

Creating Applets with Macromedia Director: AAPT Workshop, January 2003



Raman Pfaff (pfaff@explorelearning.com)
ExploreLearning, P.O. Box 2185, Charlottesville, VA 22902

This workshop is designed to give those from a physics/math background an introduction to creating interactive web-based content using Macromedia Director in a mere four hours — a daunting challenge, but let s have fun this morning!

Macromedia Director is an authoring tool for multimedia ranging from video, to CD's, to the Web. When learning Director almost all books and workshops start by teaching the basics of animation with no programming involved. This workshop takes the opposite approach and will start from a programming point of view rather than the animation point of view.

Workshop Overview

During the four hours we have available today we will focus on the following topics:

TABLE 1. Topics that will be covered.

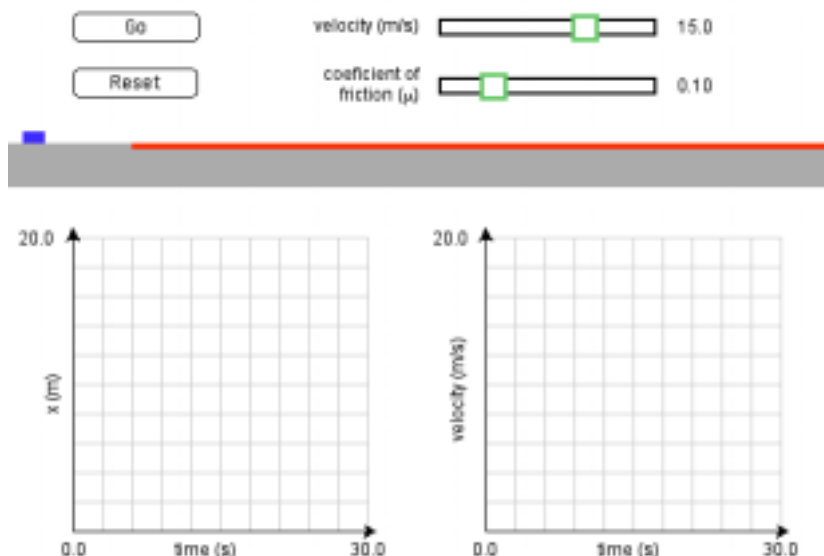
Hour	Subject	Time (minutes)
1	What is Director? Why do I use it?	10
	Introduction to Director	20
	Writing a Behavior/Simple Simulation	35
2	Simulation con't., adding friction, sliders	65
3	Simulation con't., creating a graph, publishing to web	55
4	3D Worlds, Havok Xtra, wrap up	45

Of course, depending on how things go, the times may vary from those shown above.

1.0 AAPT Workshop: Creating Applets with Director

The computers in the room have all had a trial version of Macromedia Director installed on them. If you would rather have it installed on your own personal computer I have trial versions available on CD for both platforms (Mac and PC). **The goal for the day is to build a simulation consisting of a puck that will slide from a frictionless surface with an initial velocity on to a surface with an adjustable coefficient of friction.** A sample “look” for this applet is shown below.

FIGURE 1. Possible Appearance of Puck on Frictional Surface Applet



1.1 Hour One: Introduction

We will start with introductions, some of the basic functionality of Director, and will create a very simple simulation by the end of this first hour.

