20 characters:

```
<input type=text size="24">
```

TYPE

Defines the type of data the field accepts. Defaults to free text. Several types of fields can be defined with the type attribute:

checkbox	Used for simple Boolean attributes, or for attributes that can take multiple values at the same time. The latter is represented by a number of checkbox fields each of which has the same name. Each selected checkbox generates a separate name/value pair in the submitted data, even if this results in duplicate names. The default value for checkboxes is "on".
hidden	No field is presented to the user, but the content of the field is sent with the submitted form. This value may be used to transmit state information about client/server interaction.
image	This type is not currently supported by Web Tracker.
password	The same as the **text** type, except that text is not displayed as it is entered.
radio	Used for attributes that accept a single value from a set of alternatives. Each radio button field in the group should be given the same name. Only the selected radio button in the group generates a name/value pair in the submitted data. Radio buttons require an explicit VALUE attribute.
reset	A button that when pressed resets the form's fields to their specified initial values. The label to be displayed on the button may be specified just as for the **submit** type button.
submit	A button that when pressed submits the form. You can use the VALUE attribute to provide a non-editable label to be displayed on

	the button. The default label is application-specific. If a **submit** button is pressed in order to submit the form, and that button has a NAME attribute specified, then that button contributes a name/value pair to the submitted data. Otherwise, a **submit** button makes no contribution to the submitted data.
text	Used for a single line text entry fields. Use in conjunction with the SIZE and MAXLENGTH attributes. Use the <textarea> element for text fields which can accept multiple lines.

VALUE

The initial displayed value of the field, if it displays a textual or numerical value; or the value to be returned when the field is selected, if it displays a Boolean value. This attribute is required for radio buttons.

ALIGN
SRC

These attributes are not currently supported by Web Tracker.

Examples

The following example defines a form:

```
<form method=post action="http://www.chaco.com/htbin-post/myscript">
<p>Please answer the following questions:
<ol>
<li>Name: <input type=text size=40 name=Name>
<li>How did you hear about our product? (check all that apply):<br>
<input type=checkbox name=Source value=Newspaper>Newspaper
<input type=checkbox name=Source value=Radio>Radio
<input type=checkbox name=Source value=TV>TV
<input type=checkbox name=Source value=Person>Friend/colleague
<li>How many would you like to order? <input type="number" name="Quantity"
       size=4 value=1>
<input type=radio name=CC value=Visa checked>Visa
<input type=radio name=CC value=Mastercard>Mastercard
<li>Number: <input type=number name=CCNum size=16>
and expiration date (mm/yy): <input type=text name=Expiration size=5>
</ol>
<input type=submit value=" Send this form ">
<input type=reset value=" Reset all values and start over ">
</p>
</form>
```

Also see

<form>

<option>

<select>

<textarea>

<li> element

The List Item element is used to list a single item in a list. The element begins following the <li> tag and continues until the next <li> tag or until the list is terminated.

Attributes

TYPE

The TYPE attribute may be used on the <li> tag to specify how list items should be marked. The following values are supported:

type=disc	Places a round bullet by the list item.
type=circle	Same as type=disc.
type=square	Places a square bullet by the list item.

In addition, the following values are supported for ordered lists:

type=A	Capital letters. e.g. A, B, C ...
type=a	Small letters. e.g. a, b, c ...
type=I	Large roman numerals. e.g. I, II, III ...
type=i	Small roman numerals. e.g. i, ii, iii ...
type=1	The default numbers. e.g. 1, 2, 3 ...

VALUE

The VALUE attribute works in ordered lists and changes the value for the list item and all subsequent list items.

Examples

The following example creates an ordered list, starting the first element with the value ‘5’ and using capital roman numerals. The last item is numbered with the value 10:

```
<ol>
<li type=I value=5>Fifth item
<li>Sixth item
<li value=10>Tenth item
</ol>
```

Also see

<dir>
<ol>
<ul>

<listing>...</listing>

The Listing element has been replaced by the <pre> element.   Web Tracker supports this element for compatibility with older documents.

Also see

- <code>
- <pre> (preformatted text)
- <samp> (sample)
- <tt> (teletype)
- <xch_mudtext>

<menu>...</menu>

The Menu element is functionally identical to the <dir> element.

Also see

<dir>

<ol>...</ol>

The Ordered List element is used to present a numbered list of items, sorted by sequence or order of importance. An ordered list must begin with the <ol> element which is immediately followed by a <li> (list item) element.

A <ol> element is rendered with a slight extra left indent and each list item is numbered.

Attributes

COMPACT

The ordered list type can take the COMPACT attribute, which suggests that a compact rendering be used, because the list items are small and/or the entire list is large.

START

The START attribute specifies the value for the first item in the list. The value is always specified using normal digits.

TYPE

The TYPE attribute may be used on the <li> tag to specify how list items should be marked. The following values are supported:

type=A	Capital letters. e.g. A, B, C ...
type=a	Small letters. e.g. a, b, c ...
type=I	Large roman numerals. e.g. I, II, III ...
type=i	Small roman numerals. e.g. i, ii, iii ...
type=1	The default numbers. e.g. 1, 2, 3 ...

Examples

The following example creates an ordered list, starting the first element with the value '5' and using capital roman numerals. The last item is numbered with the value 10:

```
<ol type=I start=5>
<li>Fifth item
<li>Sixth item
<li>Seventh item
</ol>
```

Also see

<li>

<ul>

<option> element

The Option element can only occur within a <select> element, and represents one choice for the <select> element.

The contents of the Option element is presented to the user to represent the option.   The contents are used as a returned value if the VALUE attribute is not present.

Attributes

SELECTED

Indicates that this option is initially selected.

VALUE

When present indicates the value to be returned if this option is chosen.   The returned value defaults to the contents of the Option element.

