Director 8.5 Tutorial

Date Produced: Jan 02

Exploring Environments

Author: Martin Agombar

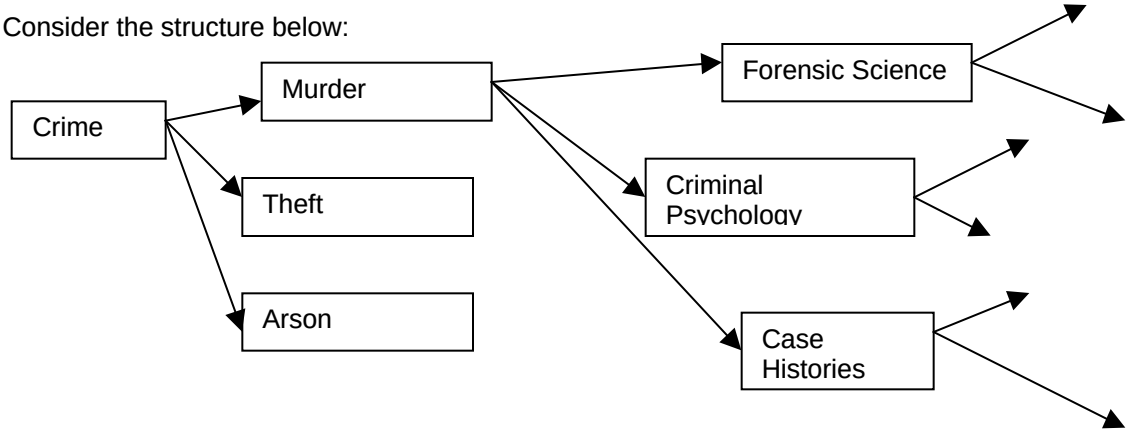
Introduction

This tutorial aims to explain some approaches to making environments explorable, using Director, to enable the end user to interact with the presentation.

Structure

One method of making a presentation explorable is to jump from part of the Score to another. This can be done using markers and simple Lingo commands. If the presentation is large however, this method can become unwieldy. A much better method, for large presentations, is to jump from movie to movie.

Consider the structure below:



If you tried to make the above presentation using markers and Lingo the resulting Score be very complicated and confusing. Not only this but the **Projector** would be very large since it would have to contain all the elements of the presentation at once. Far better to have an opening movie called “Crime”, which was then linked to other movies such as “Murder” or “Theft”. Providing all the **movies** you created lived in the **same folder** as the Projector, your presentation would play as if it was only one movie.

To link to another movie

- You need to make at least two movies. In the case above, “Crime” would be the opening of the presentation. “Murder” would be the opening of the murder section. “Case Histories” would be a sub-group of that and so on.

- Ensure all movies are saved in the same folder.

· To jump to a new movie you need to create a button. Add the following script:

```
on  mouseUp me

go to movie “Murder”

end
```

You need to add the name of the movie you want to open. Use only the name, there is no need to include the extension.

· Make a Projector. **File > Create Projector**

Add the movie that will open your presentation. Make sure you are **saving** to the **folder** that **contains** all your Director **movies** for the project, then go **Create…**

Reminder: all movies must be in the same folder as the Projector

Using Lingo to make a moving backdrop

You may wish to create an environment that can be explored. For example, supposing you wanted to create an underground cave. You might want the user to be able to move around inside this cave. To do this you could move the background horizontally so that the user could see what was to the left and right of the Stage.

In order to make a background move under the control of Lingo, we need to look at some simple examples to see how this might be done.

Moving spites using Lingo

Sprites on the Stage are located using coordinates. These refer to the registration point of the sprite. You can find the position of any sprite on the Stage, or change its position, using Lingo.

For example:

Create a simple sprite and make sure it is in **Sprite Channel 1**. Add the following script to this sprite:

```
On exitFrame me

xpos=sprite(1).locH
ypos=sprite(1).locV

put xpos, ypos

end
```

