2-19-2000

Lines Behavior 1.0

Event id’s: 200050 - 200052

By Matthew Peterson

matthew@pinoko.berkeley.edu

Introduction

Lines Behavior consists of two simple functions for drawing lines in a wired movie. To use these functions, apply this behavior to a single sprite in your sprite track. It doesn’t matter which sprite it is. Set the global variables delineated below, and execute the event.

The lines are drawn by stretching sprites into long rectangular shapes. Before drawing a line, you must specify which sprite will be used as the line by supplying the ID. Once can use the LiveStage MakeNewSprite( ) action to make new sprites during run time. You must call MakeNewSprite( ) yourself, this Lines behavior doesn’t call the MakeNewSprite( ) function for you.

Here are the Lines Behavior custom events See Quick Reference below for a description of how to use these functions:

1) **DrawLine** -- Supply the starting and ending point, then execute this event

2) **DrawLinePolar** -- Supply the starting point, the angle and the length.

Purpose

The ability to draw dynamic lines are useful for many purposes. They can be used to draw graphs, connect circuit diagrams, multiple choice tests, and in flow and connectivity diagrams. They can also be used in simple drawings, or for underlining, crossing-out and pointing to objects.

Quick Reference

• **DrawLine**: Draws a line in Cartesian coordinates.

[input variables]
1) LineSpriteID: The ID of the sprite being morphed into a line.
2) LineThickness: The thickness of the line in pixels.

3) LineImageIndex: Effectly this is the color of the line. Set this to the image index of some flat colored graphic. Several of these graphic images are supplied with this behavior.
4) DoThicknessOffset: When set to true, the line is slightly extended at both ends. This is the typical behavior for most drawing programs. It makes multiple lines starting at the same place have a slight overlap.
5) LineStartx: x position of the start of the line.
6) LineStarty: y position of the start of the line.
7) LineEndx: x position of the end of the line.
8) LineEndy: y position of the end of the line.

Example: Make spriteofid(2) into a line of thickness 3.
Draw the line from (7,5) to (25,10).

```
GlobalVars  LineThickness LineSpriteID LineImageIndex DoThicknessOffset lineStartx
lineStarty LineEndx LineEndy

LineSpriteID = 2
LineThickness = 3
LineImageIndex = 5 //set the image to index 5, some colored square image.
DoThicknessOffset = TRUE //This adds a slight lip to the ends of the line. It helps in
// appending multiple lines. TRUE in most drawing programs.
lineStartx = 7
lineStarty = 5
lineEndx = 25
lineEndy = 10

SpriteOfID(1).ExecuteEvent($DrawLine) //Assuming this behavior is in
spriteofid(1).
```

- **DrawLinePolar**: Draws a line in polar coordinates.
  [input variables]
  1) LineSpriteID: The ID of the sprite being morphed into a line.
  2) LineThickness: The thickness of the line in pixels.
  3) LineImageIndex: Effectively this is the color of the line. Set this to the image index of some flat colored graphic. Several of these graphic images are supplied with this behavior.
  4) DoThicknessOffset: When set to true, the line is slightly extended at both ends. This is the typical behavior for most drawing programs. It makes multiple lines starting at the same place have a slight overlap.
  5) LineStartx: x position of the start of the line.
  6) LineStarty: y position of the start of the line.
  7) LineAngle: The angle of the line. 90 degrees points down. 0 degrees points to the right.
  8) LineLength: The length of the line in pixels.

Example: Make a line of length 20, and thickness 1 pointing up and to the right (-45 degree angle). Have line start at (50,50)

```
GlobalVars LineThickness LineSpriteID LineImageIndex DoThicknessOffset lineStartx
lineStarty LineAngle LineLength

LineSpriteID = 2  //the id of the sprite that will form our line. This sprite must exist.
LineThickness = 1
LineImageIndex = 1 //set the image to index 1, some colored square image.
DoThicknessOffset = TRUE //This adds a slight lip to the ends of the line. It helps in
// appending multiple lines. TRUE in most drawing programs.
lineStartx = 50
lineStarty = 50
lineAngle = -45
linelength = 20

SpriteOfID(1).ExecuteEvent($DrawLinePolar) //Assuming this behavior is in
spriteofid(1).
```

NOTE: DrawLine will update the global variables LineAngle and LineLength to match the properties of the newly drawn line. But DrawLinePolar will not update LineEndx and LineEndy. Having LineAngle and LineLength update is useful in several situations. For instance say you want to draw a line between LineStart and lineEnd, but then you want to draw another line of the same length but rotated 30 degrees. After drawing the first line with DrawLine, you only need to add 30 to LineAngle, change the LineSpriteID, and execute DrawLinePolar.

Reserved Variable Names for this Behavior:

Input/Output globalvars: LineSpriteID, LineThickness, LineImageIndex, DoThicknessOffset, LineStartx, LineStarty, LineEndx, LineEndy, LineAngle, LineLength

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

Technical Notes:

I have included the trig functions necessary to draw lines in this behavior (event MP_DoLineTrig). It is used by DrawLine to determine the length and angle of the line between the start and end points. DrawLine then calls DrawLinePolar to actually draw the line.

The DoThicknessOffset attribute adds half of the thickness of the line to both ends of the line, resulting in a line of length: LineLength + LineThickness. This is standard behavior for lines drawn in other software packages such as

Adobe Illustrator. The extra length ensures that the start and end points of the line are covered by line, instead of being on the edge of the line. This improves the appearence when multiple lines share an end point.

Revision History:

2-19-2000 Version 1.0 first written. The line algorithms are an improvement to the line routines that I have used over the past year. The improvements include speed, accuracy, and the addition of the DoThicknessOffset attribute.