Examples

The following example defines a form that contains a single <select> element and a submit button:

```
<form method=post action="http://www.chaco.com/htbin-post/post-query">
What would you like to do today?
<select name="what-to-do">
<option>Drink Coffee
<option selected>Use Pueblo
<option >Use VR Scout
<option >Read A Book
<option>Take A Walk
<option>Buy A Bagel
</select>
<input type=submit value=" Enter my agenda ">
</form>
```

Also see

<form>

<input>

<select>

<textarea>

<p>...</p>

The Paragraph element defines a paragraph. The exact indentation, leading, etc. of a paragraph is not defined and may be a function of other elements, style sheets, etc.

Typically, paragraphs are surrounded by a vertical space of one line or half a line. This is typically not the case within the Address element and or is never the case within the Preformatted Text element.

Attributes

ALIGN

The ALIGN attribute may be used on the <p> tag to align paragraphs.   The value of this attribute may be **left**, **right**, or **center**.

Also see

<center>

<plaintext>...</plaintext>

Between the <plaintext> begin and end tags, all HTML tags are ignored and the text is rendered in monospace font.   No line wrapping is performed.

This is an obsolete element.

Also see

<code>
<pre> (preformatted text)
<samp> (sample)
<tt> (teletype)
<xch_mudtext>

<pre>...</pre>

The Preformatted Text element presents blocks of text in fixed-width font, and so is suitable for text that has been formatted on screen.

Within preformatted text:

- Line breaks within the text are rendered as a move to the beginning of the next line.
- The <p> element should not be used. If found, it should be rendered as a move to the beginning of the next line.
- Anchor elements and character highlighting elements may be used.
- Elements that define paragraph formatting (headings, address, etc.) must not be used.
- The horizontal tab character (encoded in US-ASCII and ISO-8859-1 as decimal 9) must be interpreted as the smallest positive nonzero number of spaces which will leave the number of characters so far on the line as a multiple of 8. Its use is not recommended however.

Attributes

WIDTH

The WIDTH attribute specifies the maximum number of characters for a line and allows Web Tracker to select a suitable font and indentation. If the WIDTH attribute is not present, a width of 80 characters is assumed.

Also see

<code>

<samp> (sample)

<tt> (teletype)

<xch_mudtext>

<samp>...</samp>

The Sample element indicates a sequence of literal characters; and is rendered as monospaced.

Examples

```
You may see the following text when you log in: <samp>This world is Pueblo
    1.0 Enhanced</samp>.
```

Also see

<code>

<pre> (preformatted text)

<tt> (teletype)

<xch_mudtext>

<select>...</select>

The Select element allows the user to chose one of a set of alternatives described by textual labels.   Every alternative is represented by the Option element.   The Select element is rendered as a pop-up list.

Attributes

MULTIPLE

The MULTIPLE attribute is needed when users are allowed to make several selections, e.g. `<select multiple>`.

NAME

Symbolic name used when transferring the field's contents.

SIZE

Specifies the number of visible items.   If this is greater than one, then the resulting form control will be a list.

Examples

The following example defines a form that contains a single <select> element and a submit button:

```
<form method=post action="http://www.chaco.com/htbin-post/post-query">
What would you like to do today?
<select name="what-to-do">
<option>Drink Coffee
<option selected>Use Pueblo
<option >Use VR Scout
<option >Read A Book
<option>Take A Walk
<option>Buy A Bagel
</select>
<input type=submit value=" Enter my agenda ">
</form>
```

Also see

<form>

<option>

<input>

<strike>...</strike>

The Strike element indicates strikethrough, a font style in which a horizontal line appears through characters.

Also see

- <b> (bold)
- <em> (emphasis)
- <i> (italic)
- <strong>
- <u> (underline)

<strong>...</strong>

The Strong element indicates strong typographic emphasis, and is rendered in bold.

Examples

```
Have you become a member of the <strong>Chaco Developers' Program</strong>?
```

Also see

- <b> (bold)
- <em> (emphasis)
- <i> (italic)
- <strike> (strikethrough)
- <u> (underline)

<textarea> element

The Text Area element is similar to <input type=text>, except that this element lets users enter more than one line of text.

Attributes

NAME

Symbolic name used when transferring the field's contents.

ROWS
COLS

The ROWS and COLS attributes determine the visible dimension of the field in characters.   The field is rendered in a fixed-width font.

Examples

The following example defines a form containing a <textarea> field that may be used to enter comments:

```
<form method=post action="http://www.chaco.com/htbin-post/myscript">
<p>Let me know what you think of my web page!
What is your name: <input type=text size=40 name=Name>
What is your email address: <input type=text size=40 name=EMail>
<textarea name=Comment rows=5 cols=60>
<input type=submit value=" Send my comments ">
</p>
</form>
```

Also see

<form>
<input>
<option>
<select>

<title>...</title>

Note: This tag is not necessary for World output to Pueblo. Web Tracker supports this element so that normal web pages may be displayed.

The title should identify the contents of the document, and may be used in history lists and as a label for the windows displaying the document. Titles are not typically rendered in the text of a document itself.

The Title element must occur within the <head> of the document and may not contain anchors, paragraph elements, or highlighting. Only one title is allowed in a document.

Note: The length of a title is not limited, however, long titles may be truncated in some applications. To minimize the possibility, titles should be fewer than 64 characters. Also keep in mind that a short title, such as “pictures” may be meaningless out of context. A more meaningful title might be “Coyote’s Vanagon pictures”.

Examples

The following code defines the title of a web page:

```
<head>
<title>Coyote’s Vanagon pictures</title>
</head>
```

Also see

<head>

<tt>...</tt>

The Teletype element specifies that the text should be rendered in fixed-width typewriter font.

Examples

```
You may see the following text when you log in: <tt>This world is Pueblo 1.0
    Enhanced</tt>.
```

Also see

<code>

<pre> (preformatted text)

<samp> (sample)

<xch_mudtext>

<u>...</u>

The Underline element specifies that the text should be rendered in the document underlined.

Also see

- <b> (bold)
- <em> (emphasis)
- <i> (italic)
- <strike> (strikethrough)
- <strong>

<ul>...</ul>

The Unordered List element is used to present a list of items, typically marked by bullets.   An ordered list must begin with the <ul> element which is immediately followed by a <li> (list item) element.

An <ul> element is rendered with a slight extra left indent and each list item is marked by a bullet.

