# Director Manual

**Æliens**

**eliens@cs.vu.nl**

original design: H. vd Lubbe
accompanying: A (not so) gentle introduction to multimedia

draft version 0.5 (12/3/2001)

# Introduction

This manual includes a description of the different user functions, that is the different windows (eg. Control Panel and Cast window) which can be opened in Director. Also the basic concepts Director uses are explained. The different tools (like Paint en Text) are described. It is important to study this pages carefully, before starting with the actual assignments (you could open Director also, to get a better view on the things explained.

Another thing, which is quite important, is the HOWTO's page. Here are practical examples of certain things (eg. let text zoom in the screen or dynamically add a background), that can be done with Director. These can of course easily be integrated with the assignments. We think the students can benefit from these examples, so try them.

For eager students, who would like to move on, during and after the assignments, with Director, we've included a section about the 'programming'-language behind Director, Lingo. Also there is a so called 'advanced' section, with advanced examples and concepts.

The last page is reserved for pictures, which can be used for the presentations that are to be made.

# Windows

### The Stage window

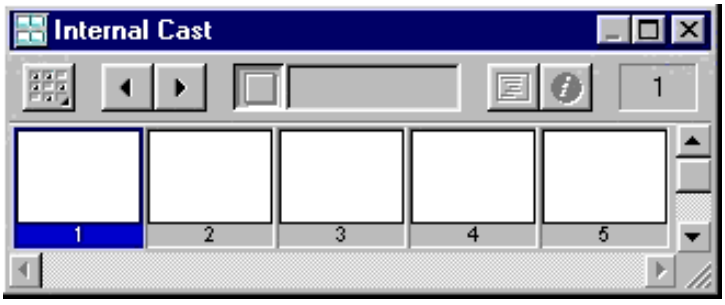
The central empty window is the Stage window, where the end result of your work is displayed. It's the screen on which Director movies are projected.

If you create a self-running piece of software, the Stage window is the universe in which that software will exist. Quite literally, this is where the action is.

What can you do to the Stage? You can change its color and size, and you can reconfigure it in a number of ways to suit the needs of a project. You can change the Stage size of any movie at any time, but each movie can have only one Stage configuration. If you want your project to incorporate Stages of different shapes and sizes, you'll need to create several movies and link them together.

### The Cast window

The theatrical metaphor that dubs director's playback screen the Stage continues with the Cast window. Actually, it is here that the metaphor begins to break down: the Cast might more accurately be called the Cast/Scenery/Props/Musical Instrument Department.




Essentially, everything that goes into a multimedia production can reside in the cast:

cast members

- still graphics (artwork/photo scans)
- sounds (in digitized form)

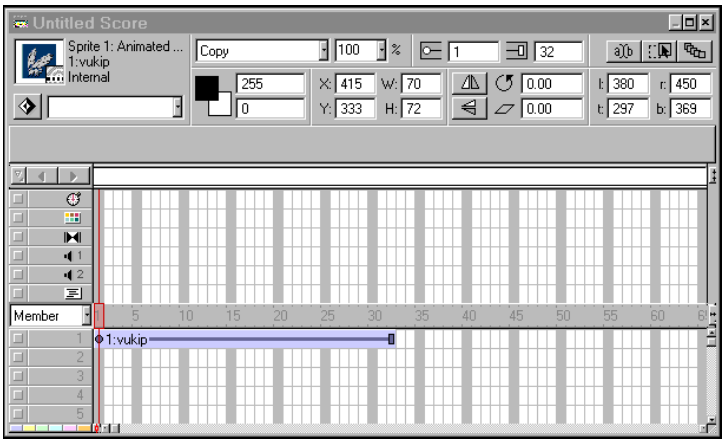
- interface elements (such as buttons and icons)
- text
- digital video (quicktime/AVI movies)
- animation (gifs/Director film loops)
- palettes (certain group of colours)
- other director movies
- scripts/behaviours.

All cast members come into being in one of two ways: You can create them directly in Director, or you can import them from documents by other applications. In either case, unless you specify otherwise, each is automatically assigned a location in the Cast database and given a cast member number (you can also give them names if you like). Once in a Cast, a cast member can be cut, copied, pasted, deleted, relocated, and modified. Any changes made to a cast member are automatically reflected in that cast member's appearance on the Stage. This means, for example, that if a cast member 12 was originally a blue dot and you change it to a red one, all instances of the dot in the Director movie will be changed from blue to red.

### The Score window

The score window is where the whole project really takes shape. When Director runs a movie, all it's doing –for the most part– is interpreting the information in the Score and whisking elements on and off the Stage accordingly.

It is possible to create commands that overrule Score information (Lingo).



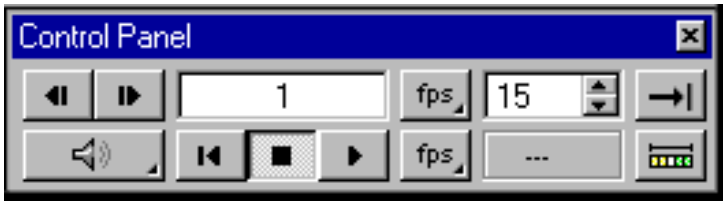

As you can see, the Score resembles a spreadsheet with lots of individual cells divided into rows and colums. The rows are called channels, the colums are called

frames. Each column has a number associated with it, and each channel begins with either a number or a distinctive icon.

