Reflective viewing and Interactive Synchronization in Hypermedia Editing

Kazuo Kajimoto, Tomoyuki Nonomura

KANSAI Information and Communications Research Laboratories
Matsushita Electric Industrial Co., Ltd.
1006, Kadoma, Kadoma-shi, OSAKA, 571, JAPAN

ABSTRACT

In this paper, some concepts that are important for hypermedia editing are discussed.

At first, the concept named "reflective viewing" is introduced. Reflective viewing is such a concept that when an user do something, then he/she can immediately get the result of that operation. This reflective viewing gives users trial and error hypermedia editing.

Next, the concept named "interactive synchronization" is introduced. This interactive synchronization concept has three features as follows:
(a) Dynamic Buttons.
(b) Objects an user focuses are interactively set front in the hypermedia.
(c) Relationship of synchronization is independent from clip video’s play speed.

Developing interactive synchronization concept, we introduced event network graph representation.

After that, we implemented a hypermedia editor based on above concepts, which can treat text, graphics, clip video and buttons as objects. As a result, it achieves programless and "trial and error" hypermedia editing.

INTRODUCTION

Recently, personal computers have grown up so powerful that they have the ability to handle hypermedia. The hypermedia enables us to represent information with proper media as text, graphics, sound, video and so on[1]. Moreover it offers us a flexible retrieval method according to our association process[1]. Hypermedia is useful. But it is still difficult for many users to built up hypermedia itself.

DATABASE SYSTEMS FOR ADVANCED APPLICATIONS '91
Ed A Makino
©World Scientific Publishing Co.

REFLECTIVE VIEWING

On DTP, "trial and error" editing is implemented with WYSIWYG concept. The key of "trial and error" processing is that the result of an user's trial is reflected immediately on screen, and he/she can confirm it. We call this concept "reflective viewing".

Reflective viewing is important not only on DTP, but on hypermedia editing. Mentioned above, reflective viewing is popular on text and graphics editing. There is, however, no hypermedia editing systems which realize all operations both in space and in time according to reflective viewing concept.

Our objective is offering such hypermedia editor which can treat text, graphics and "clip video" as objects and realize reflective viewing concept, where clip video means the video object clipped its time range within full time range of an optical video disk and it can be played as moving pictures.

DIRECT VIDEO MANIPULATION

To handle video, we’ve already developped the video interpolation hardware[2], and the video manager which is a set of system calles in operating system layer[3]. We implement a hypermedia editor on BTRON1 specification operating system[4] that provides multi task and multi window environment. BTRON1 specification recommends application software which treats text and graphics to observe a manner of human machine interface to guarantee compatibility of operations[6]. The manner is called TRON manner, and it consists of "direct manipulation" and "noun-verb model", that means, after objects selection, an user can do something on it.

We adopt this TRON manner for text and graphics, and apply it for clip video. So if an user has already used some application software on BTRON1 specification operating system, he/she can treat clip video by analogy.
Fig. 1 shows an example sequence to move a clip video between windows. This sequence is done in our system inside as follows:
(a) The source window asks the video manager to move the clip video to the destination window.
(b) The video manager informs the destination window of the clip video moving request.
(c) The destination window replies the acceptance.
(d) The video manager draws the clip video in the destination window.

We call this interface "direct video manipulation". Direct video manipulation provides the methods of selecting, moving, coping, deleting, and resizing clip video as well as graphics only with the pointing device. And the manner is extended, that is, the methods of cutting focused area from clip video and changing play speed with the pointing device are added. All of these operations according to reflective viewing concept are implemented thanks to the video manager.

CLIPPING VIDEO TIME RANGE

In this section, the human machine interface in time axis, that is, how to clip video time range is described.

The sub-processes of making hypermedia are as follows;
(1) Planning
(2) Making objects (typing text, drawing graphics, clipping video time range and so on)
(3) Composing a hypermedia with the objects
(4) Making links between hypermedia

Focussing the second sub-process, it is divided into three sub-sub-processes as follows;
(2-a) Selecting proper scene in video disk as clip video
(2-b) Checking both of start and end frame numbers of the selected scene
(2-c) Setting both of numbers in editing software

Most of conventional clip video handling software adopts the table setting interface to clip video time range[7]. This interface leads such problem that an user cannot understand the relationship between frame number and the time range intuitively.
To resolve this problem, we design new interface according to reflective viewing concept. The interface must process from (2-a) through (2-c) as any order. Fig.2 shows the sequence of clipping video time range with our new interface.

With this interface, an user can clip time range of video by direct manipulation on screen as follows;
(a) An user searches proper scene as clip video through optical video disk. Fig.2-a corresponds to sub-sub-process (2-a).
(b) When he/she chooses proper scene, resize the video time range rectangle in the tool box from left to right until the start frame with the pointing device like Fig.2-b. This operation corresponds to both sub-sub-process (2-b) and (2-c).
(c) About the end frame setting, he/she can do in the same way. (Fig.2-c) Clipping video time range is done as this.

INTERACTIVE SYNCHRONIZATION

There are two approaches to handle time axis in hypermedia, that is, scenario approach and interactive approach. Now, these two approaches unite with each other. We started from interactive approach, and introduce time relationship between objects. The approach is called "Interactive synchronization".

Interactive synchronization has three features as follows;
(a) Dynamic buttons[1]
(b) Objects related to the clip video an user focuses are shown front in the hypermedia.
(c) Objects synchronize with clip video at any play speed.