1.1.1 Why do I use Director?

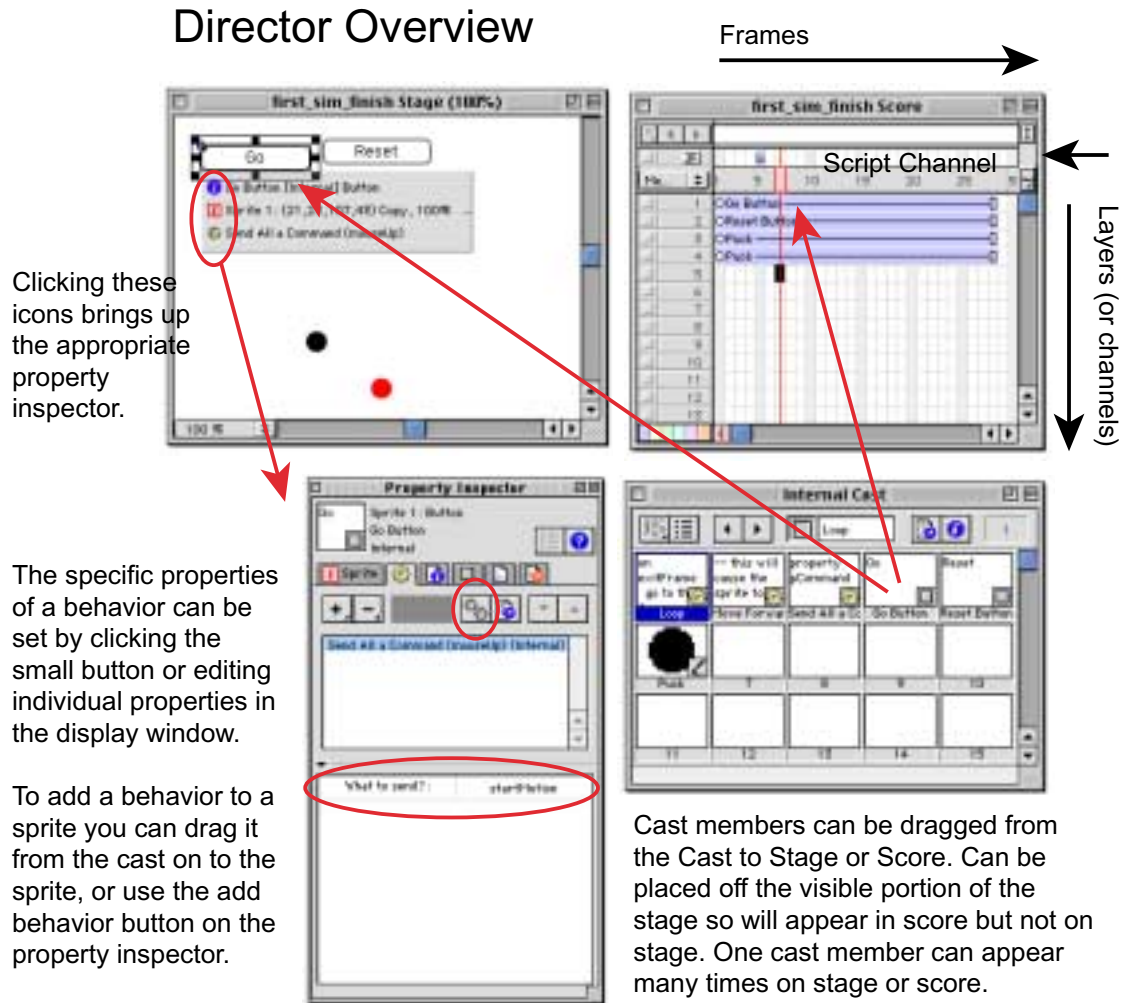
I began using Director back in 1995 when it first became possible to put the interactive multimedia on the Web with the Shockwave Plug-in. The other tools available at that time did not allow you to create anything else with comparable quality.

Now there are far more tools that allow you to create interactive applets, but Director continues to improve, and in my opinion is the best tool (personal opinion!). Some of the competitive products include: Macromedia Flash, a variety of Java tools, QuickTime Scripting, Visual Basic, and many others. Director has a powerful programming language, allows for quick development time, author once to deliver on Mac & PC's, creates very small web-based files, and has a relatively common Plug-in for web viewing.

1.1.2 Introduction to Director

When I first started using Director the program was very straight forward for me, but this is rarely the case for most people! The Figure below shows my view of Director.