Score information is organized in a strictly linear fashion, even when the project is a nonlinear interactive movie. Each frame maps out a certain instance in time during the planned playback; it's not a *specific* time but a *relative* one. For instance, frame 15 isn't necessarily 15 seconds in your movie, and it doesn't necessarily represent 1 second in Stage time. Frame 15 is simply a set of instructions Director should place on the Stage before frame 16, but after frame 14.

### The Control Panel

To the new user, the Control Panel seems like a mix of the obvious and the arcane, If you've ever operated a cassette or videotape deck, the purpose of the main arrow buttons are clear. But what about the other items?




Most important is that the Control Panel shows the frames per second (fps) rate. The top one is the predefined fps, the bottom one shows the actual tempo. This is important for online movies of course, but not for this practical assignment.

# Basic Concepts

**Sprites**

You could conceivably make dozens of different incarnations of one cast member, each for example a different size or place on the stage, each with its own score channel. these incarnations are called sprites, and they're the basic building blocks of Director animation.

Understanding the role of sprites (and their potential) is a key step in working with Director.

You'll create a sense of animation either by changing the sprite position from cell to cell or by switching one sprite for another. Sprites are usually refered to by the number of the channel they occupy. Changes to the sprite don't affect its source cast member, but changes made to that cast member will be reflected in all sprites derived from it.

**Registration point**

When you place a sprite on the Stage by dragging a cast member to the Score, that sprite appears centered on the Stage. To achieve that nice balanced effect, Director has to calculate the physical center of the sprite as well as of the Stage. That is why every graphic cast member has a registration point, which you can see in the Paint window.

If you push the registration point button, crosshairs appear to indicate where the registration point is. Just drag the crosshairs to a desired point. This point will be the center of the cast member.

Alway remember when you're working on the Cast level, keep in mind that your changes here affect all sprites!

**Keyframing and auto-tweening**

Keyframing makes it easy to designate any point in a sprite segment as a keyframe. Auto-tweening enables Director to automatically *"tween"* (as in 'in between') the cells between keyframes to show any movement that should occur. In other words, with auto-tweening you can lengthen or shorten a sprite segment, and the motion of sprites within that segment will be compressed or extended accordingly.

**Space to Time animation**

With Space to Time you place all the necessary sprites in a single Score frame, arranging them in the form of the action. Once you're satisfied with the flow of the sequence, a single command converts the arrangement into a segment suitable for playback.

**Reversing sequence**

The reverse sequence command will retain all of our sprite placement information but will simply flip the order within the segment.

Reverse sequence is a good tool for orchestrating exits: you can animate a cast member's entrance onto the Stage, use In-between to keep the cast member steady for any duration, and then paste the entrance animation and select Reverse Sequence to make the exit.

**Switching cast members**

You can switch cast members corresponding to any sprite while retaining that sprite's placement information. Applying such a substitution to a segment of sprites effectively gives you the power to save the motion while changing the image.

Exchange Cast Members is an especially powerful tool, as it can allow you to build your movie first and then refine the graphic elements later.

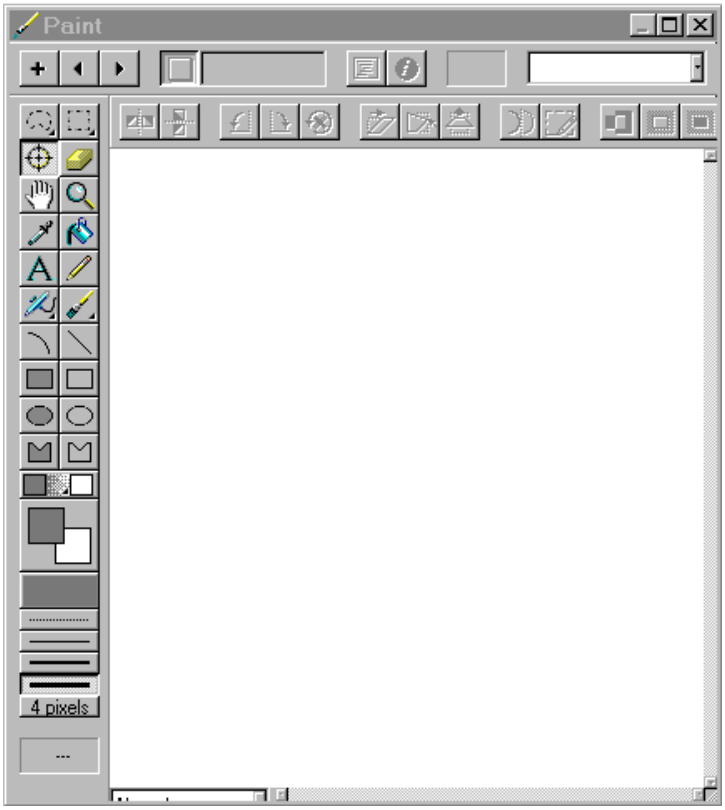
**Ink effects**

We've already established that each sprite is an individual copy of a cast member. Well, ink effects can change the *nature* of that copy by dictating how it's drawn on the screen. The various "inks" are actually modes of display; some change the sprite's appearance radically, while others make subtle changes that show up only when one sprite interacts with another.

