

**BOKA EDUARD ZORAN**



- SwishMax presentation
- Using effects in SwishMax
- Introduction to Swish Script
- Integrating SwishMax with PHP
- Creating web pages in SwishMax
- Creating games in SwishMax
- Tips and Tricks
- Three free templates with explanation
- 100+ examples for a quicker learning
- 100+ images for a better explanation

2004

## Foreword

Thank you for purchasing **SwishMax unleashed**, the ultimate guide into the wonderful world of SwishMax.

My name is Eduard Boka, I live in a country named Romania, in a city called Timisoara and I am 23 years old. I like working with SwishMax a lot. I used Swish 2.0 too. The facilities of this little program are huge. I have my own freelance company, called Idea Designs, and the current webpage is: [www.IdeaDesigns.as.ro](http://www.IdeaDesigns.as.ro) .

I am a manager and moderator on Swish DB, one of the largest Swish communities on the internet. You can visit Swish DB: [www.Swish-DB.com](http://www.Swish-DB.com) .

I have created this book to come to everybody's help regarding this wonderful program. There is no other book as well structured and explained like this one, so you made the right choice. SwishMax build in help is the only "book" available but it has another structure.

SwishMax unleashed covers almost all aspects of SwishMax. Although SwishMax is in continuing development, I hope this book will help you out. All examples that come with explanation are made in SwishMax build date: 2003.10.30.

I would like to thank the Swish DB team that helped directly and indirectly to the development of this book. Special thanks go out to: Ali Roman, Ali Imran, Charly Pixels, MC (Mario Carbonelli), Jarid, Robert Wells, Damoon, Ovidiu, B1ade, Vincent, timdav83 and many others.

Also thanks go out to the DBDG™ team (Driven By Design Group, Timisoara), and Design Kulture for their inspirational works and for giving me another perspective on design.

More thanks go out to all the swish and flash communities out there that help the people, free of charge. Respect to these people, because they are doing a lot of free work, just to help people and because they love what they do.

Also thanks to The SWiSHzone Team: David Michie, Hung-Hsin Chang, Roger Onslow, Gus Nalwan, Jason Bennett, John Collingwood-Smith, Hugh Boyd, Caitlin McKelvey, Susannah Huntington, Nathalie Albouze, Andrew Glover, Lana Khoury, David Petley, Dawn Barrie, Jon Herron, Deborah McRitchie, Simon Collingwood-Smith, Cameron Browne and James MacGregor, for bringing this wonderful program to everybody's lives.

To view more info about the team, visit: <http://www.swishzone.com/index.php?area=aboutus>

## Contents

<b>1. SwishMax presentation</b> .....	4
1.1. About SwishMax.....	4
1.2. SwishMax presentation.....	4
<b>2. Shapes and Objects</b> .....	11
2.1. Shapes (lines and derivatives).....	11
2.1.1. Creating shapes.....	11
2.1.2. Colors.....	13
2.1.3. Shape modeling.....	14
2.1.4. Gradient modeling.....	15
2.2. Objects (scene, movie, text, button, sprite).....	16
2.3. Grouping.....	19
<b>3. Basic actions and effects</b> .....	20
<b>4. An introduction to Swish Script</b> .....	22
4.1. Variables.....	22
4.2. GotoandPlay() and GotoandStop().....	24
4.3. Placing and removing objects.....	25
4.4. _root and _parent explanation.....	26
4.5. If and If ... else explanation.....	28
4.6. Move effect Swish Scripted. Movement.....	29
4.7. Using arrow keys to move objects.....	30
4.8. Integrating sounds.....	31
4.8.1. Setting up background music.....	32
4.8.2 Making a sound play when rolled over a button.....	32
4.8.3 Making a sound play when on a specified frame.....	33
4.9. Collision.....	33
4.10. Creating a preloader.....	35
4.10.1. Creating a simple preloader.....	35
4.10.2. Creating a percentage preloader.....	36
4.10.3. Creating an image preloader.....	37
4.11. Aging property.....	37
4.12. Time and date.....	38
4.13. Loading external files.....	39
4.13.1. Loading external text files.....	40
4.13.2. Loading external movie files (swf) .....	40
4.14. Dragging objects.....	41
4.15. Text scrollers.....	42
4.15.1. Simple text scroller.....	42
4.15.2. Complex text scroller.....	44
4.16. Image scrollers.....	45
4.17. Background sound volume and balance controller (scroller) .....	46
4.18. Mouse followers and cursors.....	47
4.18.1. The easiest mouse follower.....	47
4.18.2. A mouse follower with easing.....	47
4.18.3. Complex mouse cursor and follower.....	48
4.19. Duplicating and removing sprites.....	48
4.20. Drawing with Swish Script.....	50
4.21. Internet actions and browsing.....	50
4.21.1. get Url().....	50
4.21.2. fscommand().....	51
4.22.3. JavaScript().....	52

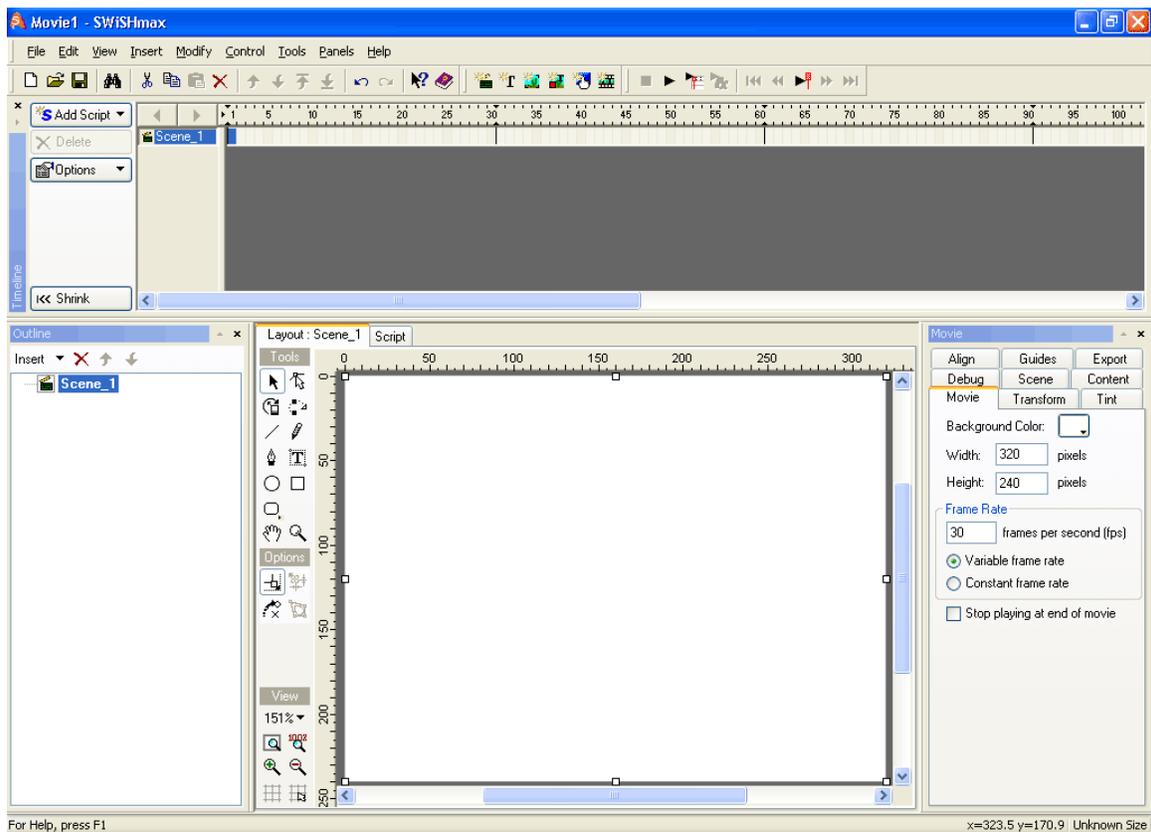
4.22.4. Mailto().....	52
4.22. Changing antialiasing of your movie.....	53
<b>5. Integrating SwishMax with PHP.....</b>	<b>54</b>
5.1. Creating a mail form.....	54
5.2. Creating a click counter.....	55
5.3. Logging IP's.....	56
5.4. Creating a guestbook.....	57
<b>6. Creating web pages in SwishMax.....</b>	<b>58</b>
6.1. Creating a simple web page.....	58
6.2. Building Idea Designs V4.0 LITE (template) .....	60
6.3. Using the old book template.....	62
6.4. Using the professional grey template.....	62
<b>7. Creating games in SwishMax.....</b>	<b>64</b>
7.1. Creating the "guess the number" game.....	64
7.2. Creating a simple click shooter game.....	66
7.3. Creating a more complex shooting game.....	67
7.4. Creating a car rally game.....	71
<b>8. Other examples provided with the book.....</b>	<b>75</b>
<b>9. Tips and Tricks.....</b>	<b>76</b>
<b>10. Quick color legend.....</b>	<b>80</b>
<b>11. Legal information.....</b>	<b>81</b>
<b>12. Bibliography.....</b>	<b>82</b>

## 1. SwishMax presentation

### 1.1. About SwishMax

SwishMax will allow you to create Flash™ content to add images, animation, sounds, and interactivity to your web site. Also you can create games and interactive applications. You can use this application to create introductions, commercials, banners, menus, and complete web sites. SwishMax revolutionized the Flash™ industry with its first release in April 2000. For the first time, complex-text effects could be created in minutes that had previously taken hours to create in Flash™. SwishMax is so intuitive and easy to use young children enjoy using it, and it is being used in schools throughout the world. SwishMax developers are dedicated to keeping the application intuitive and easy enough for the non-professional to use, while adding those features most requested by professional users.

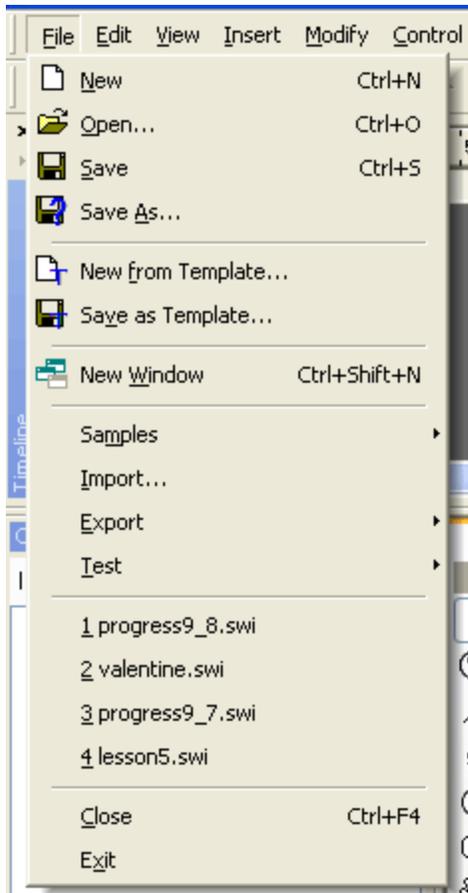
### 1.2. SwishMax presentation



In the above picture we have a sample of the SwishMax layout. We can see the menu, the timeline, the outline, the toolbar, layout and properties. We are going to take each of them and analyze them separately.

## The menu

Provides links to all possible options in SwishMax, from basic to advanced, grouped into categories. It is not used very much, because all of the most needed options appear as special icons.



### File

Provides a shortcut to options that allow us to:

Create new files: **New**  
 Open earlier created files: **Open**  
 Save our work: **Save** and **Save As...**

Create new file from a template: **New from template...**  
 Save the project as a template: **Save as template**

Open a new SwishMax window: **New window**

Open files that come with SwishMax: **Samples**  
 Import files that can be used in the project: **Import...**  
 Export files and make them ready to use: **Export**  
 Test files in the flash player, browser: **Test**

Open recently created files

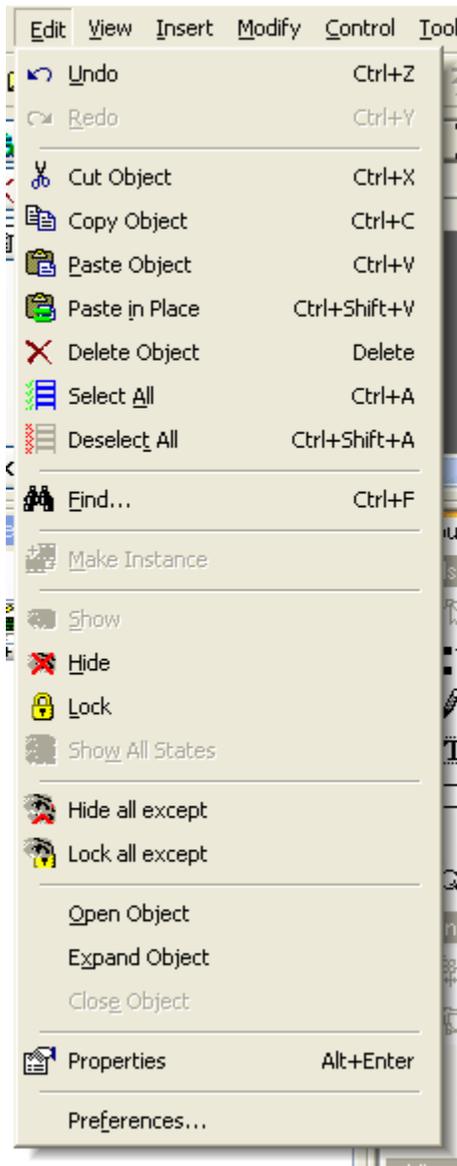
Close the file you are currently working on: **Close**  
 Exit SwishMax: **Exit**



This is the **Export** menu extension. This lets us choose for what format we want to export our movie. **Swf** is the default extension for Flash™ movies. You need the Flash player in order to play these movies. **Html + Swf** saves you work as a webpage that can be accessed by running the html file. **Exe** exporting is very useful for presentations and for people that do not have Flash™ player installed. **Avi** is the default extension used for movies. You can save your movie as such an animation too. **Copy HTML to Clipboard** lets you copy the movies html code to the clipboard.



The **Test** menu extension allows you to test the current movie you are working on in the Flash™ Player and in your default browser. **Report** is a very useful option that generates a report on your movie and allows you to see what objects are taking up the most of disk space in the movie.



## Edit

A short description of the items in the Edit menu is listed below:

**Undo & Redo** actions are commonly used in all programs to undo the very last action you took and redo is if you later decide you didn't want to undo an action. As shortcut keys to undo is Ctrl +Z and I advise you to learn this combination as you will use this option a lot.

Editing tools for the objects in the scene. Very useful to duplicate objects in the movie (using copy and after that paste), delete objects in the movie, select all objects in the movie or deselect all objects in the movie.

**Find** allows users to find a specific text in the SwishMax scripting environment.

Make Instance creates an instance of a sprite you have in you movie. Sprites will be discussed later in the book

Visibility options for the objects in the scene are very useful if you just want to specifically work on a object and don't want to see other objects, don't want the objects to interact, etc.

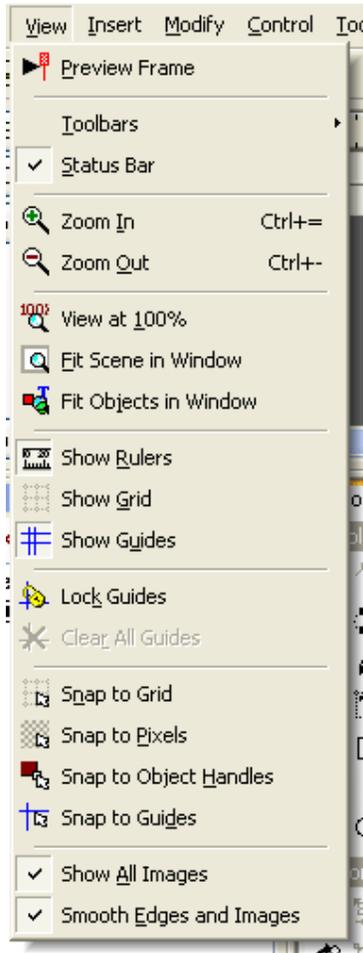
Hide/lock all except are useful for single object editing, allowing the user to **hide** all objects except those selected, or just **lock** those objects.

**Open, Expand and Close object** refer to sprites, groups and buttons that have the "⊕" character in front of them. This means that they contain other objects too. When expanding such an object the "⊖" sign will appear as a visual explanation that the object is expanded.

The **Properties** option lets the user view the properties of the currently selected object.

By clicking **Preferences**, it brings up the preferences menu for SwishMax that allow users to change interactivity options between users and the program.

**\*Note\*** It is pretty important to learn shortcut keys by heart, to make your work a lot faster and easier, accessing the wanted option only by pressing a simple key combination.



## View

The view menu provides access to options regarding viewing of objects, SwishMax toolbars, grids and rulers. A short description of the menu is as follows.

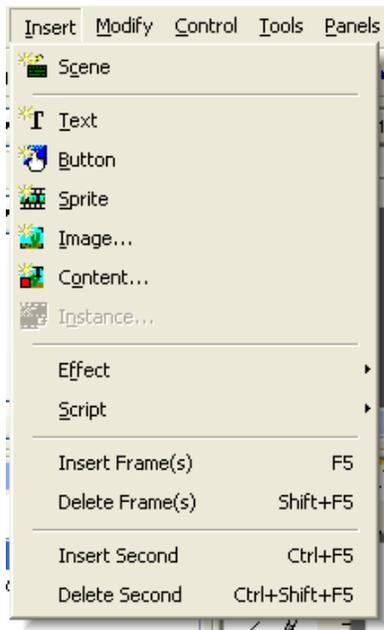
Preview Frame lets you preview a frame as it would look like at runtime.

**Toolbars** option lets the user select the wanted toolbars that are displayed in the SwishMax workspace.

The **Status Bar** is the bar at the bottom of SwishMax that displays helpful information: mouse coordinates current movie size etc.

The zooming options are very useful to work precisely on a certain point in the movie.

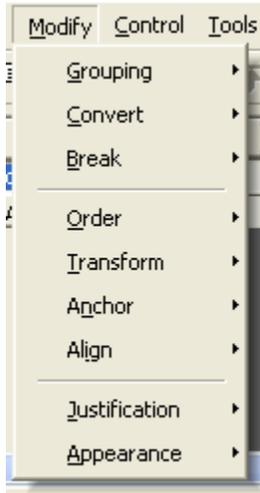
Rulers, grids and guides are very useful when creating precise applications. Snapping to grid, pixels, guides makes the object that is being built to snap (go to) to the handles. Combining these options you will get very precise graphics, object placements, etc.



## Insert

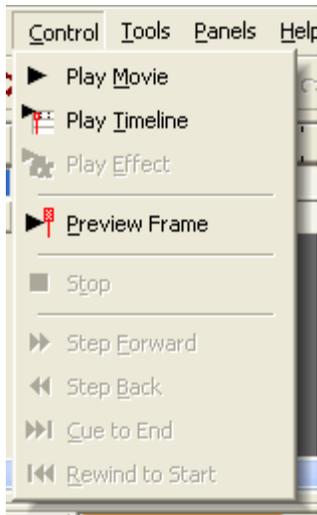
Here you can insert new scenes (**Scene**), new objects (**Text, Button, Sprite, Image, Content, and Instance**). Also you can add effects to the objects (**Effect**), or add a **Script** to objects you inserted. Also you can add frames, delete frames, insert seconds and delete seconds from the effects you just inserted.

Later on in the book there will be an explicit explanation of these objects, explanation of effects and a complete presentation of Swish Scripting.



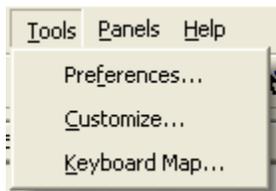
### Modify

In the modify panel we can modify object properties like grouping, conversion, breaking and arrangement of objects in the scene, preset transformations that can be applied, vertex modeling (will be discussed later in the book), alignments, justification of text and appearance of text boxes.



### Control

In this panel we can set options regarding the testing of our movie. We can play the movie, preview frames in the movie, and browse through the movie.



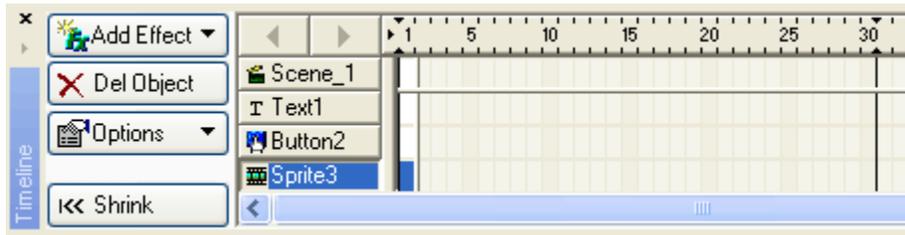
### Tools

Here you can set general preferences for SwishMax, customize toolbars for you use and a list of shortcut keys that can be used (**Keyboard Map**).

**Panels** – here you can set visibility of panels. I warmly recommend to leave them all checked

**Help** – contains help provided from the creators of SwishMax, tutorials, and useful links. Also here you can check out the version of your build of SwishMax. As the program is constantly improving, you can download the latest build from [www.swishzone.com](http://www.swishzone.com).

## The timeline



One of the most important toolbars in SwishMax allows users to view the objects in the current scene, **add effects**, and **delete objects**. Also if you notice there is a suite of numbers at the top of the timeline. This represents the frames. A certain amount of frames (can be set and will be discussed later on), form a second (30 in our case, as you see the line at frame 30).

