Working in WireFusion 3

Demicron

The author and the publisher make no representation or warranties of any kind with regard to the completeness or accuracy of the contents herein and accept no liability of any kind including but not limited to performance, merchantability, fitness for any particular purpose, or any losses or damages of any kind caused or alleged to be caused directly or indirectly from this book.

Contents

1 Preliminary

1.1 About this manual

This book will describe the basic views, windows and menues of the program. Familiarity with the essential procedures of WireFusion is a requirement for working with WireFusion. This manual covers in some detail what objects are, how you connect them, image manipulation, and the deployment of generated applets.

1.2 Conventions used in this manual

Below you'll find the typographical conventions used in this manual.

- Image in the margin indicates some special information, a **note**.

- Image in the margin, denoting a **tip** making your work in WireFusion easier.
- Text in `typewriter` font indicates an object name.
- Text in *italic* or **bold** indicates certain emphasizes. Text in *italic* also denotes directory or file names.

2 The Concept

Whether you are producing interactive product presentations, panorama views, animated logos, web site navigation controls, games or stunning visual effects, WireFusion delivers the creative tool for which you are looking. WireFusion is a drag-and-drop visual-programming tool developed for creating advanced, interactive and dynamic presentations and animations, in the form of Java applets (see Chapter 4), without knowing a single line of the Java language.

WireFusion is a graphical platform for web presentations, which can be extended with new technologies as they become available. WireFusion makes you more efficient, it helps you to create advanced, small in size and high performance presentations quickly, which also reaches a very large Internet audience with its plugin free result.

WireFusion runs on Java 2, but generates applets for Java 1.1. The importance of this statement is that while you work in the latest Java environment with all of its qualities and features, the generated applets can be used by practically anyone just having Java 1.1 in their browser. You can, hence, reach computers with browsers which do not allow for later installation. Thereby, you can produce for an audience with limited installation capacities, potentially reaching anything from airport Internet kiosks, to companies with policies that do not allow for later installation, or downloading of separate plug-ins.

WireFusion works with pre-programmed functions and operations. The ready-made set is known as WireFusion *Objects*, or just objects. The use and reuse of already made and tested code, ensures small and robust presentations and operations. The idea of programming in WireFusion is very similar to conventional programming techniques. Data or information (commonly known as parameters, such as numbers, strings, colors, images etc.) are sent from one function into another. However, the difference from conventional techniques is that WireFusion does this by connecting objects together by purely visual means. When new or updated objects become available, new functions and features, either from Demicron or a third party, can easily be installed as plug-ins to WireFusion.

Included with WireFusion are around forty objects, half of them are related to graphics, and the rest are data objects or miscellaneous tools needed for creating logic.

Developers working with JavaScript™ or JScript® (scripting implemented in browsers), can use WireFusion for interacting within webpages in which the Wire-

Fusion presentations are published. Java developers can also easily extend functionality directly in WireFusion, due to an object which allows for compilation of Java source.

Basically, the whole cycle of developing presentations are done in two steps. The first step is to visually **develop** the presentations. This includes testing, working with images in other programs, such as Adobe Photoshop®, or recording sounds. Lastly, you will deploy your final applets, publishing them on the web or an alternative purposing task such as Intranet. The second step occurs as **users** surf the webpages where the applets are embedded, and where they can see, hear or interact with the final presentations.

WireFusion presentations are suitable for use on the Internet, fairs, conventions, digital campaigns, digital advertisements, games, e-learning, teaching purposes etc.

3 How, What and Who?

3.1 How do you work with WireFusion?

Whether you are a programmer or a designer, all users should follow the typical sequence in creating a WireFusion presentation:

1. Decide what your presentation is meant to accomplish.
2. Determine how your presentation will be working for the people using or looking at it.
3. Choose the main objects you need.
4. Create the presentation by dragging and dropping objects into the *Wire View*, and by connecting the objects.
5. Test your presentation after each created function.
6. Save your project.
7. Publish the presentation as the generated applet, on a web page.

It is highly recommended to work through the book on *Getting Started with WireFusion* and accompanying exercises.

3.2 What types of presentations can you create with WireFusion?

Presentations and interactivity of different kinds can be produced with WireFusion. With WireFusion you create high quality and interactive graphical presentations. But without the requirement of any programming or script skills.

Well known tools such as Adobe LiveMotion®, Macromedia Flash® or Macromedia Director® also give you similar abilities. However, lack of precisely these types of graphical functionality and circumstances such as users will always need a plug-in, or worse an upgrade, give WireFusion a strong advantage.

Some ideas on what you will be able to do:

- Complete product simulations of electronic devices, in both 2D and 3D, such as mobile phones or handheld computers.

- Product prototype demonstrations.
- Interactive manuals, guidance of use of household appliances such as a video or how to put together a furniture.
- Interactive visualizations, e.g. show how changes in color, effects the look of a car.
- Presentations of complicated events which aren't easily described through text or pictures, e.g. in nature osmosis, or flows of control in nuclear plants.
- Menus and site maps.
- Interactive banners.
- Animated company introductions, products, or logos.
- Pure visual effects to spice up web pages.
- Web based games which require photo realistic rendering in real-time.

3.3 Who uses WireFusion?

WireFusion gives professional designers, modellers, programmers and students as well as private individuals the opportunity to develop their own graphical Java applets in just a fraction of the time needed by a professional Java programmer. As a teaching aid, it is useful for teaching purposes, teaching the general principles of object-oriented programming techniques.

WireFusion is a tool for anyone who wants to create advanced web presentations quickly. Some users of WireFusion will have experiences from using common web design, multimedia or authoring tools, others will have knowledge in Java and in using professional development tools.

Designers and modellers—for designers and modelers WireFusion gives them a virtual Java programmer in their hands. Without knowing anything about Java or programming you will be able to create advanced product presentations, logic, menus or spectacular graphics effects.