# Tools

**The Paint window**

The Paint window has some truly impressive features (intelligent lasso's/ink effects), but in terms of sheer flexibility, it's not in the same league as standalone applications such as Photoshop or Corel Draw. You can use it to create original artwork, but depending on your desired level of sophistication, you may find yourself choosing to create graphics elsewhere and then importing them.

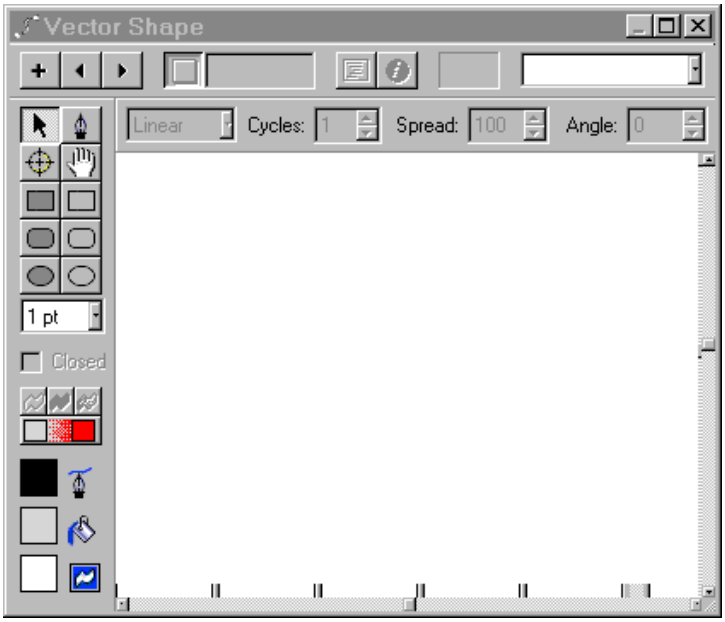



One really convenient feature of Director is that it lets you specify an external

application (i.e. Photoshop) to edit a bitmap, digital video, or sound cast member. From the File menu, choose Preferences and then Editors to specify your choice for each cast member type. When you double-click a cast member, the chosen program will now automatically be launched.

**The Vector Shape window**

To understand the difference between bitmapped graphics and vector shape graphics, imagine a simple line: You can either physically draw the line on a piece of paper (bitmaps), or you can jot down the coordinates of point A (the starting point) and point B (the ending point). Vectors take the latter approach, juggling the *mathematical representation* of a line rather than the line itself. Although the computer screen will display an actual line in either case, the vector format is a lot more versatile.

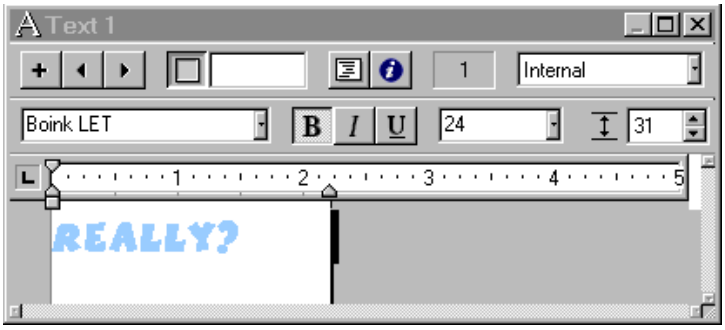



A vector's mathematical description isn't just length and shape; it also includes fill color and thickness of the line. This makes it possible to resize them without introducing any distortion. Moreover, they require a lot less disk and memory space and will download and animate much faster when used on the Web.

**The Text window**

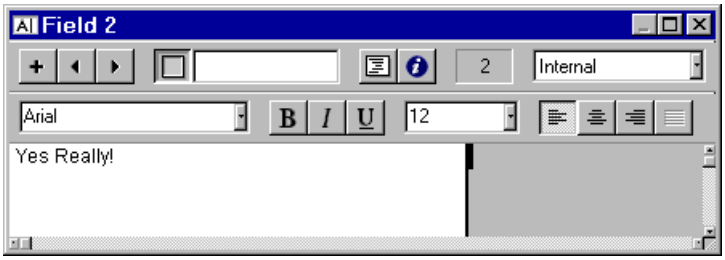
Along with specifying font, size, and style, you can set tabs, margins, leading (line spacing), and kerning (letter spacing) values. What's especially impressive is Director's ability to display text on the screen in anti-aliased form, which means the characters are smooth edged rather than jagged. and it's all done in the Text window. You can type directly in the Text window, or you can import files saved as plaint text (ASCII), saved in Riched Text Format (RTF), or saved as

HTML documents. RTF is supported by many word processing applications and preserves some formatting niceties. Director 7 recognizes most HTML tags.




### The Field window

Text is handled slightly differently in the Field window. there are no tabs, paragraph formats or typographic controls, and when placed on the stage, a field cast members won't animate as quickly as a text cast member. The main advantage of fields is that they take up a lot less space.




### The Tool Palette

The Tool Palette is designed to deposit its creations directly on the Stage. If you use Tool Palette's Text Tool, your subsequent typing will make an automatic entry in the Text window. Click the field tool, and the text will go into a Field window instead.


Some of the tools in the Tool Palette look identical tro those in the Paint window, but there's an important difference: lines and shapes created here are storen in a form that takes up less file space than bitmapped graphics, and they're easily modifiable at any point after their creation.

