2-19-2000

TrackMatrix Behavior 1.0 Event id's: 200030 - 200036

By Matthew Peterson

matthew@pinoko.berkeley.edu

Introduction

TrackMatrix is a set of functions for LiveStage Pro. They allow wired movies to set the position of tracks within a movie. Previous to this behavior, setting the position of a track was very difficult for the following reasons:

1) LiveStage doesn't provide position properties for tracks. Thus there isn't a way to determine the location of a track.
2) LiveStage doesn't have a moveTo( ) function for tracks
3) The MoveMatrixBy( ) function will only accept positive constants. That means you must know how much you are going to move the track by before the movie is compiled. And good luck if you want to move the track by a negative amount!
4) There are no properties to determine the size of a movie.
5) There are no properties to determine the number of tracks in a movie.
6) There is no property to determine the index of a track.

TrackMatrix provides solutions for these issues through the following 5 custom events:

1) **MoveTrackBy** -- Move an arbitrary track by an arbitrary X,Y amount

2) **MoveTrackTo** -- Move an arbitrary track to an arbitrary X,Y location

3) **TrackLocation** -- Determine the current position of a track

4) **TrackStatus** -- returns information about the size of the movie, the number of tracks, the index of the current track, also information about the layering and visibility of tracks.
5) **IsTrackVisible** -- Determine if a track is a spatial track and is enabled with positive dimensions (what I call 'visible').

Purpose

Moving tracks can be very useful for the following reasons:

1) The location of tracks can be saved with a movie, but the location of sprites can not. Moving tracks allows users of your wired movie to customize it. They can move things around, save the movie, and the next time they open the movie, things will still in place. Imagine a railroad movie, where the user is able to layout a train track, and then have a train follow the tracks around the movie. If I spend a long time making a cool track layout, it is going to frustrate me if I can't save it and share it with my friends, maybe even put it on my web site.

(See BuildFace.mov for an example of this type of movie)

The locations of tracks don't have to directly correspond to locations of objects but can be place holders for non-visual information. For instance, I can have a QuickTime alarm clock, who's alarm is savable by mapping the hour to the x position of a blank track, and the minutes to a y position. I can also map the sound setting to the layer of the track. Doing it this way means the alarm settings can be saved when the user chooses "save" from the QuickTime Player Pro File menu.

2) Moving tracks are important in template movies. Template movies are movies that are wired up and layed out in such a way that it is easy for graphic artists QuickTime Pro to swap images and tracks in and out to make a new movie with the same functionality, but new content. But video and text tracks aren't always the same size each time. Thus it is important to have some way to align and position tracks. Currently the QT Player offers only a tedius and not-so-precise way of moving tracks. One would thing that Applescript would come in handy here, but surprisingly QT 4.1 doesn't have a way to move or get the position of tracks through Applescript. the TrackMatrix behavior offers an elegant solution for this: Let the wired template movie move the tracks itself.

Movie widgets can also be made. i.e. movies that move the tracks around in another movie.

(See TracksAlignBottom.mov for an example movie widget.)

Quick Reference

• **MoveTrackBy** : Before calling this event, you must set three Global Variables:

[input variables]

1) TrackIndex: The index of the track to move

2) MoveTrackX: The amount to move in the x direction

3) MoveTrackY: The amount to move in the y direction

Example (Move the second track by (18, -10):

```
GlobalVars TrackIndex MoveTrackX MoveTrackY
```

```
TrackIndex = 2
MoveTrackX = 18
MoveTrackY = - 10
spriteofid(1).executeevent($MoveTrackBy)
//Assuming this behavior is in spriteofid(1)
```

Note: If moving a track past the border of a movie has different results depending on where the movie is playing. If it is playing in LiveStage Pro, then the movie might scroll to keep the moved track in the movie. If it is in the player, then the size of the movie will change to fit all tracks. If in embedded in the browser, the movie size will stay the same, and a the outside portion of the track will be cropped.

- **MoveTrackTo** : Before calling this event, you must set three Global Variables:
  [input variables]
  1) TrackIndex: The index of the track to move
  2) MoveTrackX: The x position
  3) MoveTrackY: The y position

  Example (Move the third track to location (100, 100):

```
GlobalVars TrackIndex MoveTrackX MoveTrackY

TrackIndex = 3
MoveTrackX = 100
MoveTrackY = 100
spriteofid(1).executeevent($MoveTrackTo)
//Assuming this behavior is in spriteofid(1)
```

  Note: The location of the track is relative to the sprite track containing the behavior. Thus it doesn't make sense to move this sprite track as it will alway stay in the 0,0 location.