Attributes

TYPE

The TYPE attribute may be used on the <li> tag to specify how list items should be marked.   The following values are supported:

type=disc	Places a round bullet by the list item.
type=circle	Same as type=disc.
type=square	Places a square bullet by the list item.

Examples

The following example creates an unordered list with square bullets:

```
<ul type=square>
<li>Flour
<li>Salt
<li>Yeast
<li>Water
</ul>
```

Also see

<li>
<ol>

<xmp>...</xmp>

The Example element has been replaced by the <pre> element.   Web Tracker supports this element for compatibility with older documents.

Also see

- <code>
- <pre> (preformatted text)
- <samp> (sample)
- <tt> (teletype)
- <xch_mudtext>

xch_alert

This tag causes a 'beep' of a user-configurable nature to occur on the client machine.   Users can set the sound, the volume, and how frequently they want to allow alerts to come through.

This tag is intended to get the attention of another player.

Examples

This statement is very simple.   The only format currently supported is:

```
<img xch_alert>
```

Also see

xch_sound

xch_cmd

This is an anchor field which indicates that the specified command should be sent to the World (MUD, session, etc.) when the anchor is selected.

Examples:

In this example, the anchor text will be "Take a look at Jim", and when that text is selected (clicked on, etc), the command "look Jim" will be sent to the World:

```
Take a look at <a xch_cmd="look Jim">Jim</a>.
```

This example is similar, but uses **xch_hint**:

```
You can go <a xch_cmd="go north" xch_hint="Go North">North</a>.
```

Also see

<a> (anchor)

xch_hint

xch_device

Since there are potentially multiple sound devices on client machines, the xch_device must be used to specify the behavior of individual devices. The valid parameters are 'midi' and 'wave'.

Normally you would not specify the xch_device attribute for an xch_sound=play or xch_sound=loop command. The file will be played on the appropriate device. For xch_sound=stop, this attribute tells which device should be stopped.

If the xch_device attribute is not specified, it is taken to mean 'all devices'.

Examples

The following statement will set the volume for the wave device to 30:

```
<img xch_volume=30 xch_device=wave>
```

The following statement will stop any playing music on the midi device:

```
<img xch_sound=stop xch_device=midi>
```

Also see

xch_sound
xch_volume

xch_hint

Normal HTML anchors are of this form:

```
<a href="http://www.chaco.com/">Chaco's home</a>
```

They are usually displayed underlined or in a special color, to indicate that they are links to other documents. Some browsers display the URL (the http://www.chaco.com/ part) when the user's cursor moves over the anchor on the screen, but displaying raw URLs isn't of much use to naive users, so the xch_hint field can be used to give users more information about what will happen if they select the anchor their cursor is over.

Examples

When the user's cursor moves over this anchor, they might see "Go to Chaco's home page" in their client software's status bar:

```
<a href="http://www.chaco.com/"
   xch_hint="Go to Chaco's home page">Chaco's home</a>
```

Also see

<a> (anchor)
xch_cmd

xch_graph

This is an <img> field which specifies a command related to VRML.

Examples

This tag tells loads the specified URL and displays it in the VMRL window:

```
        <img xch_graph=load href="http://www.chaco.com/vrml/room.wrl">
```

The following command causes the current graphics image to be drawn before the client proceeds.   It's useful after modifying a VRML node, to force the graphics module to draw what you've sent it immediately.

```
        <img xch_graph="draw">
```

Also see

xch_graph_node

xch_graph_node

This is an <img> field which contains actual VRML code.

Examples

In the following exampe, the string is processed as VRML, then grafted into the existing VMRL scene, replacing the old node with the specified name (in this case, "cube_tr").

```
<img xch_graph_node="DEF cube_tr Transform { rotation 0 1 0 .5 }">
```

Also see

xch_graph

xch_mode

This is an <img> tag field, used to indicate the format of the text in the stream.   the parameter to xch_mode may be either 'html' or 'text'.

Examples

```
<img xch_mode=html>
<img xch_mode=text>
```

Also see

<xch_mudtext>

xch_mudtext

Text from non-HTML muds is formatted specially by the Pueblo client.   This formatting includes converting line ends to <br> tags, using a fixed-width font, etc.   Also, HTML tags seen in muds not known to generate HTML are ignored.   To cause HTML interpretation to begin, the HTML session must issue a </xch_mudtext> tag.   To turn off HTML interpretation, issue a <xch_mudtext>.

This tag is identical to <pre>, except that text is wrapped within the display boundaries.

Examples

For example, in a predominantly non-HTML system, you could do the following to make a word bold:

```
This </xch_mudtext><b>word</b><xch_mudtext> is bold.
```

Note that the logic of xch_mudtext is somewhat reversed from most HTML tags, in that the first one you issue is a </xch_mudtext>.   This is because the Pueblo client puts a <xch_mudtext> at the top of the session.   Your </xch_mudtext> ends that HTML style, then your <xch_mudtext> begins that style again.

Also see

<pre> (preformatted text)
<xch_mode>

xch_page

This tag is used in an interactive HTML session to indicate that any open HTML styles should be closed (for example, if an <em>, but no matching </em>, has occurred, the <em> will be closed by the <xch_page>).

This tag also disables any anchors in the HTML session before the <xch_page>. The intent here is to allow HTML World authors to have anchors which are local to a room in their World. When a player leaves that room, the client software should continue to show the room in the HTML scrollback window, but the anchors which were available there are no longer active.

Attributes

CLEAR

The CLEAR attribute accepts the values **links** or **text**.

The **links** value will disable all anchors in the HTML session before the <xch_page>. This is the default for this tag. Here is an example:

```
<xch_page clear=links>
```

The **text** value will delete all text in the HTML session before the <xch_page> statement. Here is an example:

```
<xch_page clear=text>
```

FGCOLOR
TEXT

The FGCOLOR attribute (which may also be specified as TEXT) is used to specify the text color for normal text on the page. The statement takes the following form:

```
<xch_page text="#rrggbb">
```

Where '#rrggbb' is a red-green-blue triplet used to specify the text color. Note that these values are in hexadecimal.