Programmers and developers—for Java programmers and developers WireFusion will speed up the work enormously. Especially as you can write your own Java code directly inside WireFusion , use JavaScript, and share all of it with other WireFusion users.

Teachers or producers of educational materials—for teachers who want to give their students or audience the opportunity to gain understanding through interactivity or demonstrations of concepts or, for example, virtual representations of physical objects. It can also be used to teach some object-oriented principles, or (visual) prototypical programming.

4 Java

4.1 History

In 1991, Sun Microsystems started a research project that resulted in the development of Java™. The project was created to develop a language for use in intelligent consumer electronic devices. In 1993, the World Wide Web exploded in popularity and Sun saw the potential of using Java to create web pages with dynamic content. In May 1995, Sun formally announced Java. Java immediately gained interest in the business community, because of the recent commercial interest in the World Wide Web. In January of 1996, a big breakthrough came for widespread Java use when the market leader of the time Netscape released a Java enabled browser, the Navigator. Sun released the first official version of Java in early 1996.

The main value of Java is its ability to connect users with information from web servers, databases, information providers, and any other imaginable source. Java's run time library provides platform independence allowing the same code to be run on different operating systems, which is necessary for Internet programming.

Sun describes Java as a *"simple, object-oriented, network-savvy, interpreted, robust, secure, architecture-neutral, portable, high-performance, multithreaded, and dynamic language"*.

4.2 Applets vs. Applications

WireFusion is a Java *application*. An application is a more general program that is written in Java, but does not require a browser to run. Java can be used to create all the kinds of applications that a more conventional programming language would normally be used.

WireFusion generates *applets*. Applets are tiny Java programs that are embedded into web pages, and automatically downloaded from a web site. They are executed within a web browser in response to instructions contained in the web page.

4.3 Java Virtual Machine

Normally, programs are created to run on an intended architecture and that is why you can't run a Microsoft Windows XP application on an Apple Machintosh. To enable Java programs (applications and applets) to run on any platform, Sun created

a 'Virtual Machine' that interprets Java code so it can be run on any platform. A *Java Virtual Machine* (JVM) is separately written for each operating system, and thanks to it you don't have to rewrite your Java code to have it working on other systems 'create once, run anywhere'.

4.4 Versions

There is an historical development of Java: Java 1.0, Java 1.1, and today Java 2. The first two main versions was implemented directly in browsers, and therefore available still on several platforms. Due to legal reasons, many new browsers still implement the first and second version (1.0 and 1.1) and minor upgrades. Some browsers have not lately directly implemented later versions of Java (such as Java 2). Therefore, there is also a Java 2 plug-in from Sun Microsystems, that will enhance these browsers with later Java versions. The plug-in is however substantial in size, which reduces the willingness for the general user to both download and install it.

WireFusion is a Java 2 application. In the installation process, Java 2 is installed. You, therefore, don't even have to think about separate installations or downloads. It's all there.

As Java 1.1 is implemented, not only in browsers but on a variety of platforms, the presentations created with WireFusion will be highly portable from and to e.g. Linux, Microsoft Windows, Apple Macintosh or Sun Solaris. The applets generated by WireFusion are Java 1.1 compatible working in all browsers supporting Java 1.1 (JVM 1.1) and upwards.

4.5 Security

Security is a main concern with respect to Java. This is because Java-enabled browsers download code across networks, and the code is executed on the user's computer. A basic rule, concerning security on the Internet, is not to download anything that you plan to execute, except when you are sure of its source. Java applets can be run in relative safety due to its safe design. The execution of code is greatly restricted.

Java **disallows** *applets* to:

- Read or write files to the local computer (the user's computer).
- Delete or rename files on the local computer.
- Create directories or list a directory locally.
- Check size, type or existence of a file locally.
- Create a network connection to another computer, other than the originating host for the applet.

- Accept connections from computers outside (besides originating host).
- Invoke any program which is loaded on the local computer.

5 How to Work with WireFusion

The way you work in WireFusion differs from other web authoring tools. WireFusion works with pre-programmed functions and operations, known as objects (WireFusion *Objects*). The use and reuse of already made and tested code assures robust applets and operations. Hence there is little need for traditional debugging.

WireFusion is not based on a timeline as many other web animation tools. However, you can use the time as a parameter to run an animation or to have events executed in a specific order.

WireFusion works best if you let the user interact in real-time with your presentations. Let them change speed, size or colors, or let them navigate, rotate or zoom a product. Everything is easily created by you and easily experienced by the user. Even if you are using image processing filters as e.g. blur or brightness and in layers, which normally are very processor intensive, WireFusion will handle those smoothly. You can also use alpha channels which let you manipulate, isolate, and protect specific parts of an image or a filter. WireFusion could in this area be described as a "programmable and interactive Adobe Photoshop®".

5.1 Principles of Working with WireFusion

How do you then typically work with WireFusion? First, you drag the objects of your choice to a special area where you work with them (Figure 5.1). Then, you connect the objects by mouse clicks, dragging symbolic connections, and connect by pointing to other objects (Figure 5.2). The connection in detail is done by choosing an out-port from an object, clicking on another object, then choosing some of its available in-port (Figure 5.3). Result: an object has been "wired" to another object. WireFusion thus in general works with a flow of data through connected objects.

Direct connections are done by not just looking at the kind of objects, but also what kind of data that will be transported from one object to another. You will learn more about this and how to apply it when creating logics, in image processing, filters, layers or alpha channels in the following chapters. In some cases though, objects are quite enough and connections are not necesary to give fully adequate presentations.

