AVI Files With Hotspots

David A. Feinleib

Revision 1: August 4, 1993

Introduction

The AVI Hotspot Editor (AVIHED.EXE) and its accompanying DLL's (AVIHVWR.DLL, AVIHAPP.DLL) provide you with the ability to specify hotspots for AVI files, much as you can using the Segmented Hotspot Editor (SHED.EXE) for DIB files. The AVI hotspot kit can be used with Microsoft Multimedia Viewer or with a standalone application by using the correct DLL.

The Hotspot Editor allows you to draw hotspots easily on your AVI file, save them in a hotspot information file, which you specify when calling the hspPlayAVI function in the AVI hotspot DLL.

The AVI Hotspot kit allows you to:

- Specify Begin and End frames for each hotspot, so that, for example, two hotspots can cover an overlapping space as long as their Begin and End frames do not overlap;

- Execute any Viewer command or send a message to your calling standalone application when a hotspot is selected; additionally, you can continue or terminate playing when a hotspot is selected, or jump to another location in the same AVI file.

The AVI Hotspot Editor

You use the AVI Hotspot Editor to create and edit the hotspots for an AVI file. Hotspots can cover overlapping areas in an AVI file as long as their Begin and End frames do not overlap. For each hotspot, you specify a command string, a hotspot ID, and optionally a beginning and ending frame.

Using The Hotspot Editor

After loading the Editor, select File | Open to open an AVI file. You will then be prompted for a Hotspot (INI) file. If you are creating hotspots for the first time for the specified AVI file, select cancel since you do not need to open an INI file.

Once the AVI file is loaded, you can start drawing hotspots on your AVI file. To do so, click the left mouse button, and while holding it down, draw the hotspot that you want. To specify hotspot attributes, double click inside the rectangle you have created. To adjust the position of a hotspot, put in the mouse cursor inside its rectangle, click the mouse button and move the rectangle while holding the button down. To change the size, click and drag one of the hotspot's edges. You can delete a hotspot by clicking inside its rectangle and pressing the delete key or selecting *Delete Specified Hotspot* from the *Hotspots* menu.

When a hotspot is selected, its rectangle coordinates will appear in the *Selected Hotspot Info* window.

Hotspot Attributes

In the Hotspot Attributes dialog box, you can specify the following information:

- A Command String, which is the command that is executed in Viewer when the hotspot is selected (for example, you might specify sndPlaySound(`hello.wav',1));
- A Hotspot ID--The ID of the specified hotspot (for example, Hotspot 1)
- The hotspot's bounding box, i.e. its rectangle coordinates;
- The hotspot's active frames (so that more than one hotspot can be specified for overlapping rectangle areas during different frames)

- Hotspot selection options, which change the action of the AVI file when the hotspot is selected, to either continue playing, terminate playing, or jump to a specified frame in the same AVI file.

Saving Hotspot Files
Once you have drawn hotspots for the specified AVI file, make sure you have specified a hotspot ID, and, if applicable, a command string for each hotspot. Then choose Save from the File menu and enter a filename. If you have not specified an ID for each hotspot, you will receive an error message.

Other Features
Because the kit allows you to specify different active frames for each hotspot, you can choose to see only the hotspots that are active for the frame that you are currently displaying or all hotspots in the AVI file, independent of what frame is displayed in the Editor. To do this, choose the *Only Show Hotspots In Current Frame* option from the *Hotspots* menu.

The Hotspot Information File Format
The AVI Hotspot Editor writes the hotspot information file using WritePrivateProfileString, so all entries in the file have the regular ini file format. The ini file must contain a [Configuration] section, a [Hotspots] section, and at least one information for one hotspot. Each hotspot is saved under its hotspot ID.

The [Configuration] section has the format
[Configuration]
Version=1.00
Editor=AVIHED

The current version number is 1.00. The Editor will normally be AVIHED. The Hotspots section is in the form HotspotID=1. Hotspot ID's can be anything as long as =1 is specified, e.g.
[Hotspots]
Ears=1
Eyes=1
Door=1
Window=1

For each entry in the Hotspots section, there must be a corresponding hotspot entry, for example,

[Door]
Rect=247,17,281,71
Command=sndPlaySound(`n.wav', 1)
BeginFrame=0
EndFrame=101
OnClick=1018
ToFrame=0

Rect -- the position of the hotspot in the form Left,Top,Right,Bottom.
Command -- the command to be executed when the hotspot is selected.
BeginFrame -- the beginning frame for the hotspot to be active.
EndFrame -- the last frame for the hotspot to be active.
OnClick -- can be any of three values: ID_CONTINUE, ID_STOP, or ID_JUMP which are defined in the resource.h file in the ..\avihed directory.
ToFrame -- the frame to jump to if ID_JUMP is specified for OnClick.