Examples

The following statement disables all previous links:

```
<xch_page clear=links>
```

So does this statement, since "clear=links" is the default:

```
<xch_page>
```

The following statement changes the color for disabled links to a dark gray:

```
<xch_page fgcolor="#404040">
```

Also see

<font>

xch_pane

This tag may be used in as part of an image (<img ...>) or anchor (<a ...>) to load a file into a specific pane. It may also be used to close an open pane.

The **xch_pane** tag may have the following values:

open	Opens a pane. The HTML string must include a **href** attribute indicating what should be displayed in the pane.
close	Closes a pane. If this value is used, the author must also specify a **name** attribute. (All other attributes to this command are ignored.)

Attributes

HREF

Indicates the file that should be loaded into the pane. This attribute must be specified with **xch_pane=open** or the pane will not be opened.

NAME

The name of the pane. This name is used internally to reference the pane. If this attribute contains no value, then the pane will be created without a name and cannot be closed by the MUD author. Panes created without an explicit name must be created with the **closeable** option so that they don't appear on the screen forever (see XCH_PANE_OPTIONS below.)

It is strongly suggested that authors use standard, simple names such as book, scroll, and map. This is because Pueblo remembers where named panes are positioned by the user. For example, if the user moves and sizes a pane named 'book' , closes the pane, and later reopens the pane, it will be positioned where the user last placed it. Users appreciate this level of control over the interface.

This attribute must be specified when XCH_PANE**=close**.

XCH_PANE_TITLE

On platforms which provide a window title, this attribute specifies the title. Note that the title is just for display, and may be any short descriptive string.

WIDTH

Suggested width for the pane. This value is used if the pane was not previously created and sized by the user. The value should be an integer.

HEIGHT

Suggested height for the pane. This value is used if the pane was not previously created and sized by the user. The value should be an integer.

XCH_PANE_MIN_WIDTH

Minimum allowable width for the pane. If the pane is sizeable by the user, the width may not be made smaller than this value. The value should be an integer.

XCH_PANE_MIN_HEIGHT

Minimum allowable height for the pane. If the pane is sizeable by the user, the height may not be made smaller than this value. The value should be an integer.

XCH_PANE_OPTIONS

This attribute specifies options for the pane. Attributes must be specified in a string delimited with spaces or commas. The allowable attributes are:

normal	Creates a pane that is a peer of the main application. The pane may be moved to the back of the application by the user. May not be used with the **floating** or **banner** options.
floating	Creates a pane that floats above all other windows. May not be used with the **normal** or **banner** options.

banner	Causes a banner pane to be created. The pane is created at the top of the window unless the bottom option is specified. May not be used with the **normal** or **floating** options.
sizeable	Creates a pane that may be resized by the user.
closeable	Creates a pane that may be closed by the user.
small_title	Creates a pane that has a smaller caption. These panes take slightly less space than normal panes.
top	Causes a **banner** pane to be created at the top of the window. This is the default for **banner** panes.
bottom	Causes a **banner** pane to be created at the bottom of the Pueblo window.
noscroll	This option suppresses the display of scroll bars in the pane.
fit	This option causes the pane to size to fit the contents of the pane. For more information on this option, see below.

If this attribute is omitted, the default values are "normal,sizeable,closeable".

Examples

The following statement opens a HTML file into a normal pane:

```
<img xch_pane=open name=book xch_pane_title="The book of the Realm"
       href="http://www.chaco.com/example.html">
```

This statement will close the floating pane opened by the previous statement:

```
<img xch_pane=close name=book>
```

The following anchor opens a bitmap file into a floating pane. The pane is not user-sizeable, and is sized to fit a 128x128 pixel image.

```
<a xch_pane=open name=map xch_pane_title="Map of the City"
       href="http://www.chaco.com/map.jpg"
       xch_pane_options="normal closeable floating"
       width=128 height=128>Look at the map</a>
```

The use of the 'fit' option

The **fit** option is very useful for making pages of information appear pleasing on different display devices. The pane is sized as well as possible to fit its displayed contents.

Because HTML wraps text to fit the width of the display window, the results of a **fit** pane are dependent on the original width of the pane. The HTML is wrapped to this width, and the pane is then sized to the result. Because of this, it is desirable to specify a WIDTH value.

Because the **fit** option is designed for use with static data, it generally should not be used with the **sizeable** option. You can do this, but the results are unpredictable.

Here is an example of using the **fit** option:

```
<img xch_pane=open name=dialog xch_pane_title="Registration Form"
       href="http://www.chaco.com/reg_dialog.html" width=640
       xch_pane_options=”floating closeable fit”>
```

Also see

Creating and using HTML dialogs

xch_prefetch

One of the big battles with Web-like information is minimizing 'chin-scratching' time of users by displaying partial documents as they are downloaded, etc.   Another way to do this is by giving clients hints about documents the user is likely to ask for next.   This is the purpose of the **xch_prefetch** tag.

Attributes

HREF

The document to prefetch.

XCH_PROB

The probability, from 0.1 to 1.0, that the document will be needed.

Examples

This tag tells the client that it's likely that the user will ask for the specified URL next.   See the XCH_PROB field for information about giving even better hints.

```
<xch_prefetch href="http://www.chaco.com/">
```

Also see

xch_prob

xch_prob

XCH_PROB is a field of the XCH_PREFETCH tag which indicates the probability of the specified URL being selected.

Examples

This tag tells the client that there is a 50 percent chance that the specified URL will be accessed next.   HTML clients could provide users with a 'Preferences' page which allows the user to specify a minimum probability for pages to prefetch.

```
<xch_prefetch href="http://www.chaco.com/" xch_prob=0.5>
```

This set of tags indicates that about half of the time, users go to 'doc1.html' next, about 30 percent of the time, they go to 'doc2.html', and 20 percent of the time, they go to 'doc3.html'.

```
<xch_prefetch href="http://www.chaco.com/doc1.html" xch_prob=0.5>
<xch_prefetch href="http://www.chaco.com/doc2.html" xch_prob=0.3>
<xch_prefetch href="http://www.chaco.com/doc3.html" xch_prob=0.2>
```

Also see

xch_prefetch

xch_sound

This is an <img> field which specifies a command related to sound.

