HM - Card

Version 1.4

INTRODUCTION

These pages represent a preliminary version of the documentation. Any questions will be promptly answered via e-mail: **nsherbak@iicm.tu-graz.ac.at**

HM-Card is a PC authoring system for hypermedia applications, and is based on a novel method of hyper-linking. The editing and linking facilities allow the creation of conventional hypermedia documents, links and tours of various complexity. HM-Card will automatically maintain the integrity of all hyper-links created. The hypermedia databases created with HM-Card can then be integrated into other systems or browsed separately using a stand-alone viewer.

The process of creation in HM-Card does not require any special programming knowledge on the author's part. Animation or complex question and answer analysis algorithms can be created without resorting to traditional programming.

Although the terminology used in this manual may be familiar, a basic understanding of HM-Data Model (the underlying concept of HM-Card) is assumed. See **Chapter 03** HM-Data Model for more information.

HM-Card runs under Microsoft-Windows version 3.1.

1. HM-Card Basics

Information native to HM-Card is stored internally in the form of a database. By using a secure form of database, HM-Card keeps track of information and maintains the integrity of links from section to section.

There are three main components of HM-Card presentations: Media Objects, Pages and Collections. Individual media objects are grouped in pages of information. Pages can then organised in various ways as collections.

2. Pages and Media Objects

Blocks of information displayed by HM-Card are referred to as pages. A page is basically a linear list of various types of multimedia information, called media objects. When a page is called upon to present its information, it will access each media object more-or-less sequentially. An object may not actually involve a visual display, but some other form of media, such as a sound.

Objects which have a visual component are built up in layers on the screen, obscuring images behind them . Non-visual objects have no effect on screen graphics. An analysis object, in conjunction with a branch objects allows the flow of object presentation to jump around, repeating or skipping sections of objects. In this way a complex interaction can be provided within a particular page.

3. HM-Data Model

Databases in HM-Card are made up of Structured Collections (S-collections or

just collections for short). Every S-collection has a unique identifier, a name used by other S-collections for addressing, and generally an associated page which will be called *content* henceforth. An S-collection encapsulates a particular internal structure. The internal structure is a set of pages and/or other S-collections (called *members* henceforth) related by a number of computer-navigable links. One of the members of the S-collection must be designated as *head* of the S-collection. Note that we connect S-collections and pages. Note also that links are encapsulated within a particular S-collection: they may only be defined between members of the same S-collection. In this sense, links belong to a particular S-collection; they do *not* belong to the hypermedia database or to either of the members related by the link, hence the "local referential integrity" of the model.

Figure 1: Internal Structure of S-collections

The HM Data Model provides a number of predefined subclasses of S-collection. Simply speaking, subclasses of S-collection define a particular topology of encapsulated links, as can be seen in Figure 1. For the purposes of this paper, it suffices to describe the five predefined subclasses illustrated in Figure 1: *Envelope*, *Folder*, *Menu*, and *Freelinks:*

1. **Envelope**: all members of an envelope are fully related, every member is linked to every other member.

2. **Folder**: an ordered set of members, each member having links to "next" and "previous" members.

3. **Menu**: a simple hierarchical structure; the head of a menu S-collection includes links to all other members, and each member is provided with a link to the head.

4. **Freelinks**: members of a freelinks S-collection may be arbitrarily connected by means of special Insert_Link and Remove_Link operations.

Four important issues must be understood:

1. Conceptually, an S-collection X can be a member of $n \geq 1$ other S-collections Y_1 ,Y_2, . . . , Y_n. This does not mean that X has to be physically stored *n* times; whether some clever pointer mechanism or other technique is used instead is part of the implementation and of no concern to the user. In this important sense the HM Data Model offers a higher level of abstraction than models involving arbitrary links: in analogy to programming languages where abstract data types offer a higher level of abstraction than data structures based on explicit pointers.

2. Not only simple S-collections but also complex S-collections can be members of more than one S-collection. In this manner, larger "chunks" of information (including their particular encapsulated navigational strategies) can be reused in various parts of a hypermedia

database.

3. S-collections need not be "stratified". An S-collection may well consist of some simple S-collections, some slots, and some complex S-collections.

