Philips Interactive Media America

Technical Note #68

The CD-I Player and the User Interface for Non-Volatile RAM

Lucy Lediaev

August 27, 1991

This document is a summary of the purpose and functions of the user interface to the CD-I player's non-volatile RAM. The information in this note was drawn from the AIM design document, "User Interface Design Criteria and Recommendations for NV-RAM," by Blake, Kaufman, van Luijt, and van Allen and from a talk given to the AIM Product Test organization by Tyler Blake. The goal of this summary to provide information on NVRUI to CD-I producers, designers, product testers, and other non-engineering personnel.

The CD-I Player and the User Interface for Non-Volatile RAM

What is Non-Volatile RAM?

Each CD-I player has a feature known as non-volatile RAM (NV-RAM). RAM is an acronym for "Random Access Memory." On most computer based-systems, RAM is used by application programs during a user session. When the user turns off the system, RAM is emptied of its contents and when the user starts up again RAM is, for all practical purposes, empty.

In contrast, the contents of non-volatile RAM are maintained even when the CD-I player is turned off. For example, from a game application, the user can save a game or game scores, store them in non-volatile RAM, and turn off the CD-I player. When the user turns on the CD-I player and inserts the game disc to play the game again, it is possible to retrieve the previously-stored information.

Non-volatile RAM on a base-case CD-I player is very small—approximately 8 kilobytes of storage. Fortunately, most files created by CD-I applications are small and do not require much storage space. However, the occasional application may create user files that tax the limits of NV-RAM, or small files from many applications may fill NV-RAM to capacity.

The concept of non-volatile RAM is somewhat unique, because it is tied to a single piece of player hardware rather than to a specific disc title. Most people are familiar with the concept of saving or recording something on a floppy disc or tape, but storing information that is tied to a specific program on a machine is a new notion for most consumers.

Let's imagine a fourth-grade classroom with three CD-I players. Each CD-I player is used by the twenty-seven children in the class when they have free time. Johnny plays a game of chess on player #1 and saves the game for another time. The next time he gets his regular work done and wants to play chess, Susan is using a story disc on CD-I player #1, and the teacher assigns Johnny to use CD-I player #3. Johnny puts his chess disc in the player and tries to retrieve his game from last time. His efforts are futile; the game he wants is stored on Susan's machine. At the same time, Anita is playing Dark Castle on player #2, and she tries to save her all-time high score. She gets a message telling her that there isn't enough room to save her score, because other children filled up the storage space when they were playing games on this machine.

The logistical problems of saving data from a specific CD-I application are obvious with players that are shared by several people. NV-RAM also presents some problems when a player is used by a single person. Over time, NV-RAM can fill up and somehow space has to be freed so the user can continue to save data.

Creating an Interface between the User and NV-RAM

Based on the current implementation of NV-RAM, AIM recommends, if at all possible, that application developers limit use of NV-RAM unless it adds significant value to a title. Of course, some applications simply demand the use of stored information. Thus, an application must allow the user not only to save information, but to manage its storage when the limits of that storage space are encountered. Clearly, applications that do not make use of NV-RAM do not need to have an interface to it.

But, what about those applications that do require its use? And what will happen when NV-RAM is full?

It became clear to the design and development personnel at American Interactive Media that a strategy had to be devised to let the user resolve the problem of NV-RAM filled to capacity. NVRUI (Non-Volatile RAM User Interface) was designed as an interface to allow the user to deal as easily as possible, and at several levels, with the problem of insufficient memory. This first implementation of NVRUI deals only with management of NV-RAM on a single user system. At some point in the future, it will probably be desirable to allow NV-RAM to recognize more than one user—thus allowing storage, indexing, and protection of files generated by more than one person. However, the trade-offs between complexity and added functionality do not seem sufficient to justify providing multi-user management capabilities at this time.

Basic assumptions were made about the users who might encounter NVRUI when using CD-I titles. Users would have no particular technical expertise; nor would they have prior training in the use of NVRUI. In fact, NVRUI represents a new facet of consumer electronics and thus has no precedent.