The timeline also has options (display options) that can be accessed by pressing the options button. If you think that the text of the buttons is taking too much of you valuable work space, you can shrink them by pressing **Shrink**.

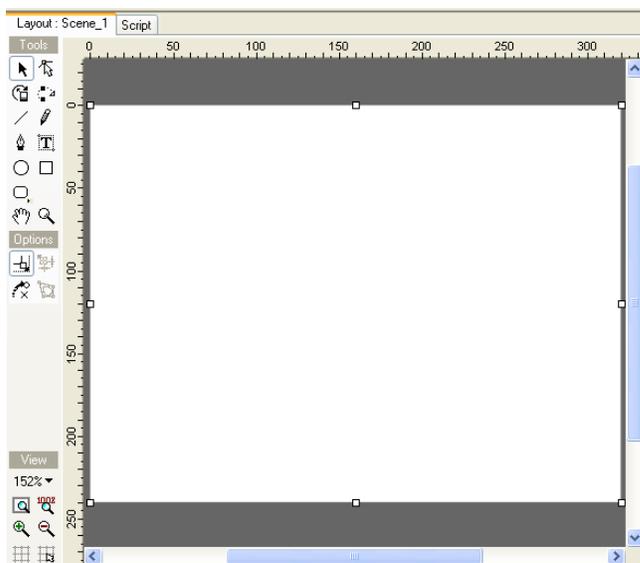
You will see later on the true importance of the timeline.

## The outline



In the outline we can see how many scenes we have, objects and their current position in the movie. Also by pressing Insert we can insert new objects, using the **X** mark we can delete objects in our scene. By using the up and down arrows we can modify the position of a certain object in the movie, relatively to other objects.

## Toolbars and layout

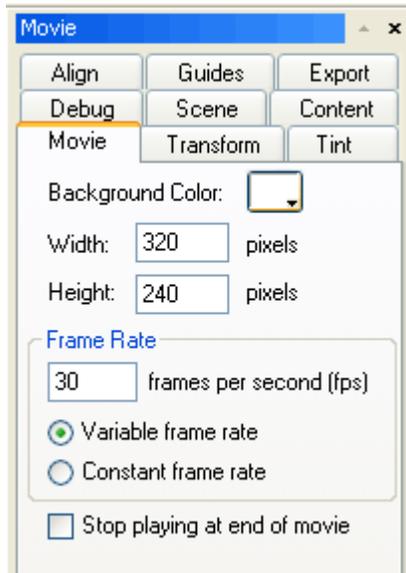


The toolbar represents easy access to creating new shapes in the movie and arrangement, modeling of shapes. The layout on the other hand is the workspace of SwishMax. Here you can view you objects, modify them after liking.

By clicking **Script** the layout will transform itself to a new workspace that lets you write SwishScript.

**\*Note\*** SwishScript is a term used for the programming language in SwishMax. Almost as complex as the Action Script used in Flash™ and with almost the same characteristics.

## Properties



As you probably noticed, in the right side of SwishMax there is this panel. It contains properties for all items used in the movie and others. It contains Export properties for the movie, properties for the movie, selected object etc. Later on in the book there will be an explanation of each item separately for a better understanding.

## 2. Shapes and Objects

### 2.1. Shapes

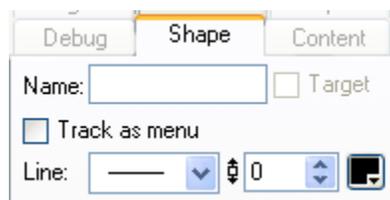
Shapes represent one of the most important things in SwishMax. Using shapes we can draw different objects that can be later modified and animated. By creating a drawing in SwishMax using shapes, we create vector graphics. Vector graphics have a lot of advantages, as they can take up very little hard disc space (or virtual space on the internet). Also they can be zoomed in very much, without losing any quality, or to become pixilated as normal bitmap graphics.

#### 2.1.1. Creating shapes

##### Line

To create a line, press the  icon from the toolbar. Now on the layout left click and drag the line till the desired length and positioning is reached.

**\*Tip\*** Holding the SHIFT key while dragging the line, you will draw only straight lines.



As line properties, we can modify the type of the line (solid, dashed, custom etc.) from the  drop down combo.

Also you can change line thickness from the  input box. Color can be modified by clicking on the black rectangle (default). You can also specify a name for the line.

##### Pencil

To create a freehand line, you can use the pencil tool. To draw using pencil, click the  icon from the toolbar. Now left click on the layout and move you mouse and draw the desired shape. The properties are the same as for line, because it is only a variation of a line (a freehand line technically). If the shape drawn with the pencil is closed, then it will gain the properties of a filled object. Then it has the properties of an ellipse.

##### Bezier

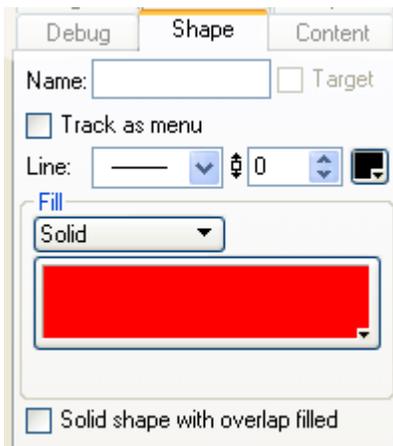
A Bezier line is a precise line that can vary from straight to curve. It all depends on how it is drawn. To draw a Bezier line or shape, click on the  icon from the SwishMax toolbar. Now left click on the layout and move you mouse. You will see that a shape outline is being formed. You can add other points to the line by right clicking in the desired places. When you want to finish your drawing, double left click on the layout in case the line is not closed, and a single click to close the shape, thus it will become a filled shape. The Bezier line has the same properties as a line, but the Bezier shape has the same properties as an ellipse.

## Ellipse

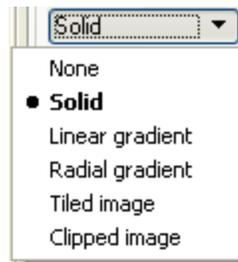
To draw an ellipse, click the  icon from the toolbar. Now left click on the layout and drag the shape till the desired shape is reached. Now release the left click mouse button.

**\*Tip\*** Holding the SHIFT key while dragging the ellipse will create only circles.

As line and filling properties for the ellipse shape we have the following:



As you can probably see we have the line properties and we have a new menu that contains Fill properties. By clicking the ▼ button, you will see the following popup:



Here you can choose the type of filling for the shape. This can be Solid, Linear gradient, Radial gradient, Tiled Image and Clipped image. For more information on colors please view chapter 2.1.2.

## Rectangle

Clicking the  icon on the SwishMax toolbar, you will be able to draw rectangles. It can be drawn exactly like an ellipse.

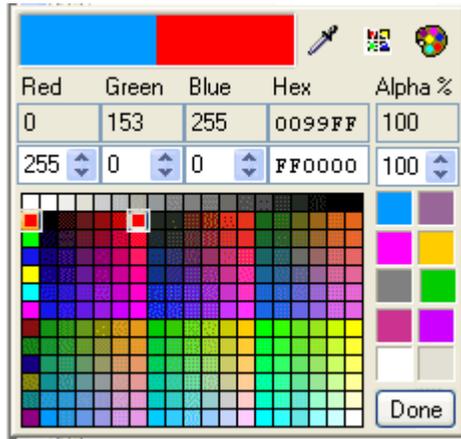
**\*Tip\*** Holding the SHIFT key while dragging the rectangle will create only squares.

The filling properties and line properties are the same as the rectangles.

### 2.1.2. Colors

Colors are widely used in SwishMax. Most commonly they can be set for shapes, but they are also used for setting background colors of the movie etc. We are going to take for example now the filling of a shape, so you can figure out colors better.

When we press the red rectangle in the **Fill** properties of a shape, we will see the following dialog box:



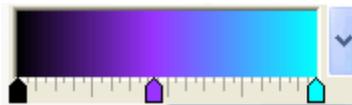
Here, you will see mostly used colors in SwishMax, marked with big blue in the top left corner of the dialog box is the old color of the object, and marked with red is the color we are currently on with our mouse, and can make a visual comparison. Also in this dialog box we have input boxes for Red, Green and Blue colors that form the RGB. These input boxes can have values from 0-255 for each color, making 16581375 unique combinations of colors. Also we have an input box for hexadecimal introduction of colors, useful to copy a color we want with just one line of code. We can also change the transparency of the shape, named alpha and it is represented in percents. This is from 0-100%, 0 being transparent and 100% being opaque.

By pressing the  icon we will be shown the Windows color chooser. Here we can also modify hue, saturation and luminosity for a specified color.

Pressing the  icon will force SwishMax to use only web safe colors, thus making your site viewable even on a 256 color depth display.

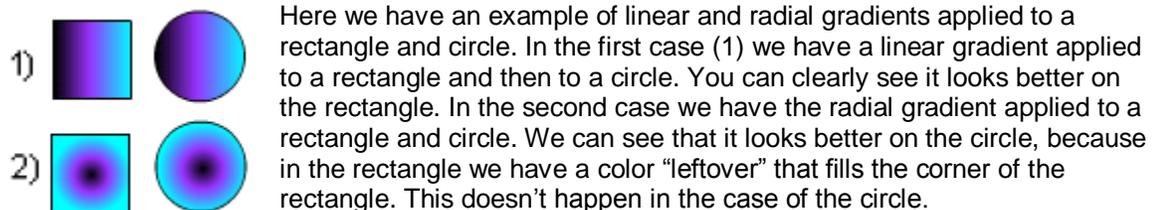
### Linear and radial gradients

Linear and radial gradients are used to make a shape change a color gradient (fade to a color) or to fade to transparency. For a shape the options look like this:



In the big rectangle we can see a preview of the gradient. Underneath this rectangle there are 3 arrows (in my example) that have different colors. As you can see the final result is a fading between the colors. To add more arrows left click on the ruler. To remove arrows, drag the desired arrow out of the rulers' domain.

The difference between linear gradient and radial gradient is that linear gradient fills a shape linearly with color, while the radial fills it circularly (radial).



To fade out the endings of a rectangle or an ellipse we can set transparency to a certain arrow in the gradient.



I have in my example a simple fading between blue and a transparent (0% alpha) white. Applying this gradient to a circle as radial gradient we'll have the following shape:



**\*Homework\*** Try to make the following star using two rectangles and a circle:

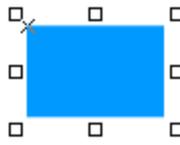


**\*Tips\*** to create it you will need a rectangle that has 3 arrows, center arrow is red and the outer arrows are transparent white. Copy + Paste it and rotate the pasted rectangle with 90°. Now create a radial gradient circle and place on top of the two rectangles. Have fun with your first vector graphic!

### 2.1.3. Shape modeling

Shape modeling is a very useful feature that allows users to modify how a shape looks like. You can create complex shapes starting from a line, rectangle, or an ellipse.

To do this, create a rectangle first, in your toolbar, the select button  will be selected and your rectangle will have eight little hollow square handles, just like in the picture below. This is in case



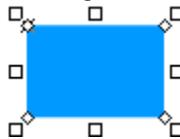
the Scale or Stretch option is activated in the toolbar: 

By dragging these squares you will modify the rectangles height, width.

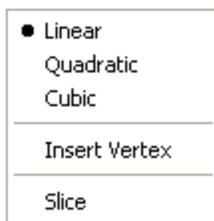
Now, if you activate the rotate or skew option  in the toolbar, these handles will turn into eight little round handles. By dragging these, the rectangle will rotate or skew as you drag it.

Using the distort transformation tool  located in the toolbar, you will see four star-like handles "⌘" in the corners of the rectangle. By dragging any of them, you will see that the whole rectangle will be distorted after the handle you are dragging.

A more complex shaping can be achieved by using key points, called vertexes. To enter in vertex mode and modify the rectangle, click on the reshape  icon in the SwishMax toolbar. Now your rectangle will look like in the image below:



To reshape the rectangle drag the little hollow circles. You know that you are doing this right, because your mouse arrow will turn to . As you can see there are infinite forms for you rectangle, you just need imagination and you can draw it.



If you want to add more vertex points, or want to delete existing vector points, right click on the outline of the shape. Your mouse arrow will turn into:  and the following menu will appear:

You can insert a new vertex or transform the outline between two vertexes to linear, quadratic or cubic. Experiment with these, and you will see the differences between them and realize the importance that they have.

Now if we right click an existing vertex, the following menu will be shown:



Here we can set the type of vertex, Cusp, Smooth or Symmetrical. Also we can sharpen the vertex in case it is smooth, or we could just remove the vertex. Usually the vertexes are cusp for rectangles and Smooth for circles. Now if you select smooth for a vertex of a rectangle, you will see the whole area around the vertex will gain a round shape and two cyan small circle handles appear “◆”. The best way to learn their meaning is to experiment. By selecting symmetrical, the “◆” handles are equally oriented from the vertex, that is situated at the half way between them.



**\*Homework\*** Try to create a shape similar to a half moon, like the one shown in the picture. Start from a circle and use all techniques that you've learned so far.

#### 2.1.4. Gradient modeling

This can be achieved by creating a filled shape and selecting a type of gradient (linear or gradient). Now click on  icon in the toolbar. Eight blue handles will appear. These can be squares or circles, depending on which icon is selected: the scale or stretch one:  (square handles) or the rotate or skew icon:  (circle handles). You can drag these handles, depending on what you want to do with the objects gradient.

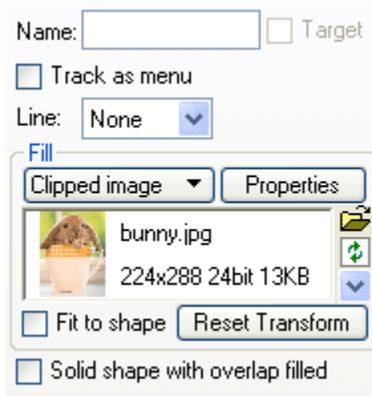
## 2.2. Objects

### The Scene

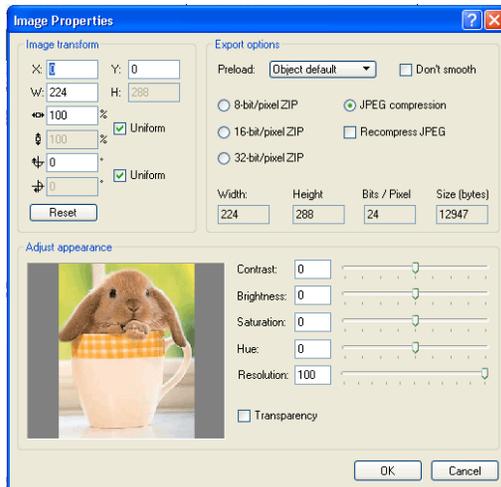
It is a “home” for all other objects present in the movie. A movie can have more than one scene, and these can be linked with one another.

### Images

You can insert images by clicking on the insert image button . The images properties appear as below:



By clicking on the properties button a new panel will be displayed:



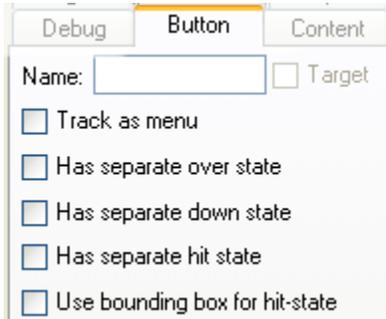
Here you can modify compression of the image, position of the image (X & Y axes), Width, Height, Rotation. Also you can modify the Contrast, Brightness, Saturation, Hue and Resolution of the image. These are very powerful build in features of SwishMax, letting you practically obtain a different image only by modifying a few values.

By clicking on the transparency checkbox, a new menu will appear, letting you choose the color you want to make transparent and the tolerance (how close to that color) should the pixels be made transparent.

**\*Tip\*** If you insert images in your movie, and you want to keep a maximum of clarity and minimum file size, try to compress large image files and save them as .jpg . This helps you movie to stay small with very little image clarity loss. For the small image files, a .png or .gif compression is recommended, the file size is very small and the clarity is the same. So my tip is to avoid .bmp extension images, as they are very large. Also in the properties panel of the image, test different compression for jpg and see how much loss you have in image quality and how it affects the file size of the new image have.

## Buttons

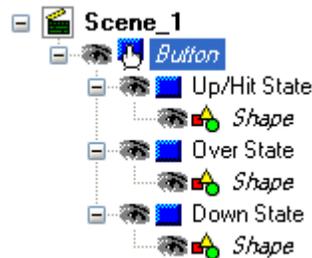
Buttons are special objects in SwishMax. You can create them by pressing the insert button icon . In the outline the buttons icon will be this one: . A button has the following properties (shown in the image):



A button has four states: Up, Over, Down and Hit state. They are referring to the properties of the button on different mouse actions: When the mouse is not over the button (Up), when the mouse is over the button (Over), when the left mouse button is pressed on the button (Down). The hit state refers only to the virtual rectangle that surrounds the button and triggers the events. To add more states to a button check the state you want in the buttons properties. The most accurate and easiest way to create a button is to create an object and group it as a button. After that, editing the objects will not represent a problem.

We would like to create a simple button that consists of a rectangle. The normal color of the button is red. When our mouse moves over the button it turns to blue and when the button is pressed it turns to black.

To do this, first create a rectangle. Set its color to red. Now group it as button. In the buttons properties panel check *Has separate over state*, *Has separate down state*. Now in the outline panel click the “+” icon. Your button will look as following in the outline panel:



On the up/hit state click on the shape . Now check its properties. The color should be red. Now click the shape in the Over state. Change the color to blue. On the down state, change the color of the shape to black. Now test your movie. Your button should work exactly like described above. Congratulations! You have just created your first button. If it doesn't look right to you, or you example didn't come out right, check out [button.swi](#) provided with the book. ★ 77

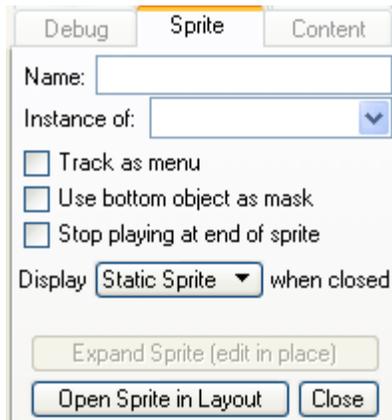
## Sprites

Sprites are the most complex objects in SwishMax. The equal to a movie clip in Flash™, the sprite can contain different objects in it. It is like a little movie in you movie. It has the ability to play a different animation and different actions in each one. That is how you can play multiple animations, play multiple sounds at once. Another good thing about sprites is that they are interactive... meaning each one can communicate with the other.

Sprites can be created by pressing the insert sprite button: . In the outline, the sprites icon will be the following: , followed by the name of the sprite.

The most efficient way to create a sprite (in my opinion... saves a lot of time), is to select an object or multiple objects and group them as a sprite. This way the objects keep their position in the layout and it can be used to lighten up the outline if there are many objects that need separate animating.

The sprite has a specific timeline and the following properties (shown in the image):



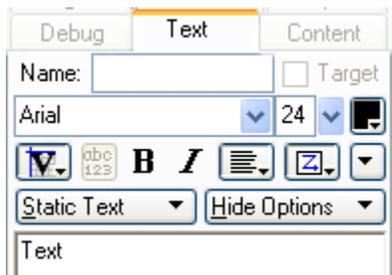
One of the power features of a sprite is masking. By checking the *Use bottom object as mask* makes the sprite show only the information as much as the bottom object lets. To be more specific, create a rectangle as the bottom object and an ellipse that surpasses the rectangle and has a different color. When the movie is tested, the sprite will display only the ellipse, trimmed by the bottom rectangle.

Another advantage of using Sprites in your movies is that you can easily reuse them throughout your movie, saving time and with only a slight increase to the file size of the movie.

★ 77

## Text

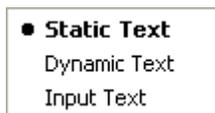
Texts are often used in SwishMax. You can use them to show from static data to dynamic data. Referring to static data, I mean text that does not change at runtime. Dynamic texts have the ability to change at runtime. The text properties look like in the image below:



You can specify a name for the text, select your font type, font size, font color, type of alignment of text, type of text (normal, bold **B**, italic *I*).

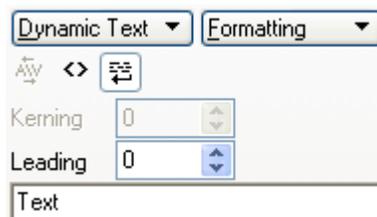
Also you can select the type of text (Vector or Pixel) from the dropdown menu drawn with a **V** in the image. Vector fonts are normal fonts and usually these kinds of fonts are used. Pixel fonts on the other hand are fonts that are very clear to read even at 8 pixels. These are used for professional presentations and web pages.

From the Static Text drop down menu you can see the following options:



Static text is generally used only to display static text.

Dynamic text is used to add interactivity to the text. It can change at runtime. Input text is used to input data that can be later used in the program.



Other options:

If we select dynamic text, and formatting we will see a new set of options. A good option to use is *Render text as HTML*. This is very useful if we import external texts that have html formatting, thus the text displayed can contain Bold, Italic, Underlined texts, etc. This option can be also used internally too. To write with bold write you text between `<b>sample text</b>`. To write with italic text, use: `<i>sample text</i>`.

For underlined text, use `<u>sample text</u>`. To begin a new paragraph with a bullet, use `<li>`



If we select Input text for text type and Advanced from the drop down menu we will see some options specific for input boxes. These are masking the text as a password, using the “ \* ” character. Also by clicking on the  icon, the input box text will be framed in a black border with a white background. By selecting the  icon, the input box accepts the enter key and supports multiple line text.