- **TrackLocation**: Before calling this event, you must specify the index of the spite you want to locate:
  [input variables]
  1) TrackIndex: The index of the track

  [output variables]
  1) MoveTrackX: The x position of the track
  2) MoveTrackY: The y position of the track

  Example (Move a sprite to the location of the second track)

```
GlobalVars TrackIndex MoveTrackX MoveTrackY

TrackIndex = 2
```

```
spriteofid(1).executeevent($TrackLocation)
//Assuming this behavior is in spriteofid(1)
spriteofid(2).moveto(MoveTrackX, MoveTrackY)
```

Note: The position of the track containing the TrackMatrix behavior is 0,0 by definition. It is the reference track.

- **TrackStatus**: Call this event and it will update 9 global variables:
  [output variables]
  1) NumTracks: Total number of tracks in the movie
  2) NumVisibleTracks: Number of visible tracks in the movie
     visible = spatial track, enabled and positive dimensions.
     Note: a track that meets the above specifications, but completely behind another track is still considered visible, a sprite track that has the visible setting deselected is considered visible because I have no way of determining the status of this parameter.
  3) MovieTop: The y position of the top of the movie relative to the reference sprite track (the one containing this behavior).
     Note: The negative of this parameter (i.e. -MovieTop) can be considered the y position of the reference sprite track relative to the movie.
  4) MovieBottom: The y position of the bottom of the movie relative to the reference sprite track.

  5) MovieLeft: The x position of the left of the movie relative to the reference sprite track.
     Note: The negave of this parameter (i.e. -MovieLeft) can be considered the x position of the reference sprite track relative to the movie.
  6) MovieRight: The x position of the right of the movie relative to the reference sprite track.
  7) MovieFront: The layer of the frontmost track
  8) MovieBack: The layer of the Backmost track
  9) ReferenceIndex: The index of the reverence track (the one containing this behavior).

  Example (find the width of the movie)

```
GlobalVars MovieLeft MovieRight
LocalVars moviewidth

spriteofid(1).executeevent($TrackStatus)
//Assuming this behavior is in spriteofid(1)
moviewidth = MovieRight - MovieLeft
```

Note: Since LiveStage doesn't have any way of knowing the id of a track, the track's index is the best way to numerically reference a track (i.e. TrackofIndex(x)). But if the movie gets added to another movie, the indexing will be changed. One way to account for this change is reference the index of tracks relative to the current tracks, and to call the TrackStatus event, to get the current index of the reference track (ReferenceIndex).

• **IsTrackVisible**: See note on NumVisibleTracks in **TrackStatus**

[Input variables]
1) TrackIndex: The index of the track

[Output variables]
1) TrackVisible: Has three possible values
1 means the track exists and is visible
*(See note on NumVisibleTracks in **TrackStatus**)*
0 means the track exists and is not visible
-1 means the track does NOT exist.

Example (determine if the third track exists or not)

```
GlobalVars TrackIndex TrackVisible

TrackIndex = 3
spriteofid(1).executeevent($TrackStatus)
//Assuming this behavior is in spriteofid(1)
if(TrackVisible > -1)
	//The track exists. Go ahead and manipulate it.
else
	//The track doesn't exist, better not try to do anything
	//to this track, or the movie might crash!
endif
```

Reserved Variable Names for this Behavior:

Input/Output globalvars: TrackIndex, MoveTrackX, MoveTrackY, NumTracks, NumVisibleTracks, MovieTop, MovieBottom, MovieLeft, MovieRight, MovieFront, MovieBack, ReferenceIndex, TrackVisible

Note: Internal variables all start with "MP_" and thus should not overlap with other variables.

Technical Notes:

I will briefly explain the general principles behind the functions in this behavior. For a more in depth understanding, look at the comments within the behavior itself.

To move a track by an arbitrary amount, I loop disable the track and loop through a sequence of preset MoveMatrixBy( ) calls, I then restore the tracks original enabled status. I make sure the track is disabled before moving it because unlike a sprite, an update to an enabled track causes a screen update.

To move a track by a negative amount. I am rotating the track 180 degrees around one point, and the rotating 180 around another point. In theory this can place the track at any location. Because this is slightly slower than the MoveMatrixBy( ) action, I first over move by a negative amount, and then correct with a positive move

To find the location of a track. I am comparing where two tracks think the mouse is located. But because this mouse location gets rotated and scaled with a rotated and scaled track, this method is only accurate when using tracks that aren't scaled or rotated. There is no easy method to determine if a track is scaled or rotated. Also the TrackWidth and TrackHeight attributes do not reflect matrix transformations. So a 200 by 200 track will always have a TrackWidth of 200 even if it is scaled to some large size on the screen.

Revision History:

2-19-2000 Version 1.0 first written.