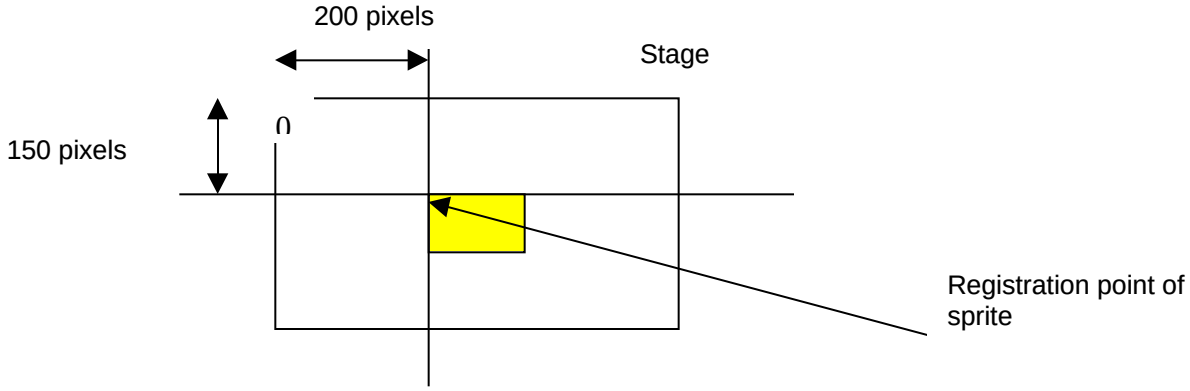
Open the **Message Window**. **Window > Message** (⌘ + M)

Run the movie. The **coordinates** of the sprite in **Sprite Channel 1** will appear.

Animate the sprite in some way and the coordinates will constantly change.

How sprite coordinates relate to the Stage in Director

The example above shows how the coordinates, and therefore position, of a sprite can be found. The meaning of the coordinates is important to understand. They relate to the Stage and how it is mapped out.



The position of the sprite is the distance in pixels measured across from the left edge of the Stage and the distance measured downwards from the top edge of the Stage, i.e. (200 x 150).

We need to look at the script in more detail to see how it works.

Explanation of script

on exitFrame me

This part of the script is called an **event handler**. Event handlers always begin with the word **"on"** An event handler is used to detect something when it happens, such as a key press. In this case it responds to the **Play Head** leaving a **frame**. Every time this happens, the handler will detect this. Whatever code is associated with the event handler will then be carried out.

xpos=sprite(1).locH

xpos is a **variable**. Its name is one I made up. It could have been anything. Its value is only determined when it is made equal to something. One way to understand variables is to imagine them as containers. They can be used to bring data from one place to another. In this case **xpos** is being made equal to the **horizontal position** of the **sprite** in **Channel 1**.

Sprite(1) is the sprite in **Channel 1**. To refer to sprites in other channels, simply change the number held within the brackets.

locH is a **property**. It is one of the properties of **sprite(1)**. Notice that **sprite(1)** is linked to **locH** by a full stop. This is very important. In Lingo a full stop is always used to link a property to an object.

ypos=sprite(1).locV

This expression makes the variable **ypos** equal to the **vertical position** of the **sprite** in **Channel 1**.

put xpos, ypos

The command **put** tells Director to display information in the **Message Window**. In this case, the variables **xpos** and **ypos** are **put** into the **Message Window**.

end

This tells Director that the code for the event handler has ended.

Using coordinates to control the position of a sprite

To make a sprite jump to a new position, using Lingo, we need to make use of coordinates. Make sure there is a sprite in **Channel 1**, positioned centrally. Create a button and attach the following script:

```
on mouseUp me

sprite(1).locH=0

end
```

Play the movie and hit the button. The sprite in **Channel 1** will jump to the left. This is because the script has changed the **horizontal position (**given by **locH)** of the **sprite** in **Channel 1,** to **zero**. This equates to the left-hand edge of the Stage.

Continuous movement sprites using Lingo

In the previous example, we moved a sprite using a button. With a small change to the scripting, we can animate the sprite.

Make sure there is a sprite in **Channel 1**. Add the following script to **frame 1** of the Script Channel.

```
property xpos

on exitFrame me

xpos=xpos+1

sprite(1).locH=xpos

put xpos

end
```

Run the movie. You will see the sprite in **Channel 1** move slowly to the right. What is happening?

To make the sprite move, the horizontal position of the sprite has to change over time. The expression **xpos=xpos+1** is used to create a number that increases by one at regular intervals. The expression works in the following way. The **on ExitFrame me** handler responds each time the **Play Head** leaves a frame. This happens 30 times per second (if the frame rate is 30 frames per second). Each time the handler is triggered, the script is run.

To begin with, **xpos** has a value of zero. Because it has been made a **property**, using the line **property xpos**, at the very top of the script, it is able to remember its value. Director moves through the script line by line until it reaches the line **xpos=xpos+1**. This expression adds **one** to **xpos**. so once the script has run, **xpos** has the value of **one**, which it then **remembers**.

In a very short time, the **on ExitFrame me** handler is triggered again. Since **xpos** now has a value of **one**, which it has remembered, when another **one** is added, its value is increased to **two**. This process continues indefinitely, increasing the value of **xpos** by one each time.

sprite(1).locH=xpos makes the horizontal position of **sprite(1)** equal to **xpos**. Since **xpos** is constantly increasing in value, the sprite is forced to move further and further to the right.