Many options are only available for different types of text boxes. It is up to you to discover which option is enabled for which type of text. 🌟 76

## The movie

The movie contains all possible objects, scenes, scripts etc. It is created when pressing the New  icon. As for properties, these are shown in the image below:



You can set the movies background color, set the width and height of the movie.

**Frame rate** is available for all types of movies. It represents the number of frames (images) displayed in a certain amount of time. In our case, frames per second (fps). Movies on you DVD, or on you TV are running from a frame rate of 24 to max 30. The human eye does not distinguish all the frames after the maximum of 30 fps. By making a slow frame rate, the movie will run choppy and it doesn't look nice. For a smooth transition use a frame rate higher than 24.

Variable frame rate (VBR) is an option used especially in music MP3 files. This is used for little quality loss and smaller files size. If the check box *Stop playing at end of movie* is checked, the movie will stop after running the animation in the timeline, it doesn't loop infinitely as usual.

## 2.3. Grouping

Grouping is one of the most useful options in SwishMax, letting the user group multiple objects together, making the outline look cleaner, and also useful for sorting different objects. For example you want to create a sun with shapes, you create a circles and some triangles around it, it would be useful to group them as one and name the group sun... right?

Grouping can be achieved in many ways, but the fastest way is to right click on the selected objects, and move your mouse over grouping. Now the following menu will appear:



Grouping as group, creates a new object called group with this  icon. Now you can rename it to whatever you like. It contains all your objects that are inside this group.

Grouping as button, creates a button that contains the selected objects.

Grouping as sprite, groups your objects into a sprite, creating a new movie inside you movie.

Grouping as shape, groups all your objects into a shape. This is a very helpful option especially for texts because texts gain the ability of filling transformations. Now you can add gradient to your text! There is only one bug to the whole problem. After grouping your text as shape, you cannot go back to text mode and modify the text.

### 3. Basic actions and effects

#### The stop action

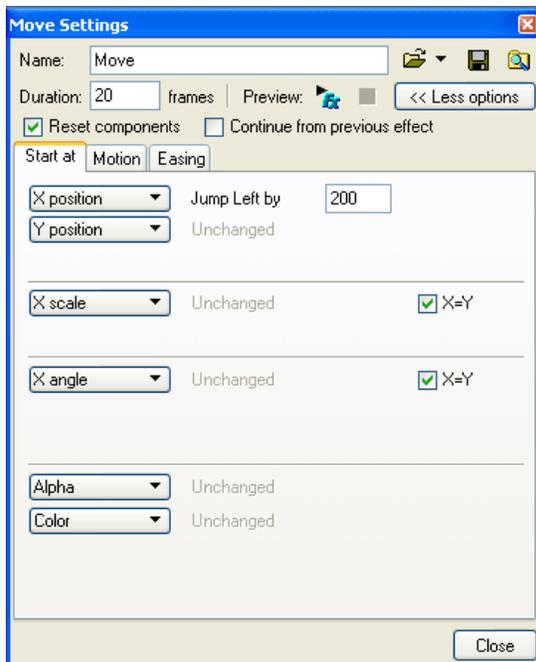
One of the most used actions in SwishMax is the stop action. The easiest way in my opinion to introduce new code in SwishMax is to right click on the timeline and select the desired option. For the stop action, right click and select: movie control → stop() . Now the movie will continue playing until the stop action is reached. In the script are you will see the following code:

```
onFrame (frame_number) {
    stop();
}
```

#### The move effect

Movement applied to any object makes it move to specified coordinates. Also it can be used to change the color of an object, change the transparency of an object, rotate an object and many other options.

To make your first movement, first create a rectangle in a new movie. Now right click on the timeline (rectangle) where you want the move effect to start and select move. A 10 frame move action will appear in the timeline for the rectangle. Double click the move effect. You will see the following dialog box:



Now click on the X position button and select jump left by. An input box will appear and you can set the amount of movement for the rectangle. In my example I entered 200. This number represents the number of pixels that the rectangle will move. For duration I set the length of 20 frames. You can choose how much you like. The more frames, the longer the animation will last. Now test you movie. You will see it repeats the moving effect. Congratulations! You have just made your first animation! Have fun and test the movement on vertical, horizontal, test scaling, angles, alpha (transparency) and color.

#### Place and remove

Probably the most used actions used in SwishMax allow users to show a certain object at runtime at a specific frame. Also useful to create animations by using sequential images. Try to import a .gif file (animated graphic). You will see it creates a sprite that contains a lot of little images that have the place and remove action attributed to them. Now let me explain a little better by giving you an example.

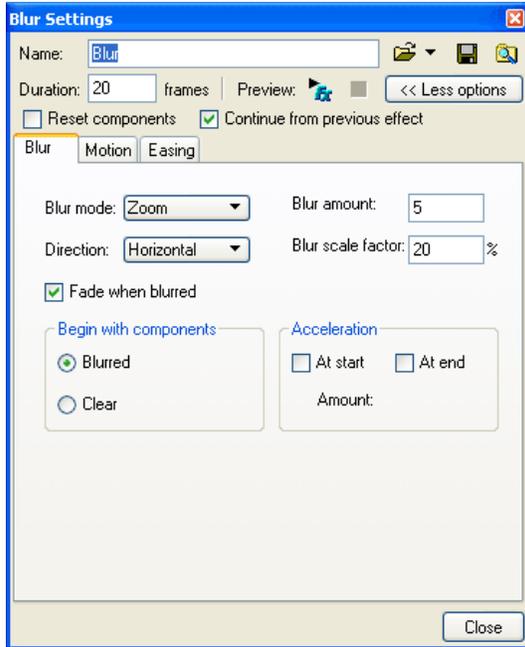
Create a rectangle in the layout and an ellipse. On frame 1 for both objects put a remove action. On frame 30 put a place action for the rectangle, on frame 40 a place action for the ellipse and a

remove action for the rectangle and on frame 60 a remove action for the ellipse. Now test you movie. You will see that after a little time the rectangle appears. When the ellipse appears, the rectangle disappears and after a while the ellipse disappears too. Easy enough right?

### Fade in & Fade out

Fade in and Fade out is used for a specific object to “fade in” (become visible) or “fade out” (become invisible). Just place an object in the layout and on the timeline right click and select fade in and after that a fade out effect. You will see that the object fades in and out.

### Blur



Blur is a processor consuming effect, but the results of a blurring an object is hardly replaced by another. You can set the blur mode, blurs direction, Blur amount and blur scale factor. Test these options out and you will see the results of the blur effect for different values.

### Core effects

Using core effects, you can almost create all of the 200 build in effects of SwishMax. Also you can design you own effects. The largest usability of these effects is found for text objects. You can edit any core effect by double clicking the effect in the timeline. As following I will explain some of the most used core effects.

#### - Transform

This core effect is very similar to the move effect but with a lot more options

#### - Snake

A very powerful effect lets you move objects after a virtual trajectory, given by a mathematical formula. Doesn't hurt if you know some math, but that isn't really a problem because you can experiment with the values.

#### - Wave

Useful on texts, makes the letters wave in different ways.

## - Typewriter

Used for text objects, it creates the idea of writing on a typewriter, by displaying one character at a time (or as you set in the effects preferences).

## 4. An introduction to Swish Script

Swish script is the new feature added to swish. It makes SwishMax very powerful and prepares it for professional use.

### 4.1. Variables

Variables exist in all programming languages, and thus present in Swish Script too. These are of three types: **STRING**, **EXPRESSION** and **BOOLEAN**. These exist to tell the programming language to remember something and use later.

Now let's take a moment to explain all of these variables.

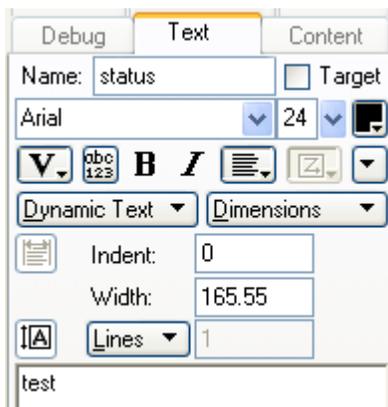
#### **STRING**

The string variable is used to identify information that is in brackets

Ex: *name* = "John"

Our programming language (well I am referring to SwishMax now) will remember that the variable **name** has assigned to it **John** information. Also we can have numbers in brackets: Ex: *weight* = "80". This tells the variable **weight** to remember **80**.

As a concise example, create a new movie. In it create a dynamic textbox, and assign it the name **status**, containing the text *test*.



*Remember that only dynamic text support variable handling and calculations.*

Now click on **Scene\_1** and add the following code:

```
onEnterFrame() {
  name = "john"}
```

Now draw a rectangle on the layout and add the following code to its script:

```
on (release) {  
status = name}
```

Now test your movie in the flash player (or html as you prefer) and click the rectangle you drew. Magic! The text **test** will transform into **john**.

Explanation: we declared a variable called name and we told it to remember john (the text in brackets). On the button action release we told the text **status** to replace its current containing information with the variables content. And that is how we made magic! If you can't figure it out exactly, or want to check my example, open the **variables1.swi** file provided in the package.

Now let's move to our next variable, **EXPRESSION**

Just like his brother, the **STRING**, the **EXPRESSION** is used to remember numbers and operations with numbers. Unlike the String, the expression does not support letters (only if they are variables that declare numbers), only numbers. You can specify the text like this(in scene\_1 script):

```
onLoad() {  
age = 22}
```

Now we told SwishMax to remember 22 as a number. To display the number we use the same tutorial from above (at STRINGS) and just replace name = "john" with age = 22. On the rectangle script, write:

```
on (release) {  
status = age}
```

When you press the rectangle, Magic again! The textbox will display 22. Just as easy. Now let's make a little modification and tell status something else:

```
on (release) {  
status = age + 10}
```

Now, when pushing the rectangle, the text **status** will display 32. Why? SwishScript automatically calculates what you told him.

Another good example (having fun are we) follows:

In Scene\_1 replace the code with the following:

```
onLoad () {  
age = 22}
```

Write as script on the rectangle:

```
on (release) {  
age += 1;  
status = age}
```

Now test your movie and push the rectangle several times. What happens? It adds up the age with 1. Why? Because we told him with += to add the variable **age**, 1.

This code can also be written like:

```
age = age + 1;
```

It's the same thing, but using += save us precious coding time and is easier to follow. To see a running example open [variables2.swi](#) . The blue rectangle exemplifies the first example, the red the second and the green the third example.

Ok, now forward to our friend the **BOOLEAN**. He is used to return TRUE or FALSE, true being of value 1 and false being of value 0. Using this we can find out information about the state of a different object (sprite) or we can tell it something.

Let's follow the next example closely:

Create a new file. Draw a circle. Convert it to sprite. Name the sprite *circle*. Draw a rectangle, and add the following code to it's script

```
on (release) {
circle._visible = false}
```

Now test your movie, and click the rectangle. Magic! The circle disappears. If you want to take it further create one more rectangle and add the following code:

```
on (release) {
circle._visible = true}
```

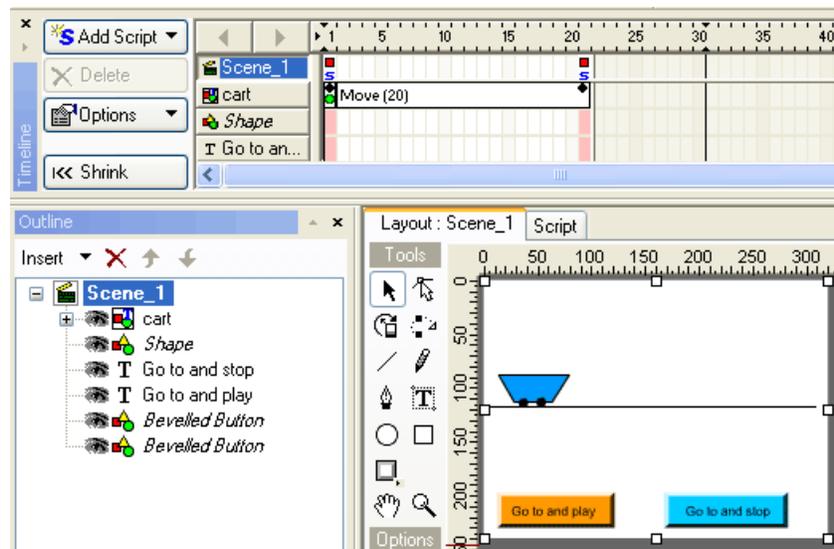
Now when testing the movie, when you click the first rectangle, the circle disappears and when pressing the second rectangle the circle will appear. How nice.

For a working example, run [variables3.swi](#) .

## 4.2. GotoandPlay(); GotoandStop()

GotoandPlay(*frame number*) and GotoandStop(*frame number*) actions are very often used in SwishMax. They are used to tell the movie to go to a certain frame number and play the action from there or just stop the action on a certain frame.

If you check out the example: [gotoand.swi](#) provided with the book you will see the following:



I made a cart (using a reshaped rectangle with 2 circles for wheels), a line (representing the ground and two beveled buttons one orange that has a text over it with the text “go to and play” and a cyan beveled button with the “Go to and stop” text over it.

In the timeline, I made the movie stop on frame1 and on frame 21(so the movie doesn't loop, this can be removed, and the movie goes to frame 1 after the move action is finished). For the cart to be viewable at the beginning of the movie, I added a place action. After that follows a move action of 20 frames, making the cart move to `_X=230`.

For the orange beveled button I have added the following code:

```
on (press) {
gotoandplay(2)}
```

and for the cyan beveled button:

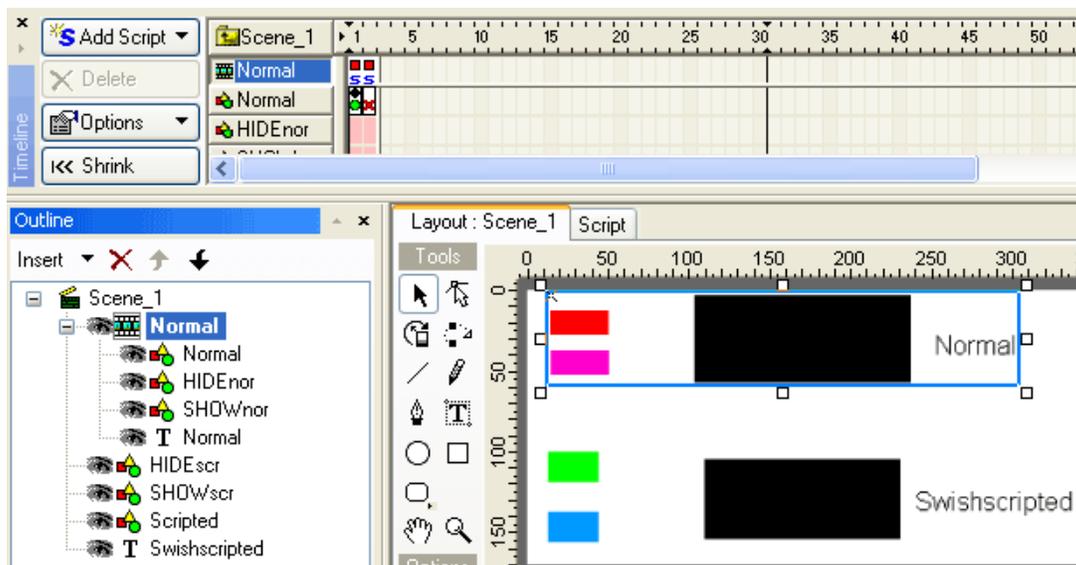
```
on (press) {
gotoandstop(10)}
```

When we test the movie, and press the orange beveled button, the movie plays the animation from frame 2. When the cyan beveled button is pressed, it goes to frame 10 and stops, thus making the cart stop in the middle of its course.

Going to frames and playing or stopping can be used to control sprites, scenes, etc. For other options regarding the goto command, use the guided mode in scripting.

### 4.3. Placing and removing objects

Open the *visibility.swf* movie. If you don't want to learn from the created example, create your own.



Create a new file. In it, create a black rectangle and name it **Normal**. Now create two other little rectangles and name them **HIDE nor** and **SHOW nor**. **HIDE nor** is red and **SHOW nor** is magenta. Also you can create a text and write in it normal. Group them as a sprite and name it **Normal**.

Inside the sprite, in the timeline put on frame 1 and 2 stop actions. For the large rectangle add a place action on frame 1 and a remove action on frame 2.

Add the following code to the red rectangle:

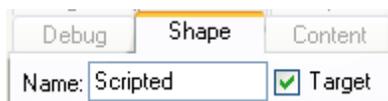
```
on (release) {
gotoandstop(2)}
```

,and this code to the magenta rectangle:

```
on (release) {
gotoandstop(1)}
```

If you test the movie, and press the red rectangle, the big black rectangle will disappear and if you press the magenta rectangle it will reappear. This is because on the red rectangle code we told it to “remove” and in the magenta code we told it to “place”.

Now for the swish scripted version of visibility (place and remove), create in the same movie, in `_root` (base layer in the scene) a big black rectangle and call it **Scripted**. Check the target check box.



This is very important, otherwise the actions we send won't be recognized. Now create 2 rectangles, a green and a blue one, called **HIDEscr** and **SHOWscr**. Add the following code to the green one:

```
on (release) {
scripted._visible = false}
```

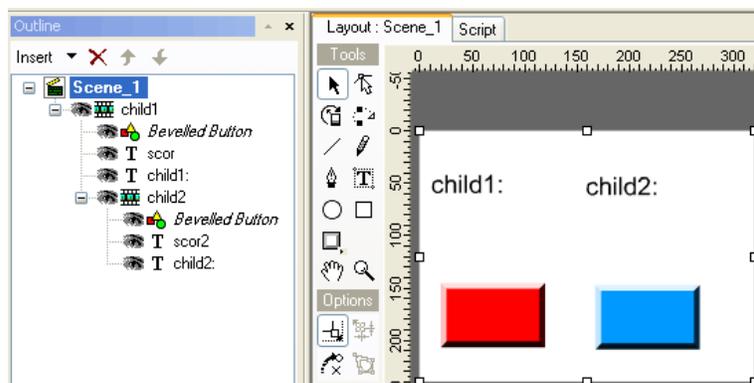
and for the blue one:

```
on (release) {
scripted._visible = true}
```

As you can see, when testing the movie, the scripted version does the same as the normal version, but more efficiently because there isn't any timeline usage.

#### 4.4. `_root` and `_parent` explanation

To exemplify `_root` and `_parent` better, we will create a little game.



Create a new movie. Now create a static textbox and enter the text **child1**: in it. Create another textbox (dynamic this time) and name it **scor**. Place them one near another. Then create a **beveled button** (using auto shapes). Now select them all (in outline using the CTRL Key and left click on them and select *Grouping --> Group as Sprite*). Now rename your newly created Sprite into **child1**. Now click on the **child1** sprite and copy it (CTRL +C) and paste it in **child1** (CTRL+V). There should be another sprite created. Move the second sprite, called **child2** as in the image above. Change the static text's content to **child2**: and name the dynamic text in **child2**, **scor2**. Also you can change the beveled buttons color to blue.

Now after all is done just like in the picture above, I will explain how `_root` works and `_parent` works:

`_root` and `_parent` describe paths, paths that we can use to control different objects. Using our newly created example, add the following code to the red beveled button:

```
on (release) {  
  scor = "OK"}
```

If you test your movie, the first dynamic text box (**scor**) will display OK. Now we want **scor2** to display OK. For this we have to give its path. This is done in the code below:

```
on (release) {  
  _root.child1.child2.scor2 = "OK"}
```

At runtime you can see that the second sprites dynamic text box changes to "OK".

Note that `_root` represents the movie, the first level in the outline.

Now we want **scor** to display OK when pressing the blue beveled button. To do this add the following code to the blue beveled button:

```
on (release) {  
  _parent.scor="OK"}
```

`_parent` represents one path level below the current sprite.

As you test your movie, you will see that **scor** will display "OK".

This can also be done by writing the exact path to **scor** like:

```
on (release) {  
  _root.child1.scor = "OK"}
```

If you are still experiencing problems please view the example provided with the book, [\\_root\\_parent.swi](#) .

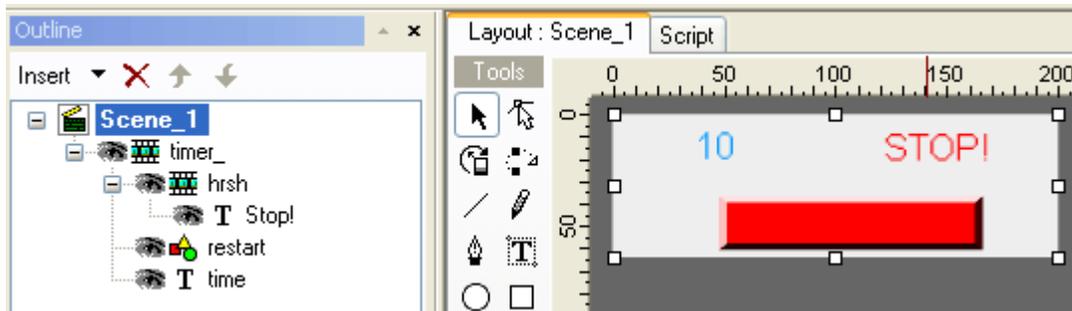
#### 4.5. If and If...else explanation

*If* is used to set a condition to a certain object in our program. *Else* is used in case we want to do something else in case the "*if*" statement doesn't satisfy the condition imposed by us.

