THE ANIMATED CURSOR EDITOR

INTRODUCTION

This chapter explains the operation of Resorcerer's Animated Cursor Editor, which lets you create and edit 'acur' resources. These resources do not themselves contain actual cursors; they are only lists of references to 'CURS' or 'crsr' resources.

The Animated Cursor Editor lets you try out animated cursors, and the Cursor Editor has a feature to let you see adjacent frames of an animation sequence while you are editing one frame. For more on this, see the "Cursor Editor" chapter.

You should also be familiar with general resource editing, as explained in the "Editing Resources" chapter earlier in the manual.

TOPICS COVERED

- Creating a new animated cursor
- Opening an animated cursor
- Using the editor
- Cutting and pasting cursor frames
- Decompiling the frame list
- Programming considerations
- Closing the animated cursor

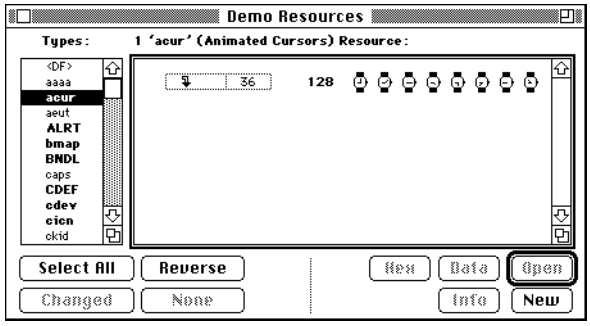
CREATING A NEW ANIMATED CURSOR

If you are viewing resources in Resorcerer's File Window, click on the **New** button or choose **New Resource** from the **File** menu. If the Types List is the Active List, Resorcerer will ask you to specify the resource ID prior to creating the resource; otherwise, it will use the next free resource ID available for 'acur' resources. The resource ID at which the search begins is specified in the **Resource ID Preferences** section of the **Preferences** dialog (for more on this, see the "Preferences" chapter later in this manual). Usually, the starting ID is 128.

If you are already viewing an open 'acur' resource, you can choose **New Resource** from the **Resource** menu. The next free ID is assigned automatically.

Once the new, empty list of cursor references has been created, the Editor opens it for editing so that you can add frames to it.

OPENING AN ANIMATED CURSOR

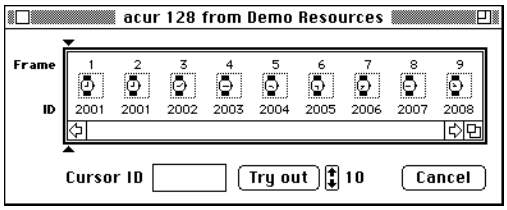


The File Window tries to display all the cursors whose resource IDs are kept in the 'acur'. If the first resource ID belongs to a color cursor ('crsr'), Resorcerer assumes that all the rest of the cursors are color ones, and tries to display them in sequence. If there is no color cursor with the first frame ID, it looks for black & white 'CURS' resources to display them. The Cursor Editor lets you explicitly say which cursor resource type you want it to resolve and display. Any cursor resources that are not in the same file get drawn as a gray square.

To open the animated cursor, select the 'acur' resource you want to open in the Resources List of your File Window and click on the **Open** button. Or double-click on the resource entry directly. If you need to get at some of the hidden placeholder fields in the resource, open it with the **Data** button to get at them with a template and the Data Editor.

USING THE EDITOR

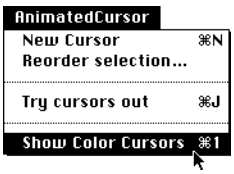
An 'acur' resource is essentially just a list of resource IDs. The Editor shows the resource ID and the cursor with that ID, all in a horizontal list. If there is no cursor then the list draws a gray square in its place.



Each cursor is labeled with its index, or frame number, from 1 up to the number of frames in the list.

Animated cursors can be either black & white or color, depending on how the code in the owning application is interpreting the 'acur' resource. However, there is no place in the 'acur' data to specify whether the resource IDs are for color ('crsr') or black & white ('CURS') cursor resources. When the Editor opens the animated cursor, for each resource ID in the list it scans the file the 'acur' is in, looking for either a 'crsr' or 'CURS' with that ID. When it finds a match, it assumes that all the rest of the frame resource IDs refer to resources of the same type as the match.

You can use the **Show Color Cursors** command in the **AnimatedCursor** menu to tell the Editor to look for and display 'crsr' resources, instead of 'CURS' resources. This should only be necessary if your file contains both color and black&white animated cursors, and you are using the same resource ID range for the cursors in both animations.