Programming Issues

Drawing on an AVI frame
The most fundamental difficulty I encountered in trying to implement hotspots for AVI files was in displaying the hotspots on top of the AVI file. Because the AVI API does not support drawing on top of an AVI window directly, there were two options: create a transparent window and keep it over the AVI

window or subclass the AVI window. Positioning the window and redrawing correctly turned out to be impractical for the transparent window, since if the transparent window's window procedure draws the rectangle on the WM_PAINT call, the rectangles will be drawn underneath the AVI frame (since the AVI frame doesn't finish painting by the time the invisible window receives its WM_PAINT message). A much better solution was to subclass the AVI "movie" window, the window in which the AVI file is played. The application that calls the DLL (your application or a Viewer application) specifies the handle of the window in which to play, so that the DLL does not have the responsibility of creating a window. It merely needs to subclass the given window and watch for WM_PAINT and WM_DESTROY messages. Because the hotspot editor needs to draw a rectangle for each hotspot, merely intercepting WM_PAINT was not enough; it had to paint the frame and then draw the rectangles. Allowing the AVI interface to do its own painting will result in the rectangles being drawn before drawing of the AVI frame is completed, with the outcome that the rectangles will not be visible on the screen. The solution was to paint the frame directly, by calling mciSendCommand with the MCI_UPDATE and MCI_WAIT flags and then *not* call the old window procedure. This way, when the editor receives a WM_PAINT message, it paints the frame itself; when painting of the frame is completed (i.e. when the mciSendCommand returns), the editor paints the rectangles, so that they appear correctly.

Unfortunately, there is still no good way to keep drawn objects on top of an AVI file while it is playing. But in any case, doing so would slow down display of the AVI file and cause flicker.

Suggested Improvements

- Undo ability;
- Printing of current hotspot information/AVI frame;
- Visual selection of hotspots;
- MDI support for dealing with multiple AVI files at the same time.

The AVI Hotspot DLL's

There are two AVI hotspot DLL's: AVIHVWR.DLL, which can be used with Viewer applications by specifying it in the Viewer application's MVP project file, and AVIHAPP.DLL, which you can link with your stand-alone application. Visual Basic support has not yet been implemented, but suggestions on how to do so are described in the section *Suggested Improvements*.

Programming Issues

Multiple Viewer/application instances with the same DLL

Because there can be only one instance of each DLL but multiple applications or instances of an application calling each DLL, the hotspot DLL associates a structure with each application (instance) that calls it. The structure it uses is called a MOVIEINFO structure, defined in *hotspot.h*. There are two ways to associate information with a window: either write your own functions to match up a global handle to data with a window handle and have a list of these associations or use the SetProp/GetProp/RemoveProp functions, which allow you to specify 16 bit values (such as handles to globally allocated memory) for a window. The DLL's exported function, *hspPlayAVI* calls SetProp; the subclass procedure calls GetProp every time it is called so that it can obtain information about the movie, such as the address of the old procedure to call. Finally, when the subclass procedure receives a WM_DESTROY message, it calls RemoveProp to remove the movie information from the window.

Visual Basic support

Although the DLL can be used with Visual Basic already, unfortunately, it has no way to send information back to a VB application when a hotspot is selected. In order either to send a message to the VB application using SendMessage or make use of a callback function from the DLL to the application, as it does with non-VB stand-alone applications, either basic VBX functionality must be added to the DLL or a separate VBX must be written that can export callbacks from a VB app or intercept new messages (ones that are not already implemented in VB) by subclassing the form on which it is placed.

Hotspots
The hotspots for an AVI file are read in from an ini file and stored in a doubly-linked list; each time the user clicks in the movie window (causing a WM_LBUTTONDOWN message to be sent), the subclass procedure calls a function that, for each hotspot in the list, calls the PtInRect function to determine in which hotspot rectangle, if any, the mouse cursor was positioned when the user pressed the mouse button. While going through the hotspot list, before determining whether the point was in a hotspot rectangle, the function checks to see if the hotspot is valid for the currently displayed frame, i.e., if the current frame is between the beginning and ending frames for the hotspot.

