AmigaGuide™ 101

by Jerry Hartzler

NOTE: AmigaGuide™ is now available to developers on the 3.0 Workbench release, the CATS Developer CD and Commodore's closed Amiga developer listings area on BIX, CIX and ADSP. Developers with a Workbench or Includes License can amend their license to distribute AmigaGuide. Currently CATS is in the process of putting together an archive file that will be made available to the general public.

Compared to other platforms, Amiga-based utilities that display on-line documentation are relatively weak. Until 1991, Amiga users had to rely on basic text viewing utilities like *More*, which lack the navigational capabilities of hypertext programs. In a hypertext environment, when the user wants more information on a subject, the user simply clicks on a word and the hypertext utility automatically cross references the subject. No type of hypertext-like utility was available on the Amiga--that is, until *AmigaGuide* came along.

AmigaGuide can display plain ASCII text files and AmigaGuide databases. An AmigaGuide database is a single file that consists of a set of documents called *nodes*. If you were to convert a book into an AmigaGuide database, a convenient way to organize the database is to make each chapter of the book into a node. Each node may contain references to other nodes (chapters) or databases (other books), using a *link*. When a user selects a link--which usually appears in the form of a button within the text--*AmigaGuide* dereferences the link and displays its node. This makes it easier for the user to find the information he is looking for because the user no longer needs to search through a document. *AmigaGuide* already knows where the information is.

Buttons within *AmigaGuide* can do other actions as well, such as execute Shell or ARexx commands. An ARexx port is also built into *AmigaGuide* providing users the means to support and control the utility. Therefore, like a book, an AmigaGuide database can display illustrations. It can go a step beyond by playing sounds and music, and by being truly interactive with the user, able to ask questions and evaluate responses.

This article was written to familiarize you with *AmigaGuide* and the format of its databases.

Setting up AmigaGuide

AmigaGuide consists of the *AmigaGuide* utility and an Exec library called *amigaguide.library*. *AmigaGuide* should be placed in the *SYS:Utilities* drawer and *amigaguide.library* placed in the *Libs:* drawer. As of Release 3, the OS comes with a utility called *MultiView* that understands and displays AmigaGuide databases. *MultiView* requires *amigaguide.datatype* and *.datatypes.library* in order to display AmigaGuide databases.

AmigaGuide can run from the Workbench or the Shell. To run from Workbench, the default tool of an AmigaGuide database or text icon should be set to ‘‘AmigaGuide’’.

From the Shell, use the following command template:

```
AmigaGuide <my_DataBase>
```

where <my_DataBase> is the name of the AmigaGuide database or text file to display.

For example:

```
1> AmigaGuide Autodocs
```

attempts to open the ‘‘Autodocs’’ database. In its search for a database, *AmigaGuide* will first look in the current directory and then through the AmigaGuide path for the database.

The AmigaGuide path is a global environment variable. *AmigaGuide* stores its environment variables in the *ENV:AmigaGuide* directory assigned in RAM:.

The variable named ‘‘path’’ contains the list of directory names that *AmigaGuide* will search through when it attempts to open a database. Directory names are separated by a space. The path variable is set and stored in the *ENV:AmigaGuide* directory through the use of the AmigaDOS *SetEnv* command (Note that as of Release 3, if the *AmigaGuide* directory does not exist, *SetEnv* will not create it).

For example:

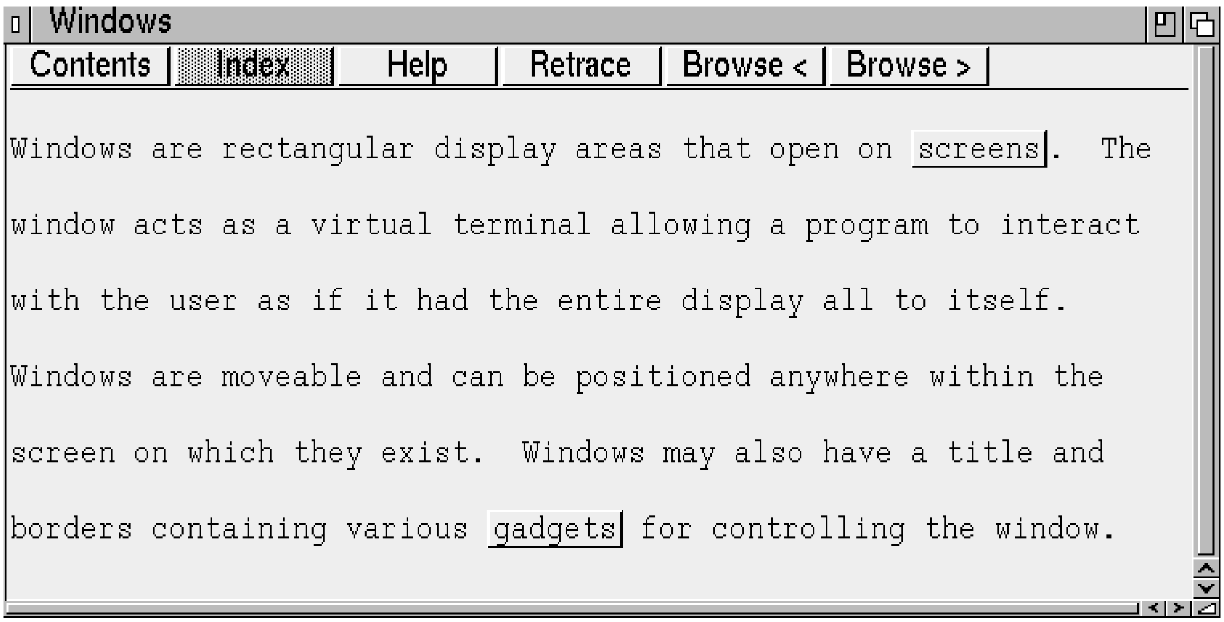
```
1> SetEnv AmigaGuide/Path "Workbench:Autodocs Workbench:Includes"
```

Of course all variables set in RAM: disappear once the computer is turned off. To prevent this, copy the file *ENV:AmigaGuide/Path* to the *ENVARC:AmigaGuide* directory. The system copies the *ENVARC:* directory to *ENV:* upon start-up.

The *MultiView* utility uses the *amigaguide.datatype* which uses this the environment variable as its path.

The AmigaGuide Window

AmigaGuide displays text within an Intuition window. The window contains scroll bars, buttons and pull-down menus making it easy to browse, search and print text.



A Sample AmigaGuide Window

Along the top edge of the AmigaGuide window is a row of six *navigation* buttons:

Contents: This button displays the Table Of Contents for the current database or node. See the @NODE and @TOC commands below.
Index: This button displays the Index for the current database. See the @INDEX command below.
Help: By deafault, this button displays the database named *help.guide* in the *s:* directory. Under 3.0, the database is named *amigaguide.guide* in the *Help:<country>/sys* directory. This database contains help in using *AmigaGuide*.
Retrace: This button goes back to the previous node.
Browse: These buttons step through the nodes in sequential order (the order they appear in the database).

AmigaGuide Databases

Creating an AmigaGuide database is quite simple. An AmigaGuide database is basically an ASCII text file, embedded with commands to tell *AmigaGuide* what to do. Let's take a look at the commands used in *AmigaGuide* and then make up a simple database.

There are three types of AmigaGuide commands. *Database commands* break up a document into nodes. *Node commands* set attributes within a node. *Action commands* are only found within a special node command called a ‘‘link point’’. As the name implies, action commands perform some action.

Database Commands

Database commands should not be used within the nodes themselves. They must start in the first column of a line. If a line in an AmigaGuide database begins with an at (@) sign, then *AmigaGuide* interprets the line as a command. Although the AmigaGuide commands appear here in upper-case, AmigaGuide commands are not case sensitive.