The tool Palette's other strength is its button-creation function. if you want to create buttons with built-in animations that underscore their 'button-ness' this is the source. But keep in mind, that anything residing on the Stage can be turned into a button.

**The Color Palettes window**

You can edit the colors of a palette in two ways. the first is by selecting the color and then using the hue, saturation and brightness arrows to modify the color. This doesn't let you see the results of your actions, so here's a better method.

It's easier to import a piece of artwork with the color values you want to use. You can then import not only the artwork but its palette, which Director will display in the Color Palettes window.

**The Digital Video window**

Director movies can incorporate QuickTime movies and AVI movies, and you can even export a Director movie as a digital movie in either format.

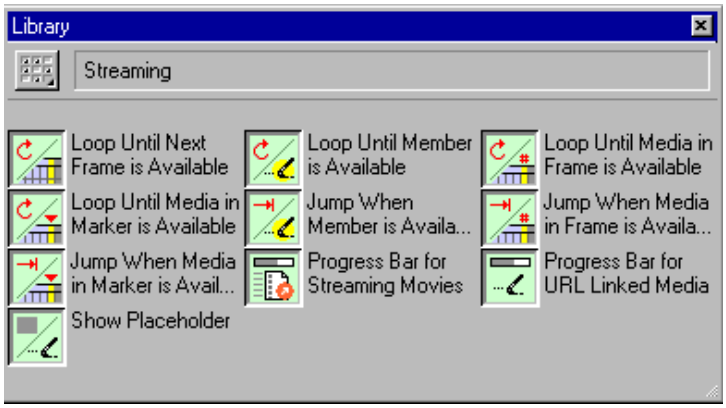
When you import a digital video file into Director it takes up residence in the Cast window and can be viewed in the QuickTime or Avi Video window. The logistics of a movie within a movie can get pretty thorny, especially since both can have independent playback rates.

**The Script window**

For scripting, look at the advanced pages.

**The Library Palette**

Director 7 has several features designed to encourage you to use readymade behaviours whenever possible, and one way this shows up in the interface is the Library Palette, which lets you drag behaviours right onto your sprites or into a frame in the behaviour channel.




Director will walk you through the process of supplying any required extra information for the behaviour. you can now include some pretty impressive behaviours for creating animation, navigation, user interfaces, and even an analogical clock.

# LINGO

In Director it is possible to manually program almost anything Director can do. The scripting language used in Director is Lingo. Lingo is a basic scripting language and is easy to master.

**How does Lingo work?**

First thing to know is that there are different types of Lingo-scripts:

- movie scripts,
- score scripts, and
- parent scripts.

The first kind of script, the movie script can be called to the screen pressing CRTL+SHIFT+U, or select from the menu.(!) You will see a completely empty text screen and it is up to you to fill it!

handlers

In Lingo you can define what are called 'handlers', these are actually functions and are similar like for example Javascript. An example of a Movie-script with two handlers is:

```
on startMovie
  alert "Multimedia is leuk!"
end startMovie

on stopMovie
  alert "bye bye!"
end
```

**Note:** Lingo is a case-insensitive language, so don't worry about those capital letters, although good programming would be making consistent and clear use of capital letters!

A handler always begins with on and then follows the 'event'. In this example two handlers are programmed, the things that must be done when the movie

starts and the things that must be done when the movie ends. A handler always closes with the word end.

**Note:** Just 'end' at the end of a handler is sufficient although you can also write 'end [handler]' which is easy when your handler becomes somewhat big.

**Note:** There are a lot of system-handlers like the ones in the example above but you can also write your own.

In this example I have issued the command 'alert [string]' which gives a popup-box with the string displayed in it (very useful for debugging your code!). Another way to debug your code is too issue some 'put ¡string¿' commands. In this case the message is send too the messenger window which can be shown on CRTL+M.

Another movie handler is 'on prepareMovie' which is executed before 'on startMovie' and is useful for setting starting-variables.

To create a variable that is accessible for more handlers, you must create a 'global' variable. You can define a global variable anywhere outside any handler in which case it is accessible by all handlers in the same script and handlers in other scripts that include the variable. Or you can define a global variable inside a script. In this case the variable is only accessible for that handler and any other handler that also defines the variable.

**Example:**

global variables

```
global startText

on startMovie
  set startText
  to "Multimedia is leuk!"
  alert startText
  global endText
  set endText = "bye bye!"
end startMovie

on stopMovie
  global endText
  alert endText
end
```

This example is actually the same as the example before except thhat the text is now stored in two global variables. Variables in Lingo do not have a type specification, the variable gets it's type when it is first set or when it is reset. Too assign a value to a variable the following notations can be used: 'set [variable_name] to [some_string_or_other_value]' or 'set [var] = [some]' for those of you who dislike programming with a lot of words.

