

# Lingo Introduction

There are several images in studentinfo on Trentdev\images

Lingo is the language you write to take more control out of Director. In this tutorial you will use some of the basic commands and start to use the code you will build on in following weeks.

There are three main terms you need to be aware of at this stage. These are

- Statement – This is one line of code and is the basic building block of Lingo. These can be as simple as `stop` or as complex as `sprite(23).blend = 100`.
- Handler – This is a group of statements put together. A handler begins with a handler name and ends with an `end`. Everything in the handler is run when the handler is triggered. The example below commands your movie to show a pop up box saying hello

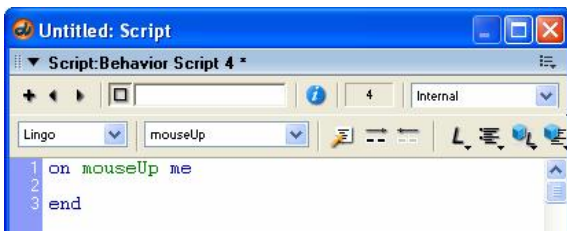
```
on hellobox
  Alert "hello"
end
```

- Event – This can be anything in a director movie. Events can trigger code to run. An event is anything such as a mouse click or move or when a certain frame is entered on the score. An example of the code in an event handler is shown below, which will cause a popup to appear when the box is pressed.

```
on mouseUp
  alert "hello"
end
```


## Scripting An Event Handler

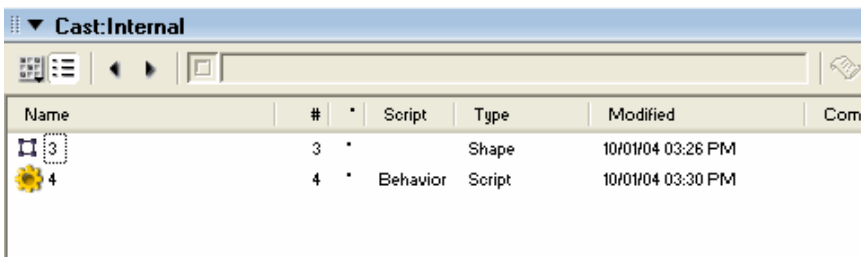
Open a new movie and add a square to the stage. Right click on the square and choose 'Script'. A box appears similar to the one shown below.



Change the code to show the following

```
on mouseUp
  alert "hello"
end
```

You will notice that by adding the code this way you have created a new behaviour, and this will appear in the cast window shown with a  similar to below.



Play the movie and see what happens

## More Mouse Up

Now lets do something more interesting when you press the box.

Add circle on the stage, and now we need to change the name of the sprite to circle1, by clicking on it in the cast and renaming it in the 'Property Inspector' as shown below



Change the code in the behaviour you have created to say the following

```
on mouseUp me
    sprite("circle1").foreColor = 123
end
```

When you run the movie again, the colour of the circle will change when you click on it.

You may also want to add a random effect to the colour of the circle. Change the code to show the following.

```
on mouseUp me
    sprite("circle1").foreColor = (random(256) - 1)
end
```

If you run the movie now and click the square, you will see the colour changes each time. There are 256 colours available for this shape and the random generator generates a number between 0 and 255.

## Different Types Of Script

There are 4 main different types of script you can add to your director movie. These are

- Behaviour Scripts – These are associated with a frame or a sprite. These are good to run code associated with that frame or sprite and are also useful if you want to add the same code to lots of different frames or sprites. Examples of these are the tasks you have done so far.
- Cast Member Scripts – These are attached to cast members, and does not independently have an instance in the cast window
- Movie Scripts – These are applied to the whole movie rather than to specific sprites or frames
- Parent Scripts – specialised scripts to create child objects. Do not worry about these at the moment.

## Movie Scripts


The example below shows you how to add a script to the movie, and in this case it will be when the movie starts and stops.

