Research Fellowship Exchange

FINAL REPORT FOR THE MOVie PROJECT

Victoria A. Burrill

Informatics Department, Rutherford Appleton Laboratory, UK
Multimedia/Hypermedia Department, ZGDV, Germany

April 1993

INTRODUCTION

The MOVie project — Mapping Objects on Video by interactive editing — is the result of a six-month fellowship exchange between the Multimedia/Hypermedia Group at ZGVD, Darmstadt and the Informatics Department of Rutherford Appleton Laboratory, UK. The first half of this exchange took place between October 1992 and April 1993; the second half is expected to take place in the latter part of 1993.

The purpose of the exchange was a mutual learning and sharing of ideas and expertise specifically in the field of multimedia, but also more generally in a wide range of institute and cultural aspects. As part of this process it was intended for the visiting research fellows to develop software which would be of research and practical benefit not only to the host but also the home institutes. It was also the intention to promote this work and both institutes by the publication of several journal and/or conference papers. The first half of the exchange has already fulfilled these aims.

This report consists of three items: an informal description of the implementation and philosophy behind MOVie (version 1.0) plus two papers intended for submission to refereed journals which describe not only MOVie but also other related aspects as applied to two other multimedia systems at ZGVD — ShareME and HyperPicture.

Acknowledgements

I would like to thank Professors Encarnacao and Hopgood for making the exchange possible, and all my new colleagues at ZGDV for making the exchange so interesting, so much hard work, but of course great fun as well! Particular mention must go to Kaisa (my friend as well as colleague), Thomas and Jochen (a.k.a Max) (suppliers of endless bewilderment and cups of coffee) and Ken (who nobly supported my absence from RAL and signed all the claim forms), as well as Benny (and the tropical fish), Christoph (for all the lunches), Elfriede, Klaus, Martin, Ute, Wolfgang and Xavier.

"Veni, Vidi, VCR." (I came, I saw, I VCRed.)

1. MOVie USER MANUAL

The prototype MOVie interface is somewhat simplistic, and ideally the mapping data should be post-processed to check for consistency errors. Nevertheless, the prototype version of MOVie is a start, and as long as the human video editor is careful, it works (most of the time!). Due to the consistency problems however there is a recommended sequence in which to use the various sections of the MOVie editor. This is described below. If you know exactly what you are doing then it is possible to go against the sequence, but this is not really recommended. (Not even *I* would do this without a lot of thought and at least one back-up.)

a. Create the directory for the film

Under the $MOVIE_PATH directory (currently hard-wired as *victoria/MOVie/Films*) create a directory named after the film you want to map plus the extension *.movie*. Use underline characters _ to space the words of the name, and distinguish between upper and lower case letters if required. For example, the directory for the film *Raiders of the Lost Ark* might be named *$MOVIE_PATH /Raiders_of_the_Lost_Ark.movie*.

b. Set up the file containing the video image sequence

Within the appropriate .movie directory, create a file called *video* containing the sequence of video images comprising the film. Currently only the tiff format is supported.

c. Set up the skeleton MOVie files

Within the appropriate .movie directory, copy and rename the following files from $MOVIE_PATH/Films:

$MOVIE_PATH/Films	$MOVIE_PATH/Films/<film>.movie
screenplay.proto	screenplay
movie.scenes.proto	movie.scenes
movie.objects.proto	movie.objects

The files should contain the following:

screenplay

1	1	1	0 0 0 0 0 0 0 0 0 0 0 0 0
X	7	2	0 0 0 0 0 0 0 0 0 0 0 0 0

movie.scenes

2	3	
1	0	Beginning_of_film
2	0	End_of_film

movie.objects

0	1

Only the screenplay file should require additional editing, replacing the X with the number of frames plus one in the given video sequence. (The "extra" frame is required to fool the system into treating the penultimate frame as part of the preceding scene, rather than as the final cut. This is a hack to provide a quick but dirty code fix.)

d. Using the MOVie editor: Defining scenes and cuts

When using the MOVie editor, first open the required film. This will contain one scene (called *Beginning of film*), one cut (called *End of film*), but no objects. Play through the video, defining the start frames of scenes and cuts as required. The name of the scene/cut can be typed into the text widget of the *Scenes/Cuts* area. It is subsequently added as a scene or cut at the current frame number by clicking the *Add scene* or *Add cut* buttons. (*Delete* deletes the currently-selected scene or cut.)

e. Using the MOVie editor: Defining objects in the film

Once the scenes and cuts have been defined, define all the objects used throughout the video. Again, the names of these can be typed into the text widget of the *Objects* area, then added by clicking the *New object* button. (*Delete* deletes the currently-selected object.) Include object qualification if required (such as *Indiana Jones — teenager* or *Indiana Jones — angry*.)