FIGURE 2. Director Overview



1.1.3 Constant Velocity Particle

For the first simulation we will create a simulation that has a “Go” button, a “Reset” button, and a particle that will move at a constant velocity in the positive x direction when the Go Button is pressed. Open the movie called **Simulation_001.dir** (choose File > Open... is one way to open the file).

Before doing any work on the simulation it is a *very good idea to storyboard* the entire simulation. Try to answer the following questions before we begin.

- What should the “Go” button do in this simulation?
- What should the “Reset” button do in this simulation?
- What should the puck (particle) do? Can the motion of the puck be written as an equation?
- When the “Reset” button is pressed what information (properties) do you need about the puck?

Here are a few of the basic Lingo terms that will be used throughout the day:

TABLE 2. Some Basic Lingo Terms

Lingo Term	Meaning
exitFrame	movie is leaving a frame
mouseUp, mouseDown	user has clicked
sprite(number).locH, sprite(number).locV	the horizontal and vertical position (location) of an object on the stage (a sprite)
sendAllSprites(aCommand, info1, info2,...)	this will send a command to every sprite that is listening

There are also three different types of script members shown in the Table below. We will primarily focus on using “Behaviors” which are scripts that can serve multiple purposes and have a drag-and-drop functionality (write once and use many times, simplicity).

TABLE 3. Types of Scripts

Lingo Term	Interpretation
Movie Script	A tool box script: receives data, processes data, returns processed data.
Behavior Script	“Drag and drop” functionality, properties.
Parent Script	For use with true Object Oriented Programming

Your file should now be open so we can begin some editing. First we will place several things on the stage. View your cast window and find the item labelled **Go Button**. Drag this on to the stage. Now drag the **Reset Button** on to the stage. Finally, drag the **Puck** to the stage. Do these items now appear in the score window? If they don't all start and end in the same frame align them by dragging (click and hold mouse down while moving) in to the proper location in the score window. We will now walk through the following steps:

- Place the **Loop Behavior** in to the script channel of the score window (not in first frame).
- Place the **Send Message on mouseUp Behavior** on the **Go Button** and set the parameters (have it send a “startMotion” message), and then the **Reset Button** (have it send a “resetSimulation” message).
- Place the **Linear Motion** behavior on the puck to control the motion.

We have three things on the screen, each of which has a behavior applied to them. Play the movie. What happens? In the score window does the movie stop advancing in the frame with

the Loop Behavior? If you press the Go Button does anything begin moving? Open the Message Window and press the Go Button (be sure the movie is still playing). Was there some text in the message window? Having scripts provide text feedback is often useful when debugging movies. If at any time you feel as if you have “broken” the movie you can always start over again by opening the original file and not saving the current work (or save it to a different file).

Currently this movie is looping in the same frame. When you press either button a message gets sent to everyone, but no one listens at this time. Stop the movie. Click (single click, not double click) on the puck and then click on the **Behavior Property Inspector Button** (the small gear, lowest little icon on the sprite overlay) to bring forward the Property Inspector window. On this window click on the small script window icon (be sure the **Linear Motion** Behavior is selected) to enable editing of the script (there are many other ways to access this script which we’ll see along the way).

You can now begin editing the Linear Motion script. When the Go Button was pressed it sent out a message to everyone. What should now happen in the puck behavior? What information will the puck always need to keep about itself (properties)? In the sample scripts (at the end of this document) you can see how properties are defined, and some other common things that can be done in Lingo. Try to develop the script for your puck so that the buttons both work as planned and the puck moves forward at a constant velocity. You can constantly check that your script has no syntax errors by clicking the small lightning bolt at the top of the script window (Recompile all scripts), and you can also run and stop your movie multiple times as needed.

At the end of this portion of the workshop be sure to save your simulation. You can then open the file **Simulation_002.dir** to see a completed version of this simulation. Spend several minutes going through the scripts associated with this movie.

1.2 Hour Two: Sliders and Simulation Development

We now have a basic simulation for a puck moving at a constant velocity, but the user can’t easily change the initial velocity, and we now want to add friction to the simulation. Our goal is to create a slider so the user can easily change the initial velocity, and have the puck slide in to a region with an adjustable coefficient of friction.

