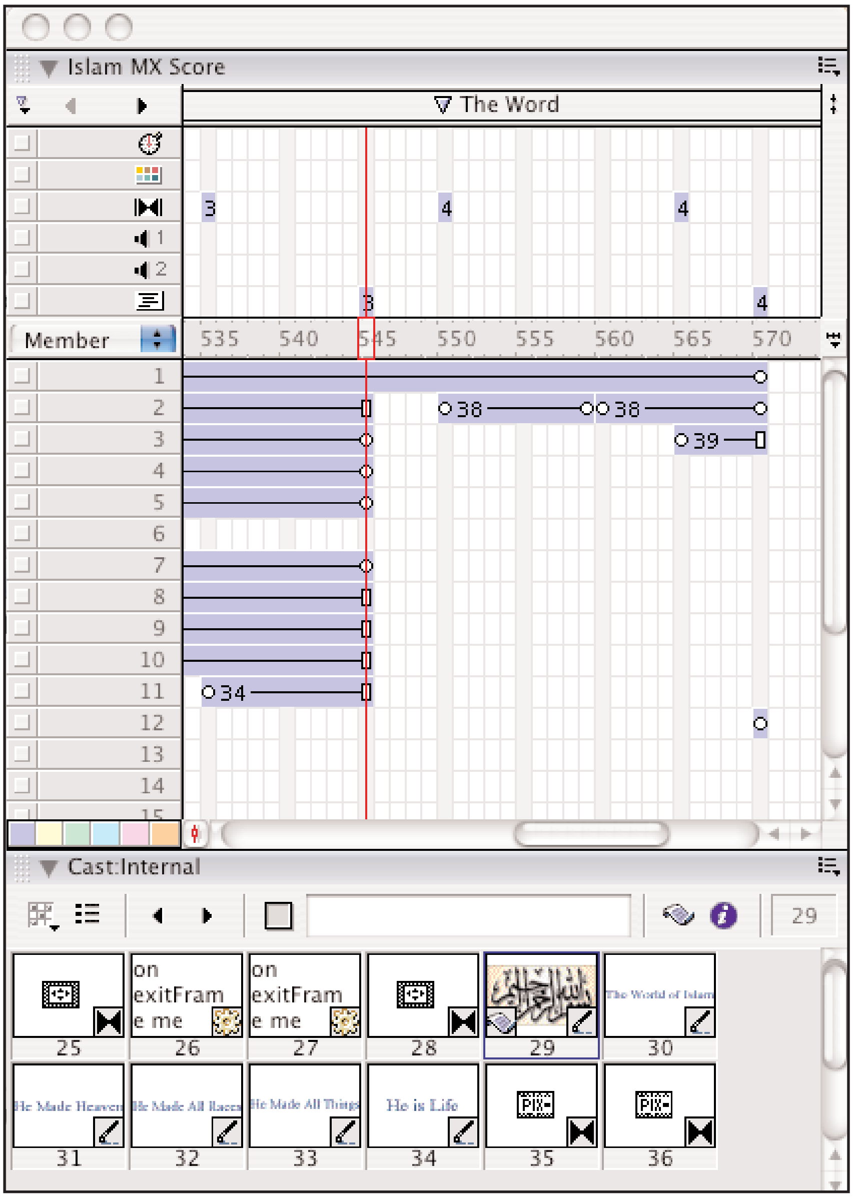
Macromedia **Director** Lingo 1

Director

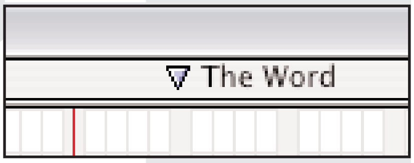
Macromedia Director is basically an animation package. The main feature of the application is the timeline. The timeline is separated into individual cells. Each cell is an individual frame of the animation and could potentially hold one segmented image of the animation. The default timing sequence for Director ie 30 frames per second. This means that 30 frames would take 1 second to play, 60 frames 2 seconds, etc.



Director, however, is much more that an animation application, it has, over the years developed into a full interactive multimedia package. With the inclusion of Lingo, Director's scripting language, it is possible to write a fully integrated and interactive application which is self standing and operates independent to Director.

Lingo

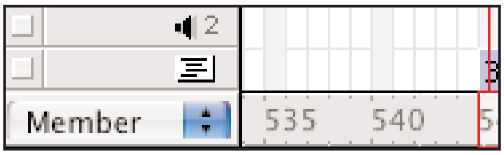
Director, with the 2004 version now allows you to script in either Lingo or JavaScript, depending on the language you feel more easy with.



Markers

To move from one frame in the timeline to another there are two methods The first is to name the frame you are going to, the second is to set up a marker in the marker channel above the timeline. The example above has been named "**The Word**". To set up a marker simply click into the channel and a new marker icon (the triangle) appears with the title "**New Marker**". Click into the title to change it to what ever you wish to call the marker. The markers set the position where something happens or changes within the structure of your work.

The Lingo commands to go to the marker would normally be set into a Cast member - for example a button - or into the scripting channel above the main timeline.