f. Using the MOVie editor: Defining objects in each scene

Once all the objects in the film have been defined, associate the required objects with their appropriate scene. (Cuts cannot be associated with objects.) To do this, select a scene, select an object (by clicking on it in the objects list), then click the *Add to scene* button. An ampersand & will appear next to that objects's name in the object list during the appropriate scene.

g. Using the MOVie editor: Defining mapping points for each object

For each (visible) occurrence of each object in each scene, select the scene (by playing the video or clicking on the scene name in the scenes list), select the object to be added to the scene (by clicking on the object name in the objects list), select the first frame in which this object is visible, draw a rectangle around the object and then click the *In point* button. Select the last frame in which this object is visible, draw a rectangle around it and then click the *Out point* button. This now defines the extreme positions of that object. Step backwards and forwards through the video, drawing rectangles and clicking the *Nudge point* button as required in order to define intermediate interpolation points to a desired level of detail. To map another object, select it by clicking in the objects list; first select the appropriate scene if this is different. As each object is mapped, the data are written out to temporary files; thus frequently switching between objects to be mapped may result in a poor response time as the files are written and read back again.

h. Other features of the MOVie editor

Displaying a frame outside the current scene results in the deselection of the objects. This is not a problem but it does mean that if you accidentally scroll outside the current scene you have to reselect the objects which takes time and patience. Therefore, the *Drawing pin* check button next to the scroll bar can be used to toggle between the modes "stay in this scene" and "allow scrolling outside this scene".

The small text area within the VCR section of the interface is used to indicate the type of the currently mapped point (if there is one). The characters -->*[]* indicate that it is an in-point; *[> <]* that it is a nudge-point; *[˜]* that it is an interpolated point, and *[]*--> that it is an out-point. When a rectangle is being drawn, the text area contains *?*.

The *Debug* check box is used to print-out (in the MOVie-invoking window, if there is one) the ID and mapping data for the current object. This is of the following form: *Object=5=7+ 14- 17+ 20. 25*- where + indicates an in-point, . a nudge point and - an out-point.

Comment on the mapping process

From limited experience with the mapping process thus far (using the polar bear and ZGDV videos) some initial, general observations can be made about the mapping process.

Firstly, if the object being mapped is small and/or moving fast (particularly randomly and fast) then make the rectangle relatively large. Whereas it is quite fun to watch people trying to click on such objects, it does not improve their tempers if they are trying to do something serious. This is probably obvious, but bears repeating.

Secondly, when mapping the ZGDV video (which panned gently from one side to the other but did not contain much action as such) it was necessary to map most of the frames in which an object was partially visible because it was just coming into or going out of shot (and hence changed in size at each frame). Once the object was fully visible however then linear interpolation was sufficient to map its path without any additional mapping points. (This method was indeed so accurate that it was possible to determine which frames had been edited out from discontinuities in the linear interpolation. Visually, however, the break in the original image sequence could not be detected.)

Thirdly, there would seem to be a limit to the number of objects which it is sensible to map per scene. For conventional videos, the number of objects per scene is usually a maximum of about 5 or 6; (conversation

scenes have fewer objects; crowd scenes more). Remember that if your application always draws all the rectangles then your user is quite likely to get information overload.

2. FILE STRUCTURES

Mapping data are stored in a series of flat ASCII files. These may be edited by hand if you are desperate. This is not something I would recommend however.

Structure of the screenplay file: *screenplay*

The screenplay file consists of a list of records each representing one scene or cut. The records are of the following structure:

<frame-num> <scene/cut-type> <scene/cut-id> <list*object-id>

where:

<frame-num>	frame at which that scene/cut begins
<scene/cut-type>	scene or cut type 0 = beginning of film 1 = scene 2 = cut 3,4,5,6 = unused 7 = end of film
<scene/cut-id>	ID of the scene/cut (index into movie.scenes)
<list*object-id>	list IDs of objects in this scene (indices into movie.objects)

Structure of the scenes file: *movie.scenes*

The scenes file consists of a header record followed by a list of records each representing one scene or cut. The records are of the following structure:

RECORD 1: <num-scenes/cuts> <next-scene/cut-num>

OTHERS: <scene/cut-ID> <application-ID> <name-of-scene/cut>

where:

<num-scenes/cuts>	number of (useful) scenes/cuts in film
<next-scene/cut-num>	next available scene/cut ID
<scene/cut-ID>	actual scene/cut ID (from screenplay)
<application-ID>	application-dependent ID (0 in MOVie files)
<name-of-scene/cut>	string name of scene/cut

Scene/Cut IDs are allocated (by the MOVie editor) in sequence using the next available scene/cut ID value (and then incrementing it). Unless you do a careful bit of editing and file deletion, scene/cut ID numbers should not be reused.