Figure 5.1: A group of objects

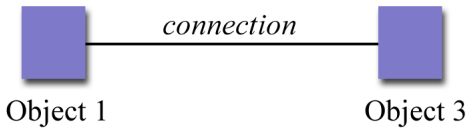

Figure 5.2: Connection between two objects

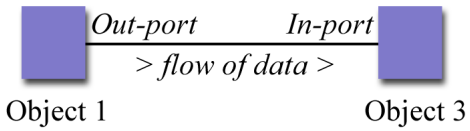

Figure 5.3: Directed connections are done by in- and out-ports

6 Layout

Some terminology, definitions and names on parts, areas or views of WireFusion are described below. In Figure 6.1 and Figure 6.2 you'll find an overview of the WireFusion general workspace.

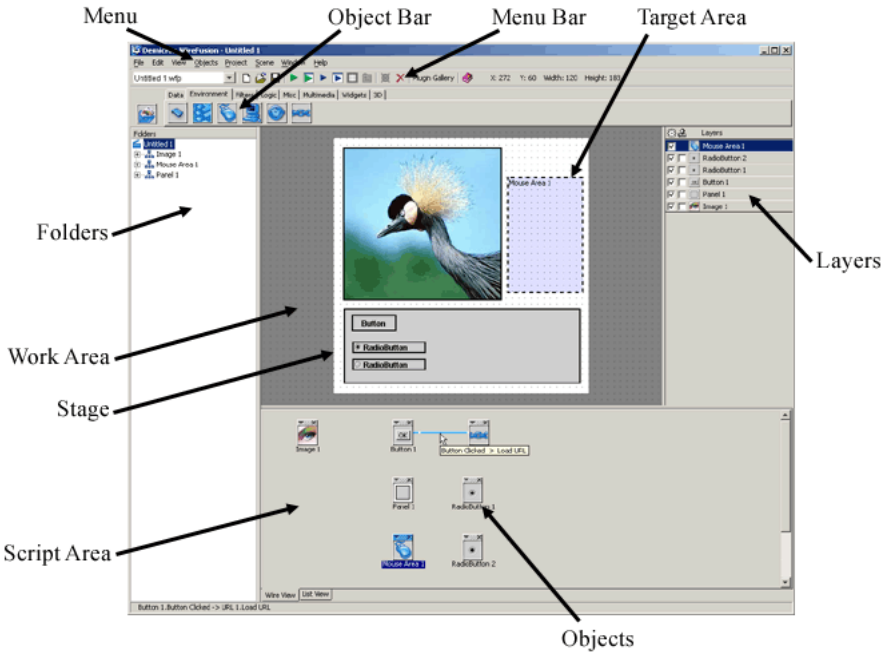

Figure 6.1: The WireFusion environment (*Wire View*)

Menu. The *Menu* consists of usual commands of the application.

Menu Bar. The *Menu Bar* contains the *Project List*, with the name of the project displayed. If there are more projects opened, the current project name will be displayed. You will also find the equivalents or duplications to relevant menus by icons to *New Project*, *Open Project*, *Save Project*, at the rightmost *Help Contents*. Other icons are *Preview Presentation*, *Preview Scene*, *Scene properties*, Cancel connection mode, Delete and Help.

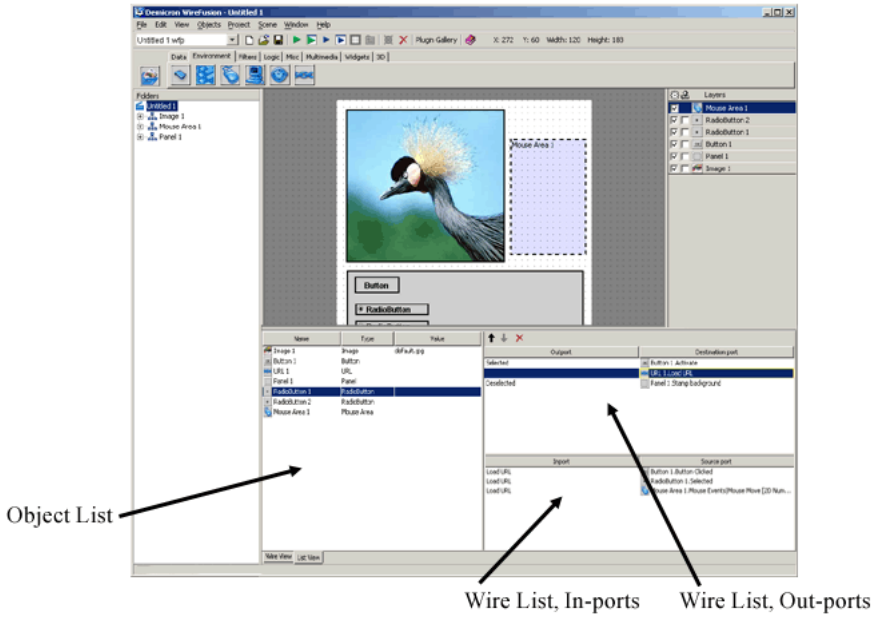

Figure 6.2: The WireFusion environment (*List View*)

Objects. The WireFusion *Objects (.wob)*, or simply the *objects*, are pre-programmed functions which are the building blocks of your presentations.

Between objects, connections are drawn. The *Connection Information* (tooltip) from the connections are displayed whenever a mousepointer is present on top of them. Extended information is present in the *Information Bar*or in the *List View*.

Information Bar Text field at the bottom left of the program window displaying connection information.

Object Bar. The *Object Bar* contains all the objects. The objects delivered with WireFusion are organized in tabs: *Data*, *Environment*, *Filter*, *Logic*, *Misc* and *Multimedia*.

Work Area. The *Work Area* is the main working area for visual objects. It contains the *Stage* and all *Target Area*s.

Stage The *Stage* is the area for the presentation. It is the area that will be seen in the browser by the user.

Target Area. Objects displaying graphics or having any visual effects have a corresponding area in the *Stage* and *Work Area* called the *Target Area*. For example, a button, an image or a text, will be displayed in a *Target Area*.

Script Area. In the *Script Area* you specify the workings of your presentation by placing objects and connecting them together. *Script Area* have two different views: *Wire View* and *List View*.