To exemplify better, think of your alarm clock. It works on the same basis. **If** a certain time is reached (alarm time) the clock sounds the alarm (in our program it executes a certain line), **else** continue in showing normally the time.

Let's have a little fun now, making our own chronometer. We will put 10 seconds for the timer and we will tell the timer to stop when it reached 0 and display stopped message.

For this, create a new movie. In it create a dynamic textbox and name it **time**. Create a beveled button and name it **restart**. Group both of them into a sprite named **timer\_**. Now create in the **timer\_** sprite a textbox (static) and enter the text "Stop!" in it. Now convert this text as a sprite and name the sprite **hrsh**. Your movie's objects should look like this:



Modify the movies frame rate to 30. Add the following code to the **timer\_** sprite:

```
onLoad () {
  _root.timer_.hrsh._visible = false;
  timp = 10;}

onFrame (30) {

if (timp>0) {
timp = timp-1;
_root.timer_.time = timp;}
else {
_root.timer_.hrsh._visible = true  }
}
```

Code explanation:

We told the sprite that when it loads for the first time, hide the "Stop!" message and declare a variable called **timp** that has a value of 10 (10 seconds in our case)

*On frame (30)* is used to define a second (30 frame rate of the movie divided with 30 equals 1 second)

If our variable **timp** is bigger than 0 then decrease it by one and display it in **\_root.timer\_.time**. Otherwise, show the "Stop!" textbox.

You were wondering what we'll do with the red beveled rectangle? Simple... we'll add a reset code to it. Add the following code:

```
on (release) {  
  timp = 10;  
  _root.timer_hrsh._visible = false;  
  _root.timer_time = timp}
```

Test your movie... it works! You could also check out the example provided by me: [if\\_else.swi](#). As I said there are infinite possibilities for if...else. But we will present some cases along the way in the following chapters.

#### 4.6. Move effect swish scripted. Movement

Movement can also be scripted. You can move, rotate, scale, change transparency for an object just using SwishScript. For a working example please view [movement1.swf](#) provided with the book. To view the source code of the file, access [movement1.swi](#).

Applying movement to an object is not that hard. Create a new file. In that new file, draw a rectangle (or any shape you'd like) and name it **dummy**. Check the target box. Create a button and tell it to do the following things for the cases that follow:

Move on X axis (use of `_X`) (in pixels)

```
on (press) {  
  dummy._x = value}
```

Move on Y axis (use of `_Y`) (in pixels)

```
on (press) {  
  dummy._y = value}
```

Change width (use of `_Width`) (in pixels)

```
on (press) {  
  dummy._width = value}
```

Change height (use of `_Height`) (in pixels)

```
on (press) {  
  dummy._height = value}
```

Scale on X (use of `_XScale`) (in percents)

```
on (press) {  
  dummy._xscale = value}
```

Scale on Y (use of `_YScale`) (in percents)

```
on (press) {  
  dummy._yscale = value}
```

Rotate (use of `_rotate`) (in degrees)

```
on (press) {
  dummy._rotate = value}
```

Transparency (use of `_alpha`) (in percents)

```
on (press) {
  dummy._alpha = value}
```

**\*Note\*** value represents a number. For pixels it is undetermined, for percent it is in the 0-100 interval and for degrees from 0-360 degrees.

These properties can be used for almost all objects.

For another great example of SwishScript movement (normal, sudden, in time) please check out [movement2.swi](#). For smooth animation it uses *if* statement to specify when the movement should stop. Until then the `_x` number is increasing.

#### 4.7. Using arrow keys to move objects

Another great implementation in SwishMax is the use of keys. You can press any key on your keyboard and add an action for it. This is especially useful for game development. I have created an example that allows you to move an object with your arrow keys. View [arrowkeys.swi](#). If you would like to create for your own an example, create a new file. In it, create an object and group it as a sprite. Call the sprite **face**. Now on script, add the following script:

```
on (keyPress("<Up>")) {
  face._Y -= 1}
```

```
on (keyPress("<Down>")) {
  face._Y += 1}
```

```
on (keyPress("<Left>")) {
  face._X -= 1}
```

```
on (keyPress("<Right>")) {
  face._X += 1}
```

Code explanation: When pressing the "Up" key, the sprite **face** moves on the Y axis upwards with one pixel. The analogue explanation is valid for the other examples.

Instead of the arrow keys you could enter whatever character you would like. A list of special characters (keys) is found below:

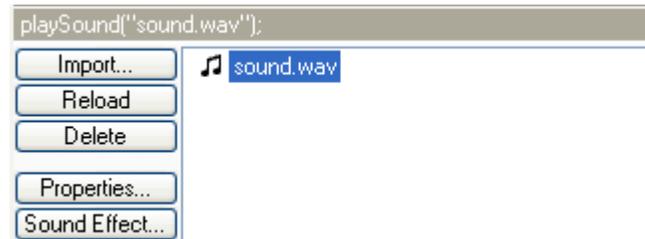
Symbol	Entered as	Description
Space	"<Space>"	Space character
Left	"<Left>"	Left arrow
Right	"<Right>"	Right arrow
Up	"<Up>"	Up arrow
Down	"<Down>"	Down arrow

Home	"<Home>"	Home key
End	"<End>"	End key
PageUp	"<PageUp>"	Page Up key
PageDown	"<PageDown>"	Page Down key
Insert	"<Insert>"	Insert key
Delete	"<Delete>"	Del / Delete key
Backspace	"<Backspace>"	Backspace key
Tab	"<Tab>"	Tab key
Enter	"<Enter>"	Enter key
Escape	"<Escape>"	Esc key

## 4.8 Integrating sounds

SwishMax has the powerful ability to import sounds and play them. Music files can be imported as wav or as MP3. Due to a limitation in the Flash Player, MP3 sounds must have sample rates of 11025Hz, 22050Hz or 44100Hz. I recommend compressing all the music files to MP3, because you will significantly make file size smaller, which is a good thing.

To import a sound, click anywhere on the timeline then go to script. Here, select AddScript → Sounds → playSound (...). The sound menu can only be accessed in guided mode. It looks like in the image below.



You can see the following options:

**Import:** Lets you import a sound file to use in your movie. Its format can be MP3 or .wav

**Reload:** To reload a sound, select the desired sound and click on reload

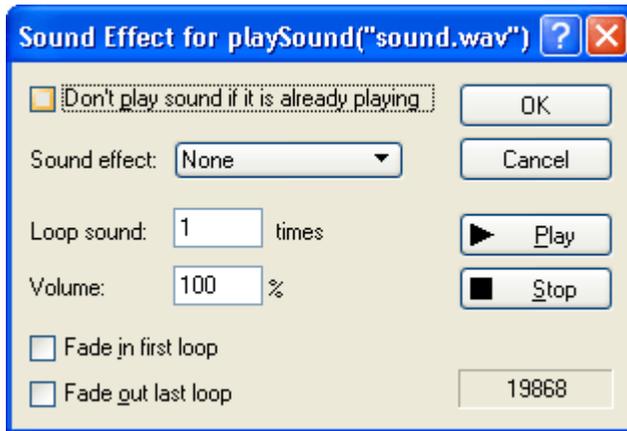
**Delete:** Select the file you want to delete and click this button to delete it from the list.

**Properties:** If you click this, you will see the following menu:

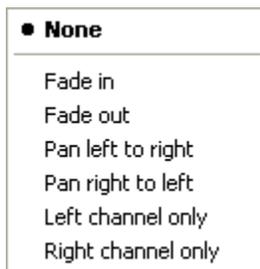


It contains the path to the file, the properties of the imported sound. You can preview the sound by clicking on the ► Play button and you can stop the sound by hitting ■ Stop. You can tell the object when to preload (will be discussed later on) and you can select export settings for the sound. These are Compression [(MP3 or WAV), MP3 recommended], Channels [(Mono, Stereo), this setting depends on what sound you want the user to hear in the speaker. The same sound in either speakers (Mono), or variant sound (Stereo). Sample rate indicates the quality of the sound. 11 kHz is used for the lowest quality, but the lowest file size. In the bottom right corner you can see the file size.

**Sound effect:** You will see the following dialog box popup if clicked:



Here you can check the upper left checkbox if you don't want to play the same sound if it is already playing, you can select some simple sound effects:



None: No sound effects enabled;

Fade in: Fades in the sound;

Fade out: Fades out the sound;

Pan left to right: You will hear the sound in the left speaker moving slowly to the right speaker;

Pan right to left: You will hear the sound in the right speaker moving slowly to the left speaker;

Left channel only & Right channel only: Plays only the selected channel.

Also you can make the sound loop X times (X representing the number of times the animation is played). Volume represents the volume of the sound file.

Fade in first loop can be checked if you want the sound to be heard slowly at first, and then the volume increases.

Fade out last loop: fades out the sound on the last loop.

You can preview the sound by clicking on the ► Play button and you can stop the sound by hitting ■ Stop.

Here too you can see the file size of the sound file in the lower right corner of the dialog box.

#### 4.8.1. Setting up background music

If you want your movie to have nice background music, create a sprite, and on frame 1 of that sprite insert the sound you want to be played in the background. Now on the sound effects properties select 9999 loop times (this makes the sound file loop almost forever). You are set with a nice background music. I suggest you get some free loops as background sounds from: [www.flashkit.com](http://www.flashkit.com) . It contains a lot of professional free lops that you can use in your web pages.

#### 4.8.2. Making a sound play when rolled over a button

For example you want a button that goes “whoosh” when rolled over, and you want that sound to stop when the button is rolled out. This can be done in a few easy steps:

First create a sprite, containing a rectangle (your button, you can design this however you like).

Then enter the sprite and add the following code to it:

```

onFrame (1) {
    stopSound("sound.wav");
    stop();
}
onFrame (2) {
    playSound("sound.wav");
    stop();
}

```

**\*Note\*** sound.wav is your music file, the file you import...

Click the rectangle and add the following code:

```

on (rollOver) {
    gotoAndStop(2);
}
on (rollOut) {
    gotoAndStop(1);
}

```

I think the code pretty explains itself. When the rectangle is rolled over, it plays frame 2 in the sprite (playSound) and when the rectangle is rolled out, it plays frame 2 which tells the sound to stop.

You can also replace the rectangle with any object you choose (sprite, button etc.) as long as it has that code to it. For an example please see [sound.swi](#).

#### 4.8.3. Making a sound play on a specific frame

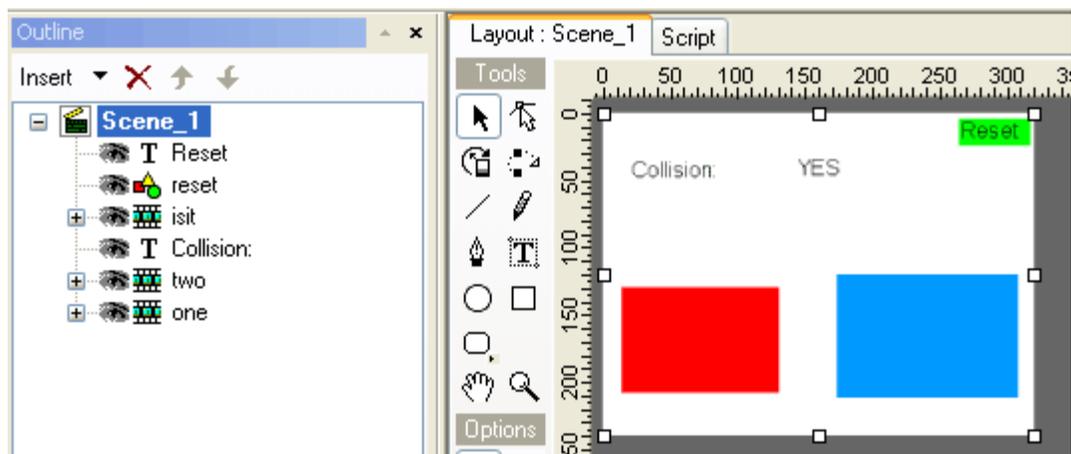
This is achieved by inserting your sound on a specific frame and tells the movie on an action to goto the specified frame.

### 4.9. Collision

There are three types of collision in SwishMax: when bounding boxes hit, when a certain distance between the sprites is reached, or if the sprite is near a certain x, y coordinates.

I will talk about the bounding boxes example, because that is the most used in my opinion...

Please open [collision.swi](#) delivered with the book file of the lesson. There you will see the following structure:



We have the text **Reset** and shape **reset** that is situated in the top right corner of our movie, the sprite **isit** that contains the text **yes**. You will see their purpose later as I explain.

We want to study... the collision between sprite **one** and **two**. Sprite **one** contains a red **rectangle** and sprite **two** contains the blue **rectangle**.

How should it work: When we move with the keys the red rectangle this gets closer to the blue one. When collision happens, it will display YES. To set the red rectangle to its original position we have created the Reset button.

Now here is the code:

The code for the sprite **one**:

```
on (keyPress("<Left>")) {
  _root.one._x=14;
  _root.isit.gotoAndStop(1);
}
on (keyPress("<Right>")) {
  one._x = one._x+1;
}
onEnterFrame() {
  if (_root.two.isNearThis()) {
    _root.isit.gotoAndStop(2);
  }
}
```

Code explanation:

We tell the sprite when we move left to reset movement by setting the default `_x` and removing the displayed **YES** (if the case). Check the contents of the **isit** sprite for details.

On keypress RIGHT... we tell our red rectangle to move right with one pixel on each key strike... but if we keep it down the red rectangle will hit the blue one in no time.

Now here is how the collision detection works :

```
if (_root.two.isNearThis()){}
```

If the blue **rectangle** (`_root.two`) is near the red **rectangle** (`_root.one` (current)) perform actions, which is in our case displaying **YES** mentioned in the sprite **isit**.

The rest I leave for you to experiment. I just got you started...

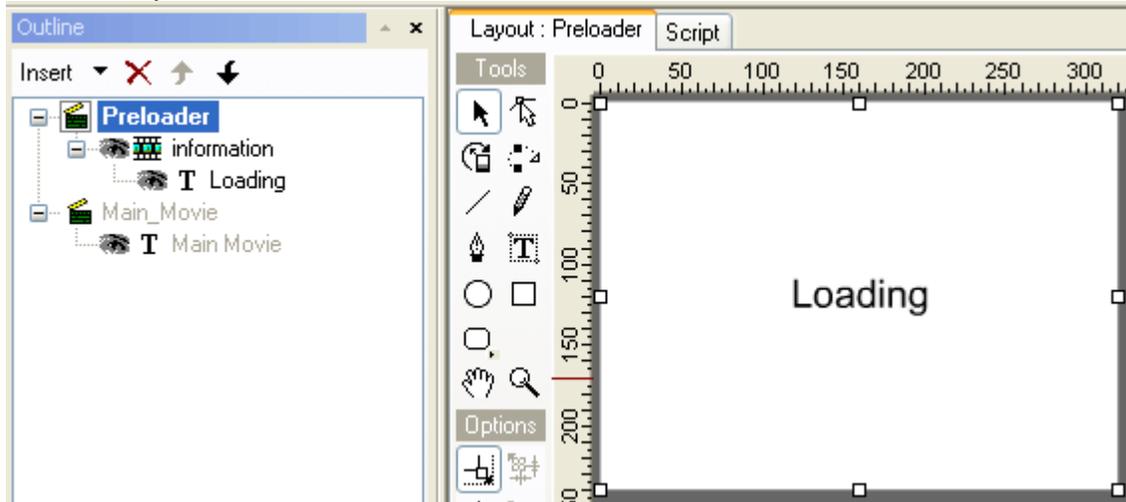
**\*Known issues\***: sprite collision does not work anymore if the path is longer than two sprites... for example if we tell: `if (_root.two.child.isNearThis())` it won't work , so relax , it isn't your fault.

## 4.10. Creating a preloader

A preloader is a scene in SwishMax that plays before the main movie, while this loads. This is very useful because users don't sit in front of a white screen and wait for the movie to load. It notifies the user that the movie IS loading.

### 4.10.1. Creating a simple preloader

A simple preloader is provided with the SwishMax program. I have modified that basic preloader for a better understanding. You can view it opening [preloader1.swi](#). This is what you will see:



All the code needed to preload a movie is found in the script of the scene **Preloader**:

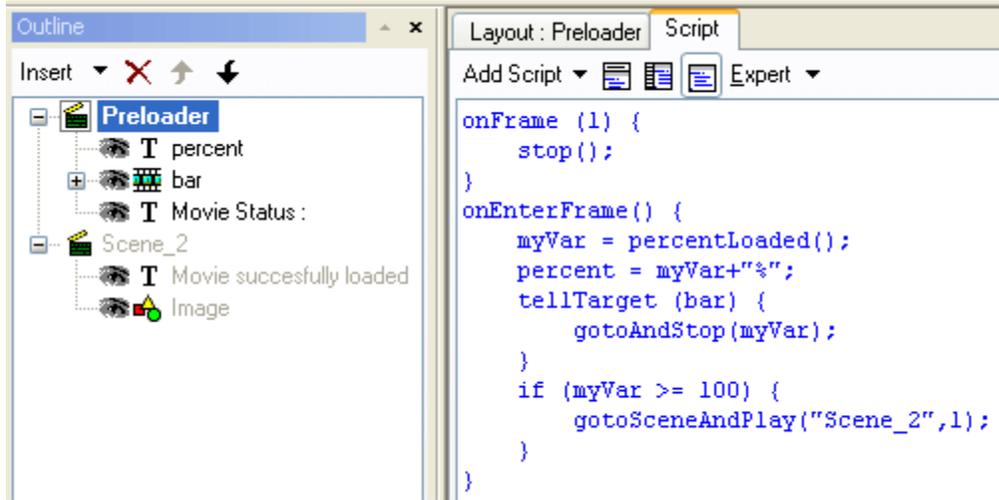
```
onFrame (1) {
    if (sceneFrameLoaded("Main_Movie",1)) {
        gotoSceneAndPlay("Main_Movie",1);
    }
}
onFrame (2) {
    gotoSceneAndPlay("Preloader",1);
}
onFrame (3) {
    preloadContent();
}
```

I have grouped the animation as one sprite called **information**. This lets the animation loop for an infinite time (until the movie loads). When creating a preloader you should consider keeping the preloader file size as low as possible, making it load fast so the user knows that the movie will eventually load. I recommend a maximum file size of 10 Kb for a preloader (a 3 second wait on dialup).

**\*Important\*** The Preloader scene must be in front of all other scenes for it to work.

#### 4.10.2. Creating a percentage preloader

SwishMax has a very powerful feature included, the `percentLoaded()` function. This function returns the percent of the movie that has so far loaded into your temporary file folder; its possible values are from 0 to 100. It always returns an integer value (means there will be no value with point as 21.4 or 34.5, it will always return a round value as 21 or 34. So it's really a handy function because this way we can let our movie goto any Frame number.



Above we have represented object management in our movie. To make this, create a new file. Rename your scene **Preloader**. Now create a static text file and write **Movie Status:** as text. Create a dynamic text box and name it **percent**. As text put "00%" in it. Now create a new sprite and name it **bar**. In it create a **rectangle**. Resize its width to 2%. Now create a movie action that has 100 frames and make it extend the width of the **rectangle** to 100%. Put a stop action in frame 1 of **Preloader** and one in the sprite **bar**. Add the following code to frame one of the **Preloader** scene:

```

onEnterFrame() {
    myVar = percentLoaded();
    percent = myVar+"%";
    tellTarget (bar) {
        gotoAndStop(myVar);
    }
    if (myVar >= 100) {
        gotoSceneAndPlay("Scene_2",1);
    }
}

```

Now your full code should look like in the picture above.

Create a new scene and insert your content there. Be sure to show the correct path in the preloader movie to your scene.

Code explanation:

`on EnterFrame()` – every time the frame is entered (continuous action)

`myVar` is a variable declared by us (to reuse it later) and contains the amount of `percentLoaded()`

`percent = myVar + "%"` – in the text percent, display the percentage of movie loaded stored in the variable and add to it's ending the "%" sign.

`tellTarget (bar) {gotoAndStop(myVar);}` – tell the sprite bar to go to that frame at which the percentage of the movie is loaded

`if (myVar >= 100)...` – if the variable is bigger or equal with 100, play the main movie.

For a finished example check out: [preloader2.swi](#)

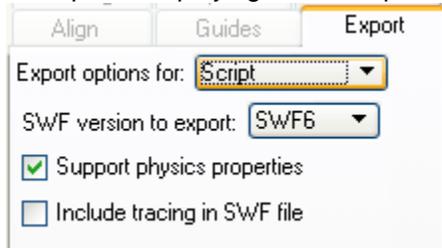
**\*Important\*** For the movie to work, export your movie to Flash Player 6. For this, go to the Export tab and set the SWF version to export: SWF6

#### 4.10.3. Creating an image preloader

Many people have asked me how to do this. Easier as you may think. First of all check out the percentage preloader example. Now the whole secret is in the **bar** sprite. Insert the image you want as a preloader display and underneath the rectangle that expands. Mask the sprite. Now you will only see the image gradually build up. For an example check out [preloader3.swi](#).

#### 4.11. Aging property

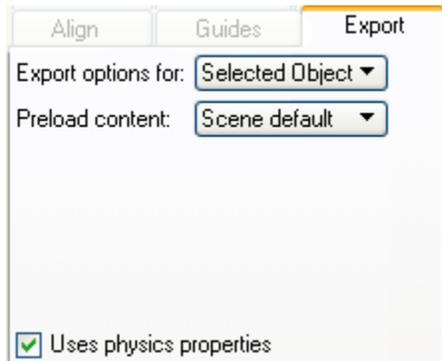
One of the many applications of this physics property (for an object in SwishMax) is the great example of displaying the time spent on a site in seconds.