The xch_sound tag may have the following values:

play	Begins playing the associated href.
loop	Begins playing the associated href. The sound will loop continuously until a 'play' or 'stop' is received.
stop	Stops any currently playing sound.

Examples

This tag loads the specified URL and begin playing the midi file:

```
<img xch_sound=play href="http://www.chaco.com/music/jazz.mid">
```

Also see

xch_device

xch_volume

xch_volume

This is another <img> field which allows the volume of the sound device to be set. The volume is in the range 0 (off) to 100 (maximum volume). You can specify the volume of the MIDI device by using **xch_device=midi** with **xch_volume**, and the volume of the waveform device by using **xch_device=wave**.

Note that the real volume range is from silent to whatever the user sets in their Sound control panel as maximum.

xch_volume may also be added to an **xch_sound** command to set the volume for a play or loop request.

Examples

The following statement will set the volume for all devices to silent:

```
<img xch_volume=0>
```

The following statement will set the volume for the midi device to maximum:

```
<img xch_volume=100 xch_device=midi>
```

The following statement will set the volume for all devices to half of the maximum setting:

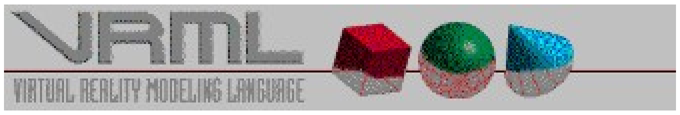
```
<img xch_volume=50>
```

The following statement will play a wave file (click.wav) at the volume setting of 30:

```
<img xch_sound=play xch_volume=30 src="http://.../click.wav">
```

Also see

xch_device
xch_sound



Background on the Virtual Reality Modeling Language

by Mark Pesce (March 1995)   (used with permission)

The Virtual Reality Modeling Language (VRML) is a language for describing multi-user interactive simulations -- virtual worlds networked via the global Internet and hyperlinked within the World Wide Web.

In 1969, the creation of ARPAnet, the forerunner of today's Internet, provided a methodology through which users could remotely manipulate computers, and the files stored on them.   While this was a powerful extension of computing, it was also confusing; without any clear sense of "what went where", access to Internet was restricted to the class of early "net surfers" who could remember where things were stored.   Most everyone else, even technically sophisticated users of computers, found it impossible to make sense of it.

In 1989, Tim Berners-Lee, a software engineer at the Center for European Particle Physics, CERN, developed a hypermedia system today known as the World Wide Web.   With the creation of the Universal Resource Locator, or URL, it became possible to tell anyone "where to go and how to get there" for almost any piece of data on the Internet.   Rather than a cryptic set of commands and accesses, the URL created a standard addressing mechanism for the data hidden in cyberspace.   In essence, it turned the entire Internet into the equivalent of a single (very large) disk drive, and made it possible to create documents which could encompass data from many different parts of the Internet, binding them together into a cohesive whole.

Even the URL required some improvement.   They're quite cryptic - for example, I can only tell you how to get to the VRML Forum home page by saying, "http://vrml.wired.com/", which is not really human-centered data, it's computer-centered data.   I need to make an effort to remember it at all.   As great as it is, the URL mechanism of the World Wide Web leaves a lot to be desired, particularly for human beings, and we comprise the user base; the Web was built for people, not for computers, and because of this, the Web has caused an enormous upsurge in Internet traffic. Very soon, the bulk of the traffic on the Internet will be Web-generated.

A few years ago, research into "sensualized" interfaces began to receive widespread attention in the press and the industry.   A wide range of technologies, which collectively came to be known as "Virtual Reality", began a fundamental change in the nature of the user interface, moving it to a human-centered design; where the space around the user became the computing environment, and the entire sensorium was engaged in the interface.   All of this was in an effort to make computers more responsive to the humans who used them, and focused around a basic realization: if something is represented sensually, it is possible to make sense of it.

Late in 1993, Mark Pesce, and Tony Parisi developed a three-dimensional interface to the Web which embodied many of the lessons learned in several years of research in both virtual reality and networking.   Upon communicating these innovations to Berners-Lee, Pesce was invited to present a paper at the First International Conference on the World Wide Web, in Geneva, Switzerland.   During a session to discuss virtual reality interfaces to the Web, attendees agreed there was a need for a common language to specify 3D scene description and WWW hyperlinks -- an analog of HTML for virtual reality.   The term Virtual Reality Modeling Language (VRML) was coined, and the group (headed by Pesce and Brian Behlendorf, of *WIRED* magazine) began work on a VRML specification immediately following the conference.

With the blessing of *WIRED*, Behlendorf set up an electronic mailing list to facilitate discussion of a specification for VRML.   The response was overwhelming; within a week, there were over a thousand members. The list membership quickly agreed upon a set of requirements for VRML, and began a search for technologies which could be adapted to fit the needs of it.

The list members proposed several worthwhile candidates, and after much deliberation the list came to a consensus: the Open Inventor ASCII File Format from Silicon Graphics, Inc. The Inventor File Format supports complete descriptions of 3D scenes with polygonally rendered objects, lighting, materials, ambient properties and realism effects. It has all of the features that professionals need to produce high-quality work, and an existing tools base with a wide installed presence.

A subset of the Inventor File Format, with extensions to support networking, forms the basis of VRML. Gavin Bell of Silicon Graphics has adapted the Inventor File Format for VRML, with design input from the mailing list. SGI has publicly stated that the file format is available for use in the open market, and has contributed a file format parser into the public domain to bootstrap VRML viewer development.

VRML is designed to meet three criteria: platform independence; extensibility; and the ability to work over low-bandwidth (14.4 kBps modem) connections. Early on, the designers decided that VRML would not be an extension to HTML, which is designed for text, not graphics.