Wire View. In the *Wire View* you get a schematic view of your objects and connections. Change view to *List View*, by clicking on the bottom tab.

List View. In the *List View* you get a table view of your objects in the *Object List* and your connections in the *Wire List*. Change view to *Wire View*, by clicking on the bottom tab.

Object List. In the *Object List* you get a table view of your objects.

Wire List. In the *Wire List* you get a table view of your connections.

Folders. The *Folders* helps to survey the object hierarchy in your projects. Use it to navigate through the objects. Only objects containing other objects are listed here. The name of the project *Project Title*, is at topmost position.

Layers. The *Layers* view lists all layers in the *Stage*. Only objects with a *Target Area*are listed. *Stamp background* and *Activate* properties can be changed directly.

7 Menus

The WireFusion main program has eight menus: *File*, *Edit*, *View*, *Objects*, *Project*, *Scene*, *Window* and *Help* (Figure 7.1).

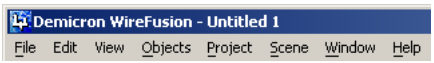

Figure 7.1: The WireFusion program menus

7.1 File

The *File* menu regulates projects and their properties. It also sets properties of the program (Figure 7.2).

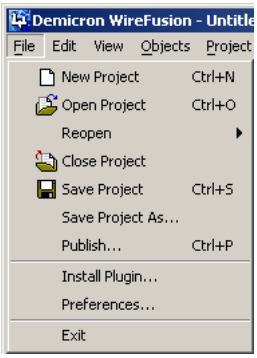

Figure 7.2: The *File* menu

New Project (shortcut: CTRL+N) creates a new project. (A duplicate command resides in the *Menu Bar*, see Section 7.9.)

Open Project (shortcut: CTRL+O) lets you open a saved project. Browse to locate your project file (.wfp) and press Open in the Open Dialog to load. (A duplicate command resides in the *Menu Bar*, see Section 7.9.) To import a singular object, see Section 9.4.

Reopen lets you reopen a saved project. Previously recently saved projects can be reopened by this short-cut command.

Close Project lets you close current project.

Save Project (shortcut: CTRL+S) saves the current project, if the project has been given a name. Otherwise a Save Dialog will be opened, to let you choose a name for the project. (A duplicate command resides in the *Menu Bar*, see Section 7.9.) To save a singular object, see Section 9.6.

Save Project As... saves the project, and lets you choose a name for it through the Save Dialog.

Publish... (shortcut: CTRL+P) deploys the presentation into an HTML file. Read more in Chapter 17.

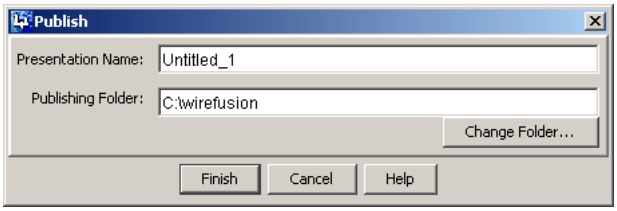

Figure 7.3: The Publish Dialog

Install Plug-in... is used to install new objects into WireFusion.

Preferences Where the program settings are set and stored.

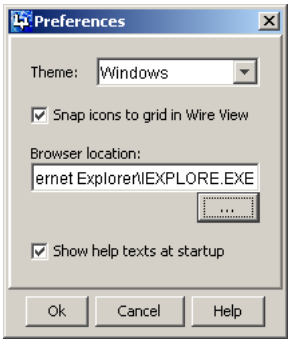

Figure 7.4: The *Preferences Dialog*

Theme: (exclusive choice) The Java environment comes with three desktop themes. Select the look that appeals to your taste: Metal (Figure 7.5), CDE/Motif (Figure 7.6), or Windows (Figure 7.7). The default theme

is Windows. (The program has to be restarted in order to update the appearance fully.)

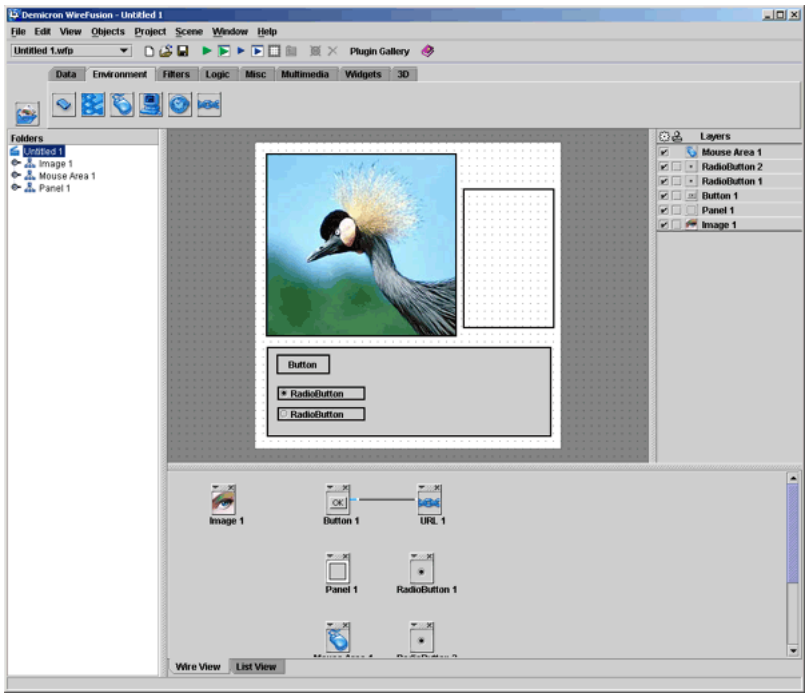

Figure 7.5: Metal Theme

Snap Icons to Grid in Wire View: (checkbox) Select this option if you want the object icons to snap to an invisible grid in the *Wire View*.