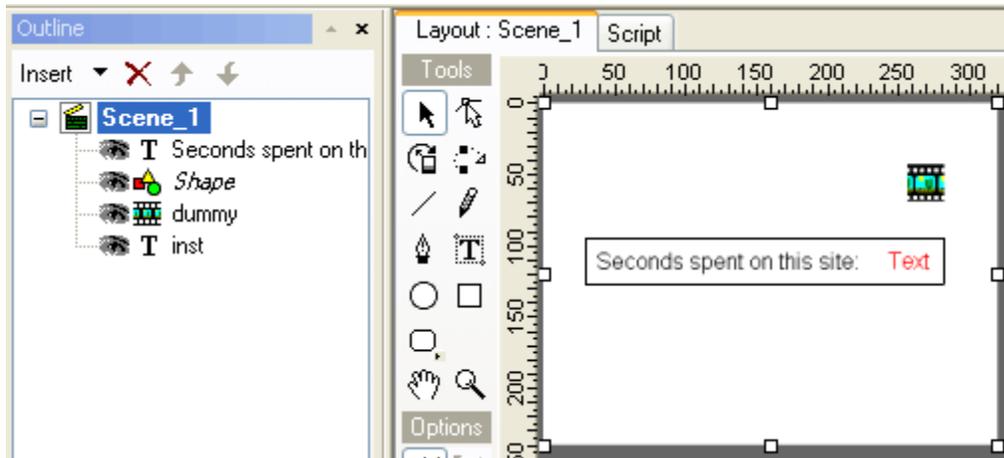


To be able to use physics properties in a SwishMax movie, you must enable it by checking the Support physics properties from the Export Panel, Script button.

For a clear example, check out the [aging.swi](#) file. Next we'll learn how to do that.

First of all create a Static text box and enter as text "**Seconds spent on this site:**" Now create a dynamic text box and as for name put **inst**, and as for text you could put anything. Arrange them nicely and create a **rectangle** for the shape surrounding the text just for aspects. Now the secret of the movie comes. Create a sprite and name it **dummy**. This is an empty sprite but has the uses physics properties selected from the Export panel, like in the image. Now all your objects and design should look similar to the one in the following image





Insert the following code in the **dummy** sprite:

```
onEnterFrame() {
    t1 = _root.dummy._time;
    _root.inst = int(t1);
}
```

This is all it takes to make it work. If you test your movie, you will see that the **inst** textbox starts counting upwards from 0.

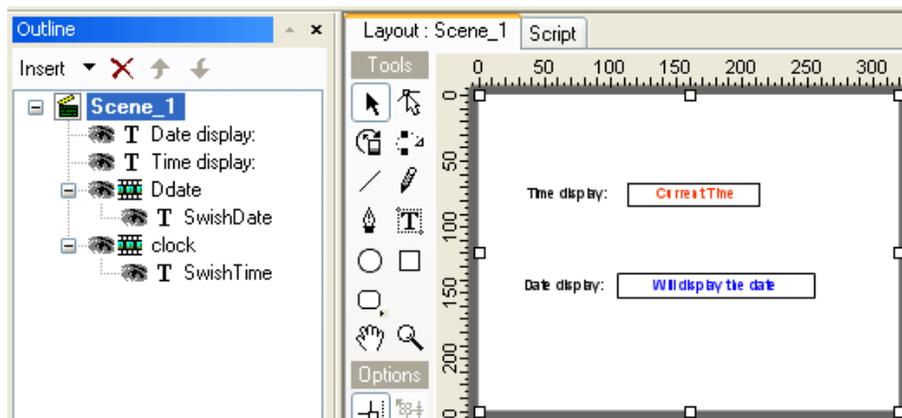
Code explanation

`t1 = _root.dummy._time;` - we declared a variable (**t1**) and told it to keep in mind the `_time` (a physics property), the actual thing that makes it all happen.

`_root.inst = int(t1);` - now we told the **inst** textbox display an integer (the integer value) of the **t1** variable. You could try to remove the `int()` from the code and leave only `_root.inst = t1;` and you will see a lot of numbers after the actual seconds. These represent milliseconds and following.

## 4.12. Time and date

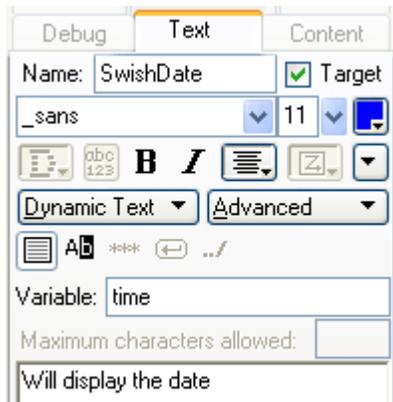
Very useful if you want the user on your site sees a digital clock (or analogue one). Provided with the book, there is **clock1.swi**. If you open it, you will see the following.



The sprite **Ddate** is used to display the date in swish and the sprite clock is used to display the time in swish. First let's analyze the code for the **Ddate** sprite:

```
onEnterFrame() {
    now=new Date();
    month = new Array('1','2','3','4','5','6','7','8','9','10','11','12');
    time= (now.getDay()+ "/" + month[now.getMonth()]+ "/" + now.getFullYear());
}
```

The code pretty explains itself. We declare a variable **now** that records the new Date() , a function that allows SwishMax to read the current date. In the array we told SwishMax to replace January, February etc. month names with numbers. You could also change these to Jan, Feb, Mar etc. , for your liking. In the last line of code we told the variable **time** (the variable of the dynamic text **SwishDate** (like in the image) to display the day, month and year using the getDay() , getMonth() and getFullYear() functions.



The time code is situated in the **clock** sprite:

```
onEnterFrame() {
    currentTime = new Date();
    _root.clock.hour = currentTime.getHours();
    _root.clock.minute = currentTime.getMinutes();
    _root.clock.second = currentTime.getSeconds();
    _root.clock.time = _root.clock.hour + ":" + _root.clock.minute + ":" + _root.clock.second;
}
```

We declared a variable called **currentTime** and using the new Date() function we read the hours, minutes and seconds just like in the example above, using date.

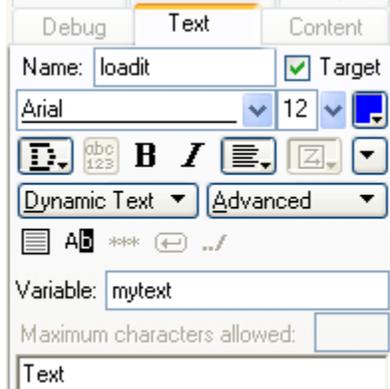
For a more complex example, view [clock2.swi](#). It displays a digital and analogue clock. The original scripting is made by Björn from The Netherlands. Thanks!

#### 4.13. Loading external files

Loading external files is especially useful if you have large movies and you want them to load separately and you don't want the user to wait without seeing a preview of things. Also good if you want to load your movie first and then play a sound file later after the main movie is loaded. When I am referring to external movies I am talking about movies with swf extension. You can also load external text files. This is useful if you don't want to edit and change your whole site just for a text. External image files can be loaded too with the same method.

### 4.13.1. Loading external text files

To load external files, create a dynamic text box and set it the following properties:



Name: **loadit** - with target checkbox selected  
 Type of text: dynamic text  
 Variable: **mytext**

Now create a new text file with notepad and add this text to it:

```
mytext=This is a text that uses
multiple lines so you can
see that it can load multiple
lined text too.
```

Now save the text file as **mytextfile.txt**.

Return to SwishMax and add the following code for the scene:

```
onframe(1) {
this.loadVariables("mytextfile.txt");
}
```

Export your swf file to the same location as your text file. Run the swf file. You should see it works just fine. For multiple line display select the **loadit** text and go to Dimensions properties. There deselect the auto height option and input the desired number of lines. If you don't want that huge space between the lines of the loaded text file, go to **loadit** Formatting properties and give a negative value to Leading. A -50 would be sufficient. Experiment for your own with different text styles.

**\*Note\*** You may load data for multiple text objects from a single text file, by adding '&' sign before each variable name

Example: **&mytext=My first text object value is 1 &mytext2=this is data of object 2**

**\*Important\*** Do not leave any spaces between the variables name and the equal "=" sign, otherwise text loading will not be possible.

For an example check out **ext\_text.swi**.

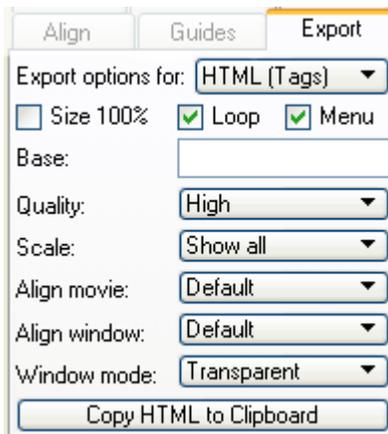
### 4.13.2. Loading external movies (swf)

For loading external swf's , create a new sprite. Add this code to it:

```
onFrame (1) {
loadMovie("movie2load.swf");
}
```

We told the sprite to load "movie2load.swf", that is our external movie. That's all there is to it. For an example check out: **ext\_movie.swi**.

**\*Tips by Charly Pixels on “How do I make an external movie load at a higher level?”\***



- make a movie with the exact same dimension of your main movie
- position everything where you want it as if you are positioning it at your main movie
- export that movie with the export option set to Transparent (window mode)
- save that movie with the name for your external movie

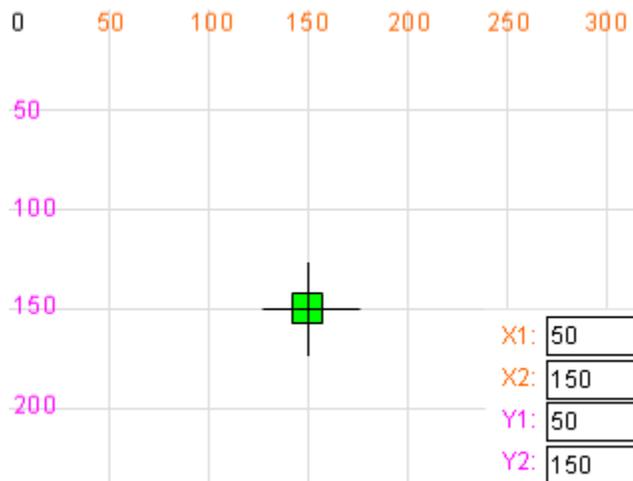
**This is the script to load a movie in a higher level:**  
Put that script on the buttons at your main movie level 0

```
on (release) {
loadMovieNum("youexternalmovie.swf", 1);
}
```

Where number 1 after the “,” represents the level we loaded the movie to.

#### 4.14. Dragging objects

Dragging is a very useful implementation in SwishMax, which allows users to drag different objects. This is useful for games, scrollers etc.



If you check out [dragging.swf](#) provided with the book, you will see the following example. It exemplifies the fact that dragging can be controlled by the user. I used a crosshair draggable object to show you that it moves perfect on the grid of the movie and coordinates given by you. You can set the X coordinates on which the object can move (from →to) and the Y coordinates (from →to). If you change these values, click on the pointer and then drag it to see its new boundaries.

To drag the object, write the following code (for the object):

```
on (press) {
startDragUnlocked();
}
```

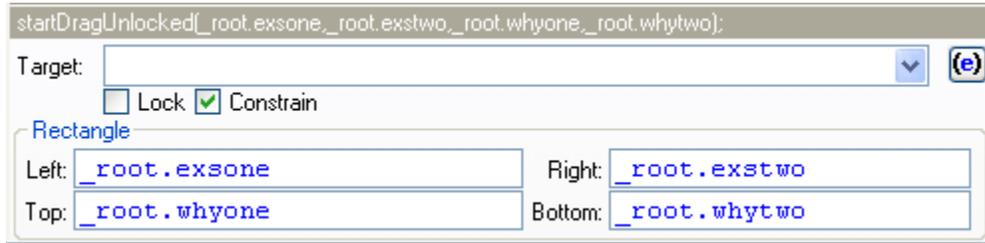
To stop dragging the object, write the following code:

```
on (release) {
stopDrag();
}
```

The `startDragUnlocked()` code can be personalized, giving it a rectangle restriction. It has the following expression:

```
startDragUnlocked(left, right, top, bottom)
```

For our example, `dragging.swi`, if we go to guided mode, we will see the following:



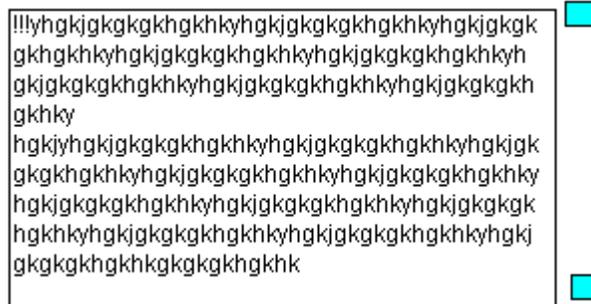
Where `exsone`, `exstwo`, `whyone` and `whytwo` represent the virtual rectangles coordinates in which you can drag the object, in our case declared in the input boxes.

## 4.15. Text scrollers

Text scrollers are used when you want to squeeze in a large amount of text in a small space. Now you can make a huge textbox and enter all your text, or you could make a small textbox and add scrollers. We will exemplify as following how to make one.

### 4.15.1. Simple text scrollers

I will demonstrate how to make the basic text scroller, just like in the image below:



The functionality is simple. When we click on the cyan buttons, the text will scroll up or down, depending on which buttons you press.

To design the scroller to do the following: Draw a rectangle, as big as you want your text to be displayed. Name it `mask`. Now create a textbox, name it `myText` and check the Target checkbox.

The text should be max as big in width as the rectangle.

Now paste the following text into the textbox:



**For the down arrow:**

```

on (press) {
  down = "true";
}
on (release,dragOut) {
  down = "false";
}

```

**And for the up arrow:**

```

on (press) {
  up = "true";
}
on (release,dragOut) {
  up = "false";
}

```

## Code explanation:

When we press the buttons, for example up, it declares a variable **up** that has “true” applied to it when pressed. The same thing is for the **down** variable and arrow.

For the code in **Scene\_1**, we have declared the variables **up** and **down** that are “false” when the movie loads, that means no button is pressed.

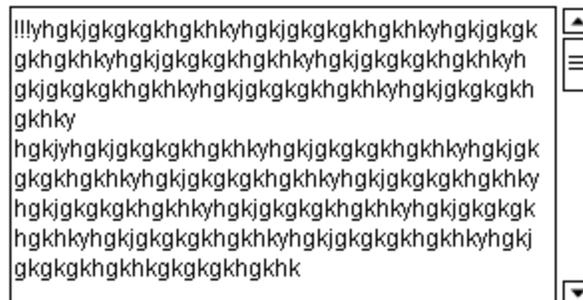
**eq** is used to exemplify equal and **||** is used to express equality of text variables. For number variables **==** is used to express equality.

Now we told the movie to check if the up button is pressed, has the “true” statement. If this is the case, move it the text down with 2 pixels ( += 2). If down is activated, move the text upwards with 2 pixels. The limit of the text in our example is -160 → 0. It could be another in your case. This defines the maximum Y on which the textbox can move. -160 represents the start of the text line and 0 means the ending of the text line.

To see a ready made example view: [textscroller1.swi](#)

**4.15.2. Complex text scrollers (adding a draggable scroll in the middle of the arrows)**

This is the more complex version of the scroller. It looks like in the image below:



To create this, we will work on the previous example a little more. Create a rectangle named **myScroller** and place it between the up and down arrows. Check the target checkbox near the name. Now group **myScroller** as a sprite and call the sprite **scrollr**. Now add the following code to **myScroller**:

```

on (press) {
  thsX = this._x;
  startDragUnlocked(thsX,thsX,57,150);
}
on (release,releaseOutside) {
  stopDrag();
}
onEnterFrame() {
  OldY =( 60 - _root.scrollr._y);
  NewY = OldY * 2;
  _root.textSprite.myText._y=math.round (NewY);
}

```

We told SwishMax that we are allowed to drag **myScroller** on the Y coordinate, from 57 → 150 in my example.

The complication appears on the `onEnterFrame()` code. Here we declared a variable **OldY**. The number 60 is good only for my example. You could change it to what case you are in. This variable describes the start point of the textbox.

$NewY = OldY * 2$  – because the scroller does not have the same movement as the up and down arrows we have to set a speed that matches the textbox movement. You determine this number by guessing. I used 2 in my case to display the whole text.

`_root.textSprite.myText._y=math.round (NewY);` - here we told the textbox to move accordingly to our scroller dragging.

The scroller works, but it doesn't update when we press the up and down arrows. For this add the following code to the `onEnterFrame()` in `Scene_1`:

```

if (down eq "true" || _root.scrollr._y < 57) {
  _root.scrollr._y += 2;
}
if (up eq "true" || _root.scrollr._y > 150) {
  _root.scrollr._y -= 2;
}

```

where 57 and 150 are the limits of the scroller (determined by the arrows position).

For the finished example view: [textscroller2.swi](#). Also a much nicer variation of this scroller, with buttons for arrows and a nicer design can be found accessing the [textscroller3.swi](#) example.

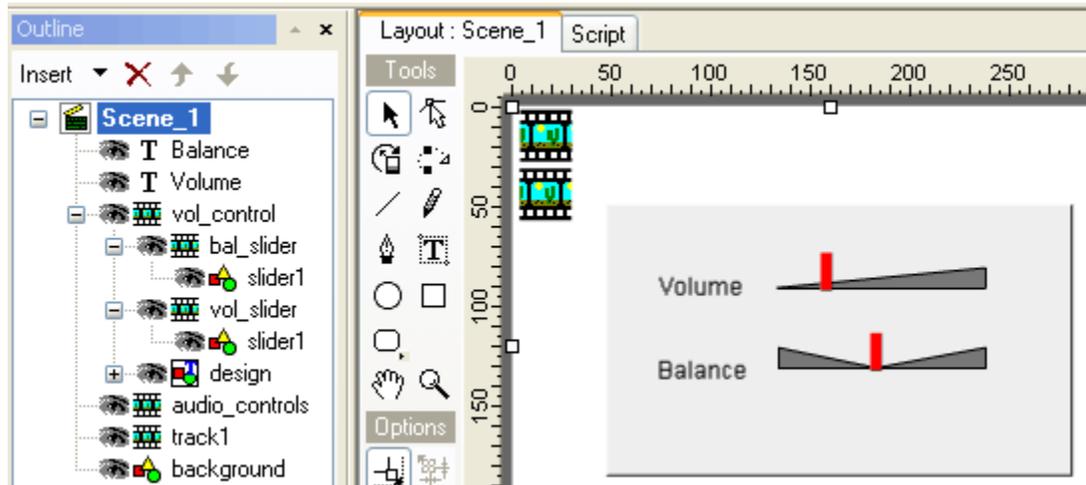
#### 4.16. Image scrollers

Image scrollers are made in the same way as text scrollers, but instead of text, they contain pictures. For a modified example of the `textscroller3` to an image scroller, see [imgscroller.swi](#).

#### 4.17. Background sound volume and balance control (scroller)

This is a must have feature if you want to create a site that has a background sound, or at least add a play and stop button for your music.

Understanding how the volume control works. Open your `vol_bal_control.swi` file. You will see the following structure:



The **balance**, **volume** texts and the **background** shape don't have any actions applied to them. The **slider1** shape in **bal\_slider** and **vol\_slider** sprites have the following code:

```
on (press) {
    this.startDragUnlocked(0,100,this._Y,this._Y);
}
on (release,dragOut,releaseOutside) {
    stopDrag();
}
```

This lets us drag the sliders from x=0 to x=100.

The **track1** sprite contains the actual background music:

```
onFrame (1) {
    playSound("Ee_rev.wav",false,100,1000);
}
```

In order to control the volume using the draggable scrollers, we must create another sprite, named **audio\_controls** in our case that has the following code:

```
onLoad () {
    track1_sound = new Sound(_root.track1);
}
onEnterFrame() {
    track1_sound = new Sound(_root.track1);
    track1_sound.setVolume(int(getProperty(_root.vol_control.vol_slider,_x)));
    track1Balance = int(getProperty(_root.vol_control.bal_slider,_x)-50)*2;
    track1_sound.setPan(track1Balance);
}
```

This tells the sprite that when it's loaded to define the track as a "sound" variable. Then when the volume slider and the balance sliders are activated, set the volume (setVolume) and set the balance (setPan) for the track.

## 4.18. Mouse followers and cursors

Mouse followers are very useful if you want to spice up a little the way your mouse cursor behaves and make cool things follow it. This can be done in many ways, but I will explain only 3 cases.

### 4.18.1. The easiest mouse follower:

Create a sprite and draw an object in it. Add this code to the sprite:

```
onEnterFrame() {  
    _X=_root._xmouse;  
    _Y=_root._ymouse;  
}
```

When you test your movie you will see the object follow your mouse button. For an example see [mousefollow1.swi](#) that came with the book.

### 4.18.2. A mouse follower with easing

Create a sprite and draw an object in it. Add this code to the sprite:

```
onLoad () {  
    _x = 0;  
    _y = 0;  
    speed = 10;  
    _alpha = 100;  
}  
  
onEnterFrame() {  
    endX = _root._xmouse;  
    endY = _root._ymouse;  
    _x += (endX-_x)/speed;  
    _y += (endY-_y)/speed;  
}
```

Now your mouse follower will smoothly follow your mouse. Change the options declared onLoad() and experiment.