1.2.1 Creating a slider

A slider is just a term used to describe a device which allows you to change a value via a device. In Director this can be done with two cast members in two channels in the score. One member is a small button that the user can drag, and the other is a long bar that the button/knob will slide on. The small knob can only be moved back and forth within the limits of the bar.

Open the movie **Simulation_003.dir** (save anything else you were working on before doing this if you wish). This movie contains a graphic element for both the knob and the slider bar, as well as a text member which will display the value of the slider. Drag the **Slider Track** on to the stage followed by the **Slider Button** and the **initial velocity** text member. You may want to position the text display so it does not cover the knob or the bar. We will now try to create a slider behavior to apply to the knob/button.

What information (properties) do we need to get a functioning slider? A minimum value? A maximum value? There are at least three more things I can think of. What about you?

-
-
-
-

How exactly should a slider work? When the user clicks (mouseDown) on the button the slider should become active. As the user moves the mouse (mouseLoCH) to different locations the button should move along the bar. While the slider button is moving the value in the display should change to reflect the new slider value. Open the script window and work to create a slider button behavior. Be sure to use the sample scripts to help you with the Lingo formats and commands. Also don't forget to save often during development. At some point you will have to attach the Slider Button Behavior to the button itself. Don't forget about that.

After about 25 minutes save your work and open the file **Simulation_004.dir** to see my sample slider script. My key features were the minimum value, maximum value, number of digits to display, the name of the text member that displays the value, and the channel number of the slider bar. In this particular case I set the initial velocity range to 0.5 to 2.0 (m/s). I also added a text cast member to label the slider so the user would know exactly what the slider was for.

1.2.2 Adding friction to the simulation

Go ahead and open the **Simulation_005.dir**. Since the goal is to study what happens to a puck as it moves across a frictional region, we will now add a slider to the stage so that we can adjust the coefficient of friction. Drag the Slider Bar from the cast to the stage, followed by the Slider Button and the text member titled **coef of fric**. Organize these on the stage appropriately. Drag the Slider Button Behavior on to the knob (either in the score window or the stage window). Choose appropriate values for the range limitations (I chose 0.0 to 0.50). Play the movie to be sure this additional slider is now functioning. You can also drag the member named **mu** on to the stage as the label for this slider.

For adding the friction there are many ways that this can be done. We'll take the approach in which the puck will start on a frictionless surface, and the frictional force will be felt after the puck has passed a specific location on the stage. In the Linear Motion Behavior the position, velocity, and acceleration are controlled through programming. Since we are dealing with a puck on a flat surface the acceleration is zero in the frictionless area, but is not zero in the region with friction. The big question is how to tell the puck to change its acceleration. How should this be done? There are several ways this could happen, both inside our current scripts with several edits, or with new scripts. Can you think of several different methods?

My choice on this topic is to have a new script (behavior) that will watch the puck and tell it to change its acceleration when it has passed in to the friction region. My primary reason for choosing this option is so that my basic Linear Motion Behavior won't have any "custom" coding in it and it could easily be used to control other objects that obey basic Newtonian motion in a future simulation. Writing generic behaviors can really aid in future development if they are well written initially!

Create a new script and name it “Puck Watcher.” Drag the image named “Friction Region” to a location so that it covers the right two-thirds of the stage. Go ahead and drag the “Table” image on to the stage so our puck is moving across the table and then hits the friction region. Drag the Puck Watcher Behavior on to the Friction Divider on the stage (or the Table) and edit the script so that when the puck has an x value that has moved in to the friction region it will set the acceleration of the puck to the appropriate negative value. As you develop the Puck Watcher Behavior it is good to save your work often. Test the movie as you add new features.