Browser location (text) Specify a path to your browser, either by directly specifying it, or by browsing through directories.

Show help texts at startup: (checkbox) By default, this option shows help texts at startup.

Exit Exits WireFusion. You can also exit by clicking on the closing icon of the program window. The program settings are saved each time you exit WireFusion.

7.2 Edit

The *Edit* menu controls simple editing with objects. You can *cut*, *copy*, *paste* and *delete* objects of your choice. You can also select objects by groups and edit them. To move a group of objects in the *Wire View*, select the objects (click and hold the mouse button down, then drag over the area where the objects are located). Then choose *Cut* from the menu, and *Paste* respectively, at the position of your choice.

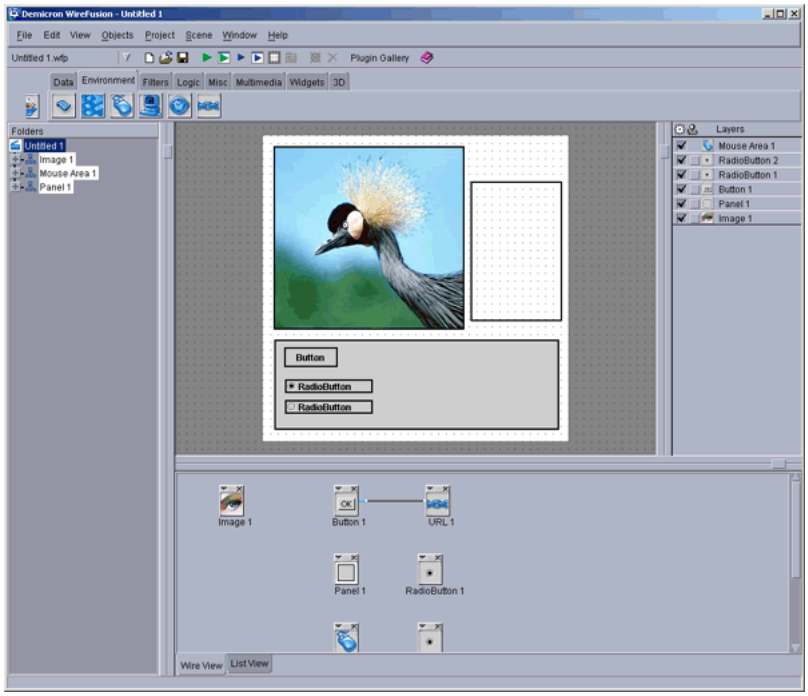

Figure 7.6: CDE/Motif Theme

Cut (shortcut: CTRL+X) Cut the objects, and place them in a clipboard.

Copy (shortcut: CTRL+C) Copy the objects to the clipboard.

Paste (shortcut: CTRL+V) Paste the objects from the clipboard.

Delete (shortcut: DELETE) Delete the objects.

To paste in a specific location in the *Wire View*, place the cursor at the desired location and click the right mouse button.

7.3 View

Toggle Folders View (shortcut: CTRL+1) Toggle the *Folders View* on and off.

Toggle Stage View (shortcut: CTRL+2) Toggle the *Stage View* on and off.

Toggle Layers View (shortcut: CTRL+3) Toggle the *Layers View* on and off.

Toolbar (shortcut: CTRL+4) Toggle the *Toolbar* on and off.

Switch Script View (shortcut: CTRL+SHIFT+1) Switch between *Wire View* and *List View*

Console (shortcut: CTRL+SHIFT+C) Display *Java Console*

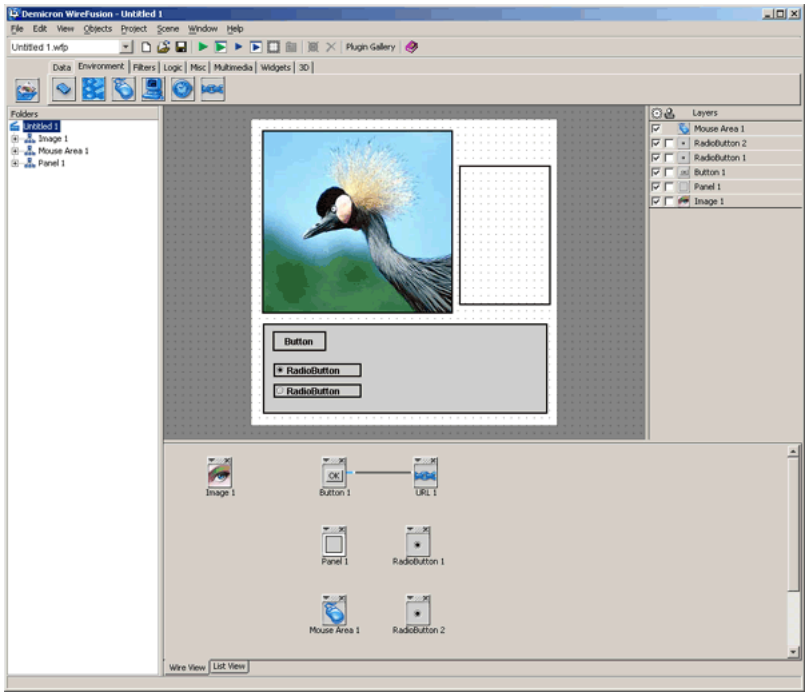

Figure 7.7: Windows Theme

7.4 Objects

The *Objects* contains all the WireFusion objects. These objects are the same as those found in the *Object Bar*, see Section 9.1.

7.5 Project

The *Project* contains relevant control over projects (Figure 7.8).

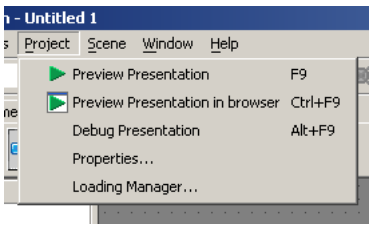

Figure 7.8: The *Project*

Preview Presentation
(shortcut: F9) Preview the current project in the viewer.