To disable the option modify the code to this:

```
onLoad () {  
    _x = 0;  
    _y = 0;  
    speed = 10;  
    _alpha = 100;  
}
```

```
onEnterFrame() {  
  if (_root.drag) {  
    endX = _root._xmouse;  
    endY = _root._ymouse;  
    _x += (endX-_x)/speed;  
    _y += (endY-_y)/speed;  
  }  
}
```

As an example see: [mousefollow2.swi](#)

#### 4.18.3. Complex mouse cursor and follower

For an example of a complex mouse cursor and follower see [mousefollow3.swi](#). The original file was created by Swish Zone. It uses the `Mouse.hide();` command to hide the real mouse cursor. Study it for a better understanding.

### 4.19. Duplicating and removing sprites

We duplicate sprites using the `duplicateSprite()` function. This lets us clone the actual sprite (makes instance of sprite, creates exact sprite) while the sprite is playing.

The syntax for `duplicateSprite()` is:

```
spritename.duplicateSprite(newname, depth)
```

it has two arguments `newname` and `depth`

`newname` = this is the first argument, we will put a new name here, for example your actual sprite name is **mySprite**, then you name your duplicated sprite **mySprite2**.

`depth` = this is the instance number.

We have to give names to the newly created sprites, because later we can remove them. It is mostly efficient when creating games.

Depth is basically the instance number of the duplicated sprite, for example if you have a sprite named as **mySprite** and you want to duplicate this sprite each time you click on a button, you can use the following script (on the buttons action)

```
on (release) {  
  _root.mySprite.duplicateSprite("mySprite2",0);  
}
```

Then it will make only one instance of that sprite, each time you will click on button it will make one instance (duplicated sprite) on it, and if you drag your sprite to any other location and again click on button, the sprite will be removed. That's why we use `depth`, for creating as many instances we want.

To create many duplicated sprites, we will do the following:

Create a new sprite and put some text in it, name it as **mySprite**. Make a button directly on the scene and in its action, paste the following script.

```
on (release) {  
  _root.mySprite.duplicateSprite("duplicatedSprite",depth);  
  depth++;  
}
```

On **mySprite** 's action write the following code.

```
onFrame (1) {  
  depth=1;  
}  
  
onSelfEvent (press) {  
  startDragUnlocked();  
}  
  
on (release) {  
  stopDrag();  
}
```

The new thing is the "depth++;" that this simply adds one more to depth variable.

```
onSelfEvent (press) {  
  startDragUnlocked();  
}  
on (release) {  
  stopDrag();  
}
```

This code is used to drag **mySprite** .

To remove any duplicated sprite we will use the RemoveSprite() function.

This function removes any instance of the duplicated sprite; it simply removes the latest instance of the duplicated sprite.

Add a new button in above example, and on its action paste the following code

```
on (release) {  
  _root.duplicatedSprite.RemoveSprite();  
}
```

It will simply remove the instances of sprite **mySprite**.

To check this, first click on button, which duplicates **mySprite**, 2 or 3 times and change their positions and then click on the button which removes the sprite.

Now test your movie. It should work. For an example made by me, see [duplicate.swi](#).

## 4.20. Drawing with SwishScript

Did you know you can draw with SwishScript? Yes you can. You can draw lines using the `lineStyle` command.

Expression: `lineStyle(line_width, line_color, line_alpha)`

To view an explicit example, view [drawing.swi](#). You don't need any objects in the scene. Just add this code to your scene:

```
onLoad () {
  _root.lineStyle(0, 0x000000, 100);
}
on (press) {
  draw = true;
  startX = _root._xmouse;
  startY = _root._ymouse;
  _root.moveTo(startX,startY);
}
on (dragOut,releaseOutside,keyPress("<Space>")) {
  draw = false;
}
onEnterFrame() {
  if (draw) {
    newX = _root._xmouse;
    newY = _root._ymouse;
    if ((newX != startX) or (newY != startY)) {
      _root.lineTo(newX,newY);
      // reset location for new time
      startX = newX;
      startY = newY;
    }
  }
}
```

Drawing starts when you left click the movie and it stops when you hit the space key. The line's width in the movie is 0, the color is black (in hexadecimal code 000000), and it's fully opaque (100%). Play with this value to see changes.

## 4.21. Internet actions and browsing

SwishMax has integrated series of commands useful for Internet browsing and actions.

### 4.21.1. `getUrl()`

**Syntax:** `getURL(url [,window])`

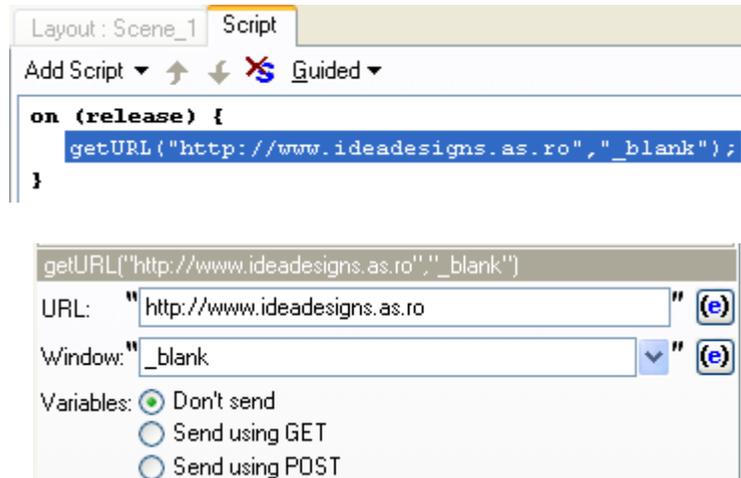
**Arguments :**

URL: Name of the URL to load (text / string).

window: Optional parameter defining which Frame to load URL into. One of "\_self", "\_parent", "\_top" or "\_blank".

**Function description:** Loads the specified URL into the specified Frame. Note that Frame is an HTML Frame, not a Movie Frame in this situation.

**Example:**



#### 4.21.2. fscommand()

**Syntax :** fscommand("command", "Arguments")

##### Arguments

command: A string passed to the host application for any use or a command passed to the stand-alone Flash Player.

Arguments: A string passed to the host application for any use or a value passed to the Flash Player.

##### Description

Action: Allows the Flash Movie to communicate with:

- The .exe Flash Player.
- The program hosting the Flash Player, such as a Web browser.
- Other programs that can host ActiveX controls such as Visual Basic or Visual C++.

**Usage 1:** To send a message to the Flash Player, you must use predefined commands and Arguments. The following table shows the values you can specify for the command and Arguments Arguments of the fscommand Action to control a Movie playing in the stand-alone Flash Player (including projectors):

Command	Arguments	Purpose
quit	None	Closes the projector

fullscreen true or false Specifying true sets the Flash Player to full-screen mode. Specifying false returns the player to normal Menu view

`allowscale true or false` Specifying `false` sets the player so that the Movie is always drawn at its original size and never scaled. Specifying `true` forces the Movie to scale to 100% of the player

`showmenu true or false` Specifying `true` enables the full set of context Menu items. Specifying `false` dims all the context Menu items except About Flash Player

`exec Path to app.` Executes an application from within the projector

`trapallkeys true or false` Specifying `true` sends all key events, including accelerator keys, to the `onClipEvent(keyDown/keyUp)` handler in the Flash Player

**Usage 2:** To use the `fscommand` Action to send a message to a scripting language such as JavaScript in a Web browser, you can pass any two Arguments in the command and Arguments Arguments. These Arguments can be strings or expressions and are used in a JavaScript function that 'catches', or handles, the `fscommand` Action.

In a Web browser, the `fscommand` Action calls the JavaScript function `movienamename_DoFScommand` in the HTML page containing the Flash Movie. The `movienamename` is the name of the Flash Player as assigned by the `NAME` attribute of the `EMBED` tag or the `ID` property of the `OBJECT` tag. If you assign the Flash Player the name `myMovie`, the JavaScript function called is `myMovie_DoFScommand`.

**Usage 3:** In Visual Basic, Visual C++ and other programs that can host ActiveX controls, `fscommand` sends a VB event with two strings that can be handled in the environment's programming language.

### Samples

In the following example, the `fscommand` Action sets the Flash Player to scale the Movie to the full monitor screen size when the button is released.

```
on(release){
    fscommand("fullscreen", true);
}
```

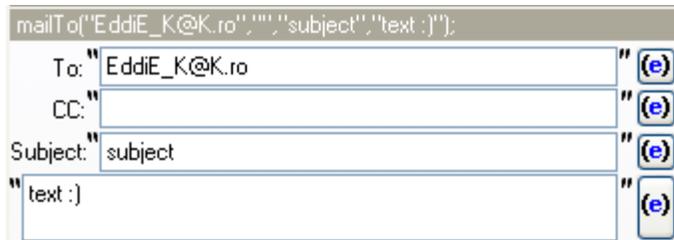
### 4.21.3. JavaScript ()

Use this Action to directly execute some JavaScript. For example if we want to change the border of a movie use:

```
on (release) {
    javascript('document.bgColor = "#000000";');
}
```

### 4.21.4. Mailto ()

Use this Action to send email.



When the Mailto Action is executed, the Flash Player will start the default email client (client's default email program). The fields you have specified will be already filled in. The user can edit any of these or fill in any that are blank and then send the message.

To: You can put in the email address of the recipient.

CC: You can put in the email address of the recipient of a copy of the message.

Subject: You can supply a subject line for the message.

Message body: You can supply text for the message. New lines can be embedded for formatting.

In the example above, the Movie will allow the user to send an email to [EddiE\\_K@K.ro](mailto:EddiE_K@K.ro).

To view an example of all the actions specified in chapter 4.21., view: [inet\\_actions.swi](#).

**\*Note\*** Some text used in chapter 4.21. is taken from the SwishMax help.

## 4.22. Changing anti-aliasing of your movie at runtime

Anti-aliasing has an impact concerning the quality of your movie. This can be changed at runtime and can be done in two ways. It is the same as choosing quality of the movie, low quality → no anti-aliasing and high quality → anti-aliasing activated.

The first method is by using two buttons to set and remove anti-aliasing. It is done by using the:

`_highquality = value`

, command. Value can be 0 or 1. 0 means low quality and 1 means high quality

Sample: `_highquality = 1; // apply anti-aliasing`

Another method used, to actually toggle anti-aliasing is:

`toggleHighQuality()`

By applying this to a button, it will toggle anti-aliasing.

Sample:

```
on(release) {
    toggleHighQuality();
}
```

For a ready made example view: [antialiasing.swi](#).

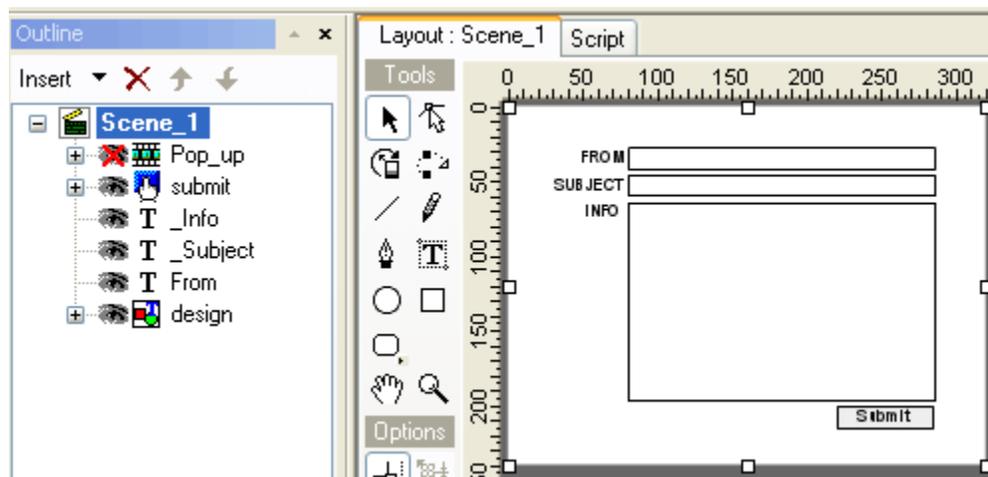
## 5. Integrating SwishMax with PHP

One of the power features of SwishMax is the possibility of its integration with PHP (Hypertext Preprocessor). This allows SwishMax to interface with a database and add more functionality. To use the following examples in this chapter, you must have a server (hosting) that has PHP support.

### 5.1. Creating a mail form

A mail form allows users to send mail to you directly from your website. This is a great thing, saving your potential customer precious contacting time. As hard as you might think to do this, it isn't.

We will create a form like in the image below.



We have the **design** group, that contains the rectangles that surround the **\_info**, **\_subject** & **from** input texts. All the input boxes have the target checkbox selected and as variable, **\_info** has "**InfoVariable**", **\_subject** has "**SubjectVariable**" and **from** has "**FromVariable**".

Now we must create the php file. Create a new text document and add the following code to it:

```
<?
mail("Youremail@site.com", "$SubjectVariable", "$InfoVariable",
    "From: $FromVariable");
?>
```

, where [youremail@site.com](mailto:youremail@site.com) is the email that will receive the mail. Save it as mailform.php.

Now we have the button submit that has the following code:

```
on (press) {
    Pop_up.gotoAndPlay("Here");
    loadVariables("mailform.php", 'POST');
}
```

This code allows SwishMax to load the variables (marked with a \$ in the php file) and post the ones we have declared in SwishMax.

The **pop\_up** sprite just tells us that the mail has been successfully sent and when we press the **OK** button we have the following code:

```
on (press) {
  _root.FromVariable = "";
  _root.SubjectVariable = "";
  _root.InfoVariable = "";
  gotoAndPlay("Not_Here");
}
```

, this clears the input boxes and removes the **pop\_up** sprite.

To see the complete example, open **mailform.swi** and **mailform.php** located in the Mailform folder.

## 5.2. Creating a click counter (view the number of visitors on your web page)

A click counter shows the number of visitors that visited your site. This can easily be created in SwishMax.

First of all create a dynamic textbox, named **\_counter**. Now check the target button and in the advanced panel set the variable to **counter**. For your **scene** add the following code:

```
onLoad() {
  this.loadVariables("counter.php");
  stop();
}
```

Now, create a new text file and insert the following code in it:

```
<?

$filename="counter.txt";
if(!(@chmod($filename,0666)))
  report("Cannot write to file - $filename");

$counts=intval(loadfile($filename));
$counts++;
if(savefile($filename,$counts))
  report("$counts");
else
  report("Cannot write to file - $filename");

function savefile($fname,$data) {
  if(!file_exists($fname))
    report("File not found - $fname");
  $f=fopen($fname,"w");
  fwrite($f,$data);
  fclose($f);
  return true;
}
function loadfile($fname) {
  if(!file_exists($fname))
    report("File not found - $fname");
```

```

    $f=fopen($fname,"r");
    $data=fread($f,filesize($fname));
    fclose($f);
    return $data;
}

function report($msg) {
    die("&counter=$msg&");
}

?>

```

Save the file as counter.php. Create a blank text file and name it counter.txt. Export the swi file to swf, upload counter.swf, counter.php and counter.txt to your PHP enabled server in the same directory and run counter.swf.

To view the examples created by me, see the Counter folder, containing [counter.swi](#), [counter.php](#) and [counter.txt](#).

### 5.3. Logging IP's

This example shows you how to communicate with php and send data (such as IP) to Swish. And after displaying the IP in SwishMax it saves it to an ip.txt which you can open up.

To do this, create a new file. In it place a dynamic text. Name it "**ipdisplay**" and check the target checkbox. Set the variable for the text: **myvar**. Create a button and add the following code to it:

```

on(release) {
loadVariables("ip.php",'POST');
}

```

Name this button **getIP**. Now create another button and name it **Openfile**. Add the following code to it:

```

on (release) {
    getURL("ip.txt","_blank");
}

```

This ends the SwishMax part. Now for the php file, create a new file and name it ip.php. Add the following code to it:

```

<?

$ip = $REMOTE_ADDR;
echo "&myvar=$ip";
$filename = 'IP.txt';

$f = fopen( 'ip.txt', 'a+' ) or exit( 'Please check permissions. You may need to Chmod 777' );
fwrite( $fp, $ip."n" );
fclose( $fp );

?>

```

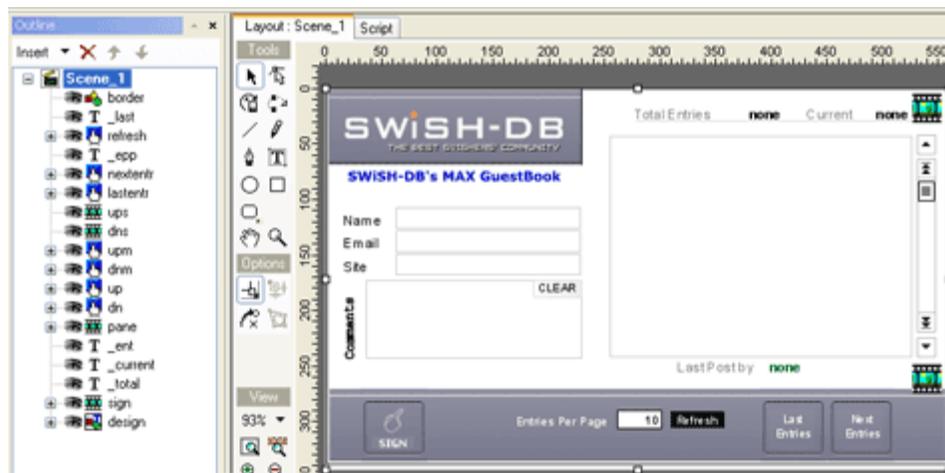
Save your php file. Now export your swi to swf, Upload the swf and php to a server that has PHP and run your movie. It should work.

To see the examples created by me, check the LogIP directory that contains [logIP.swi](#) and [ip.php](#).

#### 5.4. Creating a guestbook

One of the most challenging experiences in SwishMax is to create a guestbook. Of course variables are sent from SwishMax, also the design is made here, and the server part and file writing is done in the PHP file.

A guestbook is a program that let's the visitor comment on your site, or tell their opinions. You can open the [guestbook.swi](#) file located in the PHP folder to change the design of the guestbook.



If you haven't got the hang of SwishMax yet, I advise you to stick only to the **design** groups created in the movie and modify them. They contain designing shapes and texts.

It is pretty hard to create this from scratch, so I am not going to explain all steps. I am just going to say that the variables are stored in the input boxes and then sent to the PHP file. The PHP file writes all the info in MAXGbook.txt file (blank in the beginning). After that, it reads it through PHP and sends the variables back to SwishMax. This is the principle of the Guestbook.

An interesting option in the PHP file is the word filtering, where you can add curse words that you don't want to be displayed.

Open the file, dissect it and learn from it. Also studying the PHP file doesn't hurt too. If you read carefully the book so far, you should understand all actions in the book.

If you have any questions regarding this subject, please visit [www.swish-db.com](http://www.swish-db.com) for support.

**\*Info\*** Original coding and design by Ali Imran & Ali Roman.

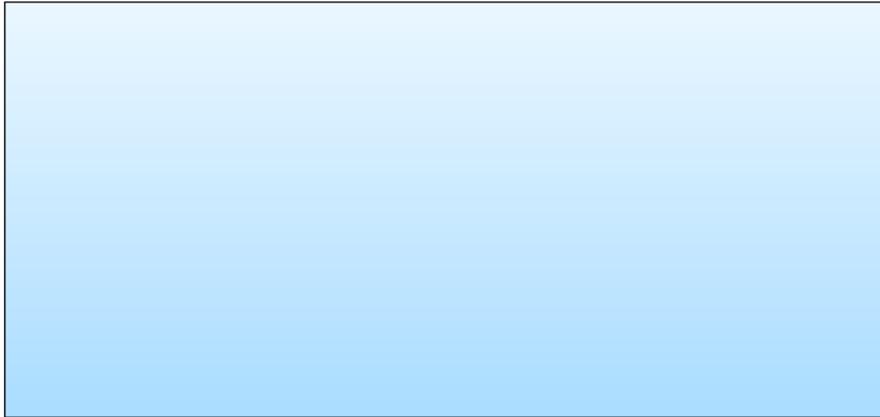
## 6. Creating web pages in SwishMax

Almost the whole purpose of this book is to teach you the basic things you need to build a great web page. As following I'll present the making of a simple web page and then more complicated web pages. As you have got the hang of SwishMax by now, you just need an idea to transform it into reality.

### 6.1. Creating a simple web page

Create a new movie. Make its width = 640 and height= 300

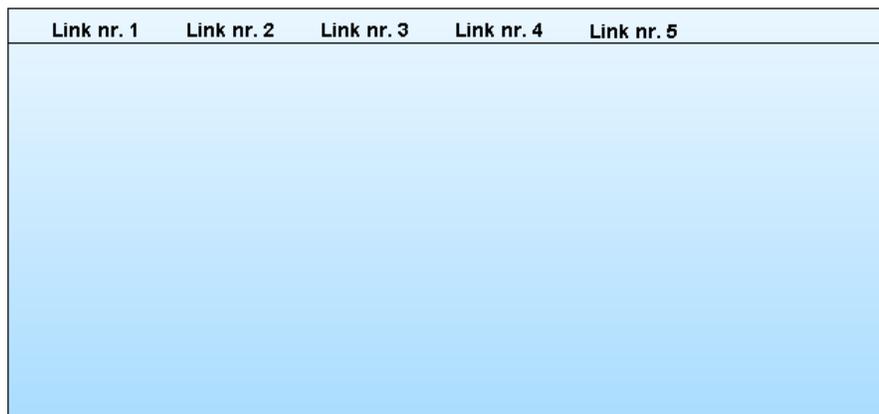
Create a rectangle, set the anchor point to Top Left and set the X=0; Y=0; W=637; H=297. Add a line to it and select linear gradient. Make the gradient fade from AADDFF to EAF7FF both with an alpha of 100. Modify the gradient, so it looks like in the image below. Name it **background**.