The Scripting Channel

The script command set into the Cast member would be:

```
on mouseUp
   go to “The Word”
end
```

or -

```
on mouseUp
   go to marker “The Word”
end
```

When you are going to a marker name - or:

```
on mouseUp
   go to 30
end
```

or -

```
on mouseUp
   go to frame 30
end
```

When you wish to stipulate a specific frame number.

When the command is set into a cast member the script window automatically writes the “on mouseUp” it sees the Cast member is an active button. If however the command is set into the Scripting channel it would read:

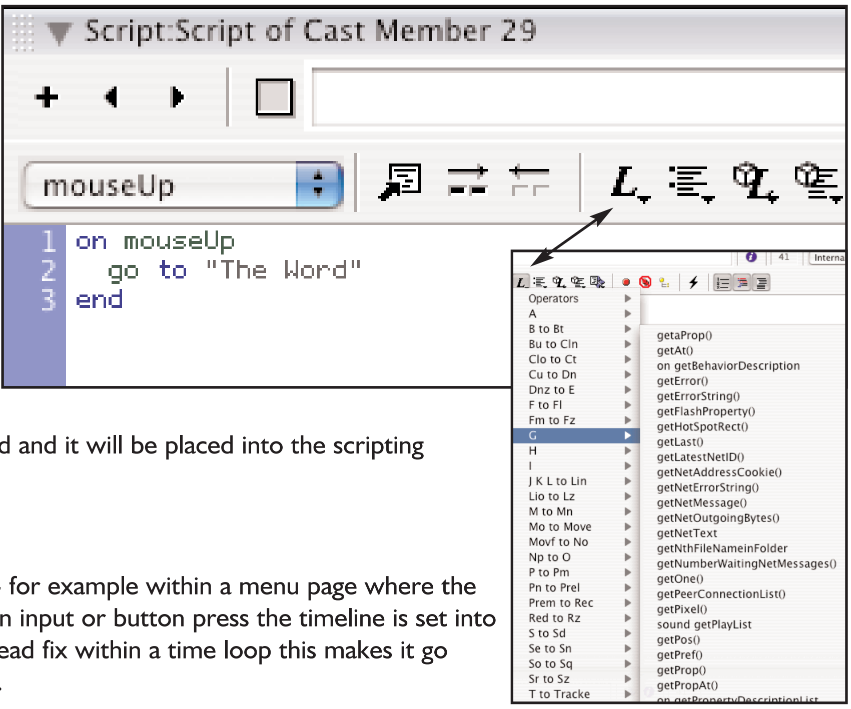
```
on exitFrame me
   go to “The Word”
end
```

Or as above including the words either “**Frame”** or “**Marker”** into the script.

The Script Window

The script window contains all of the facilities to write scripts. If you know the exact syntax and wording of the script you wish to use you can simply type it into the window.

If however you are not sure you can go to the “**L”** icon in the script window menu bar, click into it, a sub menu appears on-screen which gives you all the scripting commands alphabetically. If you wish to use the - “go to” - command scan down to G and highlight the “go to frame” command and it will be placed into the scripting window.

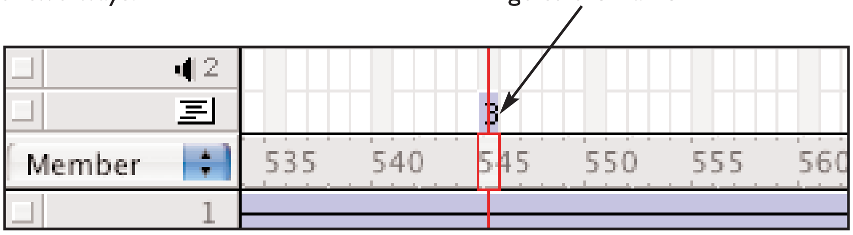


```
on mouseUp
  go to "The Word"
end
```

Loop

If you want to make the timeline active - for example within a menu page where the programme is working, but waiting for an input or button press the timeline is set into a loop. The loop makes the video play head fix within a time loop this makes it go round in a circle until the loop is broken.

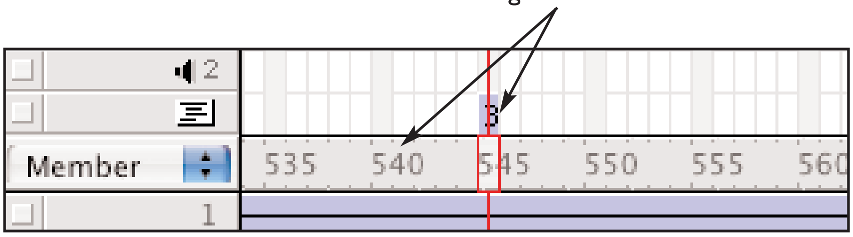
The loop is set into the time line in one of two ways:



```
on exitFrame
    go to the frame
end
```

This command seemingly freezes the video play head on the frame where the command has been set into the scripting channel.

or



```
on exitFrame
    go to frame 540
end
```

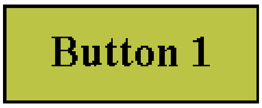
This command is used when the video head reaches the frame where the command has been set into the script channel. The script tells the video head to go back to a frame before the one holding the command (if the command is on 545 you tell the head to go to 540). This sets the head within a loop as above. The programme is now waiting for a mouse click or a stipulated event.