The next generation of Web browsers will understand and interpret VRML; here are three examples of the kinds of projects VRML will enable:

The Interactive Media Festival Gallery Tour

The Interactive Media Festival (http://www.arc.org/) is an annual event of world-wide scope, which attracts some of the best talents of the new media. Their annual gallery and awards event, which takes place in Los Angeles in early June, will also be modeled in VRML and linked into their own Web site. People anywhere in the world will be able to tour the gallery space, examine the contestants' works, and follow links from these works other items of interest. In this way, IMF can have an international scope and an international reach.

WaxWeb 2.0

David Blair, the avant-garde filmmaker of *WAX: or The Discovery of Television Among the Bees*, and Tom Meyer, a doctoral candidate at Brown University, spent the last two years developing WaxWeb (http://bug.village.virginia.edu), a hypermedia web version of the film. David and Tom have spent the last 8 months bringing WaxWeb forward into VRML; at the release of WaxWeb 2.0 in the beginning of April, it acquires a VRML component - when a user goes to the WaxWeb VRML site, WaxWeb generates an assortment of rooms and links - but no two journeys through WaxWeb are ever completely the same. At last count, they'd created over 9000 (!) possible rooms to walk through, each of which is rich with content and anchors.

Virtual SoMa

The heart of the nation's multimedia industry is San Francisco's "Multimedia Gulch", located in its "South of Market" neighborhood, nicknamed SoMa. Several organizations have initiated a project to model SoMa in VRML and make it accessible through the Web. As a pilot, the 10-block area between 1st and 3rd Streets, from Howard to King, is being modeled as a VRML world. Many of the organizations in this neighborhood already have a web presence; this model links to their web pages. You can take a stroll down the virtual 3rd Street, to the block between Bryant and Brannan, find the offices of WIRED, click on the building, and go to their home page in the web.

There are some areas where VRML is still incomplete. Except for the hyperlinking feature, the first version of VRML does not support interactive behaviors. This was a practical decision intended to streamline design and implementation. Design of a language for describing interactive behaviors is a big job, particularly when the language needs to express behaviors of objects communicating on a network. Support for arbitrary interactive behaviors is critical to the long-term success of VRML; it will be included in the second revision of the VRML specification, which will be completed by December of 1995.

Mark D. Pesce

Moderator of the VRML mailing list

mpesce@netcom.com

Other Resources

The following pages are good places to look for more information on VRML:

- The Chaco VRML page (http://vrml.chaco.com/vrml/) contains a VRML test suite.
- The *Wired Magazine* VRML forum (http://vrml.wired.com/)
- The VRML Repository (http://www.sdsc.edu/vrml)
- The VRML Suppository (http://www.virtpark.com/theme/supp/) is fun and *definitely* interesting.

How to move through a VRML scene

You can move through a VRML scene using either the mouse or the keyboard. How movement works depends on what movement mode you are using.

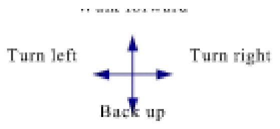
Walk mode

Walk mode is very similar to being in the scene and walking around.

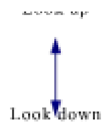
Mouse controls

To use the mouse, click on the scene and move the mouse in the direction you want to walk. Dragging in a direction has the following effects:

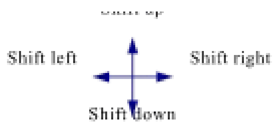
Normally, dragging the mouse 'walks' you around in the world.



When the Control key is pressed, you can use the mouse to look up and down.



When the Shift key is pressed, your position is shifted from where you are standing, but your orientation isn't changed.



You can also drag diagonally to combine movements. The further you move away from where you originally clicked, the faster you will move. The *3D Graphics* preferences dialog allows you to degrade the quality of the image while moving, so that you move faster (but things don't look as good while you're moving.)

Keyboard controls

The keyboard movements in Walk mode are as follows:

This key:	**Does this:**	**With the Shift key pressed it does this:**	**With the Control key pressed it does this:**
←	Turn left	Shift left	
→	Turn right	Shift right	
	Walk forward	Shift up	Look up
↓	Back up	Shift down	Look down

These all work the same as described for the mouse controls above.

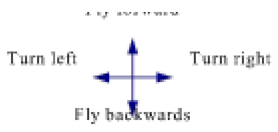
Fly mode

Fly mode is like being in an airplane. This airplane is really great... it can even hover and fly backwards. You move in whatever direction your nose is pointing. If you look up at the sky and then move forward, you start climbing. In addition in fly mode you can tilt the view from side to side.

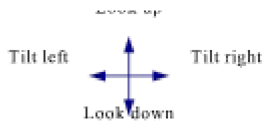
Mouse controls

To use the mouse, click on the scene and move the mouse in the direction you want to walk. Dragging in a direction has the following effects:

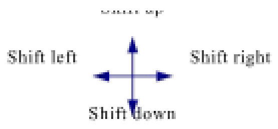
Normally, dragging the mouse flies you around in the world.



When the Control key is pressed, you can use the mouse to look up and down as well as tilt left and right.



When the Shift key is pressed, your position is shifted from where you are standing, but your orientation isn't changed.



You can also drag diagonally to combine movements. The further you move away from where you originally clicked, the faster you will move. The *3D Graphics* preferences dialog allows you to degrade the quality of the image while moving, so that you move faster (but things don't look as good while you're moving.)

Keyboard controls

The keyboard movements in fly mode are as follows:

This key:	Does this:	With the Shift key pressed it does this:	With the Control key pressed it does this:
←	Turn left	Shift left	Tilt left
→	Turn right	Shift right	Tilt right
	Fly forward	Shift up	Look up
↓	Fly backwards	Shift down	Look down

These all work the same as described for the mouse controls above.

Examine mode

Examine mode is for looking at objects modeled in VRML. Rather than moving through a scene, examine mode is like holding an object in your hand and turning and twisting it to look at it carefully. You can also move the object closer and further away from your eyes.

Imagine the entire scene to be contained in a big glass ball, which is right in front of you. By spinning this ball, you can look at all sides of the scene or object. To spin the ball, use the mouse to 'grab' a point, and spin the ball around, much as you would a large trackball. The program attempts to keep the same point of the ball under the mouse cursor.