@DATABASE <name> - This command identifies a file as an AmigaGuide database. It must be the first line of the database.

@NODE <name> <title> - This command indicates the start of a node. The first node of a database should be named ‘‘MAIN’’. ‘‘MAIN’’ is typically the master table of contents for the database (see the @TOC node command in the next section). When *AmigaGuide* displays the node, it displays <title> in the window's title bar. If there are any spaces in the <title>, it must be in quotes. A node name cannot contain spaces, tabs, colons (’:’) or slants (’/’).

@DNODE <name> - This command indicates the start of a dynamic node. Dynamic nodes are beyond the scope of this introductory article and therefore are not discussed here.

@INDEX <node name> - This command tells AmigaGuide which node to use as the index for the database. When the user hits the ‘‘Index’’ button at the top of the AmigaGuide window, *AmigaGuide* will display the node specified by this command. The node can be in another database. If the node is in another database, the node name is the file name of the database followed by a slant (‘/‘), followed by the name of the node in the other database. For example, to access the MAIN node in another database called ‘‘other_database’’, the link point looks like this:

```
@{"my label" LINK other_database/MAIN}
```

The name of the other database can contain a full path to the other database. If it doesn't, *AmigaGuide* will search the AmigaGuide path for the other database.

@REMARK <remark> - This commands lets you add programmer remarks to a database. *AmigaGuide* ignores the remarks.

Node Commands

These commands are only valid within a @NODE. They must start in the first column of a line unless otherwise noted.

@ENDNODE <name> - This command ends a node.

@TITLE <title> - *AmigaGuide* displays <title> in the node's window title bar. It must start at the beginning of a line. This command isn't necessary if you use the <title> option in @NODE.

@TOC <node name> - This command tells AmigaGuide which node to display when the user hits the ‘‘Contents’’ button while displaying this node. If a node does not have a @TOC command, the table of contents defaults to ‘‘MAIN’’. The <node name> can be in another database. See the @INDEX command in the ‘‘Database Commands’’ section for more information on the format of <node name>.

@PREV <node name> - The ‘‘Browse’’ buttons can be reprogrammed so as not to step through the nodes in sequential order. For example, if this command appears in a @NODE, *AmigaGuide* will display the <node name> node when the user selects the ‘‘< Browse’’ button while in the current node. The <node name> can be in another database. See the @INDEX command in the ‘‘Database Commands’’ section for more information on the format of <node name>.

@NEXT <node name> - Similar to @PREV but will display the <node name> node when the user selects ‘‘Browse >’’ button while in the current node. The <node name> can be in another database. See the @INDEX command in the ‘‘Database Commands’’ section for more information on the format of <node name>.

@{"<label>" <action command>} - This command is referred to as a *link point*. AmigaGuide makes the string <label> into a button. The <label> must be enclosed by quotes. When the user hits that button, AmigaGuide carries out <action command> (see the next section of this article for a description of action commands). A link point does not have to start in the first column, it can appear anywhere on a line.

Action Commands

These commands are for use with the link point command described in the ‘‘Node Commands’’ section of this article.

LINK <node name> <line#> - This command loads and displays the <node name> node at the line number specified in <line #>. The <line #> parameter is optional and it defaults to zero. The node can be in another database. The <node name> can be in another database. See the @INDEX command in the ‘‘Database Commands’’ section for more information on the format of <node name>.

The <line #> parameter is optional and it defaults to zero. The default line number is line zero.

RX <ARexx command> - This commands executes the ARexx macro named in <ARexx command>.

RXS <command> - This commands executes the ARexx string file named in <command>.

SYSTEM <command> - This commands executes the AmigaDOS command named in <command>.

QUIT - When *AmigaGuide* encounters this command is shuts down the current database.