## Start and Stop Movie Events

Go to 'Window' and choose 'Script'. Type the following in the script window that appears. This has created a movie script rather than a behaviour.

```
on startMovie
  sprite("circle1").foreColor = 26
end
```

```
on stopMovie
  sprite("circle1").foreColor = 44
end
```

This has created a movie script rather than a behaviour so you should notice in the cast that the script is signified by 

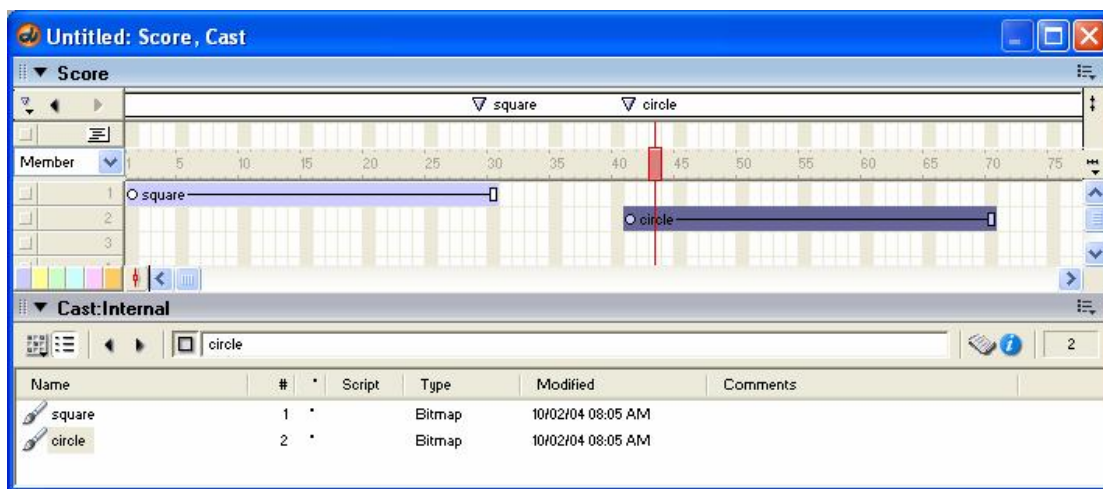
Make sure the loop playback is not ticked and test the movie. You should see the movie starts with one colour and when the movie ends another colour is applied.

## Frame Scripts


You can also add code to a particular frame of the movie. The following task shows how this is done.

Create a new director movie.

Create two shapes in the paint window. Call one circle and one square, and position them on the stage as shown below.



You should also see that I have added two markers called 'square' and 'circle' to the score. Add these to yours as well.

The frame script channel is just above the timeline, signified by 

In that channel, right click in the frame that your square marker is on and choose 'Frame Script'. You can then type in the following code.

```
on exitFrame me
  go to "square"
end
```

This will cause the movie to stop on that frame.

Now, using previous examples, add some code when the square is clicked to jump to the circle marker.

# Handlers

In director you can create your own handlers to hold several lines of code.

Create a new movie and add a circle to the stage.

In the 'Property Inspector' call the sprite circle, as shown below



Go to 'Window' and then choose 'Script' (remember this is a movie script and can be called at any time in the movie) and type the following

```
on movesprite
  sprite("circle").locH = sprite("circle").locH + 5
  sprite("circle").locV = sprite("circle").locV + 5
end
```

This moves the horizontal location of the circle 5 to the right, and the vertical location 5 to the right.

Now you need to add a button to trigger the handler. Add a small rectangle to the stage and right click it and add the following script.

```
on mouseUp me
  movesprite
end
```

This will activate the movement each time the button is pressed. Put your movie on continuous play and test it.

Add another handler to your movie to move the circle to the left rather than the right, and add this to another small square on the stage.