Cast Members and Sprites

A sprite is a cast member which has been placed on the stage (the main working screen). A sprite is made into a working button by putting a script into the cast member. A sprites image can be changed by a “**mouseEnter / mouseLeave”** command. This makes it possible for the button to toggle between two visual states simply by rolling the cursor over the button.

With HCI guidelines, buttons need to give feedback to the user to inform the user that this sprite/button is active. In some cases the button toggles between black and white - or the image changes.

The command is:



Button State 1

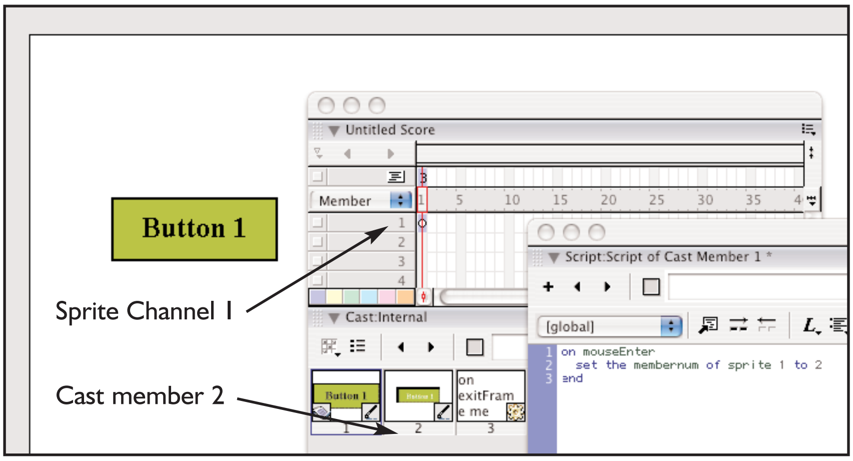
Button State 2

```
on mouseEnter
    set the membernum of sprite 1 to 2
end
```

The sprite image changes as the cursor crosses over it. The changed image state is permanent until a command is set to tell it to change back.

Note here: - “**set the membernum of sprite 1**” refers to the sprite channel “**1**” in which the image is resident, and the “**to 2**” statement refers to the cast member number “**2**” (the image)

When your cursor rolls out of the



```
on mouseEnter
  set the membernum of sprite 1 to 2
end
```

image you need to set another script command (into the image you have changed the sprite to) - so that it reverts back to the original image.
This command is:

```
on mouseLeave
    set the membernum of sprite 1 to 1
end
```

this will change the image back to the original as the cursor leaves the sprite.

The checkCursor Command

In certain circumstances you would use a "**handler event**" to make the application follow your scripts. A handler is similar to a go-subroutine where you set the command word (in this case "**checkCursor**") and then set a global movie script describing the programming to be executed when the command is initiated. The programme looks for the handler, when found it executes the code contained within the handler.

A handler could be used to activate several buttons where for example those buttons activate AVI or QuickTime films used within your work. The scripting could go so:

The command word:

```
on exitFrame me
    checkCursor
end
```

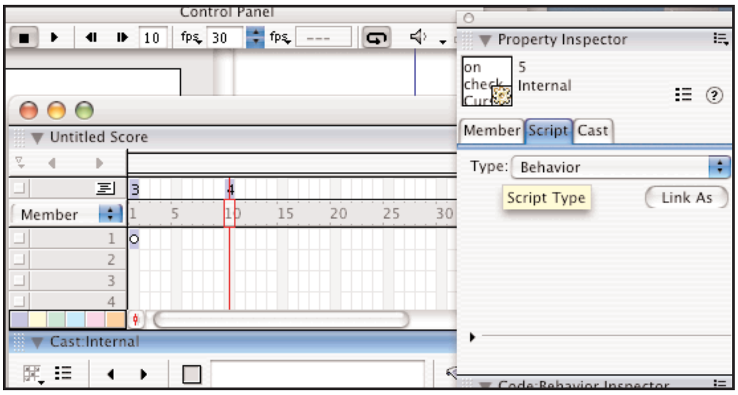
This command in itself means nothing to the programme and would set up an error message if used by itself.

The handler script set into "global" and "Movie Script"

```
on checkCursor
        if rollover(1) then
                go to frame 20 (or marker + name)
        else
                if rollover(2) then
                        go to frame 30 (or marker + name)

                else
                        if rollover(3) then
                                go to frame 40 (or marker + name)
                        end if
                end if
        end if
end checkCursor
```

Remember ! This command must be set as a movie script by selecting the **Cast member** then in the **Property Inspector** - **Script** - and change the **Type** from "**Behaviour**" to "**Movie**". When you run the script it will now work.


Control Panel
10
fps 30
fps ---
Untitled Score
Member
1 5 10 15 20 25 30
1
2
3
4
Cast:Internal
Property Inspector
on check Curs
5
Internal
Member
Script
Cast
Type: Behavior
Script Type
Link As
Code:Behavior Inspector