A Working Database

As an example, Let’s make the text file below into a simple AmigaGuide database. The simple database will consists of a single node.

```
What is a user interface? This sweeping phrase covers all
aspects of communication between the user and the computer. It
includes the innermost mechanisms of the computer and rises to
the height of defining a philosophy to guide the interaction
between human and machine.  Intuition is, above all else, a
philosophy turned into software.  See the Amiga ROM Kernel
Reference Manual: Libraries for more information.

Intuition screens are the basis of any display Intuition can
make. Screens determine the fundamental characteristics of the
display such as the resolution and palette and they set up the
environment for multiple, overlapping windows that makes it
possible for each application to have its own separate visual
context.

Windows are rectangular display areas that open on screens.
The window acts as a virtual terminal allowing a program to
interact with the user as if it had the entire display all to
itself. Windows are moveable and can be positioned anywhere
within the screen on which they exist.  Windows may also have a
title and borders containing various gadgets for controlling
the window.
```

```
Gadgets are software controls symbolized by an image that the
user can operate with the mouse or keyboard.  They are the
Amiga's equivalent of buttons, knobs and dials.

Menus are command and option lists associated with an
application window that the user can bring into view at any
time.  These lists provide the user with a simple way to access
features of the application without having to remember or enter
complex character-based command strings.
```

If *AmigaGuide* displayed the file above, it would appear without any buttons linking it to text files or databases. To mark this file as an AmigaGuide database, add the @DATABASE command. As mentioned earlier, it must start on the first line in the first column of the file.

Since there will be no index for this database, the next command can be a @NODE. For AmigaGuide to open a database, it must contain a node. Since this will be the first node in the database, it should be named ‘‘MAIN’’.

At the bottom of the file, on a separate line, add the @ENDNODE command. This tells *AmigaGuide* that the current node ends here. All nodes must contain an @ENDNODE command.

This is what we have so far:

```
@DATABASE
@NODE MAIN
What is a user interface? This sweeping phrase covers all
        .
        .
        .
complex character-based command strings.
@ENDNODE
```

The file is now an AmigaGuide database. However, since the database has only one node and has no link points, the database will still appear as plain text. Also, *AmigaGuide* will ghost all of the navigation buttons. The database can be broken into smaller nodes if desired. Normally such a short file would not need to be broken up, but it makes a good example.

One important issue to consider while designing an AmigaGuide database is how to lay it out. Some thought should go into the break-up of a document into manageable nodes, and how these nodes relate--and eventually link--to one another. Each node should consist of information dealing with one topic and should contain links to other related nodes.

Notice that the example text is about Intuition, but each paragraph discusses a different topic of Intuition. The first paragraph is an overview followed by four paragraphs: one each on screens, windows, gadgets and menus. This organization makes it convenient to make each paragraph a node with a table of contents in the beginning:

Intuition Table of Contents

Introduction
Screens
Windows
Gadgets
Menus

Since the ‘‘Table of Contents’’ node is first, it is the MAIN node. The other paragraphs follow the MAIN node, each in their own separate node. So as not to confuse *AmigaGuide*, each node within a database must have a different name.

```
@DATABASE
@NODE MAIN "Intuition Table of Contents"
Introduction
Screens
Windows
Gadgets
Menus
@ENDNODE

@NODE Intro "Introduction"
What is a user interface? This sweeping phrase...
@ENDNODE

@NODE Screen "Screens"
Intuition screens are the basis of any display...
@ENDNODE

@NODE Window "Windows"
Windows are rectangular display areas that open...
@ENDNODE

@NODE Gadget "Gadgets"
Gadgets are software controls symbolized by an...
@ENDNODE

@NODE Menu "Menus"
Menus are command and option lists associated...
@ENDNODE
```

When *AmigaGuide* loads the database above, *AmigaGuide* displays the MAIN node with the title string ‘‘Intuition Table of Contents’’ in the window’s title bar. As the user clicks the ‘‘Browse’’ buttons at the top of the AmigaGuide window, AmigaGuide steps through each node in the database, displaying each paragraph. Notice that each node in the database uses the window title parameter. As the ‘‘Browse’’ buttons step through the database displaying the different nodes in sequence, the AmigaGuide window title changes to match the title from the current node.