What happens to the puck as it slows with the negative acceleration? Does the velocity become negative? What happens to the puck under this situation? In the Puck Watcher Behavior you may need to add some coding to deal with this situation. When you change sliders should that affect the motion as the puck is moving? Should the user be able to change the coefficient of friction as the puck is moving? All these things must be considered as development of the simulation continues.

Open the file **Simulation_006.dir**. This file has the fully functional version of this simulation where the user can adjust the friction before the puck is moving (this is controlled in the Puck Watcher Behavior which checks the coefficient of friction slider value when the simulation begins), and when the velocity of the puck becomes negative it is set to zero (along with the acceleration) so it won't get in to a “bouncing” mode where it continues to jump slightly back and forth.

1.3 Hour Three: Graphs, Appearance, and the WWW

At this point the simulation consists of a puck starting on a frictionless surface with a constant velocity, which then moves on to a surface with friction where the velocity decreases. There are really just three things left that we would like to do to this simulation: 1) add a graph so that we can view a plot of velocity vs. time (or any other variables of interest), 2) improve the overall appearance so it is a bit more user-friendly, and 3) publish it to the WWW.

1.3.1 Adding a graph to the simulation

Go ahead and open the file **Simulation_007.dir**. Added to this simulation is a **Graph Dot** image and a **Grid - 1 Quadrant** image that will be used for the actual graph. Go ahead and drag the grid on to the stage followed by the graph point. For the most basic graph the requirements are rather simple: be able to instruct a graphing point to move to a selected x-y point on a grid. When we want the graph to show a line/curve as time advances one way to do this is to use the “trails” feature of Director in which an object will leave a trail of its image every time the object moves.

For our **Graph Point Behavior** we want to be able to send it a command such as `sendSprite(i, #movePointToSpot,x,y)` and have the point move to that (x, y) point. If the point is not on the graph have the point move off the screen. It would also be nice to have the ability to switch the trails on and off, and be able to clear all the trails at any given time. Go ahead and create a new script and see if you can create a Graph Point Behavior. As you add new features it is good to test them one at a time. Using the message window to send information to a behavior is a very good way to do quick testing. When you begin this process be sure to think of the information (properties) you need to keep track of in the behavior. You may want to start with a very simple behavior where the x and y axes will both have the same scale on the standard one quadrant graph (or go ahead and try a more generic graphing engine).

(Depending on time constraints we could just open the movie **Simulation_008.dir** and proceed from this point. This file has the fully functional Graph Point Behavior applied to the dot, along with labels placed on the stage.)

Once the **Graph Point Behavior** is working we now want to make use of it to view the velocity as a function of time. The best place to do this is in the **Puck Watcher Behavior** (so that we don't have to add any custom coding to the Linear Motion or Graph Point behaviors).

Edit the behavior and have it report the velocity and the time on every exitFrame to the Graph Point behavior. With all this editing of scripts are all elements still working? The Reset Button? The Go Button? Both sliders? When adding new features it is always important to make sure no other items need any modifications. With more experience in Lingo (as with other programming languages) one becomes accustomed to good ways to code so that no changes are required in basic behaviors (such as the Graph Point, Linear Motion, etc.), and most coding changes happen in just one place (such as the Puck Watcher in this particular case).

Go ahead and open the **Simulation_009.dir** movie to see the completed grapher which shows graphs of x , and v versus t which are dynamic as the puck moves.

1.3.2 Improving the appearance

Our applet at this point is rather plain looking but it is relatively easy to improve the appearance by adding just a few images to the stage. When adding images I always think it is rather important to keep the file size small (for web-based content) so minimizing images or using vector-based images (Flash) are often a good idea. In this instance the graphics were rather small in size and bitmap images were used.

Using just one background image and a few pretty images for several items on the stage can have a drastic effect. Open the movie **Simulation_010.dir**. There are several images in the cast that can be used. Drag the Background Image to a very low channel in the score. Click on your puck on the stage and then click on the image labelled Pretty Puck in the cast and click on the swap button on the toolbar. Reposition things on the stage so that the puck is properly positioned on the new sliding region. It is already looking much better!