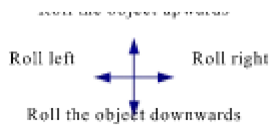
You can also zoom in or move around using the arrow keys. These do **not** affect the center of rotation. That is always defined by the sphere enveloping the world.

Probably the best way of using the examine interface is to rotate the object by grabbing it and rotating it using the mouse, and using the up and down keys to move in and move away from the object.

Mouse controls

As described above:

Dragging the mouse rotates the object you're examining:



Keyboard controls

The keyboard movements in examine mode are as follows:

This key:	Does this:	With the Shift key pressed it does this:	With the Control key pressed it does this:
←	Turn left	Shift left	Rolls the object left
→	Turn right	Shift right	Rolls the object right
	Move towards the object	Shift up	Rolls the object up
↓	Move away from the object	Shift down	Rolls the object down

Popup definitions

VRML is an acronym for Virtual Reality Modeling Language, and is a format for describing 3D scenes.

HTML is an acronym for HyperText Markup Language, and is a way of describing formatted text on the Internet.

TCP/IP stands for Terminal Control Protocol / Internet Protocol.   This is the main means of communications between machines on the internet.

SLIP stands for Serial Line Internet Protocol, and is used to connect to an internet provider using a serial line (a modem, for instance.)   SLIP is a wrapper around the TCP/IP protocol.

PPP stands for Point-to-Point Protocol, and is used to connect to an internet provider using a serial line (a modem, for instance.)   PPP is a wrapper around the TCP/IP protocol.

Ping is an internet command (available on both Unix and Windows) that allows you to see if you can reach a specified machine on the internet.   For example, `ping chaco.com` will indicate whether you can reach Chaco's main server.

About Chaco

About Chaco Communications, Inc.

Chaco builds advanced multiuser Internet software. Chaco software has been incorporated in a broad range of applications: from Internet game systems to document viewers, from Internet art galleries to geographic mapping systems. Our customers include leading network, media, and application software companies, educational institutions, art organizations, and individuals.

Chaco created VR Scout, the first VRML (3D graphics) viewer to be bundled with a commercial Web browser. VR Scout is currently the fastest complete VRML 1.0 viewer for Windows 3.1, 95 and NT. Chaco's revolutionary multiuser Internet client, Pueblo, adds the beauty of 3D graphics and multimedia to the social world of MUDs (multiuser dimensions).

Chaco supports the development of social media by donating a portion of its profits to MUD authors, artists, and musicians. Chaco donates server space to MUD producers. Chaco works closely with media tool and library vendors, operating system manufacturers, and network software companies to ensure our products conform to standards and have broad utility.

If you're interested in using VRML in a multi-user environment, check out ***Pueblo***, another Chaco Communications product. *Pueblo* is available from:

http://www.chaco.com/

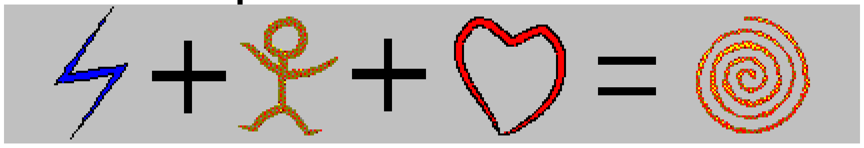
You can also send us electronic mail or write us. We'd love to hear from you.

Electronic mail:	info@chaco.com
Telephone:	408.996.1115
United States Postal Service:	Chaco Communications, Inc. 10164 Parkwood Drive, Suite 8 Cupertino, CA 95014-1533 USA

Also see

Chaco Corporate Values

Chaco Corporate Values



Chaco software brings people together.

We devote ourselves to customers, world society, our employees, and shareholders, in that order.

We recognize innovation as our lifeblood, and respect the innovations of others.

We tell the truth.

We never announce vaporware: it crushes the creative spirit in others and turns us into cynics.

Customers

We provide fair value for our customers, going the extra mile when in doubt.

We seek to understand our customer's needs, and offer creative solutions to their problems.

We avoid competing with our customers.

World society

We shrink the world.

We foster interactions and understanding between distant people and cultures.

We promote democracy, stability, happiness, and education.

Employees

We celebrate our people, because the unique talents of individuals help Chaco succeed.

We create a nurturing environment, free of discrimination, to encourage our employees to develop and contribute their talents.

We provide stable employment for employees who promote Chaco's values and health.

We encourage Chaco employees to communicate their concerns: to other employees, managers, executives, or board members.

Shareholders

We provide a fair return for investors.

VR Scout

Chaco Communications, Inc. has publicly released its VR Scout™ 1.2 VRML Plug-in viewer for Windows95 and Windows/NT. VRML is a standard for representing 3-dimensional scenes and objects on the World Wide Web.

The VR Scout 1.2 Plug-in works with your web browser as an internal viewer, integrating 3D images that you can maniplate directly into your web browser.   VR Scout works with both NetManage WebSurfer and Netscape Navigator.

VR Scout sets a new standard for VRML viewers, implementing the full VRML 1.0 standard at a high frame rate. In particular, VR Scout 1.0 supports GIF, JPEG and BMP textures, ASCII text and font displays, point-sets, and texture transformations, unlike other VRML viewers. VR Scout uses Microsoft's Reality Lab technology, making it able to exploit new 3D rendering boards for speed.

VR Scout is free for non-commercial and evaluation purposes, and can be downloaded from http://www.chaco.com/vrscout/. Commercial users and others wanting technical support may purchase VR Scout 1.1 for $49 direct from Chaco through email at buy-vrscout@chaco.com.

VR Scout is being licensed by leading software manufacturers for bundling with browsers and other multimedia products. VR Scout is also embedded in Pueblo™, Chaco's revolutionary multimedia Internet game client.