When a variable is reset the type is also reset, so the following code will give an error:

```
set myVariable to 2  -- integer
alert myVariable     -- error! for 2 is not an integer
```

Correct would be:

```
set myVariable to 2  -- string
alert myVariable     -- correct! since myVariable is a string
```

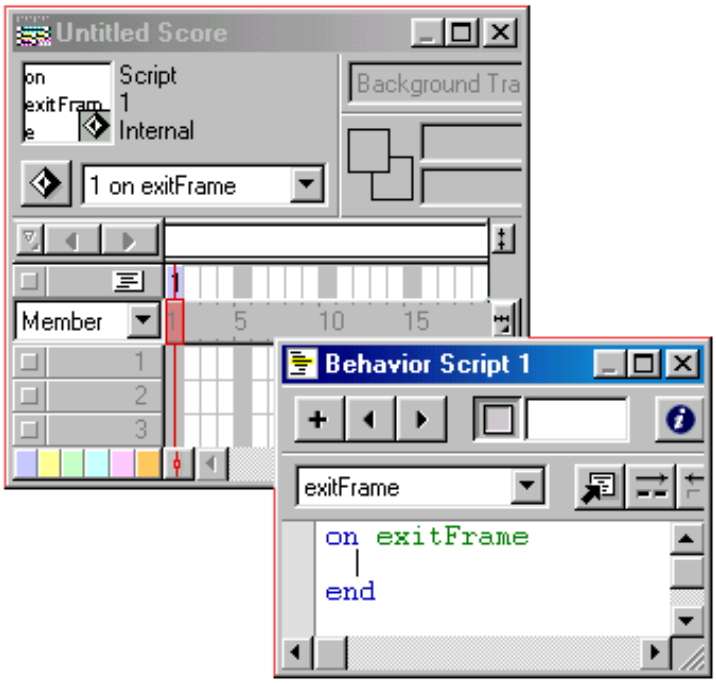
or

```
set myVariable to '2'
-- integer
alert string(myVariable) -- correct! since now the value 2 will
 be cast to a string
```

Lingo also has a lot of system variables which can be used at any time. System variables often consist of the word 'the' followed by the variable. this can be confusing when debugging if you forget the word 'the' since Lingo will create a new local variable. Some System variables are 'the mouseV' (vertical position of the mouse-pointer), 'the mouseH' (horizontal position of the mouse-pointer), 'the shiftDown' (boolean presenting wheter the 'shift' key is pressed or not), 'the key' (the last key pressed) etc One very important system-variable is 'the frame' which denotes the current frame of the movie and is much used in frame-scripts.

In addition to movie-scripts you can also use score-scripts. These are either frame-scripts, sprite-scripts or behaviors (which are actually sprite or score-scripts but more on bahaviours later).

Frame-scripts are defined in the first line of the upper-part of the score and can be accessed by double-clicking on a cell.




In a frame-script code is programmed which will only be executed when the play-back-head of the movie is at the frame. Handlers that can be used in a frame script are: 'on prepareFrame', 'on enterFrame', 'on exitFrame' and there effects are obvious.

**Note:** You can also use these handlers in movie-scripts accept that then they count for EVERY frame in your movie!

**Example:**

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

or

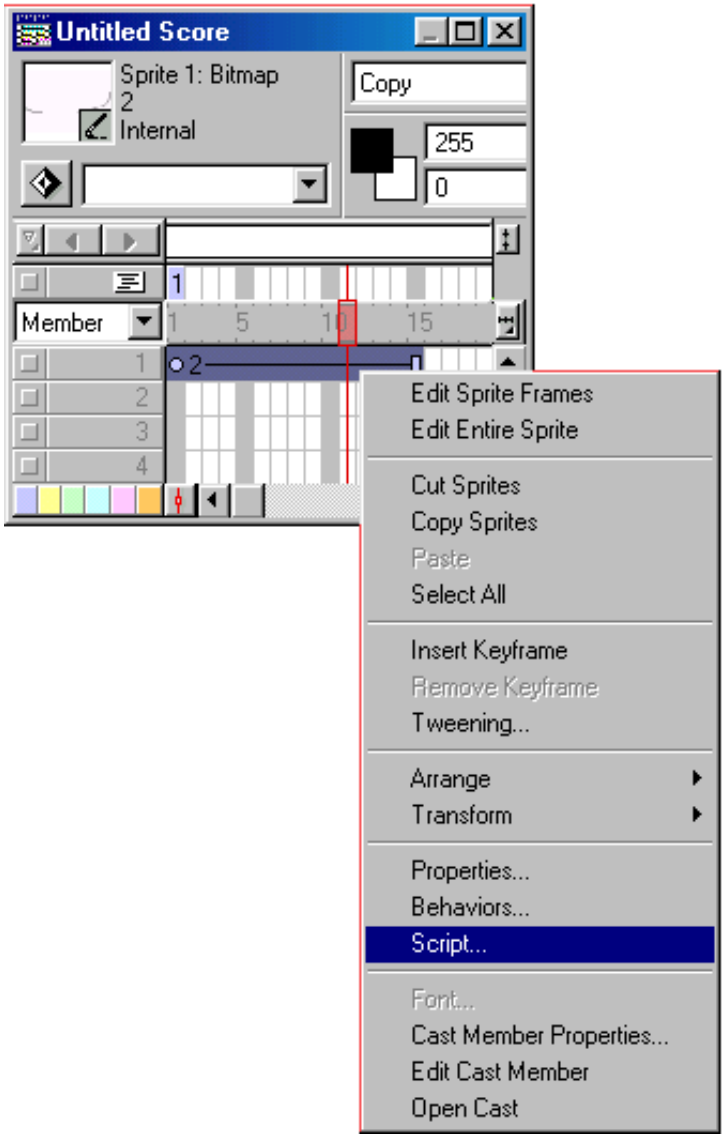
```
on exitFrame
  go to marker("the end")
end exitFrame
```

In both examples the 'go to' command is issued. This command expects a integer value which represent a frame (the first frame is number 1 and so on). In the first example the go to commands goes to 'the frame' which is the frame the

script is in so the play-back head enters the same frame again and thus a loop is created! In the second example the play-back head goes to the frame where the marker "the end" is defined.