To see a more complete version with several nice graphics additions you can open the movie **Simulation_011.dir** (although "pretty" is a very arbitrary term, and this one was not looked at by any graphics people!).

1.3.3 Publishing to the WWW

Getting your completed applet to the web is probably the easiest thing we'll deal with today. With your file open you can simply click on the Publish Button in the Director toolbar. There are several preferences that can be set for this in the **File > Publish Settings...** menu command but we can ignore those most of the time, but let us take a look at two features. Select the option. Under **Formats** disable the **view in browser** option, and under **Compression** choose **Standard Compression** for images. The reason for the former is due to a security issue (see next paragraph), and the latter is due to the compression techniques of JPEG tend to cause "oddities" when the file is viewed in a browser. Now go ahead and click OK to go back to the movie. Now when you click on the publish button an html file will be created along with a **dcr** file which is a compressed (and non-editable) version of the applet.

The reason we disabled the immediate view in browser option is due to a security issue with Shockwave content. When viewing locally (not via the http protocol) the content needs to sit in a folder named dswmedia. Take the html file and the dcr file and place them in a new folder called dswmedia. Then view the html file in a browser. The html file can be edited to add additional content around the dcr file. A sample file is located on the CD which has minimal information added to the html file which was produced by Director.

1.4 Hour Four: Exploring 3D Capabilities

We will now take a very quick look at some more advanced features of Director which include the third dimension, and a physics engine that is used in many computer games today.

1.4.1 From 2D to 3D

A significant amount of learning can occur with simple 2D simulations, but in some cases the 3D world can really add a significant feature to a simulation (such as 3D chemical structures, 3D mechanics problem, astronomy, etc.). In the current version of Director it is *relatively* simple to build a 3D world, although by no means is it trivial!

In the book listed in the Additional Resources I only list a single book, since the content on the web is much more useful than the content in books – most of the time. The book that covers the 3D abilities of Director is called Director's Third Dimension by Paul Catanese and it comes with a CD full of informative code. If you plan to use 3D content it is definitely worth the price (and it is very reasonably priced).

With the 3D in Director you control models, lights, camera, and groups. The models can be create in a 3D program or directly via Lingo. You can create planes, spheres, boxes, cylinders, and particle producers with Lingo which allows for exceptionally small file size since all content can be created at run-time. Every model can also have a texture and a shader associated with it. Despite all the nifty features, there are a few drawbacks. For performance across many platforms the “look” of the simulation can change based on the hardware/software configuration of the user system, there are no shadows created by objects (although there are some work-arounds), and you can only edit the 3D world in Director via coding (no typical 3D tools built in).

On the CD there is a simulation which allows you to create a myriad of solid objects on the screen and position them in new locations. While exploring the features see if you can create something that could be used in a physics lab simulation.

1.4.2 The Havok Xtra (Physics Engine)

The Havok Xtra is a fully integrated rigid body physics simulation engine for Macromedia Director. It enables you to assign properties such as mass and elasticity to physical objects, apply forces, impulses or torques and set velocities and momentum. You can also register interest in specific rigid body collisions or even disable them completely. Support is provided for importing Havok HKE (Havok Export) files, which allow full physics scenes to be constructed in a 3D modeler (like 3ds max) without requiring additional Lingo scripts. [ed. – this paragraph was taken directly from the Havok web site]

Many simulations can be created with the use of this Xtra. There are several files on the CD that we will discuss (if there is time) to show a few of the capabilities of this physics engine,

Conclusion

including a friction force experiment that uses several of the features of the simulation we built during this workshop. If we run out of time be sure to explore this simulation at a later time.

2.0 Conclusion

We have barely touched on the capabilities of Director. If you plan to further learn about Director it is best to design a project which has a well defined final product and then just work your way towards it rather than just trying to learn all the Lingo commands. There are many resources available online where you can find out a wealth of information. I've been using Director since 1995 and still feel as if I know very little and have much room to grow. I hope you found this workshop useful and if you have any suggestions feel free to drop me a line.