You can check **xpos** using the **Message Window**.

Making a backdrop

You can make a backdrop in two ways.

- Draw a backdrop
- Use a digital camera

Whichever method you use you need to make sure of two things:

- You create two images, each the **width of the Stage**.
- The two images form a **continuous loop**.

Making a backdrop using a digital camera

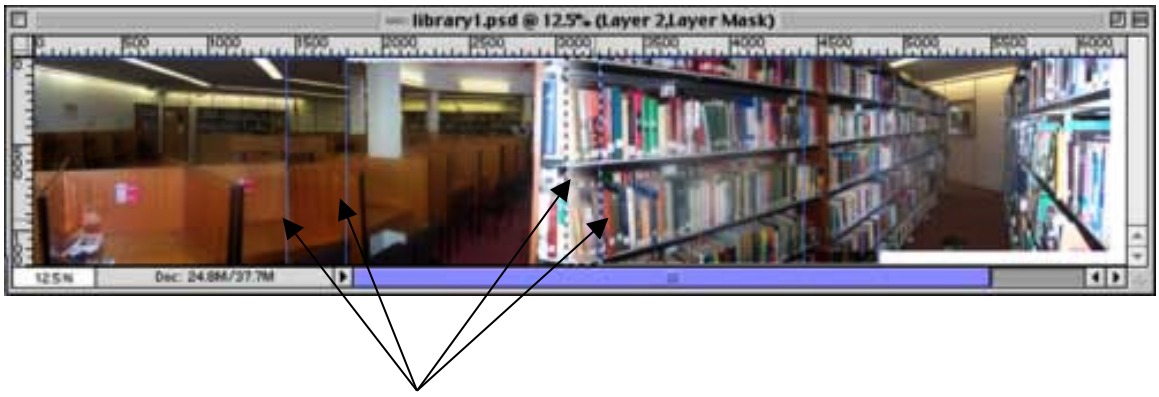
1 You need to take a series of photographs, each one slightly overlapping the last. Obviously these need to be a 360˚ panoramic view. Make sure you have enough photographs for a complete circle.

2 Import the pictures into **PhotoShop.**

3 Create a new document. Make it at least as wide as all the images arranged in a line, i.e. if you had six images, each 1800 pixels wide, make the new document about 12 000 pixels wide.

4 **Copy and paste** the images into the new document so they are in a line (on separate layers). You will find they do not match exactly. You need to blend in the edges of each one to get the desired result.

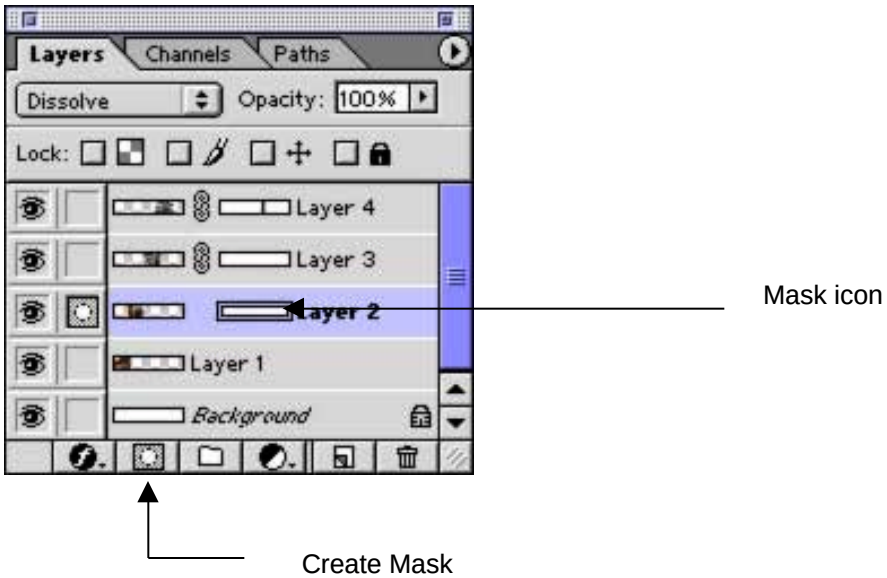
5 To **blend in** the edges do the following:



Guides

Set up **guides** to mark out the **ends** of the images. The images will overlap to some degree.

Create a **mask** for the second image. Click on the **Create Mask** icon in the Layers Palette.



Next, click on the **mask icon** within the **layer** containing the **second image**. This ensures that you are working on the mask itself and not the image.