Preview Presentation in Browser
(shortcut: CTRL+F9) Preview the current project in the browser. By default, it is run in the (default) browser. (Choose browser in the *File* > *Preferences*.)

Debug Presentation (shortcut: ALT+F9) Run the current project through the *Debugger*.

Properties... Displays the properties for the project `Scene`. On the relation between the project and `Scene` object and *Scene properties*, see Section 12.1.

Loading Manager... Controls if resources like images and sounds should be streamed into the presentation and in which order. (Figure 7.9).

Figure 7.9: The *Loading Manager...*

7.6 Scene

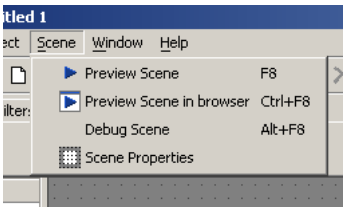

Figure 7.10: The *Scene*

 *Preview Scene*
(shortcut: F8) Preview the current `Scene` in the viewer.

 *Preview Scene in Browser*
(shortcut: CTRL+F8) Preview the current `Scene` in the browser. By default, it is run in the (default) browser. (Choose browser in the *File* > *Preferences*.)

Debug Scene (shortcut: ALT+F8) Run the current `Scene` through the *Debugger*.

Scene properties Displays the properties for the current `Scene`

7.7 Window

If several projects are open, you can switch between windows from this menu.

7.8 Help

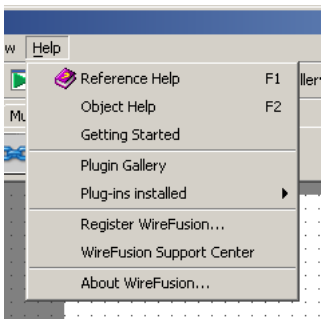

Figure 7.11: The *Help*

Reference Help (shortcut: F1) This command opens the main body of the *Help* to a table of contents. From here, or from any other help page, you can search all of the *Reference Help* for a particular topic.

Help Objects (shortcut: F2) This command opens the *Reference Help* for the *WireFusion Objects* (also called Wobs). Here basic information about the objects can be found as well as information about the *In-ports* and *Out-ports*.

Getting Started The *Getting Started* web site helps you to get started with WireFusion and how to get the most out of the program. You will find updated tutorials, extensive manuals and example files.

Plug-in Gallery On the *Plug-in Gallery* web site you will find information about the different plugins available for WireFusion.

Installed Plugins Shows installed plugins. You can also register an installed tryout plugin here.

WireFusion Support Center The *WireFusion Support Center* web site is updated regularly with the latest information on WireFusion and online forums with tips from experts.

About... Displays WireFusion's splash screen and user license information.

7.9 Menu Bar

Through the *Menu Bar* you can easily control the running and testing of projects (Figure 7.12). The *Menu Bar* contains the *Project List*, with the name of the project displayed. If there are more projects opened, the current project name will be displayed. You will also find the equivalents or duplications to relevant menus by icons to *New Project*, *Open Project*, *Save Project*, at the rightmost *Help Contents*. Other icons are *Preview Presentation*, *Preview Presentation in Browser*, *Preview Scene*, *Preview Scene in Browser*, *Scene properties*, Cancel connection mode, Delete and Help.

Figure 7.12: The *Menu Bar*

Project List A list of the loaded projects. The current project name is visible at the top of the list.

New Project
A new and empty project opens with a default name in the *Project List*, and at the top of the *Folders*. (Clicking the icon is equivalent to choosing *File* > *New Project*.)

Open Project
Opens a project through an Open dialog. (Clicking the icon is equivalent to choosing *File* > *Open Project*.) (On the other hand, to load a singular saved object, see Section 9.4.)

Save Project
Saving the project. If the project has not been saved before, a Save dialog will appear. (Clicking the icon is equivalent to choosing *File* > *Save Project*.) (On the other hand, to save a singular object, see Section 9.6.)

Preview Presentation
Preview the current project in the viewer.

 *Preview Presentation in Browser*

Preview the current project in the browser. By default, it is run in the (default) browser. (Choose browser in the *File* > *Preferences*.)

 *Preview Scene*

Preview the current `Scene` in the viewer.

 *Preview Scene in Browser*

Preview the current `Scene` in the browser. By default, it is run in the (default) browser. (Choose browser in the *File* > *Preferences*.)

 Delete

To delete an object (or a connection) in the *Wire View* or *List View*, select the object (or the connection) to be deleted and then click the Delete button (or the *Delete Key* on your keyboard) to remove the object (or the connection).

 Cancel connection mode

If you have selected, e.g. an *Out-port* in an object, the program expects you to choose an *In-port* in another object in order to finish the connection procedure. After selecting a port (out or in) the mouse pointer will change to connection mode and the Cancel connection mode button and a text saying 'Select an in-port' will be visible on the toolbar. Click the Cancel connection mode button in order to discontinue the connection. Read more in Chapter 10.6.

 *Scene properties*

Displays the current *Scene properties* dialog. (See Section 12.1).

 *Help*

Shows the *Reference Help*. (Clicking the icon is equivalent to choosing *Reference Help* from the *Help* menu, see Section 7.8.)

