What MindMap can be Used for

First and foremost, MindMap is an environment in which applications can be developed. Since computers ultimately execute machine instructions, a development environment must eventually generate such instructions. Over the brief history of computers, major advances have been accomplished in isolating the developer from the low level of machine code. Each new level of isolation has been accompanied with more abstraction. At each new level the developer has to deal less and less with machine based concepts. In the very early days, it used to be AND and OR switches. Then came registers and machine instructions. These were followed by variables, conditionals, loops, and so forth.

MindMap is a member of the newest development level - graphical development. It is a language, albeit a graphical language.

The price we have paid in general for the increase in power has been a loss of control. The more complex the entities that are manipulated, the less influence one has regarding the details. All in all, more can be accomplished, but at the expense of being limited by the scope of the environment.

Lets briefly look at the power issue. What MindMap offers is twofold. Persons competent in development, in general, are now able to perform most tasks much easier, faster and will be less prone to errors. Persons not having a background in development, can now build their own applications, without having to resort to agents to do the work for them.

MindMap is an environment in which applications are actually assembled. Existing building blocks (*component*s) are used, arranged and connected, such that the result is an application. MindMap is not intended to be used as an environment in which building blocks are created.

Traditional development environments offer a language, a debugging facility and a variety of supplemental tools. The language is supplied in the form of a compiler or interpreter. The developer enters statements into an editor and has the statements translated - either up front (*compiler*) or at runtime (*interpreter*). The statements input reflect a certain, and hopefully intended, procedural flow of things. Therefore, the developer must conceive of the flow (*semantics*) and then mold it into the language (*syntax*). In order to easily locate and eliminate syntactic and semantic errors in the code, a debugger is most often supplied. This helps the developer step through the application and view isolated states. Finally, most environments offer various other tools that ease the general process of building applications (*linkers, profilers, etc*.).

Here, MindMap differs dramatically from the traditional approach to development, in that it shifts the emphasis of development. The syntax of MindMap is very limited. Statements are not actually keyed in. They are selected from various lists, which are displayed depending on the situation. It is not possible to create a syntactically invalid statement. MindMap is also non-procedural, so that the concept of mapping the flow to a series of statements is also completely different. The process is much more comparable to painting a picture. What You See is What You Get (*WYSIWYG*) best describes it.

The assembler of an application selects a building block, defines various attributes and then links it to other blocks. Think of it as a grown-up version of an erector set or Lego® blocks. First, one selects the **type** of block from an assortment of different blocks (*wheels, plates, blocks, windows, etc*.). Next, a decision is made regarding the **attributes** of the selected block type (*color, size, number of panes, etc*.). Finally, the block is **combined** with other blocks (*nobs are plugged into holes, parts are screwed together, etc*.).

Obviously, the types of applications that can be built with the environment is a function of the available components, their attributes and the ability to connect the components. This is fundamentally a question of expandability. MindMap ships with a large assortment of components, having many attributes, and numerous methods of being combined with one another. Nevertheless, the system offers diverse means of incorporating new components.

What we Assume you Know

The first major assumption is that you know how to handle MS-Windows. We assume that you are familiar with MS-Windows concepts and associated conventions. This implies that you know how to deal with either a mouse and/or the equivalent keyboard entries.

If you will be using MindMap in conjunction with databases, we assume that you are knowledgeable of general database concepts . This means that you are comfortable with the concepts of fields, records, and tables. If the underlying database engine you are using supports SQL (*Structured Query Language*), and it becomes necessary to execute an action by employing SQL, we also assume that you are either familiar with the SQL language or have access to other sources of information about SQL.

If you will be using MindMap to construct multimedia applications and will be using MCI commands (*Media Control Interface*), we make the assumption that you are either already knowledgeable or have access to other sources of information on MCI.

It will be helpful if you have some experience in application development. This does not mean that we expect you to be able to write code, but having experience in programming usually means that you:

- know how to decompose complex tasks into smaller, more manageable pieces;
- are accustomed to first analyze and then design, before you jump to the actual realization;
- have been exposed to the concepts of variables, loops, subroutines, etc.

Please do not misunderstand this point. It is absolutely *not* necessary to be a programmer in order to be able to build applications with MindMap. It simply makes it easier, in some cases, in that general programming concepts can be utilized. We must point out, though, that sometimes programming experience is a hindrance, in that it becomes difficult to think in non-procedural terms.

Throughout this manual and your use of MindMap, we will discuss various options or alternatives in the use and configuration of the components. You are highly encouraged to try the alternatives discussed. Only by experimenting, will you truly come to realize the depth of possibilities offered in MindMap. We look forward to your comments, suggestions and general feedback.