The **Try Out** button is the same as the menu's **Try Out** command. It builds an animated cursor, either color or black&white depending on the **Show Color Cursors** setting, using the frames in the list with resource IDs that refer to actual cursors. It then attaches the series of cursors to the mouse and animates them cyclically. The small up and down arrow click control next to the **Try Out** button lets you change the wait time between frame changes. The number to the control's right is the number of ticks (60th's of a second) to wait. Lower numbers mean faster animation.

Note: There may be a slight jerkiness to the animation, due to the Editor periodically letting the System have some time.

OPERATING THE FRAME LIST

You can grow the list and the dialog that holds it to any size when you click and drag the list's grow box in the lower right corner of the list. The Zoom Box grows the list to the widest it can be on its current screen.

In the Editor window, either the cursor ID list or the Cursor ID text box is the active item. When the list is the active item, it is framed with a bold border. The active item is the one to which the Editor directs all editing commands.

On the top and bottom of the list are two triangular handles. These indicate the position of the list insertion caret. If no cursor IDs are selected, the caret will also display a blinking vertical line. To move the caret, click on either handle and drag it to the position you want. If you drag it to the right or left of the list edges, the list entries scroll automatically.

To select a single cursor in the animation, click once on it. Its cursor resource ID is installed in the text box for you to edit.

Sorcery: ⌘N is the keyboard equivalent of **New Cursor**.
Double-clicking on a list insertion caret handle is also equivalent to clicking on the **New Cursor** command.
⌘A is the keyboard equivalent of **Select All** in the **Edit** menu's **Select** sub-menu.
Shift-clicking extends the current selection of cursors.
⌘-clicking on a cursor ID toggles its selection status.
The Right or Left Arrow cursor key collapses the current selection and selects the first frame to the right or left of the former selection, respectively.

To add a new frame to the animated cursor, place the list insertion caret at the position in the list where you want the new resource ID inserted, and choose **New Cursor** from the **AnimatedCursor** menu. The resource ID the Editor assigns to the new frame is the first free ID in the list. This does not create the referenced cursor resource, so the list will either display a gray box, or show you any cursor that already exists with that ID.

To edit a referenced cursor with the Cursor Editor, or to create a referenced cursor if the list is showing it as a gray box, double-click on the list entry. The state of the **Show Color Cursors** command determines whether any new cursor will be a 'CURS' or 'crsr' resource.

CUTTING AND PASTING CURSOR FRAMES

When the frame list is the active editing item, you can **Cut**, **Copy**, **Clear**, and **Paste** any selection of cursor frames. When copying or cutting all selected cursor IDs, they are assembled into the same format as an 'acur' resource and placed in the clipboard as a piece of scrap with type 'acur'. When pasting cursor frames, the Editor always places them at the position of the list insertion caret.

Note: If you are transferring frames between two 'acur' resources in two different open files, remember that you are only transferring resource ID references, not the cursors themselves.

To delete a set of cursor IDs, select them and tap the Delete key.

To quickly reorder a contiguous selection of frame resource IDs in the animated cursor, use **Reorder…** in the **AnimatedCursor** menu. This performs Resorcerer's standard list reordering algorithm, which invalidates each selected frame's index by marking it with "??", and then waits for you to click on the list items in the order in which you want them. The Editor does all the cutting and pasting for you.

Note: An invalid index is only a reminder to you that you haven't yet repositioned the frame resource ID in your new ordering. If you close the resource while some frames have invalid indices, the Editor saves the patterns in display order.

Should you want to stop reordering prior to clicking on all the cursor IDs with invalid indices, choose **Reorder…** again. All cursor IDs with invalid indices are reassigned indices corresponding to their current list position.

DECOMPILING THE FRAME LIST

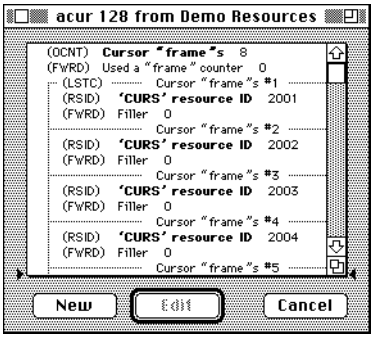
The AnimatedCursor Editor will decompile the list of frame resource IDs into a Rez language declaration. The text is placed directly in the clipboard so that you can switch to your development system's text editor and paste the declaration into your Rez file.

To do this, first ensure that no cursor is currently selected in the list. Then choose **Copy** from the **Edit** menu to decompile the entire list to the clipboard.

PROGRAMMING CONSIDERATIONS

Animated cursors are an important user-interface technique. They give the user a simple and unobtrusive visual cue that your application is still working during lengthy calculations that would otherwise involve no user-interface activity. Without an animated cursor, your user cannot tell whether their computer is doing real work, or whether it is hung indefinitely. User-interface testing has shown that without any visual feedback or other response from the computer, many users will assume something is wrong after very short periods of time (such as 10 seconds!), and proceed to do nasty things like rebooting the computer.