8 Projects

Steps in Creating a WireFusion presentation

Before you deploy your project as a Java applet, you must first use the WireFusion objects to create a presentation. Your presentation is inactive as you build it, which means that its behavior in the *Stage* window is not identical to the behavior of the finished presentation. You have to test and preview your presentation along the development process. However, you can test and refine your presentation without having to leave WireFusion.

i. Choosing the Objects

Before you start, you need to identify the objects you will need. Since each objects has a specific task or function associated with it, try to break your overall design plan into small pieces that can be addressed using one or a few objects. If there is no specific object for the function you need, then, in most cases, it is possible to create the needed function with other objects, by using your imagination, or by building new functions by connecting objects together. It is therefore important to learn what the different objects can achieve. The *Object Bar* is composed of a series of tabs for groups of related objects.

ii. Placing Objects

To place an object into your project:

Click its icon in the *Object Bar*. Then drag and drop it in the *Wire View*. If it is an object with visual properties, then it will be visible in the *Work Area* window. You can click and drag the rectangles in the *Work Area* window to define both the size and location of these objects. These types of objects also appear in the *Layers* view. Further, if it is an object that can contain other objects, then it will be visible in the *Folders* view as well. The *Wire View* window will always display all of the objects in your presentation.

iii. Customizing Objects

When you place an object with settings possibilities into the project, the object's properties dialog is displayed. Each object's properties dialog has one or more tabs where you can change the properties of the object. For example, some properties allow you to change the visual size of an image, while others allow you to set numbers or colors. If you need more information about an object, click *Help* in the properties dialog.

iv. Connecting Objects

Each object has one or more ports that send or receive parameters. A port can communicate with one or several ports on another object, or on itself, if necessary.

To connect one object to another:

Open the local object menu, by right clicking the object, and select an out port option. Then open the local object menu of another object and select an in port option. A connection has been made between the two objects. Each port may be connected to an unlimited number of other ports.

v. Adjusting the Wire View

When you connect two objects in the *Wire View* window, a wire appears between the respective our port and in port. You can rearrange your objects to make it easier to view the logical layout of your project. You can also change the wire colors, or even make them invisible, to make the layout easier to understand.

vi. Testing your presentation

Your presentation is inactive as you build it. That means that objects do not begin to function until you preview (or publish) your presentation. Some objects can however show a preview of its visual contents, as for example the `Image` object. It is important to preview the presentation along with the development of new functions, in order to achieve a well working presentation.

vii. Deploying your presentation

When your presentation is working as it should and when you have saved your project, then you can publish it. Your presentation will be published into a default HTML page as a Java 1.1 compatible applet.

8.1 Saving and loading projects

You don't need to publish Java applets each time you create a presentation. You may want to save your project and work on it later, or to send it to another WireFusion user so that they can use or modify it.

8.2 Saving your project

To save your project and preserve the contents:

- Choose from *File*, *Save Project As...*.
- Enter a name and choose where to save your project.
- Click *Save Project*.

WireFusion creates a file with a .wfp extension. For example, if you type *Myfile* as the file name, WireFusion will save your project in the file *Myfile.wfp*.

8.3 Loading a project

To load a project that you previously saved:

- Choose from *File*, *Open Project*.
- Browse the .wfp file in which you saved your project.
- Click *Open Project*.

8.4 Navigating Projects

In WireFusion, some objects contain other objects. These types of objects can be seen as folders, and they are visible in the *Folders* view when dragged into your project (Figure 8.1).

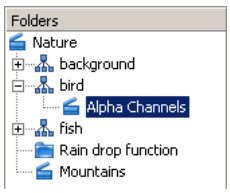

Figure 8.1: The *Folders* view

You navigate through your WireFusion project by using the *Folders* view and the two different views in the *Script Area*; called *Wire View* and *List View*. The *Folders*

view, together with the *Script Area*, works in a similar way as the *Windows File Explorer*. You navigate through your folders (folder objects) to the left, and your files (objects) to the right (Figure 8.2). When browsing your files in the *Windows File Explorer*, you can choose to view your files as *Large Icons* or *Details*. This is analogous to switch between the two Script Area views *Wire View* and *List View* in WireFusion (Figure 8.3).

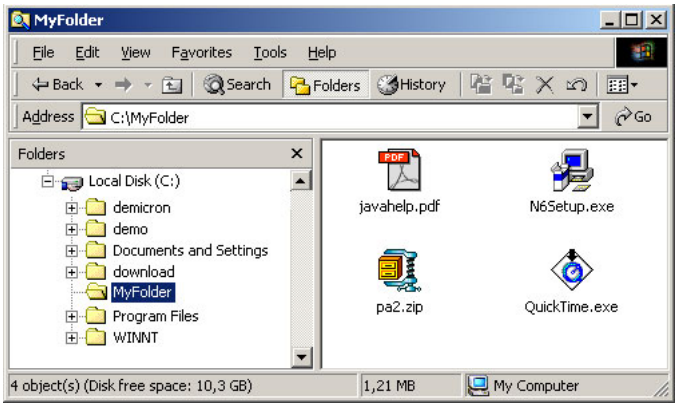

Figure 8.2: *Windows File Explorer* with *Large Icons* view

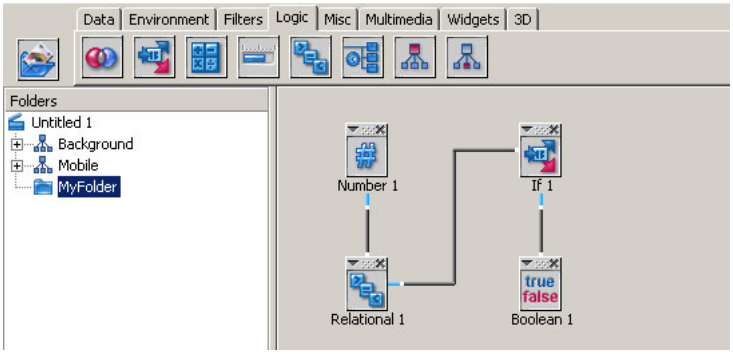

Figure 8.3: WireFusion in *Wire View*

In the *Wire View*, you can see your objects as large icons. You can also see the connections, if any, visually between the objects here.