An important limitation of the database above is the only way to access the different nodes is using the ‘‘Browse’’ button to step through them in the order they appear in the database. The database still has no buttons within the nodes to link to other nodes.

Using Link Points

The link point command is probably the most commonly used command in a database. With it you can make a word in one node reference another node. For example, in the ‘‘Table of Contents’’ node from the previous example, you can make each entry from the table into a button that references its respective node.

The template for a link point is:

```
@{<label> <action command>}
```

The label parameter is the word or phrase in a database that is made into a button, and the action command is the action *AmigaGuide* takes when the user clicks the button. For our example, we want each topic in the table of contents to be a label and the action command for each to be a LINK command. The LINK command loads and displays another node.

```
@{<label> LINK <name> <line#>}
```

For example, if you change the word ‘‘Introduction’’ from the Table of Contents node into a link point it would look like this:

```
@{"Introduction" LINK Intro}
```

The <label> string must be in quotes. If you load the database into *AmigaGuide* after making the change above, the word ‘‘Introduction’’ would appear as a button on the Table of Contents screen. If the user clicks on the ‘‘Introduction’’ button, *AmigaGuide* displays the node named ‘‘Intro’’.

Unlike other AmigaGuide commands, link points do not need to start in the first column. This makes it easy to indent a button from the left edge of the window and also to embed a button within a block of text.

Link points can also link to nodes in other databases. As an example, consider the following excerpt from the ‘‘Intro’’ node:

```
philosophy turned into software.  See the Amiga ROM Kernel
Reference Manual: Libraries for more information.
```

If an AmigaGuide database of *Amiga ROM Kernel Reference Manual: Libraries* was readily available, it would be convenient if the word ‘‘Libraries’’ from the excerpt was a button, linking the word ‘‘Libraries’’ directly to the *Libraries* manual database. If the *Libraries* manual database was named *Libraries_Manual* and it was in the current AmigaGuide path, to make the word ‘‘Libraries’’ from the ‘‘Intro’’ node into a link point that opens the *Libraries_Manual* database, change the excerpt above to the following:

```
philosophy turned into software.  See the Amiga ROM Kernel
Reference Manual: @{"Libraries" LINK Libraries_Manual/Intro} for more information.
```

In the example above, when the user selects the ‘‘Libraries’’ button, *AmigaGuide* will try to display the MAIN node from the database called *Libraries_Manual*. *AmigaGuide* will first look for *Libraries_Manual* in the directory where the example database resides. If it’s not there, *AmigaGuide* searches through its path for the database. Alternately, you can supply a full path name to *Libraries_Manual*.

A link point can also be used to display a picture. For example, to display an ILBM image of a Workbench screen from the ‘‘Screen’’ node, use the SYSTEM action command:

```
@{"Picture of a Workbench Screen" SYSTEM sys:utilities/Display sys:Workbench.pic}
```

From the command above, when the user click the ‘‘Picture of a Workbench Screen’’ button, SYSTEM executes the *Display* utility. *Display* shows the ILBM file *sys:Workbench.pic*. Because *Display* is merely a shell command, you can substitute your own ILBM viewer for *Display*.