In spite of their importance, there is no toolbox support for animated cursors. You usually have to write your own code to process the 'acur' resource, and to do the actual animation. The following is an overview of what you need to do in case you are unfamiliar.



First, take a look at the actual structure of the 'acur' resource by opening one with a template, which you do by using the **Data** button in the File Window instead of the **Open** button. You will notice that for each frame there is an extra two-byte placeholder field next to each resource ID field, as well as an empty field (labeled "Used a frame number") just after the frame count.

To initialize the 'acur' resource, read it into memory, and ensure that its handle isn't purgeable. Scan the list for each resource ID and use it to retrieve the handle to the appropriate cursor (use `GetCursor()` for black & white 'CURS' resources, or `GetCCursor()` for color 'crsr' resources). Then replace the resource ID field, along with its adjacent placeholder field, with the 4-byte handle to your cursor and continue to the end of the list.

You need to write a routine that gets called regularly during any lengthy calculation. Each time it is called, it should increment the current frame index, wrapping it back to the beginning if it gets larger than the frame count, and use the index to retrieve the handle in the 'acur' list of cursor handles. You may want to include a governer that doesn't allow the current frame to change until a minimum amount of time has passed, which helps keep the animation looking the same regardless of the machine's processor speed. The following is a sample routine that illustrates how to do this:

```
/*
 * Call this with FALSE to put up a simple non-animated watch, or with TRUE to
 * either start or continue the animated cursor, as specified by 'acur' 128.
 * This code continues to work if the 'acur' resource is missing or empty, in
 * which case it defaults to displaying the static system watch cursor.
 */

typedef struct {
  short cursID;
  short filler;
  } CursorEntry;              /* 4 bytes */

typedef struct {
  CursHandle cursorHndl;
  } CursorEntryHandle;        /* 4 bytes */

void PleaseWait(int animate)
{
  static CursHandle watchHandle;
  static Handle acur;
  static Boolean once = TRUE;
  static long then;
  static short nCursors,cursIndex;
  short i,*word;
  CursorEntry *idTable;
  CursorEntryHandle *handleTable;

  /* One time initialization: swap cursor IDs for CursHandles in list */

  if (once) {
    acur = Get1Resource('acur',128);
    if (acur!=NIL && ResError()==noErr) {
        LoadResource(acur);
        HNoPurge(acur);

        word = (short *)(*acur);      /* Scan the resource header */
        nCursors = *word++
        cursIndex = *word++;          /* Usually 0 */
        if (cursIndex < 0) cursIndex = 0;
        idTable = (CursorEntry *)word;
        handleTable = (CursorEntryHandle *)idTable;

        for (i=0; i<nCursors; i++) {
            /* Use temporary so we don't have to lock acur/handleTable down */
            CursHandle tmp = GetCursor(idTable[i].cursID);  /* Moves memory */
            handleTable[i].cursorHndl = tmp;
            HNoPurge((Handle)tmp);
            }
        }
     else
        nCursors = 0;         /* Nothing to animate */

    watchHandle = GetCursor(watchCursor);    /* From system resources */
    then = TickCount();
    once = FALSE;
    }
```

```
  if (animate && nCursors>0) {
     long now = TickCount();
     if ((now - then) > 3) {
        /* Been a while, so install next frame in cycle */
        CursHandle cursH;
        handleTable = (CursorEntryHandle *) ((*acur) + 2*sizeof(short));
        if (cursIndex >= nCursors) cursIndex = 0;
        cursH = handleTable[cursIndex++].cursorHndl;
        if (cursH!=NIL && *cursH!=NIL)
            SetCursor(*cursH);
        then = now;
        }
     }
   else
    SetCursor(*watchHandle);
}
```

This type of subroutine works for both black & white cursors and color cursors. For color cursors you have to use color analogues of `GetCursor()` and `SetCursor()`: `GetCCursor()` and `SetCCursor()`.

Note: Make sure you test for the presence of Color QuickDraw before using any of the color cursor routines. Also, the technique of using a VBL task to install an animated cursor is not recommended for color cursors, because `SetCCursor()` moves memory, which is disallowed during VBL interrupts. For more on VBL-animated cursors, see Apple's sample code.

CLOSING THE ANIMATED CURSOR

When you have finished editing an animated cursor frame list, click in the editing dialog's GoAway box to close it and save the changes you have made to the resource. If you have made any changes, and if your **Confirm resource saves** preference is set (for more on this, see the "Preferences" chapter later in the manual), Resorcerer will ask you to confirm saving the changes.

The changes you save when closing the resource will be saved to disk when you save or close the open file later on.