4. Recursive membership is both possible and meaningful.

3.1 Navigational Operations

There are several things that a collection can do, of which the access function is the most used. When a collection is accessed it presents its content, the page associated with it. By wrapping pages with successive complex collections, the structure of a multimedia presentation is created.

Thus, Navigation within the HM Data Model is accomplished via three operations Access, Zoom_In, and Zoom_Out , which are addressed to a particular S-collection.

All S-collections can respond to the message Access. It implies executing the S-collection´s content (i.e., presenting some text, picture, audio, video clips etc.). Complex S-collections without content forward the message to their head.

Link following within the HM Data Model is simply a form of message passing. At any particular moment in time, the user can navigate only through a single, specific S-collection called the *current container*. Only members of the current container can receive messages during navigation. Links within the current container serve as a message passing paradigm. A concrete member of the current container is the *current member* for each particular navigational step. More precisely, the member that most recently received the message Access is the current member. Only members related (linked) to the current member can be accessed (can receive the message Access) in the next step of navigation.

Since links are encapsulated within an S-collection, they become available for navigation only when the S-collection has been "entered" by means of the Zoom_In operation, which is available for all complex S-collections. The Zoom_In message is automatically addressed to the current member. For example, if the current member is S-collection "e" in Figure 2, the user can apply the Zoom_In operation in order to make "e" the new current container. After a Zoom_In operation, the head of the new current container automatically becomes the current member and is visualised appropriately (i.e. receives message Access).

The Zoom_Out operation is the complement of Zoom_In. Zoom_Out restores the current container and current member to the state they had before the *most recent* Zoom_In. Extending the functionality of Zoom_Out to give access to any S-collection of which the current collection is a member is provided by the operation Zoom_Up. Together, Zoom_In, Zoom_Up and Zoom_Out provide users with the capability of navigating in a direction orthogonal to the conventional plane of link-based browsing. Thus, we say that the HM Data Model

supports an additional dimension of browsing hypermedia databases.

Figure 1: Operations "Zoom In" and "Zoom Out"

The three operations Access, Zoom_in, Zoom_Up and Zoom_out constitute the navigational element of an S-collection´s public interface. Navigation in our model leads to the activation of part of the public interface, not directly to information content (i.e. primitive nodes or anchors) as in many other models. In fact, this object-orientation eases many of the problems associated with the visualisation and aggregation of composites.

3.2 Operations for Modifying the Database

New S-collection instances are created with the operation Create, whereby the name, content, and head are given as parameters as shown in Figure 1. Once an S-collection has been created, new members can be inserted and existing members removed by means of the operations Insert_Member and Remove_Member. All links within S-collections belonging to the classes Menu, Envelope, and Folder are maintained automatically in accordance with their associated regular structure.

Of course, the regularly structured subclasses do not restrict us from using S-collections having arbitrarily connected members. Users can create an S-collection of class Freelinks and explicitly define its link structure using the messages Insert_Link and Remove_Link.

At any point, an S-collection can be deleted with the Delete operation. In this case, its content and all links encapsulated within it cease to exist. If a deleted S-collection has been reused (defined as a member) by other S-collections it is removed from S-collections by means of the Remove_Member operation .

4. The Editor, Linker and Viewer

Three different programs make up the HM-Card authoring package. The are the Editor, Linker and Viewer.

The Editor is used to create and edit the media objects contained in a page. These pages are the base, upon which complex collections are built. All objects native to HM-Card are available for editing and the author is provided with powerful facilities for incorporating materials already existing in electronic form (text files, graphic images, sound, etc.) into presentations.

The editor supports a conventional set of operations with information pages, a particular page can be created, stored into a database, copied via clipboard into another HM-Card database or deleted from a database.

Media objects can be created, µ §deleted, modified, copied and/or moved within a particular page.

Following is a brief description of the 16 types of media objects available.

Vector Graphics: These consist of common graphical elements such as lines, rectangle, curves, splines, etc.

Text: These can be loaded into HM-Card in two ways, by typing the information in directly, or by importing it from an existing text file.

Message and Mailbox: A message object is used to display variable information, such as that stored in a mailbox. A mailbox can remember states or information. This information can then be used to control the flow of page execution.