Copy paste **background** and modify the new shapes X=0, Y=0, W=639; H=299. Make this a solid shape, black color, without the border line and put it behind the **background**. Name this **bshadow**.

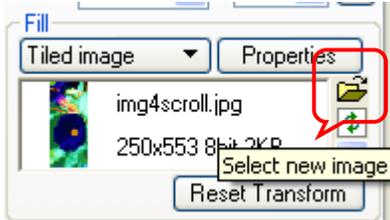
Now create a line and add the following to it: Top left; X=0; Y=25; W=640. Name it **line**.

Create a black static text and set the font to: Arial, 14, pixel font (smooth), bold and add the following text to it: "**Link nr. 1**". Move it to: X=32; Y= 8. Now copy the text and paste this four times. Now modify the texts so you should have **Link nr.1 – 5**. Move them to the top of your movie, where Link nr. 1 resides, preferably higher. Now select all texts and select Align→ Bottom. This makes all your texts at the same level. Now right click again on the selected texts and select Align→Space Evenly→Horizontal. This makes space between texts the same. You should have this by now:



Now group each text separately as button and name them **button1-5**. Activate the “has over state menu” checkbox for all buttons. For the over state, of each button, set a red color to the text. Now when you roll over your buttons, they will change their color to red.

Create a rectangle, and make its width= 543 and height = 235. Set it to tiled image and load **back.jpg** from the sample “simple web site” folder provided with the book. Now rescale the gradient, so only little flowers can be seen. Set the following properties to the image: 8bit pixel zip compression; saturation = 50; resolution=10. Your image file should have 2084 bytes now. Name the rectangle **backgr**, select the target checkbox and group **backgr** as sprite. Name the sprite **Sprite1**. Duplicate **backgr** and make the same x and y as **backgr**. Name this **border**.

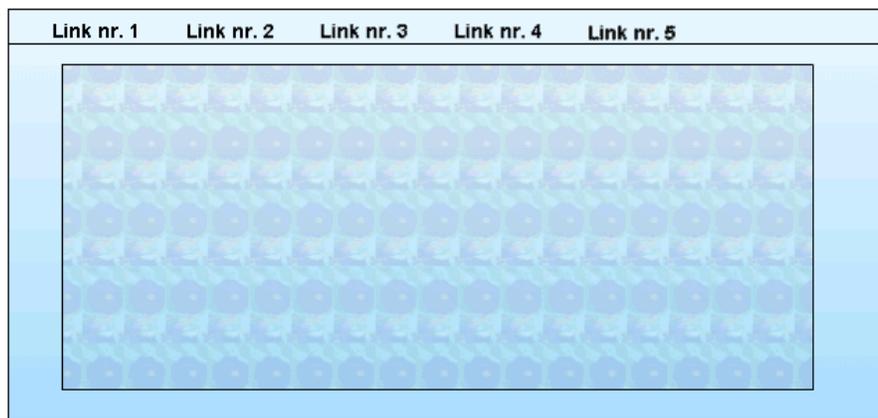


Make the fill style for this NONE and set a line of 0 widths.

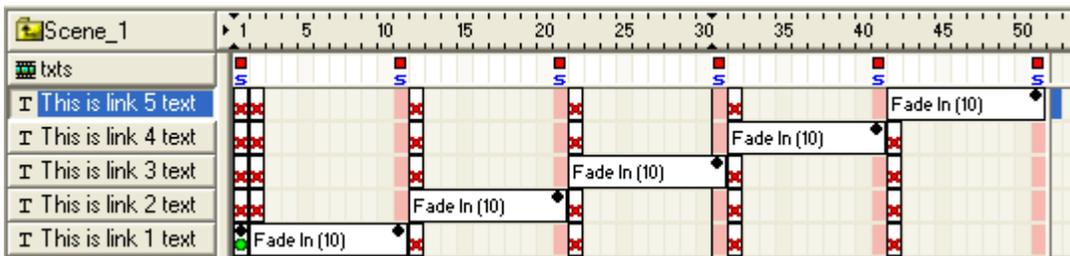
Add the following code to **Sprite1**:

```
onLoad () {
backgr._alpha = 5;
}
```

When you test your movie, it should look like this:



Hide **Sprite1**, so it doesn't get in your way. Now create a static textbox, with any characteristics you want, smaller than **Sprite1**, in the main scene. As text put: “**This is link 1 text**”. Copy and paste this 4 times and set the same x and y for all the texts. Modify the text by modifying “**This is link 2-5 text**”. Now group all these texts as a sprite. Name the sprite **txts**. Now add the effects show in the image below to the sprite **txts**:



Now add the following code to **button1**:

```
on (release) {  
  _root.txts.gotoandplay(2);  
}
```

Add the following code to **button2**:

```
on (release) {  
  _root.txts.gotoandplay(12);  
}
```

Add the following code to **button3**:

```
on (release) {  
  _root.txts.gotoandplay(22);  
}
```

Add the following code to **button4**:

```
on (release) {  
  _root.txts.gotoandplay(32);  
}
```

Add the following code to **button5**:

```
on (release) {  
  _root.txts.gotoandplay(42);  
}
```

Test your movie in HTML browser. Congratulations. You have just made your first site! To view a report on the file, select File→Test→Report.

To see the example made by me, view [simpleweb.swi](#)

## 6.2. Building Idea Designs v4.0 LITE (template)



One of my best sites ever, with an approximate 18000 unique visitors in two months, more than 80000 clicks on links visited, Idea Designs v4.0 Lite revolutionized Idea Designs web site design. The whole site is only 40Kb, so it loads in max 10 seconds even on dial-up. The secret of this page is its simplicity, interactivity and design. It was selected as staff pick on Swish Zone.

I will not help you create the page from scratch, but I will explain the making of the site step by step and special things used in it. If you are thinking I got these results from the start, think again. I visualized the idea and all, but more than an idea was needed to create this. A great site cannot be completed in hours. It takes days, even months to develop. You can find the files that make up the site in the ..Templates/ID v 4LITE folder that comes with the book. These are [contour.swf](#), [ID4LITE.swi](#), [preloader.jpg](#), [rose\\_JPEGGG.jpg](#) and [rose\\_mask.swf](#).

First step: Creating the main design of the page. That means creating the outer border, insider border, and image insertion. Image effects were done in the first steps. How it is done? A contour of the image was created by me in Flash™ and then imported into SwishMax (contour.swf). This could have also been done in Jasc Paint Shop Pro, or Adobe Photoshop. Now I created a linear gradient rectangle that covered this contour and masked the whole sprite, obtaining the nice effect of the moving margins. Another sprite is the rose masked.

Second step: I created the main navigation bar. This was created by a masked rounded rectangle. Then came the buttons, that are made with sprites, that when are rolled over, change their color.

Third step: This is when I added the content. First of all I created one sprite, called **Home\_m**, you can find it in the **display** sprite. This is where I set up all the text and design for the content. I made this appear masked by a circle, for the flashing appearance effect. The next step was easy, duplicating this sprite and changing the content of the texts.

Fourth step: Created the options. This was pretty difficult, because I had no options in mind when I implemented the button. I said I'll improvise as I went along. And that I did. I created a mouse cursor (found in the mouse follower chapter). I also created a shadow for the window; created background color change and a display resize using JavaScript. Background color changing is done with move effects that have change color applied to them. Almost all links and options work with the gotoandStop() or gotoandPlay() script.

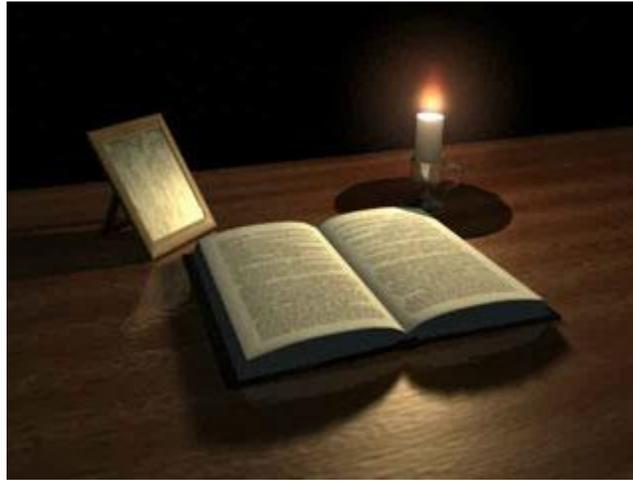
Fifth step: Creating that wonderful image preloader. See the image preloader chapter for more information. Also I fixed last minute bugs and minor adjustments.

By purchasing this book, you have received this template for free. This is for personal use only. You can modify the contents, images, break into pieces, and learn from it. That is its purpose.

Before running the file, make sure that you have the fonts installed on your computer. If not, just copy the fonts that came with the template in your ..Windows/Fonts directory.

If you modify this template and upload it as your own on the internet, please send me a link at [EddiE\\_@K.ro](#), because I would be more than happy to see how you made it better (I hope).

### 6.3. Using the old book template

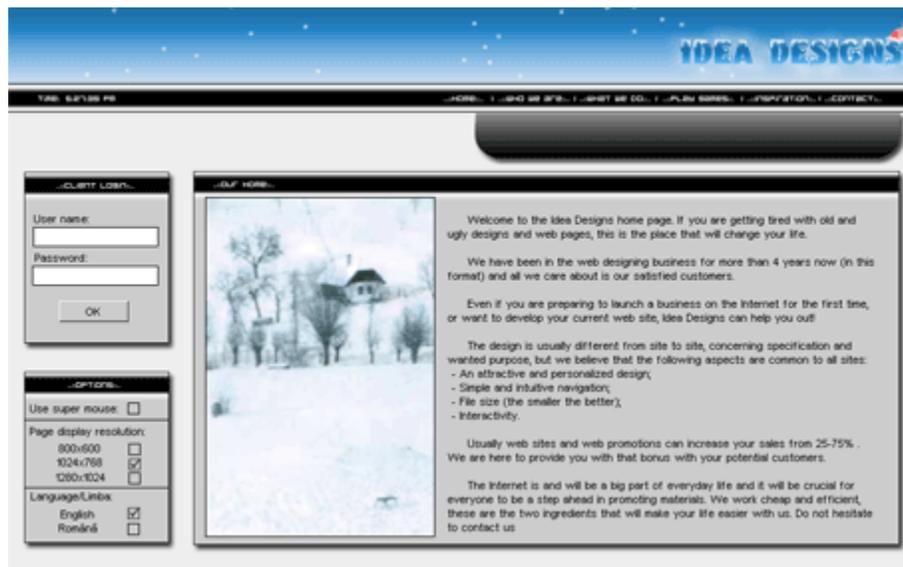


A clear integration of SwishMax with 3D is the old book template. The graphics are made by my brother, Boka Arthur in Maya a revolutionary 3D program. I got the idea “over night”, drew him my idea, and he made it 3D. You can change the text for your liking and also the portrait image. Hope you have fun by learning how it was done. You can study the swi file by opening [old book template.swi](#) .

Before running the file, make sure that you have the fonts installed on your computer. If not, just copy the fonts that came with the template in your `..Windows/Fonts` directory.

If you modify this template and upload it as your own on the internet, please send me a link at [EddiE @K.ro](mailto:EddiE@K.ro) , because I would be more than happy to see how you made it better (I hope).

### 6.4. Using the professional grey template



You can open the file, study it, and modify it for personal use. I think you have some things to learn from it. This was the Idea Designs home page for the December month in 2003. The file is: grey **template.swi**, located in the Templates folder.

Before running the file, make sure that you have the fonts installed on your computer. If not, just copy the fonts that came with the template in your `..Windows/Fonts` directory.

If you modify this template and upload it as your own on the internet, please send me a link at [EddiE @K.ro](mailto:EddiE@K.ro) , because I would be more than happy to see how you made it better (I hope).

## 7. Creating games in SwishMax

Due to the new features implemented in SwishMax (SwishScript), users can create from simple to complicated games, in a very short time. Games are fun to create and also can attract visitors to your webpage. What better way of understanding how games are made than to create our own games.

### 7.1. Creating the “Guess the number” game

Our purpose is to create a guess the number game.

How does it work?

The computer generates a random number from 0-100 (set by the programmer) and you must guess it by 7 tries (set by programmer).

It also displays number of wins, loses, number of tries left and if you lose the number that you didn't guess.

Open your **guess number.swi** file from the Games folder.

I won't go into designing in this case, I will try to explain only the code and how the game works.

You have five dynamic boxes and one input box, as following:

**was** – dynamic text box. Will display the number if it isn't guessed;

**loses** – dynamic text box. Displays the number of loses;

**wins** – dynamic text box. Displays the number of wins;

**tries** – dynamic text box. Displays the number tries left;

**status** – dynamic text box. Display status, if you win or lose;

**number** – input text box. Here you can insert your number, and then press “Okay”.

Buttons:

**start** has the following code:

```
on (press) {
  nr = Math.randomInt(100);
  _root.status= "And the game has begun!";
  nrtries = 7;
  _root.tries = nrtries;
  _root.butons.start._height=0;
  _root.butons.okey._height =22;
  _root.was = "";
}
```

This initializes the game. Generates the random number stored in the variable **nr**. The **status** dynamic text box displays “And the game has begun!” Numbers of tries are defined in the variable **nrtries**. Then it hides the **start** button and shows the **okey** button. Also clears the **was** textbox.

**okey** has the following code:

```
on (press) {
  if (_root.number == nr) {
    _root.status = "You guessed it! :)";
    winz=winz+1;
    _root.wins = winz;
    _root.butons.okey._height =0;
    _root.butons.start._height =29;
    _root.number = "";
  }

  if (_root.number > nr) {
    _root.status = "The number is too big!";
    nrtries = nrtries -1;
    _root.tries = nrtries;
  }

  if (_root.number < nr) {
    _root.status = "The number is too small!";
    nrtries = nrtries -1;
    _root.tries = nrtries;
  }

  if (nrtries ==0) {
    _root.status ="You lost! The number was: ";
    _root.was = nr;
    losez=losez+1;
    _root.loses = losez;
    _root.butons.okey._height =0;
    _root.butons.start._height =29;
    _root.number = "";
  }
}
```

Pressing the **okey** button, verifies the number you entered.

If it is guessed, it displays so in the **status** text, adds one to your **wins**, hides the **okey** button, shows the **start** button and clears the **number** field.

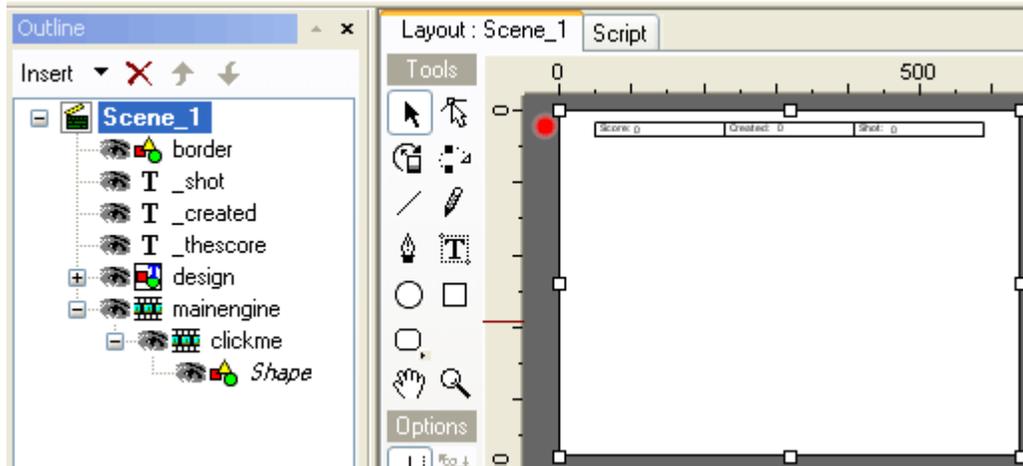
If it is too big or too small, it subtracts one from your number of tries (**nrtries**). Also tells you in the **status** how's the number relatively to the number you entered.

If the number of tries reach zero, it displays that you lost, and shows the number. Also adds one to your losses. Hides the **okey** button and displays the **start** button. Also clears the **number** field.

As you can see the code is not that complicated. Just give it a try. I dare you to try and make a two player game.

## 7.2. Creating a simple click shooter game

The main idea of click shooter games is to give you points when you “shoot” an object. I will explain how to make such a game. For support, open your [clickshoot.swi](#) , located in the ...Games/ directory in the swi examples.



You will see the following layout. We are not interested in the design part, but more interested in how the game works. The objects that make the game functional are the following:

Dynamic texts: **\_shot** ; **\_created** ; **\_thescore**  
 Sprites: **mainengine** and **Shape**

You will see the following code for the **mainengine**

```
onLoad () {
  Screated = 0;
}

onFrame (1) {
  _root.mainengine.clickme.duplicateSprite("duplicatedSprite",depth);
  depth++;
  duplicatedSprite._x = Math.randomInt(600);
  duplicatedSprite._Y = Math.randomInt(460);

  screated +=1;
  _root._created = screated;
}

onFrame (20) {
  duplicatedSprite.RemoveSprite();
  gotoAndPlay(1);
}
```

Explanation: When the movie loads, we declare the **Screated** variable. Now on frame 1 we tell the movie to duplicate the sprite and randomly position it in our movie. After the sprite has been created, add 1 to the **screated** variable and display it in the **\_created** textbox.

On frame 20 (the larger this number is, the slower will the sprites be removed), it removes the created sprite, and then it restarts the duplication process.

The **shape** object is nothing more than a gradient circle. You can put any object here, from button to sprite, adding the following code to it:

```
onLoad () {
yourscore = 0;
shot = 0
}

on (press) {
_root.mainengine.duplicatedSprite.RemoveSprite();

yourscore += 33;
_root._thescore = yourscore;

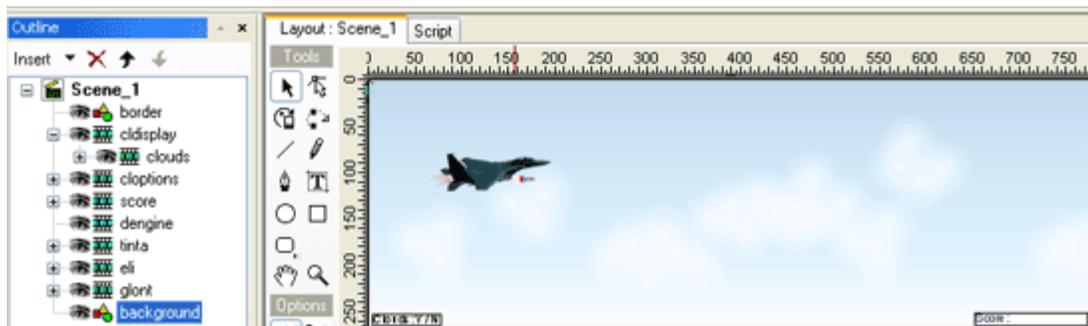
shot +=1;
_root._shot = shot;
}
```

When the shape loads, it declares the **yourscore** and **shot** variable. Now, when we click on the object, it removes it and adds 33 to your score (then it is displayed) . Also number of shot objects increases and it is displayed in the **\_shot** textbox.

This is just the basic code. You could work on it, so it creates only a certain number of sprites and display your score, shots as a final result. Give it a go. It's a simple if usage.

### 7.3. Creating a more complex shooting game

As the name of the chapter says it, it's all about a complex shooting game. To view the ready made example, view: **complex shooter.swi** from the Games folder.



This is your main movie, a combination of sprites that contain a lot of goodies. Test your movie. Watch out for this movie because it is processor hungry. I developed the game on an Athlon XP 1900+ (1, 6 GHz) and when I start the clouds options, processor usage in windows goes up to 100%, even if I stop anti-aliasing.

The game isn't that complicated as it may look. The complication starts from the design. Let me explain the code on **Scene\_1**.

```
onLoad () {
  tras="true";
  bullet._alpha=0;
}

onEnterFrame() {
  if (Key.isDown(Key.space)) {
    if (tras eq "true") {
      bullet._x=eli._x;
      bullet._y=eli._y + 10;
      bullet._alpha=100;
      tras="false";
    }
  }
  if (tras eq "false") {
    if (bullet._x<750) {
      bullet._x = bullet._x+30;
    }
  }
  if (bullet._x>750) {
    bullet._alpha=0;
    tras="true";
  }

  if (Key.isDown(Key.UP)) {
    if (eli._y>40) {
      eli._y=eli._y-5;
    }
  }

  if (Key.isDown(Key.DOWN)) {
    if (eli._y<250) {
      eli._y=eli._y+5;
    }
  }
  if (Key.isDown(Key.RIGHT)) {
    if (eli._x<280) {
      eli._x=eli._x+5;
      eli._rotation=2;
    }
  }
  if (Key.isDown(Key.LEFT)) {
    if (eli._x>85) {
      eli._x=eli._x-5;
      eli._rotation=-2;
    }
  }

  if (!Key.isDown(Key.LEFT) and !Key.isDown(Key.RIGHT)) {
    eli._rotation=0;
  }
}
```

When we load the movie, we declare the variable **tras** and assign it "true". Also we tell **bullet** to be transparent.