3.0 Additional Resources

There are many web sites that provide a wealth of source code, tips, and tricks for using Macromedia Director. Several that I commonly use are shown below.

- Director Online has many lessons with source code (<http://www.director-online.com>)
- DirectorWeb – great mailing list (<http://www.mcli.dist.maricopa.edu/director/>)
- Havok Site contains all the Havok Xtra info (<http://www.havok.com/xtra>)
- DirectorDev (<http://www.directordev.com>)
- Macromedia's Director Site (<http://www.macromedia.com/software/director/>)
- Macromedia Forums are a good Q & A site (<http://webforums.macromedia.com/director/>)
- LingoProgrammer (<http://lingoprogrammer.com/>)
- MediaMacros (<http://www.mediamacros.com/>)
- Making a 3D world (<http://www.dmu.com/3dd/dd0.html>)
- Stormsky (<http://www.stormsky.com/>)

There are also many books available on Director but the content on the web is generally more informative since most books tend to just discuss commands and don't delve in to actual development. There are a few exceptions to this statement, and sometimes others find books more tangible, so feel free to see what is out there (new ones continue to show up). For any 3D development in Director I highly suggest this book:

- Director's Third Dimension: Fundamentals of 3D Programming in Director 8.5, Paul Catanese.

It is an exceptionally well written book with excellent sample code.

4.0 Sample Behavior

The following is a sample behavior that will move an object across the screen from left to right at a specified number of pixels each time an "exitFrame" handler is called.

Sample Behavior

```
-- SAMPLE BEHAVIOR
-- In Lingo any time two minus signs are together this indicates a comment
-- and content after the two signs is ignored

-- This behavior will cause the sprite to move across the stage to the right (positive)

property spriteNum
-- spriteNum is the channel number of the sprite (automatically set if declared as property)

property pX -- will keep track of x value
property pNumOfPixels -- how many pixels will it move each frame

-- the beginSprite just happens once when this sprite first
-- appears on the stage in the score
on beginSprite me
  put "I began" -- this will place text in the message window
  pX = sprite(spriteNum).locH -- the horizontal location of sprite in channel spriteNum
  -- locV is the vertical location of the sprite (measured from upper edge of stage)
  pBar = 25 -- a "hard coded" property where a bar is in channel 25
end

-- an exitFrame occurs every time the movie tries to exit from this frame
-- when looping in this same frame it happens until you leave the frame
on exitFrame me me -- using an enterFrame is another possible option
  pX = pX + pNumOfPixels -- determine new location
  sprite(spriteNum).locH = pX -- move the sprite to this new location
  sendSprite(pBar, #reportValue, pX)
  -- the line above calls the handler called "reportValue" in the behavior on sprite 25 (pBar)

  sendAllSprites(#reportValue, pX)
  -- the above line would send the value pX to every sprite that contains
  -- the handler named reportValue
  go to frame "information" -- this would cause the movie to jump to the frame which labelled "information"
end

on placeInformation me, aValue -- this is a handler in this behavior that could be called
  put aValue -- this line would put the aValue in to the message window
  -- whenever the handler placeInformation is called
end

on mouseUp me
  -- when the user clicks up on this sprite the scripts here will run
  put sprite(spriteNum).left -- this would put the left side of the sprite (in pixels) in to the message window
  -- other options are right, top, and bottom to get the x or y values in pixels of the sides of a sprite
end

on mouseDown me
  beep -- this will make a system level beep happen when the user clicks down
end

-- when this behavior is dropped on an object the author
-- will be prompted for the values which are in the propList
-- (any name can be used, such as myPropList, aList, etc.)
-- many formats are allowed, such as integer, boolean, string, member, sound, etc
on getPropertyDescriptionList me
  propList = [:]
  propList[#pNumOfPixels] = [#format:#float,#default:"5.0",#comment:"Pixels to move?:"]
  return propList
end getPropertyDescriptionList
```