Below is the finished database with a few more interactive link points.

```
@DATABASE
@NODE MAIN "Intuition Table of Contents"
@{"Introduction" LINK Intro}
@{"Screens" LINK Screen}
@{"Windows" LINK Window}
@{"Gadgets" LINK Gadget}
@{"Menus" LINK Menu}
@ENDNODE

@NODE Intro "Introduction"
What is a user interface? This sweeping phrase covers all
aspects of communication between the user and the computer.  It
includes the innermost mechanisms of the computer and rises to
the height of defining a philosophy to guide the interaction
between human and machine.  Intuition is, above all else, a
philosophy turned into software.  See the Amiga ROM Kernel
Reference Manual: @{"Libraries" LINK Libraries_Manual/Intro} for more information.
@ENDNODE

@NODE Screen "Screens"
Intuition screens are the basis of any display Intuition can
make.  Screens determine the fundamental characteristics of the
display such as the resolution and palette and they set up the
environment for multiple, overlapping @{"windows" LINK Window} that makes it
possible for each application to have its own separate visual
context.

@{"Picture of a Workbench Screen" SYSTEM sys:utilities/Display sys:Workbench.pic}

@ENDNODE

@NODE Window "Windows"
Windows are rectangular display areas that open on @{"screens" LINK screen}.  The
window acts as a virtual terminal allowing a program to interact
with the user as if it had the entire display all to itself.
Windows are moveable and can be positioned anywhere within the
screen on which they exist.  Windows may also have a title and
borders containing various @{"gadgets" LINK Gadget} for controlling the window.
@ENDNODE

@NODE Gadget "Gadgets"
```

```
Gadgets are software controls symbolized by an image that the
user can operate with the mouse or keyboard.  They are the
Amiga’s equivalent of buttons, knobs and dials.
@ENDNODE

@NODE Menu "Menus"
Menus are command and option lists associated with an
application @{"window" LINK Window} that the user can bring into view at any
time.  These lists provide the user with a simple way to access
features of the application without having to remember or enter
complex character-based command strings.
@ENDNODE
```

Although this is a small sample and only a few basic commands are used, it is a good start to making your own AmigaGuide database.

The Doctor Is In

To make it easier to create a bug-free database, below is a list of common mistakes made when editing an AmigaGuide database.

As with any type of programming, errors are bound to crop up. One of the most common mistakes is spelling errors. Always double check your spelling of commands. Also, make sure all commands that should begin in the first column do so.

Symptom: *AmigaGuide* displays a database as a text file. For example, instead of a button, AmigaGuide displays the link point command that was supposed to display the button.

Cure: *AmigaGuide* does not think the file is an AmigaGuide database. Check that the first line of the file is the @DATABASE command and that it starts in the first column.

Symptom: When displaying a database node, *AmigaGuide* displays nothing in its window.

Cure: Either the @ENDNODE command is not present in the node or the @ENDCODE command does not start in the first column.

Symptom: *AmigaGuide* does not display a button or its label.

Cure: The <label> parameter in the link point is missing or not enclosed in quotes.

Symptom: When the user selects a button, *AmigaGuide* flashes the screen and displays the error message ‘‘Couldn’t locate <node>’’.

Cure: *AmigaGuide* cannot locate the node specified in the button's link point command. If the link point command points to a node in another database, the database must be in the AmigaGuide path or the node must contain the full path name to the database. The link point command also require a NODE name as the last parameter in the path separated by a slant (/). If the file is only text and not an AmigaGuide database, its path name should end with ''/MAIN''. If a node or file has any spaces in its name, the path name must be enclosed within quotes.

Symptom: *AmigaGuide* flashes the screen when a button is selected but displays no error message.

Cure: *AmigaGuide* does not recognize the <action command> parameter in the link point, or *AmigaGuide* cannot locate the specified node (see the cure above).

Symptom: *AmigaGuide* does not fully display a button.

Cure: Check that all of the link points end with a closing brace (}). Note that there is a bug in *AmigaGuide* that can crash the system if a link point does not end with a closing brace.

Symptom: *AmigaGuide* will not load a database and displays a ''Can't find Node'' requester.

Cure: All databases must contain at least one node. If the file is short, all the text can be in one node called ''MAIN''. If you do not wish to make the file a database at all, remove the @DATABASE command.

Symptom: *AmigaGuide* does not allow the the user to browse the database to its end. It either stops browsing before reaching the last node, or it gets caught in an endless loop, cycling through a number of nodes.

Cure: At least two nodes within a single database have the same name. Every node must have a different name unless they are in separate databases.