VR Scout 1.2 supports:

- All of VRML 1.0
- Microsoft Reality Lab, for fast software rendering and hardware acceleration
- Plug-in integration with Netscape 2.0beta3 and an upcoming NetManage WebSurfer
- Windows 95 and Windows NT
- Many Open Inventor nodes, so not-quite-compliant VRML files will work
- GZIP and ZIP files, automatically and transparently
- Multi-threading (on Windows 95 and NT)
- A headlight with a brightness control
- Walk/Fly/Examiner viewing modes with heads-up toolbar
- Textures (GIF, JPEG, BMP, and SFImage)
- Extensive help

New in VR Scout 1.2:

- Plug-in support for Netscape Navigator and Netmanage WebSurfer
- Support for the Netscape EMBED tag, for multiple VRML files inline in HTML
- Reality Lab 3d rendering.   Wow, many times faster than version 1.1!
- Toolbar, for heads-up navigation changes
- Tooltips anchor displays (hold the cursor over an anchor object to see the tooltips)
- Improved AsciiText support
- Transparency
- Transparent textures
- Support for concave faces
- Warnings for multiple top-level nodes and other common VRML file errors

Known bugs in this version:

- Navigation.   We'd like your feedback on navigation.   Tell us what we need to be doing, and we'll do it.   The planned changes for the release are: Be sure left/right navigation feels the same as forward/back navigation.   Slow down mouse navigation so it's a little less "tight".

On occasion (perhaps 1 in 100 scenes), the beta will silently fail to render a scene. This is a thread problem we're looking for, and will be fixed in the release. For now, just hit 'Reload' and all should be well.

Other stuff. Hey, it's a beta. Let us know if you run into problems, and we'll get them fixed as soon as possible. The address to email bugs to is: scout-support@chaco.com. People who report bugs first will receive, ummmm, VRML Jack-O-Lanterns or something. (Hey, we don't have as much money as Netscape for bug bounties! ;-)

VRML Support:

VR Scout renders the image progressively in pieces, so you can get started using a VRML scene before the whole scene is done being downloaded, parsed, and rendered.

VR Scout supports WWWAnchor, including hints, links to VRML documents, links to CGI scripts, links to HTML documents, etc.

VR Scout supports WWWInline, including relative URLs, nested WWWInlines, etc. If there are WWWInlines in the VRML scene, they are downloaded while you are viewing, walking around in the scene, etc. As the downloads of the WWWInlines are completed, they are added to the scene.

VR Scout supports Textures of GIF, JPEG, and BMP files. Again, these are downloaded while you're walking around in the scene, and as the downloads complete, the textures are added to their surfaces.

VR Scout supports the LOD (Level-Of-Detail) node.

VR Scout supports VRML "hints", like background color, etc.

VRML Extensions Supported:

BackgroundColor. This Info{} node allows scene authors to set the background color of the scene.
Example (sets background to black):

```
DEF BackgroundColor Info { string "0.0 0.0 0.0" }
```

Recipes

Recipe Index

In the spirit of sharing, here are some of our favorite recipes from the Chaco Communications kitchens:

Dan's Kalamata Olivada

Garlic is to cooking as madness is to art. Anon.

2 cups Calamata olives (before pitting)
3 tablespoons extra virgin olive oil
2 tablespoons toasted pine nuts
2 large garlic cloves

Pit the olives. (This is easy if you put a few olives at a time on a cutting board and smash the olives first with the flat of a large knife, then pick out the pits.) Blend all ingredients in a food processor. Season the paste with salt and pepper. Cover and refrigerate.

This recipe may be made a week ahead of time. It actually tastes better after a few days, when the flavors have blended together.

Ron loves this deep purple spread on toast for breakfast. (You have to have a high tolerance for garlic, though!

Ventana Eggplant Burgers

1 cup extra virgin olive oil
3 cloves garlic (of course)
2 teaspoons fresh rosemary
2 teaspoons fresh oregano
2 medium eggplants

A day before your meal, chop the herbs and mix into the oil. Cover and let the flavors blend overnight. Before cooking, slice the eggplant into 3/4 inch thick patties. Salt each side of each slice and let them stand about 10 minutes. With a paper towel wipe off the slices. (The salt leaches excess water from the eggplant.)

After the grill is fired up, dip each eggplant slice into the oil for one second on each side. Don't let the slices sit in the oil... eggplant will absorb oil amazingly quickly. Lay the slices on the grill, turning once to brown each side. Serve on a bun with the standard accoutrements: ketchup, mustard, and onion slices (best when sauted).

These are really simple to make once the oil is prepared, and they're amazingly tasty.

Cupertino Cilantro Pesto Linguine

1 cup pinolas (pine nuts)
2 bunches fresh cilantro
1 small or medium jalapeno
2-4 cloves garlic
1/4 cup extra virgin olive oil
2 tablespoons capers, drained and rinsed
2 tablespoons pink peppercorns
1 pound dry linguine

Toast the pinolas on a cookie sheet under the broiler, tossing a few times until brown. You don't need to oil the sheet, as the nuts have their own oil. Set the toasted nuts aside. (Toasted pinolas make a great snack on their own, but save some for the pasta!)

Cut off the stems from the cilantro. (Don't be picky. just hack off most of the stems from the bunch using a very big knife. Don't hack off your hand!) Throw the leaves into a food processor and chop them into a paste. Remove the seeds and stem from the jalapeno and add it to the food processor. Add the garlic and olive oil. You should end up with a bright green paste (*pesto* in Italian.)

Cook the linguine until it is chewy, but not soft (*al dente* is the Italian term.) Drain, then toss with the pesto, the capers, and the toasted pinolas. Sprinkle the peppercorns on top and serve with a nice red wine.

Mom's Baked Apple Pancakes

Madeleine Lussier

2 tablespoons butter
4 tablespoons sugar
3/4 teaspoon cinnamon
1 large sliced apple
1/3 cup flour
4 eggs
2/3 cup milk
1/2 teaspoon salt

Preheat the oven to 400 degrees.

Melt the button in a 10 inch skillet with an oven-proof handle. Combine 3 tablespoons of sugar with the cinnamon and sprinkle evenly over the butter. Place the apple slices in the pan and cook over medium heat for 3 or 4 minutes. Cool slightly.

Meanwhile, beat together the eggs, milk, flour, remaining 1 tablespoon of sugar, and salt until smooth. Pour evenly over the apples. Bake until brown and puffy, about 15 minutes.

Serve at once!