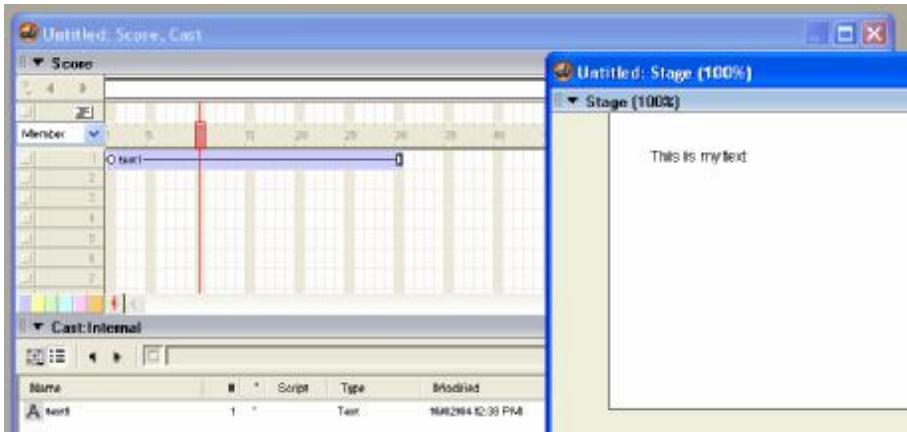
# Coding Basics

## Variables

Variables are bits of space to store values in them. Variables can be local – just accessible in the code you use them, or global – accessible throughout the movie.

To use variables it is easy as is shown in the examples below.

Create a movie and add a piece of text called text1, so the movie looks like the one below



Then add a black circle and add the following code to its script

```
on mouseUp me
    member("text1").text = "hello"
end
```

This changes the text to hello when the circle is pressed. Now we will use variables. Create a movie script and add the following code

```
global colourcircle
on changetext
    member("text1").text = colourcircle
end
```

Do not run it at this point, but change the code on the black circle to read the following

```
global colourcircle
on mouseUp me
    colourcircle = "black"
    changetext
end
```

If you run it now, you will see the word changes to black when the circle is pressed.

Now add a green circle on the stage, and add the following code to it

```
global colourcircle
on mouseUp me
    colourcircle = "green"
    changetext
end
```

Run the file, and click the circles. Now add a circle for red and one for blue on.

## ***The Message Window***

The message window is very good at showing you what is happening. Start a new director movie and go to 'Window' and choose 'Message' and the following screen pops up.



This can help you by showing what is happening.

Create a new movie script and type the following code

```
global gNumber

on prepareMovie
    gNumber = 23
end
```

In this movie you can see the global variable has a number put in it and hence is a numeric variable, and uses the preparemovie event.

Now add some frame script to the 1<sup>st</sup> frame, and type in the following.

```
global gNumber
```

```
on exitFrame me  
  put gNumber  
end
```

The word 'put' puts the gNumber into the message window. Run the movie and you should see 23 appear for each time the movie loops.

Go back to the movie script and alter the code as follows

```
global gNumber, gNumTotal  
  
on prepareMovie  
  gNumber = 23  
  gNumTotal = 0  
end
```

You will see you can declare more than one global variable on the same line.

Go back to the frame script, and change it so it reads the following

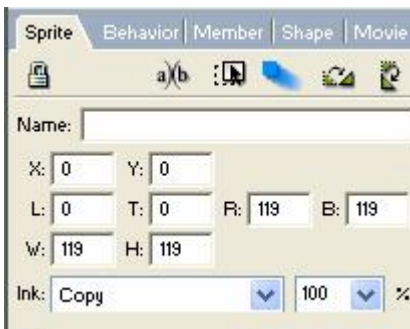
```
global gNumber, gNumberTot  
  
on exitFrame me  
  gNumberTot = gNumberTot + gNumber  
  put gNumberTot  
end
```

Run the movie and see what happens.

## ***Properties***

You can also set the properties of a sprite using Lingo. Start a new movie and add a circle to the stage.

Move it to the top left hand corner of the stage. If you click on the sprite you can see the properties of that sprite similar to the ones shown below



Name the sprite circle1. Place a red square in the other corner and change the script of that square to the following.

```
on mouseUp me  
  sprite("circle1").locH = 24  
end
```

You will see that the circle moves when you click the square. Go back into the code of the square, and change some other properties such as locV, foreColor, blend, blendlevel, width, height, flipH, flipV. Try these plus any others you can find in the director help.

Next try the example below

Create a new director movie and create a new sprite on the stage.

Create a new movie script, and declare 2 global variables gSpeed and gSprite. Add a on prepareMovie handler, where you set gSprite to the number channel the sprite is on, and gSpeed to a number between 1 and 10.