For the Viewer DLL (avihvwr.dll), the function calls *VwrCommand* as follows:

```
VwrCommand (VwrFromHinst (GetWindowWord (hwnd, GWW_HINSTANCE) ),
        NULL, pHotspot->pszCommand, cmdoptNONE)
```

Because Viewer requires a VWR structure to be specified as the first parameter to VwrCommand, the function obtains the instance handle by using GetWindowWord; the hwnd parameter is obtained from the following calls:

```
hwnd = GetParent(pMovieInfo->hwndParent);
if (!hwnd) hwnd = pMovieInfo->hwndParent;
```

where pMovieInfo is a pointer to a MOVIEINFO structure passed to the function from the subclass procedure.

For the stand-alone application DLL (AVIHAPP.DLL), the DLL executes the callback function passed as the last parameter to the hspPlayAVI DLL function. The callback function is defined in the application by exporting a function in the DEF file and then calling MakeProcInstance and passing its return value to the hspPlayAVI function. When the application receives the WM_DESTROY message, it should call FreeProcInstance. Using SendMessage with a WM_USER+xxxx message or (as a hack) WM_COMMAND with a certain value for wParam is another possibility, but a callback appears to be the most straightforward and efficient solution.

Using the DLL's
There are separate DLL's for Viewer (AVIHVWR.DLL) and for stand-alone applications (AVIHAPP.DLL).

With Viewer
To use the hotspot DLL with Viewer, specify the following command in the [CONFIG] section of the your Viewer application's MVP project file:

```
RegisterRoutine("AVIHVWR","hspPlayAVI","I=USS")
```

In your Viewer .RTF file, specify some text or bitmap and then, in hidden text specify ! hspPlayAVI(hwndContext,`AVIFILE.AVI',`HSPFILE.INI')
where AVIFILE.AVI is the name of the AVI file to play and HSPFILE.INI is the name of the INI file containing hotspot information for the AVI file. hwndContext is a Viewer defined variable; see the Viewer documentation for further details.

With stand-alone applications
To use the hotspot DLL with Viewer, add AVIHAPP.LIB to your link statement. Include *avihapp.h*, which defines the prototype for the hspPlayAVI function. In order to call the hspPlayAVI function, you will need to export a callback function that the DLL will call to notify you that a hotspot has been selected. To play an AVI file with hotspots, call the hspPlayAVI function with the name of the AVI file as the first parameter, the name of the ini file containing hotspot information as the second parameter, and the address returned from MakeProcAddress called with the name of your exported callback function. When your application's window receives a WM_DESTROY message, it should call FreeProcAddress.

The callback function must have the following syntax:

```
BOOL __export hspAVICallback (HWND hwndParent, HWND hwndMovie, WORD wMessage,
                                                    LONG FAR * lParam1, LONG FAR
```

*lParam2)

The callback should handle the WM_SIZE and and WM_LBUTTONDOWN messages. (Note that these are not used as messages but as command identifiers, defining what action should be taken on the given parameters, if any). If it processes a message successfully, it should return TRUE; if it does not, it should return FALSE.

If wParam is WM_SIZE: The callback is being given an opportunity to adjust the size of the window in which the movie will be played before the movie is displayed. lParam1 is a far pointer to a RECT structure that gives the dimensions of the movie.

If wParam is WM_LBUTTONDOWN: The user has selected a hotspot. lParam1 is a LPSTR containing the Hotspot ID; lParam2 is a LPSTR containing the Command for the hotspot.

The AVIHRUN sample (stand-alone)
The AVIHRUN sample program demonstrates how to call hspPlayAVI, how to implement a callback function, how to process the messages in the callback function; in addition, it implements a simple command language, which supports the following two commands:

sndPlaySound
ExecProgram (or WinExec)

Both commands have the same syntax as their Viewer counterparts.

Suggested Improvements
The Hotspot Editor and DLL's provide the framework and basic, but usable functionality for an AVI hotspot interface, in Viewer and stand-alone applications.

- Visual Basic support by means of a VBX would be an important improvement.

- Multiple command support, so that, for example, you could play a *wav* file and execute a command.

- Better device independence; support for specifying coordinates.

- Extending the command language of AVIHRUN into a full-fledged run-time application that can be distributed with AVI and hotspot files would add tremendous functionality. Some sort of extensible module interface, beyond merely calling WinExec, i.e. support for external functions in DLL's would be valuable.

- Using RIFF to combine hotspot information with AVI files, and perhaps commands implemented by the AVIHRUN application would be an excellent extension to the AVI interface.