A sprite script is also a score-script. Sprite-scripts can be called from the menu at sprites or cast members.

sprite script




Sprites can have multiple scripts on them as long as there aren't more then one of each handler. Hanlders often used in sprite scripts are: 'on beginSprite',

'on endSprite', 'on mouseOver', 'on mouseLeave', 'on mouseDown', 'on mouseUp' etc.

**Example:**

sprite script

```
on mouseUp
  alert nextSentence()
end mouseUp

on nextSentence
   set myNounList to ["Multimedia", "Assisting",
       "College", "The manual"]
   set myVerbList to ["stays", "is, "is not",
       "continues to be", "will be"]
   set myProverbList to ["stupid.", "fun.",
       "exhilarating.", "too great too put in words."']
   set myRandomSentence to
            myNounList[random(count(myNounList))] &&
            myVerbList[random(count(myVerbList))] &&
            myProverbList[random(count(myProverbList))]
return myRandomSentence
end nextSentence
```

**Note:** I've written a small handler in this sprite script which generates random sentences from the given lists.

**Note:** To let a handler return some value just put a return command somewhere along with the variable too be returned.

**Note:** When a return command is issued the rest of the script is NOT omitted. The same holds for the 'go to'-command and others.

You can also alter your sprites and cast members run-time with Lingo. To select a cast member type 'member "[yourMember]" of castlib "[yourCastLib]" or member [A_Number]', to select a sprite just use 'sprite [yourSpriteNumber]'. You only have to specify your castlib if it isn't the internal castlib. You can either reference cast members by there name (string!) or by there place in the castlib (int!).

Sprite-scripts are often referred to as behaviors. In effect these scripts do not differ accept that behaviors can be attached to multiple sprites without being copied. So if the behviour script changes the script on all sprites with the behavior changes. In contradiction sprite-scripts are private too the sprite.

*Reference*: To know more about lists in Lingo choose help lingo dictionary Lists in Director.

*Reference*: To know more about cast members in Lingo choose help lingo dictionary Cast members in Director.

*Reference*: To know more about sprites in Lingo choose help lingo dictionary sprites in Director.

# Howtos

howto

- How to make a transition effect
- How to add a keyframe
- Tweening
- How to let text zoom into screen
- How to make a background transparent
- How to create a repetition
- How to pause your movie

### Transition effects

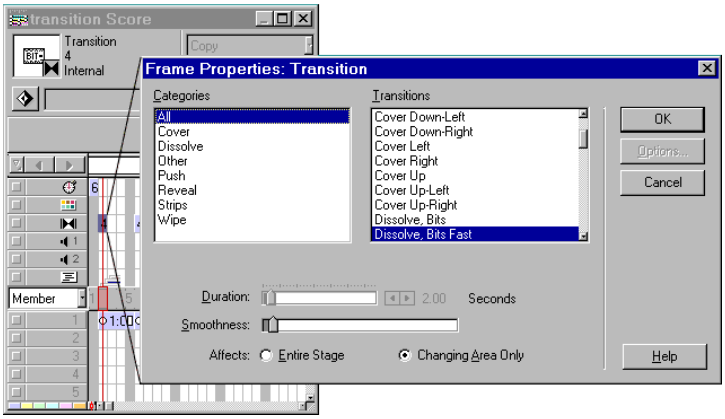
**Name**
How to make transition effects

**Purpose**
A transition effect is used to gradually go from one frame to another. (eg. one frame slides over one another or pixel per pixel etc..)

**Use**

1. In the transition channel, double-click on the frame where you want the transition to occur to display the dialog box.
2. In the categories list, select a categorie (eg. All), and in the transition list, the desired transition.
3. Click OK.
4. In addition the transition becomes a member of the Cast.