Structure of the objects file: *movie.objects*

The objects file consists of a header record followed by a list of records each representing one object. The records are of the following structure:

RECORD 1: <num-objects> <next-object-num>

OTHERS: <object-ID> <application-ID> <name-of-object>

where:

<num-objects>	number of (useful) objects in film
<next-object-num>	next available object ID
<object-ID>	actual object ID (from screenplay)
<application-ID>	application-dependent ID (0 in MOVie files)
<name-of-object>	string name of object

As for Scene/Cut IDs, Object IDs are also allocated in sequence.

Structure of the individual objects files: *movie.object<id>*

The individual objects files consist of a list of records each representing one interpolation point. The records are of the following structure:

<frame-num> <shape> <interp> <point-type> <coords>

where:

<frame-num>	frame number at which this point applies
<shape>	shape drawn to define the object ("R" = rectangle)
<interp>	interpolation method to use ("L" = linear)
<point-type>	mapping point type + = in-point . = nudge-point - = out-point
<coords>	coordinates of bottom-left/top-right corner of rectangle

Unless something has gone very wrong with the MOVie editor, there should always be matching pairs of in- and out-points, with a variable number (including 0) of mid-points inbetween.

The application-dependent scenes and objects files: <*applic*>*.scenes* and <*applic.objects*>

Once the scenes, cuts and objects used in the video have been defined, the *movie.scenes* and *movie.objects* files can be copied and edited according to the requirements of the application. Thus an application can resolve to the cinematic context (such as *Indiana Jones*) or its own context (such as *Man in hat*) as required.

To do this, first determine a unique name for the underlying instance of the application that will use these new names. (For example, the tourist information database about Rostock used by ShareME is known to MOVie as *Die neuen Bundesländer*.) Copy the *movie.scenes* file and/or *movie.objects* file to this unique name with the extension *.scenes* or *.objects* as appropriate. Use the underline character _ as a separator. Then edit the application-dependent version of the file(s), changing the names of the scenes/objects to their application-dependent forms. Again, use the underline character _ as a separator if required. When creating a MOVie OO object you will need to specify the cinematic name of the video (for example, *Temple_of_Doom*) and (if application resolution is required) the application-dependent name as well (for example, *Die_neuen_Bundesländer*).

In addition, the application-dependent versions of the scenes and objects file can be used to define an application-significant call-back string. MOVie Scene and Object OO objects can read-in this call-back data and, if required, can pass it back to the controlling application.

For example, if an entry in the *application.objects* file was edited to contain:

```
...
5  display_node_123456  Indiana_Jones
...
```

then when the user clicked on a rectangle representing object 5, the associated MOVie OO object could also be interrogated to return the call-back string *display_node_123456* which the underlying application could then use to display some relevant piece of information.

Note that because of the way the data is read-in by MOVie, words within string names in these files must be separated by some character other than blank; the underline character _ is recommended.

Temporary files

During the mapping process, any object selected for mapping will cause a temporary version of its mapping data file to be created. This file will be named, for example, *movie.object3.tmp*. If the MOVie editor is then abandoned and exited, these .tmp files will be removed (and any mapping data created will be lost); if the MOVie editor is saved and exited, they will be renamed to their corresponding *movie.object<id>* name.

If file protections on the film directories are not set up correctly, it may not be possible for the operating system to do this, in which case the .tmp files will have to be deleted or renamed as required (and change the protections!).

Adding new mapping points, objects or scenes

It is recommended that the MOVie editing sequence described above is followed. Failure to do so may result in inconsistencies within the data for reasons described below.

When a new in- or out-point is added, its corresponding out- or in- pair is automatically created (with the same extents), at the last or first frame in the current scene, or just before or after the next in- or out- point. (Mapping point ranges cannot overlap each other or cross scene/cut boundaries.) The extent of this automatically-created point may be altered by displaying its corresponding frame, drawing the correct rectangle and then clicking the *In point* or *Out point* button as appropriate. In- and out-points may be redefined within their current range but not outside it. (You can make a mapping range shorter by bringing the points closer together, but not longer.) Deleting a nudge point results in the removal of that nudge point; deleting an in- or out-point results in the deletion of their whole mapping range.