Hardware and Software Requirements

MindMap appreciates powerful hardware, but it does not give up on systems that are not state-of-the-art. The faster, bigger and better your system is, the faster your MindMap applications will run. Being a little more specific…

The more **RAM** you have, the better it is. MindMap will function on a 4MB system, although you might not enjoy the performance. We suggest a minimum of 8MB - obviously the more, the better.

Additional RAM is more important to MindMap than a faster **microprocessor**. Obviously, the faster the processor, the faster MindMap will execute. MindMap will function on a 386SX, although we would suggest a 486 as a minimum processor.

Since MindMap itself, and most applications built with it, are very user interface oriented, and thus utilize graphics extensively, we suggest a fast **graphics card**. MindMap will function in 4-bit, 640x480 pixels graphic environments, but we again suggest at least 256 colors and a fast graphics card.

Since MS-Windows is a virtual memory operating system, swapping from memory to the **hard disk** and back is quite common. MindMap and applications built with MindMap are generally compact in size, but, nonetheless, must often be swapped to a hard disk. Once MindMap has been installed, it does not make special demands concerning the hard disk. The faster the drive, the more free space it has, and the less fragmented it is, the better. MindMap consumes approximately5MB of hard disk space after installation.

Since MindMap is an MS-Windows application it does require one of the currently available MS-Windows environments. MindMap currently supports the 16-bit interface in MS-Windows, so it will run on the following versions of MS-Windows:

MS-Windows 3.11

MS-Windows for Workgroups

MS-Windows 95

If you plan on using the supplied database interface, you will require Microsofts **ODBC** (*Open Database Connection*). MindMap installs a version on your system, but please take caution here; there might be a conflict with other ODBC applications on your system. MindMap supports the 16- and 32-bit implementations of ODBC. In order to have access to some of the more sophisticated features (*scrollable cursors, etc.*), you must use ODBC 2.x.

If you use videos (*AVI files*) or other **multimedia features** in MindMap, it will be necessary to have Video for Windows and/or any other required drivers installed (*Sound card etc.*). Video for Windows is included in MS-Windows 95 and MS-Windows NT 4.0. Please refer to the vendor of your hardware for additional information.

If you have installed other software products, you might have opted to also install some supplied **graphic filters**. In this case, they will be recognized by MindMap and offered, along with the standard MindMap filters, when you attempt to import graphic files. These filters are denoted with an asterisk in front of the file type in the corresponding dialog box, while the MindMap filters appear without the asterisk.This applies to 16-bit versions of import filters which support the Lotus/Intel/Microsoft standard.

Suggested Reading

If you will be dealing with databases in your applications, we strongly recommend that you make yourself familiar with SQL. Although MindMap attempts to isolate you from SQL as much as possible, it might become necessary to generate more complex statements using SQL. A number of books and general introductory literature is available. Some of the titles we suggest include:

Author	Title
Taylor	**SQL for Dummies**; IDG, 1995
Bowman, et al	**Practical SQL Handbook**; Addison-Wesley, 1996
Gruber	**Understanding SQL**; Sybex, 1990

If you plan on creating applications that include multimedia features (*sound, video, animation, etc*.), you will be using Microsofts MCI (*Media Control Interface*) language. The scope of this language is basically determined by Microsoft, but it has provisions for vendor specific extensions. Beside referring to the appropriate Microsoft literature, it might also be necessary to read the information provided by the vendor of your additional hardware/software.

Technically, MindMap is a visual programming language. It includes a syntax which is graphical and does not permit you to create syntactically incorrect statements. It has a domain which is limited to building a rather broad scope of applications under MS-Windows. Since it does not do away with underlying concepts commonly found in application building (*loops, variables, subroutines, etc*.), we recommend that you eventually devote some time to these concepts. If you have a programming background, these concepts are more than familiar to you. If you are a novice, you might run into some roadblocks, while building applications. They most likely will be of a more conceptual nature. It is therefore recommended that you spend time at least browsing through some fundamental literature dealing with these basic concepts.

MindMap directs a major focus towards the user interface. Building appealing and functional user interfaces is part science and part art.

The scientific part is dealt with in the following representative publications:

Author	Title
Aaron Marcus, Nick Smilonich, Lynne Thompson	**The Cross-GUI Handbook**; Addison-Wesley, 1995
Susan L. Fowler, Victor R. Stanwick	**The GUI Style Guide**; Academic Press, 1995
The artistic side can be found in books such as:Ray Kristof, Amy Satran	**Interactivity by design**; Adobe Press, 1995