**Look at**




**Example**
transition

**Reference**
Macromedia director7 and Lingo authorized, Phil Gross,p.100.

## Keyframes

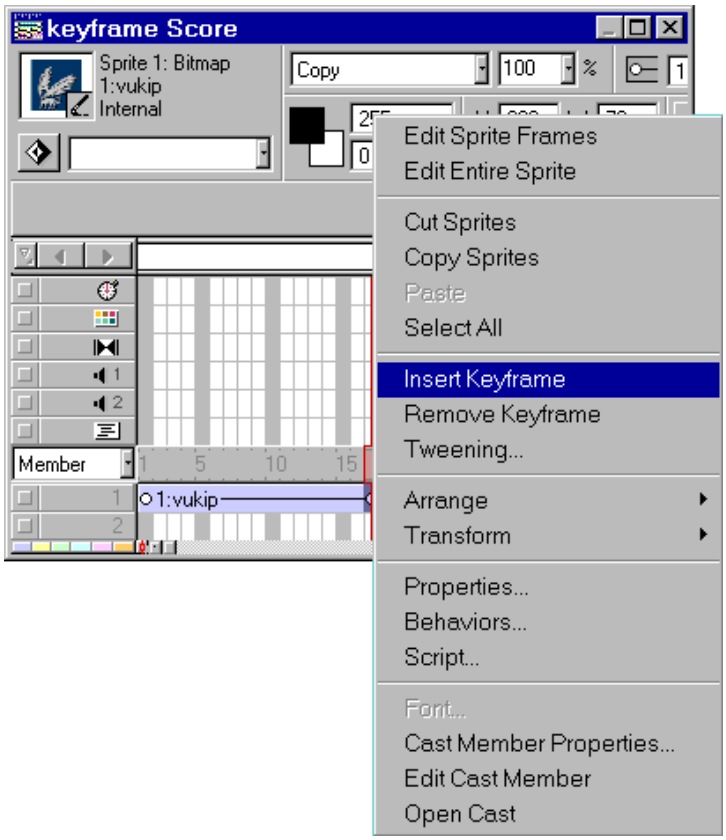
**Name**
How to add a keyframe

**Purpose**
Keyframes can be used as reference points for tweening.

**Use**

1. Select a sprite in the score.
2. Right-click on the frame in that sprite where you want a have a keyframe.
3. Select 'insert keyframe'.
4. The sprite will now show that is has a keyframe.

**Look at**




**Example**
keyframe

**Reference**
Macromedia director7 and Lingo authorized, Phil Gross,p.60.

**Tweening**

**Name**
How to do tweening

**Purpose**
Tweening can be used to animate sprites.

**Use**

1. Select a keyframe in the score. (the start of a sprite is also a keyframe).
2. Go to the stage and drag the selected sprite frame to a desired point in the stage.
3. Playback and watch
4. Hint: holding down SHIFT while dragging will keep the spriteframes aligned!

**Example**
tweening

**Set backs**
Working with keyframes and tweening may be rather confusing. You should save your work a lot, because you could end up dragging the wrong keyframes and making a mess of your whole movie.

**Reference**
Macromedia Director7 and Lingo authorized, Phil Gross,p.58-62.

### Zooming text

**Name**
How to let text zoom into screen

**Purpose**
The purpose is quite obvious. It's could look really cool!!

**Use**
There are at least two ways of letting text zoom in. One way is to make several text objects one larger than the other, and place them in sequence on the Stage. But an easier and prettier solution is to make a text bitmap with the paint window or, even better, an external editor such as Adobe Photoshop.

**how to use a bitmap:**

1. Make a bitmap. This bitmap has to be the largest enhancement of your zoom-sequence

2. Place it on the Stage

3. Click on the first sprite of the segment (be sure it is a keyframe).

4. Grab a corner of the object and, while holding down the CRTL and SHIFT key (to keep the proportions accurate), make the object smaller.

5. Now drag the reference point of the first sprite precisely over the last one

6. Playback.

**Example**
zoomin.dcr

**Set backs**
One setback is, that text can't be used to zoom, you have to make seperate cast members and place them one after another one the Score. Or you can use a bitmap, but the anti-aliasing doesnt work that smooth in director, so again you may want to use seperate cast members.

## Backgrounds

**Name**
How to make the background of an imported picture transparent

**Purpose**
First of all it's just prettier. But also, it would be nice, that if two images overlap only the images do so and not also the background of these images...

**Use**

1. select the desired sprite in the score

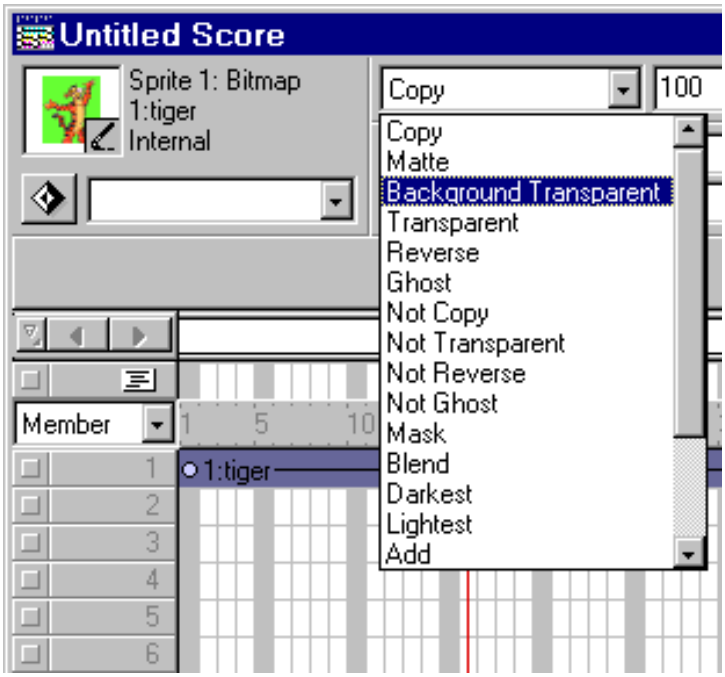
2. select the choice 'background transparent' from the ink-effects

3. double-click on the background-color indicator

4. select the color that has to be transparent

5. you're done! (Look at the Stage to see the changes)

**Look at**




## Pause a movie

**Name**

How to pause your movie

**Purpose**

For the practical assignment, you won't have to make an interactive movie, but it may come in handy to pause your movie. And it's really quite simple

**Use**

1. Double click in the tempo channel on the sprite where you want the pause to occur.

2. Here you can choose either how long a pause you want to have or a pause until the user presses a mouse or keyboard button.