The first feature, dynamic buttons, is the buttons which have time attribute. Using dynamic buttons, an user can retrieve related information directly from objects appearing in clip video.

The second feature represents the concept that objects, an user is interested in, should be shown prior to the others. It can be determined what clip video an user is interested in, that is the one played as moving pictures. Of course, an user can select which clip video is played as moving pictures, interactively.

The third feature means that the time relationship between clip video and the other objects depends not on the play speed, but on the scene of clip video.

Interactive synchronization concept gives an user good interface to understand the information represented with hypermedia, because it has interactive ability in time axis handling.

EVENT NETWORK GRAPH

Implementing interactive synchronzation concept, we introduce "event network graph" which represents the relationship of synchronization.

Window's open/close, object's appearance/disappearance and so on can be regarded as events, then a scenario is defined as a set of the relationship between these events. When events and the relationship between them are represented as nodes and directed arc, respectively, a scenario is represented as a directed graph. This graph is called "event network graph".
In event network graph, there are two kinds of arc. One is "invoking arc", the other is "successive arc". An event network graph has two special nodes. One is "entrance node", the other is "exit node".

When scenario starts, the entrance node invoked at first. Next, the events related to the entrance node are invoked. As this sequence, the events which related to invoked events with invoking arc are invoked successively. On the other hand, successive arc makes the condition whether an event can be invoked. Finally, the final event, that is, exit event is invoked and the scenario finishes completely.

Text, graphic and button objects are usually represented by two nodes and one successive arc between these nodes. Each node represents appearance/disappearance event, respectively. When an user sets a clip video object's time as a set of successive events in event network graph, the graph can represent the synchronization relationship between the clip video object and other ones. This derives three features of interactive synchronization concept.

(a) The definition of event network graph proves that the synchronization between a clip video object and buttons can be represented.

(b) The second feature depends on the interpretation of the graph. We regard a clip video object in the hypermedia as active video. This active video can be changed by an user interactively at any time. And the successive events of a clip video object are effective only when the clip video is active video.

(c) When the system adopts frame numbers as successive events of clip video, the relationship depends not on its play speed, but on the scene of it.

Fig. 3 shows an abstract of event network graph.

The logical data format of edited hypermedia is defined to be interpreted as event network graph. Fig. 4 shows RNF expression of a subset of this logical data format.

[Hypermedia data] ::= (Version segment) (Entrance node segment) (Hypermedia body) (Exit node segment)
[Hypermedia body] ::= [Stream data] (Block body list)
[Block data] ::= (Block start segment) (Block body list) (Block end segment)
[Block body list] ::= NULL (Block body) [Block body list]
[Block body] ::= (Appearance event node segment) (Block component) (Disappearance event node segment)
[Block component] ::= [Text block group] (Graphic component) (Clip video component) (Button component)
[Clip video component] ::= (Clip video start segment) (Clip video segment) (Clip video additional information components list) (Clip video end segment) (Frame events list)
[Frame events list] ::= NULL (Frame event node segment) (Frame events list)

where [ ] means non-terminal symbol, ( ) means terminal symbol.

Fig. 4 Logical data format of hypermedia

Our hypermedia editor interprets this logical file format, and construct event network graph in memory.

MAKING EVENT NETWORK GRAPH BY AN USER

The last problem about our hypermedia editor is how to make an event network graph as easily.

Conventional software for desk top presentation adopts programming interface to describe of synchronization because of its flexibility. But this interface forces an user to remember commands, syntax and so on, as a result, only a few users can use it.

We've already explained event network graph representation is useful for interactive synchronization description. So, we adopted the method of event network graph directly drawing instead of programming.

Fig. 5 shows the tool box of implemented interface.

At first, when an user makes an object in the window, our hypermedia editor automatically creates two nodes and one successive arc between them. (Fig. 5-a) Secondly, an user plays clip video with buttons in the tool box. When an user finds the scene he/she wants to adopt as invoking other object's appearance/disappearance event, then he/she draws invoking arc in the tool box with pointing device directly. (Fig. 5-b) Then the editor inside interpretes the user's operation, divides the clip video's successive arc at the frame position, creates new node between divided arcs and sets one invoking arc from the node to the appearance/disappearance node. These sequence will be continued until all invoking arcs are set. (Fig. 5-c)
(a) In the tool box, skelton of event network graph is drawn.

(b) Making event node, and drawing invoking arc from it to destination node.

(c) Final result

*Fig. 5 Implemented tool box*

Of course, an user can retry to set arcs at any time. that is, reflective viewing concept is also available in this interface.

Both fig.6 and fig.7 are hypermedia examples played according to fig.5's event network graph. Fig.6 shows the living area of lions, when a lion appears in the clip video. On the other hand, fig.7 shows that of elephants, when elephants appear in the same clip video instead of a lion.

CONCLUSION

In this paper, two concepts are discussed, that is, reflective viewing and interactive synchronization. The concepts are important for hypermedia editing.

A hypermedia editor which can treat text, graphics, clip video and buttons is implemented on BTRON specification operating system. This editor is named "AV Super-sheet". Our "AV Super-sheet" provides programless and "trial and error" hypermedia editing.

After more studying the mathematical characteristics of event network graph, our next interest is extending the graph and designing human machine interface to built complete scenarios without any programming, either water-fall approach.

REFERENCES