All following actions are in the `onEnterFrame()` code:

```

if (Key.isDown(Key.space)) {
  if (tras eq "true") {
    bullet._x=eli._x;
    bullet._y=eli._y + 10;
    bullet._alpha=100;
    tras="false";
  }
}
if (tras eq "false") {
  if (bullet._x<750) {
    bullet._x = bullet._x+30;
  }
}
if (bullet._x>750) {
  bullet._alpha=0;
  tras="true";
}

```

If the “Space” key is pressed, do the following:

Check if **tras** is true, if so, move the **bullet** after the **eli** (plane) and assign **tras** the value “false”

If **tras** is “false”, and the **bullet** ‘s X is smaller than 750 (end of movie), move the **bullet** right with 30 pixels per frame.

If the **bullet** ‘s X is larger than 750, then hide the **bullet** . (alpha=0 ;transparent). Set **tras** to be “true” enabling us to shoot again.

```

if (Key.isDown(Key.UP)) {
  if (eli._y>40) {
    eli._y=eli._y-5;
  }
}

```

If we press the UP arrow key,  
check if the **eli** ‘s y is bigger that 40, and if so  
move the **eli** up 5 pixels

```

if (Key.isDown(Key.DOWN)) {
  if (eli._y<250) {
    eli._y=eli._y+5;
  }
}

```

If we press the DOWN arrow key,  
check if the **eli** ‘s y is smaller than 250, and if so  
move the **eli** down 5 pixels

```

if (Key.isDown(Key.RIGHT)) {
  if (eli._x<280) {
    eli._x=eli._x+5;
    eli._rotation=2;
  }
}

```

If we press the RIGHT arrow key,  
 check if the **eli** 's x is smaller than 280, and if so  
 move the **eli** Right with 5 pixels;  
 rotate **eli** clockwise with 2 pixels

```
if (Key.isDown(Key.LEFT)) {
  if (eli._x>85) {
    eli._x=eli._x-5;
    eli._rotation=-2;
  }
}
```

If we press the RIGHT arrow key,  
 check if the **eli** 's x is bigger than 85, and if so  
 move the **eli** Left with 5 pixels;  
 rotate **eli** counter clockwise with 2 pixels

I have created the **cdisplay** sprite (contains the moving clouds) in order to remove the clouds when I press the buttons located in the **cloptions** sprite.

To control the duplication of the enemy sprites, I have created the duplication engine, stored in the **engine** sprite.

```
onFrame (1) {
  _root.tinta.duplicateSprite("duplicatedSprite",depth);
  depth++;
  _root.duplicatedSprite._Y = Math.randomInt(200)+50;
  _root.tinta.duplicatedSprite.gotoAndStop(1);
}
onFrame (100) {
  _root.duplicatedSprite.RemoveSprite();
  gotoAndPlay(1);
}
```

On the first frame, duplicate the enemy (**tinta**), position it randomly on the Y axis and send it to play the first frame (this makes the enemy plane fly in).

On frame 100 (1 second normally, but you need a super fast processor to run this game in real time), remove the duplicated sprite and go to first frame (duplicate it again).

For the **tinta** sprite (enemy airplane) we have the following code:

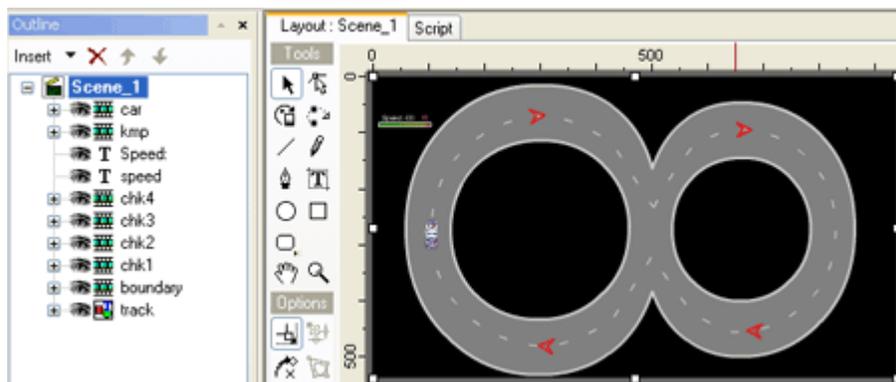
```
onEnterFrame() {
  if (_root.bullet.isNearThis()) {
    _root.score.nimerit++;
    gotoAndStop(2);
  }
}
onFrame (1) {
  stop();
}
onFrame (2) {
  stop();
}
```

On enter frame we check if the **bullet** is near the enemy plane and if so, add 1 to our score (**nimerit** variable), and go to the explosion sprite located also in the **tinta** sprite (frame2)

Feel free to experiment on the ready made movie, or create your own example. I bet you surely can do better than me. Also you can add enemy shooting, lives and I am sure you can think of more.

## 7.4. Creating a car rally game

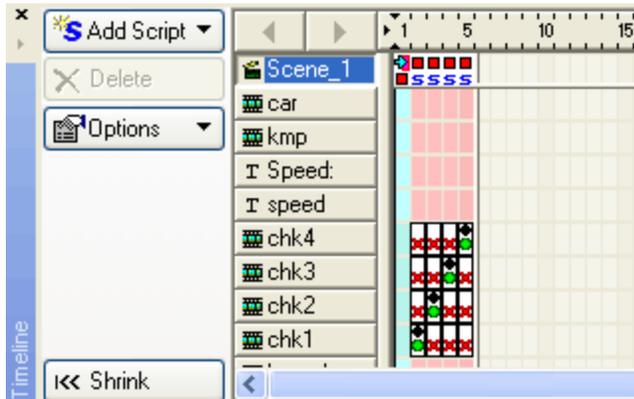
Next, you'll learn how to create a car rally game. You know, you have the little car and have to drive it along the track. Open the **Car rally.swi** located in the Games folder to follow my explanations more closely.



For **Scene\_1** we have the following code:

```
onLoad () {
  _root.chk1._visible = true;
  _root.chk2._visible = false;
  _root.chk3._visible = false;
  _root.chk4._visible = false;
}
onFrame (1) {
  gotoAndStop(2);
  stop();
}
onFrame (2) {
  stop();
}
onFrame (3) {
  stop();
}
onFrame (4) {
  stop();
}
onFrame (5) {
  stop();
}
```

On load, visibilities of the checkpoints are defined. You may ask yourselves why I added the following (see main timeline):



I am referring to the stop actions and the lots of place and removes actions. Well this is for the checkpoints. I wanted to make the checkpoints change their visibility by using the `_visibility` option. But to my surprise if a checkpoint is invisible, it still works, so I had to use place and remove actions, and `gotoandstop()` actions for the checkpoints.

The main code is found in the car sprite:

```
onEnterFrame() {
    spe = speed*2;
    spee = speed*9;
    _root.speed = int(spee);
    _root.kmp.gotoAndStop(int(spe));

    if (int(spe)==0) {
        _root.kmp.gotoAndStop(1);
    }
    if (int(spe)==-1) {
        _root.kmp.gotoAndStop(19);
    }
    if (int(spe)==-2) {
        _root.kmp.gotoAndStop(20);
    }
    if (int(spe)==-3) {
        _root.kmp.gotoAndStop(21);
    }
    if (int(spe)==-4) {
        _root.kmp.gotoAndStop(22);
    }
    if (int(spe)==-5) {
        _root.kmp.gotoAndStop(23);
    }
    if (int(spe)==-6) {
        _root.kmp.gotoAndStop(24);
    }
    if (int(spe)==-7) {
        _root.kmp.gotoAndStop(25);
    }
    if (int(spe)==-8) {
        _root.kmp.gotoAndStop(26);
    }
}
```

```

    if (Key.isDown(Key.UP)) {
        speed += .5;
    }
    if (Key.isDown(Key.DOWN)) {
        speed -= 0.25;
    }
    if (Math.abs(speed)>20) {
        speed *= .7;
    }
    if (Key.isDown(Key.LEFT)) {
        _rotation -= 7;
    }
    if (Key.isDown(Key.RIGHT)) {
        _rotation += 7;
    }

    speed *= .95;
    x = Math.sin(_rotation*(Math.PI/180))*speed;
    y = Math.cos(_rotation*(Math.PI/180))*speed*-1;

    if (!_root.boundary.hitTest(_x+x, _y+y, true)) {
        _x += x;
        _y += y;
    } else {
        speed *= -.6;
    }
}

```

All the code till the key.UP action is used to display the speed in the movie, and control the **kmp** sprite, that contains a bar that goes up and down depending on the speed. After that we have the arrow key codes, that multiply the speed, or modify the rotation of the **car**.

The magic happens here:

```

speed *= .95;
x = Math.sin(_rotation*(Math.PI/180))*speed;
y = Math.cos(_rotation*(Math.PI/180))*speed*-1;

```

This math formula allows the **car** to move as it should.

Also here, we have the collision detection with the sprite **boundary**:

```

if (!_root.boundary.hitTest(_x+x, _y+y, true)) {
    _x += x;
    _y += y;
} else {
    speed *= -.6;
}

```

For the checkpoints we have the following code:

```
onEnterFrame() {  
    if (_root.car.isNearThis()) {  
        _root.gotoAndStop(2→5);  
    }  
}
```

This tells the program to move to the desired frame when the car collides with the checkpoint mark.

I have limited myself to give you the start. Now go ahead and make more levels, add lap times and why not more cars.

**\*Thanks\*** to the Swish Zone forums for the main engine of the game.

## 8. Other examples provided with the book

File name	Author	Description
external txt.swi	EddiE K	External loaded scrolling text
multipletext.swi	EddiE K	Multiple loaded text from one file
balls.swi	EddiE K	Interactive balls
colorchangeonmove.swi	B1ade	Star color change on movement
countdown.swi	B1ade	Complex countdown, letting you set countdown time
cursors_dragging.swi	B1ade	Using multiple cursors for different actions
dogchase.swi	B1ade	A "dog" chasing the mouse
draw_curve.swi	MC	Action script curve drawing
eyes follower.swi	B1ade	Eyes looking at mouse
fading_scripted.swi	EddiE K	Test of scripted fading
following mouse.swi	B1ade	Objects following mouse
killertextscamble.swi	B1ade	Scrambled text
Load_Picture.swi	EddiE K	Dynamically loaded pictures
magnify glass.swi	EddiE K	Magnifying glass
Press-And-Release.swi	EddiE K	Tests if a key is pressed or released
randomcolor.swi	damoon	Assigning random color to an object
rewinding.swi	damoon	Rewinding a sprite
Scripted 3D.swi	B1ade	Test of 3D display in SwishMax
scripted balls.swi	B1ade	Mouse touch- breaking balls
scriptedtypeandtext.swi	B1ade	Swish scripted typewriter effect
SnowFlakes.swi	EddiE K	Scripted snow flakes
supermouse.swi	B1ade	Animated mouse cursor
swapDepths.swi	EddiE K	Test of swapping depths of objects at runtime
window_wooble.swi	MC	Cool window appearance

**\*Thanks\*** I would like to thank B1ade for his wonderful conversions from Flash to SwishMax. We can all learn from his examples. Thanks also go to damoon and MC of swish DB for their free examples. I go by the nick of EddiE K, so I am not going to thank myself ☺ . My examples can be opened in SwishMax build date: 2003.10.30. For the others you need updated versions of SwishMax.

**\*Want more?\*** For more than 180 great examples go to the [www.Swish-Db.com](http://www.Swish-Db.com) forums.

## 9. Tips and Tricks

### **My movie doesn't work right although I followed all your steps. Why?**

I advise you to export all your made movies to Flash 6. For this go to Export panel and view the export options for SWF.

### **I am looking for some designing tips. Can you help me?**

1. Take a look at other sites and gather ideas from those at first (just ideas, don't copy them).
2. Then decide on the content of your site. What are you going to be presenting to the world.
3. Decide what colors and images you want use. These colors will somehow match your content. For instance, gaming sites usually use the colors red, black, and dark blue. The images on them are usually darkish.
4. Organize your data into sections. Where are you going to put the data about you? About the site? Where will the files be? The news, your projects, and so on.
5. Start laying out your site, perhaps on paper at first. Then draw it in swish or whatever program you're using. (If you're a good drawer, you could scan your picture/web page you drew on paper and insert that into your web design program)
6. Decide what effects you want to use. If you want to grab visitor's attention, you got to add some spice to your site, whether it's html or swish animation (buttons, images rotating, lines moving in and out, etc). Remember not to overuse the effects. Use them wisely.
7. Make your design simple and attractive. Also do not forget about file size. The smaller the file size is the better. A lot of users have dial up, that means they have a download speed of max 5,6 Kb/s. Imagine a 1 Mb file loading. (1 Mb = 1000 Kb).

### **What are the main types of text available in SwishMax?**

#### **- pixel fonts**

Pixel fonts are special fonts that even at 8px display very clearly. Also they give a professional look to your web site. To display them correctly, make sure:

- They are justified to "top left" and not "center" or any other;

- You have selected the render as pixel font option for texts: 

- The "X" and "Y" coordinates are set to whole numbers (integers)

Example: "X"=30,"Y"=40 not "X"=30.1;"Y"=40.6;

- Make sure the uniformity of the text is of "100%" horizontal, and vertical.

- This makes your fonts crisp and scales the fonts after a virtual grid.

#### **- vector fonts**

Vector fonts are fonts that are SwishMax renders as vector shapes. Vector fonts can be selected by pressing the  icon, in the text properties. They are especially effective on pixel sizes larger than 16px.

**- device fonts**

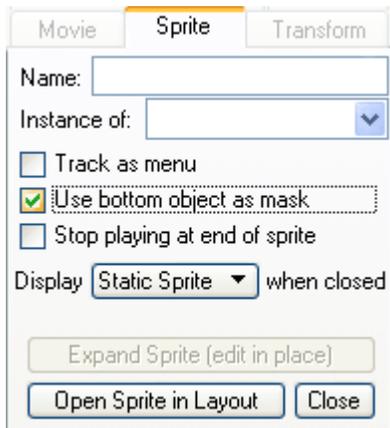
Device fonts are fonts that tend to have the crispiness of a pixel font. Use them for small text. A big disadvantage of this kind of font is that it can't be masked. They can be used by choosing the  icon in the text properties.

**- system fonts**

System fonts are fonts that “come” with SwishMax. These are: `_sans`, `_serif`, `_typewriter`. A big advantage of system fonts is that they display well on all pixel sizes. A big disadvantage is that you cannot mask them.

**I want my loaded external text to display special characters. How?**

Save the text file as UNICODE and not as ANSI (in your text editor). For Notepad, when you click save as, you will see a combo box at the bottom of the option screen, reading Encoding.

**I keep hearing of this sprite masking think... How can it help ME?**

Sprite masking, as you know can be activated by pressing the “Use bottom object as mask” checkbox. The advantages of a masked sprite are that you can animate the object that is masked and you can also animate the mask. This is very useful for creating cool effects and hides unwanted objects at runtime.

Try to make a movie that contains such a sprite and animate first the object and then animate the mask. See the results.

**I want to make great animated buttons. How?**

There are two ways to create animated buttons. One is by using buttons and the other one by using sprites.

**- using buttons**

Create a sprite (with an object in it), and add some animation to it. Now group it as button. Select has separate mouse over state. You have an animated button now. Remove the animation on the UP state of the button. Now when you roll over your button, it will animate.

**- using sprites**

Create a sprite (with an object in it), and add a stop effect on the first frame. Now create another effect starting from frame 2. Add a stop effect at the end. Now, on the sprites code, just tell it to go to the desired frame (animation frame) when rolled over.

To see an explicit example, view [ani\\_buttons.swi](#).

**I update my website very frequently. What can I do so users have the latest version of my site?**

If your website has content that is changing on a regular basis, and you have visitors that come to your site on a regular basis, you want to ensure that your new content is visible to these visitors each time they come. Most browsers will cache webpage's to speed up loading on each subsequent visit. With constantly changing content, this could be a problem for your regular visitors.

One solution is to use Meta tags to prevent pages from being cached. In this way, every time someone visits your site, they will be getting the latest content delivered to their browser.

```
<HEAD>  
<META http-equiv="Pragma" content="no-cache">  
</HEAD>
```

This simple line of code will tell the browser not to cache the page. This will slow down the loading of the page each time a visitor comes (it reloads the page entirely), but the benefit is the displaying of the fresh content.

**I want to load a movie into SwishMax. How?**

Well you can't do this directly. One way is to make a swf file from your mpeg/avi/wmv movie file using Flix (<http://www.wildform.com>), and then use load it in your main movie.

Another way (in case you have Animation Pro) is to load your movie there, and export it as .gif. Now you can import this into SwishMax just like an image.

**Don't forget about Machintosh users when linking to the flash plugin at Macromedia.**

I've seen already a lot of sites where in there intro a link is provided to Macromedia for downloading Flash player. You have to know that a lot of people are not able to find that player on the site of macromedia because they can hardly surf on the net. So I suggest when making a link, make the link directly to the exact page where they can download the player and DO NOT FORGET that there are still a number of people who are using a Macintosh computer. The flash player for Windows is not compatible for MAC and so, when providing a link to macromedia, provide one for WIN and another for MAC.

Windows Flash player link:

[http://www.macromedia.com/shockwave/download/download.cgi?P1\\_Prod\\_Version=ShockwaveFlash](http://www.macromedia.com/shockwave/download/download.cgi?P1_Prod_Version=ShockwaveFlash)

Macintosh Flash player link:

<http://www.macromedia.com/shockwave/download/alternates/>

**SwishMax crashed while I was working. My file is lost forever!**

Don't worry, there is no problem, just rename you SBK file (you will find this file in the same folder as your SWI) to SWI and the problem is solved. SBK comes from Swish Backup File. This is automatically created when you save your SWI (made changes to it and saved).

**I want to change the display size of my movie at runtime.**

This can easily be done using Java Script. Add the following code to a button:

```
on (release) {  
    javascript("javascript:window.parent.document.body.style.zoom= NUMBER");  
}
```

Where number represents scaling size. If NUMBER =1 then the movie displays normally. Increasing the number enlarges the movie and decreasing reduces the movie size.

For example you create your website for 1024x768. Use .74 as NUMBER to fit on 800x600 display and 1.34 to display it on 1280x1024 monitor resolution.

**Can I swap depths of objects at runtime?**

Yes you can change object layering (depths) by using the `swapDepths()` syntax. For an example please see [swapDepths.swi](#) located in the Other examples folder.

**Can I change the scrollbar color in Internet Explorer?**

Yes, this can be changed by inserting the following code in the html code of your page:

```
<STYLE type="text/css">  
<!--  
BODY {  
scrollbar-face-color:#000000;  
scrollbar-highlight-color:#000000;  
scrollbar-3dlight-color:#000000;  
scrollbar-darkshadow-color:#000000;  
scrollbar-shadow-color:#000000;  
scrollbar-arrow-color:#000000;  
scrollbar-track-color:#000000;  
}  
-->  
</STYLE>
```

Insert it in the <HEAD> tag of your movie. The #000000 represents the hexadecimal code of the color.

**I want an image to explode. How do I do that?**

Select the image and then from the Modify menu, select Break → Break into pieces. Now you can add cool effects to your image like explode!

**\*Thanks\*** go out to Charly Pixels, Damoon and Tim for some of the tips.

## 10. Quick color legend

- – Normal text;
- – SwishScript code;
- – PHP code;
- – HTML & Java Script code;
- – Tips text;
- – Homework text;
- – Notes and important text;
- – Objects and variables;
- – Example files;
- ★ **XX** – More info in the tips and tricks section. See page number XX.

## 11. Legal Information

THIS BOOK IS PROVIDED AS IS WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE WHICH ARE HEREBY DISCLAIMED. IN NO EVENT WILL EDUARD BOKA BE LIABLE FOR ANY SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM POSSESSION, USE, OR MALFUNCTION OF THIS BOOK.

THIS BOOK AND ALL EXAMPLES HEREWITH ARE COPYRIGHTED. ALL RIGHTS ARE RESERVED. NO PART OF THIS BOOK MAY BE COPIED, REPRODUCED, TRANSLATED, OR REDUCED TO ANY ELECTRONIC MEDIUM OR MACHINE-READABLE FORM WITHOUT THE PRIOR WRITTEN CONSENT OF BOKA EDUARD, [EDDIE\\_K@K.RO](mailto:EDDIE_K@K.RO). ALSO THE EXAMPLES ARE COPYRIGHTED, BUT THEY ARE ALLOWED TO BE REPRODUCED, COPIED, AND TRANSLATED BY THE USER OF THE BOOK FOR **PERSONAL USE**. DISTRIBUTION OF THESE FILES IN ANY WAY, THAN SPECIFIED, RESULTS IN SEVERE LAW PROSECUTION.

© 2004 IDEA DESIGNS & SWISH DB PARTNERS

ALL RIGHTS RESERVED.

## 12. Bibliography

**SwishMax help** – [www.SwishZone.com](http://www.SwishZone.com) – used it for general development

**Swish Db forum** – [www.Swish-Db.com](http://www.Swish-Db.com) – used it for general development

**Swish-Tutorials forum** – [www.Swish-Tutorials.com](http://www.Swish-Tutorials.com) – used it for the volume and balance info

**Swish Zone forum** – [www.SwishZone.com](http://www.SwishZone.com) – used for the car rally basic engine



Copyright 2004 - Boka Eduard, Idea Designs director and Swish DB Partners

[www.IdeaDesigns.as.ro](http://www.IdeaDesigns.as.ro) , [www.Swish-Db.com](http://www.Swish-Db.com)