3. Choose the second one.

4. Play the movie and see how the movie pause and waits for a button to be pressed.

# Behaviors

Behaviors are most of the time little scripts which can be added easily to a sprite or frame to increase the functionality.

There are a lot of behaviors out there, either in director like 'mouse-over-change-cursor' or on the internet:

- http://www.behaviors.com,
- http://www.director-online.com.

As mentioned before, behaviors can have properties. These properties are declared outside the hanlders at the top of the screen. When a behavior is dragged from the library palette and released on a sprite a pop-up box can appear to set these properties to specific values for that sprite. The handlers which makes this possible is `on getPropertyDescriptionList`. In this handler you can define what and how properties are set.

**Example:** properties

```
property anInteger

on getPropertyDescriptionList
  set thePropertyListDescription = [:]
  addProp thePropertyDescriptionList,
  #anInteger, [ #default: 0, #format: #integer, #range:
  [ #min: 0, #max: 10], #comment:'Enter an integer']
  return thePropertyDescriptionList
end
```

The property anInteger gets filled in by the method specified by the `addProp` command. In this case a slider is used with possible values 0 to 10. The handler `on getBehaviorDescription` gives a description of the behavior when the mouse roll overs the behavior in the library cast.

**Note:** To add a behavior to the library, add or create a new external cast and put the cast in the libs folder of the macromedia directory.

**Note:** To view the library choose window library palette.
Next to do is just write the behavior.

**Example:** behavior

```
                                                    -- mouseOver
property pMouseOverCursor
property pEnteredMouseCursor

on getPropertyDescriptionList
  set theDescriptionList = [:]
  addProp theDescriptionList, #pMouseOverCursor,
    [ #default: 0, #format: #cursor, #comment: "de cursor"]
  return theDescriptionList
end
```

example (cont'nd)

```
on mouseEnter me
  mySprite = sprite (me.spriteNum)
  pEnteredMouseCursor = mySprite.cursor
  cursor pMouseOverCursor
end

on mouseLeave
  cursor pEnteredMouseCursor
end

on getBehaviorDescription
  set description = "defines de mouse over cursor"
  return description
end
```

This behavior let's you select a cursor which appears when the mouse enters the sprite and turns the mouse to it's normal cursor when the mouse leaves the sprite.

**Note:** To make behaviors fast, you can use a program which makes the basic behavior for you in which you only have to type the specific code. Such a program is available at. (behavior_writer).

# Director Overview

four main windows

- Stage – where the action is
- Cast – that play their role(s)
- Score – for the director
- Control Panel – for playback

movies and actions

- Director files are called movies.
- All actions in a Director movie take place in the Stage window.
- An action is created in the Score window. The process of experiencing the action is known as playback; clicking the Play button on the Control Panel sends a cursor known as the playback head through the Score, which effectively runs your movie by sequentially processing the Score information.

cast members

- The multimedia elements are stored and accessed in the Cast window. They are known as cast members. Some cast members are embedded, which means that they reside entirely in the director movie. Others are linked, which means that the actual data remains in an external file.

the score – timing and channels

- Each column in the Score represents a relative moment in time. Cast members placed in cells in a column will show up on the Stage at that given moment during playback.
- Each channel in the Score represents a layer on the screen. There are specialized channel for sounds, transitions, tempos, and color palettes.

the cast – members and sprites

- Onscreen text can be created in three different ways: as formatted text, as a field, and as artwork.

- The distinction among cast members, sprites, and sprite segments: Cast members are media elements in a Cast database, sprites are their representations on the Stage, and sprite segments are those representations in sequence across the timeline of the Score.



- The registration point is what Director considers the physical center of a bitmap cast member. You can relocate the point in the Paint window to affect where the cast member appears on the Stage.
- Each physical element placed on the Stage has a bounding box, a square area that's as large as the outer perimeter of the object itself. It's always a rectangle, and it appears only when a sprite is selected. You can change the proportions of the box to change the appearance of a sprite, without changing the original cast member from which the sprite is derived.



- The main method of animation in Director is step recording. It involves placing the same cast member in multiple Score frames, in slightly different positions.
- Step recording is made somewhat easier with auto-animation tool such as Extend Sprite, Sprite tweening, and Space to Time.
- Other ways to introduce a sense of animation are through the use of ink effects to vary modes of display for individual sprites, and by modifying the size of the sprites.



- Sprite sequences can be saved and reused as a unit, known as a film loop.
- For straight lines and shapes (as well as buttons), the Tool Palette is as alternative to the Paint window.
- When inporting new cast members from external files, make sure the pop-up menu in the Import window is set to the type of file you're looking for. Otherwise, the file may not appear in the list.
- To swap in a new cast members while retaining the Stage placement of the old one, use the Exchange Cast Members command.
- Markers are useful tools for identifying and navigating the frames in your movie.



- The tempo channel can be made to repeat indefinetely by enabling the Loop option in the Info window.
- Transitions can apply either to the overall Stage or only to those elements that are different in the two frames straddled by the transition.

# Resources

**A small database of pictures**

- photo's
- air
- drawings

**Resources**

- Amsterdam Monumenten
- Amsterdam - The Channels