Create a new frame script for frame 1, and again declare the two global variables. Add the following line to your on exitframe handler

```
Sprite(gSprite).locH = gSpeed
```

Test the movie. Now alter the code to read as below.

```
on exitFrame me
  sprite(gSprite).locH = sprite(gSprite).locH + gSpeed
  go to the frame
end
```

Test it again and see what happens. Add another line to change the blend as the sprite moves (HINT: blend = blend - gSpeed)



## If, Then, Else Conditions


Lingo can have conditions set in it such as if, then else. If you think of it in English

```
If I mow the lawn
then I get £20
Else I get nothing
End
```

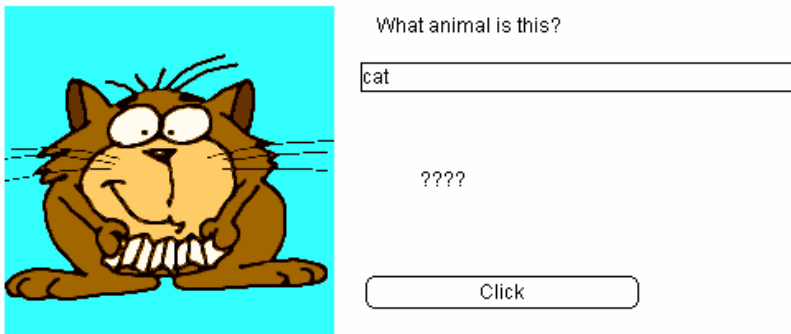
The example below should show how this works.

Create a new director movie, and change the drop down list from 'default' to 'classic' (the button is shown below)



Add a field , and put a border around it. Call the member field1. Make sure that it's editable property is selected.

Add a picture of a cat (or anything else) to the stage. Add a button, so the stage should look something like this.



The ??? text field should be a member called 'answer1'. Add the following code to the button.

```
on mouseUp me
  if member("field1").text = "cat" then
    member("answer1").text = "Correct!"
  else
    member("answer1").text = "Incorrect..try again!"
  end if
end
```

Try the movie.

## Operators

The table below explains the symbols allowed in a if statement in Lingo

=	Equal to
<>	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

## Case Statement

If you have multiple values to compare in a large nested if statement, you may be better using a case statement.

An example of this is shown below. This example also covers swapping of sprite contents and making sprites invisible or visible.

Import 4 pictures into a director movie. I have used a cat, mouse, Tigger and fish.

Place one of these on the stage and call its sprite 'background'.

Add 5 buttons on the screen and label 4 of them the same as the pictures and one called 'waiting' so it looks like the picture below.



Add a movie script to the movie and code it similar to below (depending on your pictures)

```
global gPic

on preparemovie
  sprite("background").visible = FALSE
end

on choosepic
  sprite("background").visible = TRUE
  case gPic of
    "waiting":
      sprite("background").visible = FALSE
    "cat":
      sprite("background").member = "cat"
    "fish":
      sprite("background").member = "fish"
    "mouse":
      sprite("background").member = "mouse"
    "Tigger":
      sprite("background").member = "Tigger"
    otherwise
      nothing
  end case
end
```

Try and figure out what this code is doing.

Now you need to add code to each of the buttons. An example of one buttons code is shown below.

```
global gPic  
  
on mouseUp me  
    gPic = "Tigger"  
    choosepic  
end
```

Code each of the 5 buttons, and test the movie.

# Director Practical

The aim of the practical is to demonstrate the basic use of Macromedia Director. The screens you are going to generate will look like:



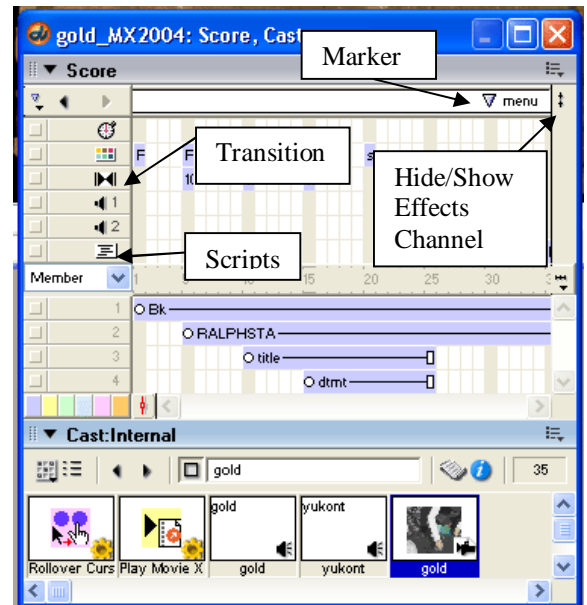
All the files are in the following folder:  
Studentinfo on Trentdev:\DIM\Tutorial Media\gold