Import: This object is a powerful aspect of HM-Card. It allows many types of media to be loaded and displayed, from simple textual information, to complex information from any OLE (Object Linking and Embedding) compatible application.

Group: By importing a section of an existing page as a group, complex graphics or animation objects can be re-used in other pages.

Pause and Branch: These objects control the flow of object display in a page. They allow the flow to be halted temporarily, or diverted to a different section of objects.

Beep and Flash: To attract or alert the user these object supply visual or audio warnings.

Input and Analysis: The input object is used for input from the user in the form of text or as a selection of options. This input can then be analysed in a complex fashion and results used to control page execution.

Snapshot: This object allows the a portion of the page to be captured, saved and restored at a later date.

Fill: Complimenting the vector graphic objects, this objects will fill an enclosed area with colour.

Animation: Any visible object or group can be animated in timed sequences involving arbitrary trajectories.

Navigational objects: basic navigational operations of the HM-Data Model (i.e. "Zoom_IN", "Zoom_Out", "Next" and "Prior" can be incorporated into information pages in the form of objects).

The Linker is used to create and link complex collections. It allows all collections to be linked together to form a hypermedia presentation and it will maintain the integrity of all links created.

New S-collection is created with the operation Create the name, content, and head and type of the newly created S-collection should be defined on this stage.

There exist the following types of S-collections:

1. **Envelope**: all members of an envelope are fully related, every member is linked to every other member.

2. **Folder**: an ordered set of members, each member having links to "next" and "previous" members.

3. **Menu**: a simple hierarchical structure; the head of a menu S-collection includes links to all other members, and each member is provided with a link to the head.

4. **Freelinks**: members of a freelinks S-collection may be arbitrarily connected by means of special Insert_Link and Remove_Link operations.

Various attributes of collections may be set and altered with this program.

Thus, such attributes define a particular type of link visualisation.

The system supports the following types of link visualisation (i.e. types of anchors).

- Link visualisation in the form of scrollable list.
- Link visualisation in the form of pushbuttons.
- Link visualisation in the form of clickable areas ("hot" areas).
- Link visualisation in the form of "hot"-words.

Additional attributes of an S-collection can install special regimes for the S-collection: "Automatic Zoom In", "Automatic Zoom Out", "Permanent Presence of a Predefined Layout", "Transparent members" and "Form Filling".

The regime of "Automatic Zoom In" means that when a member of the S-collection is accessed in some way, the operation "Zoom In" is carried out automatically.

In analogy, the regime "Automatic Zoom Out" means that when the head of such S-collection is accessed in a way different from "initial opening" of the S-collection, the operation "Zoom Out" is carried out automatically. For instance, an S-collection of type "Folder" is closed automatically after visualisation of a "last" member.

The predefined Layout is a particular information page which is visualised as a first part of the screen layout during browsing of the current container. For instance, if a particular S-collection (say, S-collection "A") is defined as a member of two different S-collections-containers (say, as a member of "B" and "C" having different predefined layouts), then visualisation of the S-collection "A" depends on a particular context (i.e. it is visualised differently as a member of "B" and as a member of "C"). In analogy, some information common to all members of a particular S-collection (say, a "help" message) can be defined once as such a predefined layout.

Form is a named combination of attributes which are attached to every member and/or head of a particular S-collection.

When a certain member (or head) is visualised on the screen, a form defined for the S-collection (container) is also visualised as a part of the screen layout. Fields attached to members of the S-collection are filled by the user when a particular

member is inserted into the S-collection. Attributes attached to the head are dynamically calculated.

When members are inserted into an S-collection defined with this attribute, the system demands filling predefined fields with particular values .

Once an S-collection has been created, new members can be inserted and existing members removed by means of the operations Insert_Member and Remove_Member.

Links within an S-collection of class Freelink and explicitly defined using the operations Insert_Link and Remove_Link .

Of course, existing S-collections can be modified (operation) or deleted (operation) in due course.

The Viewer is an independent program used to view presentations previously created using the Editor and Linker. Although the user can alter many aspects of the presentation they are viewing, no changes may be saved. Thus, key-word search, annotations and macro (tour) recording and playback facility are available.