Use the rectangular marquee tool to draw a rectangle where the first and second images overlap. Make sure you have **Snap to Guide** selected. **View > Snap To > Guides**

With the **overlap** between images 1 and 2 still **selected**, use the **gradient fill** tool to produce a gradient from **black to white** within the **selection**. The direction of the gradient should be from **black** at the **end** of the image to **white** as you move towards its **centre**. You should have created a rectangular shaped gradient mask. This will blend image 1 to image 2.

Repeat this operation for the remaining images.

6 Flatten the image. **Layers > Flatten**

7 You now have one layer containing all of your picture. You now need to make sure the ends will match when they are put together.

8 Set up a **guide** to mark the **beginning** of the image. Do the same to mark a position **just before** the **end** of the image. Select the **end portion** of the image using the **Rectangular Marquee** tool. Jump this to a new layer. **⌘ + J**

Use the **Move** tool to position the **end portion** of the image to just before the beginning so the two images appear to **butt up**. Move the **end portion** further to the right so that it **overlaps** the first image. Find a position where the two will **blend together** successfully.

9 Use the mask technique explained above to blend the two images.

10 Flatten the image. **Layers > Flatten**

11 Use the **guide** at the **beginning** of the image to **select** the area you have added when you moved the **end portion**. Use the **Rectangular Marquee** tool to select this area. Use the **Move** tool to drag this area rightwards to the end of the image, where you set up a guide. Its end will now match the beginning of the image.

12 **Crop** the **top** and **bottom** of the image to **tidy up** any problems.

13 **Resize** the image to exactly **twice the width** of the **Stage** you are working on in **Director**. **Image > Image Size**

14 Set up a **guide** exactly **half-way** along the image.

15 **Copy and paste** each half of the image into a new document.

16 Save each half. **Go File > Save for Web**. **Optimise** appropriately.

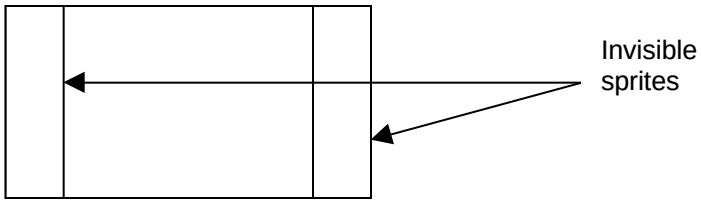
Building the moving backdrop in Director

The following example assumes a stage size of 640*480. Other stage sizes will require the final script to be adjusted accordingly.

1 Import the two images you have created. Position them so they are butted up against each other.

The movement of the two halves needs to be controlled by a user action. Two invisible sprites, at either end of the Stage, will trigger the movement.

2 **Create an invisible sprite**. Use the **Rectangular Shape** tool in the main toolbar. Make sure the type of line used is **none**. Position the sprite at the left-hand edge of the Stage as shown. Reuse the cast member to create the right-hand sprite.



Add the following script to the left-hand sprite:

```
global gLeft

on mouseEnter me
  gLeft=TRUE

end

on mouseLeave me
  gLeft=FALSE

end
```

This sets the global variable gLeft to TRUE, if the mouse has entered the invisible sprite and FALSE if it has left the sprite. A global variable is one that can be read anywhere in the movie. It is a way of communicating to other sprites. By doing this the two images of the backdrop can be told that they need to move.

Add the following script to the right-hand invisible sprite:

```
global gRight

on mouseEnter me
  gRight=TRUE

end

on mouseLeave me
  gRight=FALSE

end
```

We now can tell the two sprites, which comprise the backdrop, whether they need to move right or left.

3 Finally, we need to add a script to each of the two sprites to make them move. The script does two things. Firstly, it moves the sprites according to whether the left or right invisible

sprite has been rolled over. Secondly, if a sprite has moved beyond the limits of the Stage, it is shifted to the other end of the Stage. This gives the impression of a continuous, flowing backdrop.

Add the following script to both of the sprites comprising the backdrop. Remember, you only need to write the script once. Drag the script in from the Cast to reuse.

Remember to name your scripts appropriately

```
global gRight, gLeft,
property rate, sprt, sWidth

on beginSprite me
  sprt=me.spriteNum
  sWidth=sprite(sprt).width
end

on exitFrame me
  rate=5

  if gRight =TRUE then

    sprite(sprt).locH=sprite(sprt).locH+rate
  end if

  if gLeft =TRUE then

    sprite(sprt).locH=sprite(sprt).locH-rate
  end if

  if sprite(sprt).locH <= -640 then
    sprite(sprt).locH=sprite(sprt).LocH+1280
  end if

  if sprite(sprt).locH >= 640 then
    sprite(sprt).locH=sprite(sprt).LocH-1280
  end if


end
```