Because the user would not necessarily have previous knowledge, experience, or training with NVRUI, or anything resembling it, the design for NVRUI had to provide an interface that would be easy to interpret and use. To date, most CD-I titles use a visible cursor; therefore, NVRUI uses a visible cursor. The assumption is that a user who has previous experience with an invisible cursor would adapt more readily to the use of a visible cursor, than vice versa. In addition, the design should assure that the user's first encounter with NVRUI would be successful.

The basic purpose of NVRUI is to:

- Simplify the user's encounters with NVRAM by intelligently suggesting files for deletion.
- Allow the user to manually delete selected files from NV-RAM
- Allow the user to confirm the system's choice of files to be deleted from the current application to make room for additional information.

AIM has developed a hierarchy of strategies that developers should use in dealing with the memory constraints of NV-RAM. From the simplest to the most complex, the approach to NV-RAM should be to:

1. Avoid the use of NV-RAM altogether.
2. Use NV-RAM, but don't burden the user with its management.
3. Give the user limited access to its management.

Note: A slightly different approach is recommended for children's titles that use NV-RAM. The application should set up a mode of operation in which storage is all nwed only if there is available RAM. If no space were available, the program would simply fail to store the data (with appropriate warning messages).

When the User Encounters NVRUI

In giving the user limited access to management of NV-RAM, NVRUI takes a "stepped" approach, allowing the user to deal with the simplest cases first. The user progresses to the next and more difficult step of management only when the previous step fails to resolve the problem of freeing sufficient space in NV-RAM for the user to complete the operation of saving data. Thus, NVRUI acts as an intelligent adviser to the user. To do so, NVRUI uses the following priority of selection criteria for files to be deleted:

1. Look for files in NV-RAM from the application in which the user is currently working:

   a) Look for the oldest file that is equal to or larger in size than the current file.
   b) If the first file found is of insufficient size, then look for newer files until one of sufficient size is found.

2. If, in using criterion #1, no size match is made, NVRUI looks for any other file that is not in a format recognized by NV-RAM. (Only designated NV-RAM files and preference files—extremely small files created when the user sets preferences within an application—are recognized by NV-RUI.)

3. If there is still no match, select the oldest file from all other applications that is equal to or larger in size than the current file.

4. If there is still no match, leave the mode in which NVRUI plays an advisory role and enter a mode that lets the user manually view and delete files. In this mode, the user can sort the files by date, size, and name. The user can also "protect" a file. Protecting a file requires the user to explicitly "unprotect" it to be able to choose it for deletion.

It is expected that NVRUI's default mode, in which the system advises the user, will be used most of the time. NVRUI is designed so that the user will only rarely encounter and have to use the "manual" mode.

If the user does have to select files for deletion manually, it is possible to sort the files in several ways—by date, by size, and alphabetically by name.

NVRUI also has a simple "bubble" help facility that consists of text messages to advise the user.

Other characteristics of NVRUI's design also prevent the user from having to deal with computer jargon and unfamiliar concepts. For instance, memory capacity and space available are represented in percentages of the total NV-RAM space; these percentages are shown graphically. A convention of rounding up the size of the file to be saved and rounding down the size of a previously stored file frees users from having to deal with fractional file sizes. It also avoids situations where a user may try to replace one 3% file (really 2.2%) with another 3% file (really 2.7%). Although this approach may result in the loss of a small amount of storage (probably less than 2% of the total), the benefits of a simple approach outweigh the potential inaccuracies.

How is NVRUI Implemented in CD-I?

In its current implementation, one of the ways NVRUI can be implemented is as an application separate from the CD-I application. When the user wants to perform an operation that requires use of NV-RAM, the CD-I application sends the user to the NV-RUI application via chaining or forking. When operations requiring access to NV-RAM are completed, the user is sent back to the main CD-I application. In addition, the software engineer can link NVRUI directly into the application, so that NVRUI becomes part of the CD-I application.

Producers and designers who need more information on implementing NVRUI in a title should contact Tyler Blake at (213) 444-6504. Software engineers who have engineering questions about NVRUI should contact Charles Golvin at AIM (213) 444-6515.