**Note Save the Movie after completing each section.**

1. Open Macromedia Director MX 2004.
2. Select Create New Director File.
3. Make sure the following windows are open: **Stage, Score, Property Inspector and Cast.** You will need to use the window menu throughout the task to make these windows visible. (Use **WINDOW** menu to open them)
4. Set the movie stage size to 800x600 (Modify | Movie| Properties) Make sure the Movie tab is selected.

## Importing Cast Members

5. Import all the images you need for the first screen. Use FILE | IMPORT and ADD the 5 files BK.JPG; RALPHSTA.BMP; TITLE.BMP; DTMT.BMP; STAR.BMP They will be added to the CAST window when you press the IMPORT button. Click OK for the dialog box. **Save the movie.** To see the cast members as thumbnails, click on the Cast View Style button.



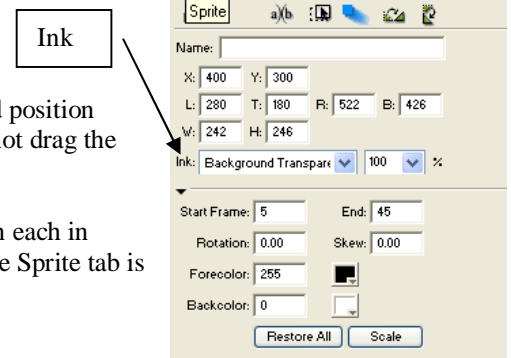
## Adding Cast Members to the Stage

6. Make sure the score window is visible. Drag the background (BK) from the CAST on to CHANNEL 1 FRAME 1. Pull the slider so that it fills frames 1 to 80 or set the end frame in the property inspector.

Put RALPHSTA into CHANNEL 2 FRAMES 5 to 45.  
Put TITLE into CHANNEL 3 FRAMES 10 to 25.  
Put DTMT into CHANNEL 4 FRAMES 15 to 25.  
Put STAR into CHANNEL5 FRAME 20 to 25.

Click in FRAME 25 so that you can see all the images, then select the stage and position them by dragging them to the correct positions (see above). Make sure you do not drag the spot in the centre of the image.

In the score window select the all objects, except the background (Shift and Click on each in turn). Choose Background Transparent from ink in the property inspector. Make sure Sprite tab is chosen)



## Adding Transitions

7. Make sure the effects channels are visible in the score window. Double Click the transition channel on frame 5. Choose Dissolve | Dissolve pixels. Select the same transition for frames 10, 15 and 20 by dragging from the cast. Save your file.

## Playing the Movie

8. Play the movie. Use the rewind and play buttons on the toolbar or use the control menu. The graphics should come on one at a time with a transition effect. If your movie keeps running then turn off **Loop Playback** in the control menu.

## Setting up the Menu Screen

9. Repeat the same as above to set up all the graphics on this screen make sure you set up the EXIT and GOLDEN FACTS buttons especially as you will be scripting these. Each image is labelled by its position. TOP.BMP(Help); TOPRIGHT.BMP(GoldenFacts); MIDRIGHT; BOTRIGHT; BOTTOM; BOTLEFT; MIDDLEFT; TOPLEFT. These images will act as menu options. Import these images. Click in frame 30 in the score then add all images (shift click to select them all) to the stage. Dragging a cast member directly to the stage will also add the cast member to the score. In the score select all the images and set them all to run until frame 35 using the property inspector. Rearrange the buttons on the stage in approximately the right place. Select all the buttons and set Ink | Background Transparent from the score. Save the Movie. Play the Movie.