If a new scene is to be added, it is important to check that its frame number does not occur within the range of any mapping points or this will result in an inconsistency of the data. For example, if an object has the mapping points: *7+ 14- 17+ 19. 25-* then adding a new scene at frame 16 will maintain the integrity of the data; adding a new scene at frame 20 will corrupt it. Hence, it is recommended that the MOVie editing sequence described above is followed.

Use great caution when adding or removing objects from scenes. It is possible to remove an object from a scene for which there is mapping data. Thus the data become inconsistent.

3. CODE DOCUMENTATION

Code files

INTERFACES	MOVie.nib
CLASSES	Animator.m MapPoint.m Mobject.m MovieController.m
	MovieMatrix.m OOMobject.m Scene.m VideoView.m
MFILES	MOVie_main.m
TIFF_FILES	movieicon.tiff

MovieController.[hm]
The main controlling object (obviously!). Has responsibility for handling call-backs from all the interface widgets, enabling/disabling parts of the interface, opening the required film and associated files, saving/abandoning the mapping data on exit, and performing the interpolation calculations.

Scene.[hm]
Object representing each MOVie scene/cut. Has knowledge about its starting frame number, ID, scene/cut type, name and the list of objects contained in it. Can also be queried about, and has capabilities to alter, the list of objects it contains.

Mobject.[hm]
Object representing each MOVie object. Has knowledge about its IDs, cinematic and application names, and whether or not it is in the current scene.

MapPoint.[hm]
Object to store mapping record data (frame number, mapped shape, interpolation method, coordinates etc).

MovieMatrix.[hm]
General purpose matrix class handling the lists of scenes/cuts and objects.

VideoView.[hm]
Handles the display of the video images read-in by the MovieController. Also handles the drawing of the rectangles and draws the appropriate mapped shape on top of the current frame.

Animator.[hm]
The animator object code borrowed from Heiko, and in turn borrowed from RE Crandall. Generates the clock ticks when playing the video.

OOMobject.[hm]
One of the object-oriented objects. Included in MovieController for testing purposes only.

Known bugs

1. The use of an extra, nonexistent frame at the end of the sequence

2. The mapping data can get corrupted if you really mess about with it

3. u.s.w...

Desirable improvements

1. Fix bugs
2. Handle other image file formats
3. Clear error messages (on a mouse move?)
4. Prototype other OO objects
5. Provide an extra button to display all extents associated with the current frame
6. Soft-wire MOVie directory name using $MOVIE_PATH
7. Shell script for creating raw MOVie data files
8. File consistency checking
9. Improved MOVie interface, especially of the extents mapped for the current object.
   (Use a bar with lines marking in/out/mid points?) Also for scenes within the whole video
10. Other mapping shapes and interpolation methods. (Modify code to cope with these.)

4. OBJECT-ORIENTED MOBJECTS

The following sections describe some of the functionality required by the various MOVie object-oriented objects — *OOFilms*, *OOScreenplay*, *OOScenes* and *OOMobjects*. Note that as yet, only MOVie objects have been implemented (though there are a few omissions in this code).

Object-Oriented Films

initialisation	OOFilms();
destruction	~OOFilms;

get number of films in MOVie	(int) get_number_films();
get titles	(char*) q_title:(int)ordinal;
get video name	(char*) q_video_name:(int)ordinal;
string search	(list) q_films:(char*)search_string;

Object-Oriented Screenplay

initialisation	OOScreenplay(char* filmname):
destruction	˜OOScreenplay;
get number scenes/cuts	(int) q_number_scenes()
<various>	conversion functions between frame numbers, scene ordinals and scene IDs
number of objects in scene	(int) q_number_objects(sceneID);
list objects in scene	(object_list) q_objects(sceneID);
list scenes containing object	(scene_list) q_scenes(objectID);

Object-Oriented Scenes

t.b.d

Object-Oriented MOVie objects

// initialisation	setup:(char*)applicname:(char*)videoname:(int)objid;
// access routines	
get name of application	(char*) get_application_name;
get name of current video	(char*) get_video_name;
get objects application name	(char*) get_application_dependent_name;
get objects video name	(char*) get_video_dependent_name;
get objects movie ID	(int) get_object_id;
get objects frame size	(NXSize) get_original_size;
get objects callback	(int/char*) get_call_back;
// query routines	
// object visible in frame?	(BOOL) q_object_visible:(int)frame;
// coords inside object?	(int) q_object_pointed_to:(int)frame:(NXPoint)p;
// give coords for object	(NXRect) q_object_extent:(int)frame;
// give map records for object	q_object_records:(int)frame:(MapPoint*)before:(MapPoint*)after;