## Controlling the Movie

10. Double click the script channel in frame 25. Add the script:  

```
on exitFrame me
    delay 2*60
end
```

Close the script box. Delay is specified in terms of *ticks* where 1 tick is 1/60<sup>th</sup> of a second.
11. Open Window | Library Palette. Select Navigation using the pop-up menu button at the top left. Scroll down and drag the **Hold on Current Frame** button to the script channel of Frame 35.  
Play the Movie from the start. You should get a 2 second delay before the menu appears on screen. The Movie also stops on frame 35.

9 uses Lingo to control the Movie and 10 uses a behaviour.

## Exiting the Application

12. Setting up the EXIT button. Select the EXIT (topleft) button in the CAST window and click the script button(top right).  
Add the script:  

```
on mouseUp
    halt
end
```

Save the Movie. Run the Movie and click on EXIT. The application stops. If you use QUIT then the application quits out of the Movie and Director.

## To name a frame

13. Click the markers channel above frame 50. A text insertion point appears which allows you to type in the name of the frame. Name frame 50 "gold facts". Name frame 30 "menu". Note: to delete a marker, drag it out of the channel.

## Setting up "Gold facts" screen

14. Import the following files: TEXTBOX.JPG; GOLDFACT.JPG; MENUBUB.BMP; MENUBTN; DISPRBTN.BMP; DISNBTN.BMP; GFACTS.RTF. Click OK for all the dialog boxes. Click in frame 50 and drag the above on to the stage. Set the end frame to 59  
Position TEXTBOX on the left of the frame; GFACTS.RTF on top of the textbox; GOLDFACT.JPG at top right and MENUBUB;DISPRBTN;DISNBTN and MENUBTN at bottom right (see image on page 1).  
If you cannot see your text or buttons you may need to use Modify | Arrange | Move Backward on the text background and menu bubble. Alternatively lower numbered channels are behind higher numbered channels so import items into the appropriate channel.

Put a pause script in the script channel for frame 50. Drag script for **Hold on Current Frame** from cast window to the script channel of frame 50.

## **Navigation**

You can use behaviours to move to different parts of the movie. To turn a graphic into a button you can drop a behaviour on to the graphic sprite.

### **Navigation buttons**

15. To go to the "Gold facts" marker when the gold facts graphic is clicked on, select **Controls** from the Library Palette. Drag the **Jump to Marker** button on to the **Golden Facts** graphic on the menu frame. In the dialogue box, select **golden facts** from the **Jump to Marker** list.
16. Set up the menu graphic on the Golden Facts page to return to the menu. This time use the Jump Back Button behaviour.
17. Check your buttons work.  
You may need under other circumstances to deselect the **Remember Current Marker** box and set up buttons individually.
18. Set up a marker on frame 60.
19. Set up **Gold Fever** graphic on the menu page to move to frame 60 from the menu screen. Set up frame 60 so that it has the same menu controls as the Gold Facts screen. Set up a hold on current frame behaviour as well. Set up the behaviour for the menu button on frame 60 to **Jump Back Button**. Check that you can move to and fro between these two pages from the menu. Director remembers the last screen because it is set by the **Jump to Marker** button.

### **To attach the same behavior to several sprites**

20. In the score window, select all the menu buttons in frame 30. In the Library Palette select Animation | Interactive | Rollover Cursor Change and drag it onto the selected sprites. Select the Hand cursor from the dialogue box. Run your movie and check that the cursor changes when you move over the buttons.

### **Making your movie modular**

21. Make sure you have saved your movie. Start a new Movie.
22. Put the background in Channel 1 and RALPHSTA in channel 2.
23. Check looping is off. Control | Loop Playback.
24. Save your movie.
25. Open your original movie.
26. Drag the **Play Movie X** behaviour from **Library Palette | Navigation** on to the **Going for Gold** button on the menu screen. Type in the name of your second movie.
27. Save your file and run your movie. When you click the **Going for Gold** button your second movie should run and then return to your menu.

## **Sound**

28. Add the YUKON.WAV sound to your "gold" movie in one of the sound channels.
29. Use the tempo channel (with the clock icon) to wait until for cue point **end** before you move to the next frame.
30. Remove the tempo and use in-between to make the sound play over several frames.
31. Some files do not have cue points so you must inbetween or wait for the correct time

## **Video**

32. Import a video (gold.avi ) and use the tempo channel and in-betweening to play the video as before.