In the *List View*, on the other hand, you get detailed information about the objects. The *List View* is split into two windows; *Object List* to the left and the *Wire List* to the right. In the *Object List* you can sort your objects and see if the objects contain any predefined data. If connections have been made to an object, then they will be listed to the right in the *Wire List*. Out port connections are listed in the upper half and in port connections at the lower half.

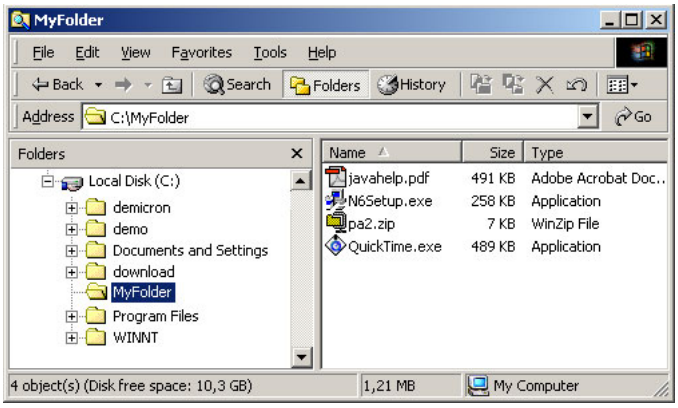

Figure 8.4: *Windows File Explorer* with *Details* view

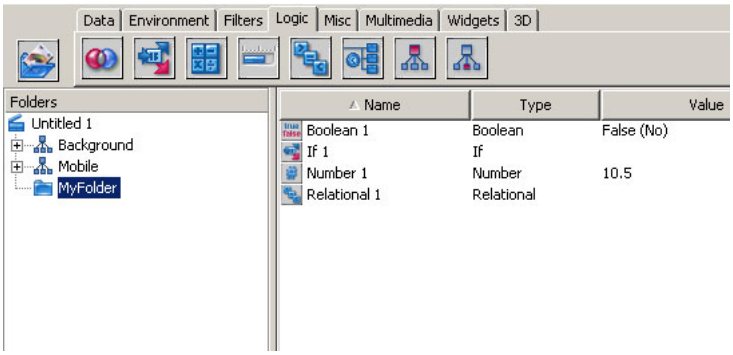

Figure 8.5: WireFusion in *List View*

Figure 8.3 shows the *List View* and how a `Slider` object, called 'Change opacity', sends its parameters through its *Out-port Slider Set Value [Number]* to an `Image` object, called '`Background`', and its *In-port Set Opacity [Number]*.

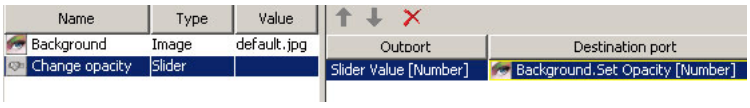

Figure 8.6: *Object List* and *Wire List*

To simplify the navigation of your project, it is recommended to name your objects in a pedagogical way, i.e. give the objects names that explain their functions.

8.5 Project properties

Scene properties can be set through the *Scene properties* dialog (Figure 8.7). You can always change the properties of the local scene, i.e. the scene you are located in, by pressing the *Scene properties* button in the *Menu Bar* or by choosing *Scene* > *Scene properties*.

To change the scene properties for the project, i.e. properties for the main scene, then select either *Project* > *Properties...*, or select the *Scene properties* button in the *Menu Bar* when you are located in the topmost `Scene` object.

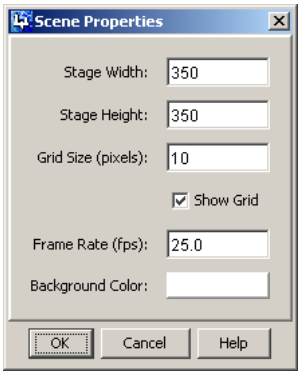

Figure 8.7: *Scene properties* dialog

Stage Width (number) Sets the *Stage* width in pixels. (Default 240 pixels.)

Stage Height (number) Sets the *Stage* height in pixels. (Default 180 pixels.)

Grid Size (number) It is possible to adjust the *Target Area*s to grid points in the *Stage* and *Work Area*. This option allows you to set the distance between the grids in pixels. (Default 10 pixels.)

Show Grid (checkbox) Shows and activate the grid.

Frame Rate (f/s) (number) You can set the maximum number of frames displayed per second in your presentation. The user's computer will try to show your presentation at this frame rate. Default is 25.0 frames per second.

Background Color (color) Sets the *Stage* background color.

9 Objects

9.1 Predefined Objects

There are preprogrammed functions and operations in WireFusion known as objects. They are located in the *Object Bar* (Figure 9.1), or equivalently, in the *Objects* (Figure 9.2). In the *Object Bar* there is also a tooltip displaying the name of each object. On the far left hand side of the tabs in the *Object Bar*, there is an `Import a saved object` object.

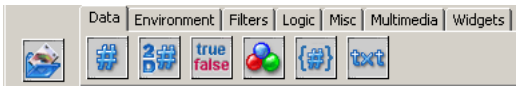

Figure 9.1: The *Object Bar*

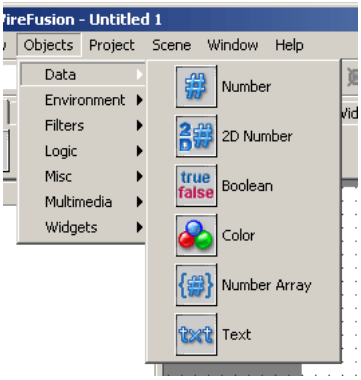

Figure 9.2: The *Objects*menu

WireFusion comes delivered with around fifty objects and half of them are related to graphics, and the remainder are data objects or miscellaneous tools needed for creating logic. The objects that come delivered with WireFusion fall into the categories of *Data*, *Environment*, *Filter*, *Logic*, *Misc*, *Multimedia* and *Widgets*.

Updated or new objects, either from Demicron or from third parties, can be installed in WireFusion as plug-ins, to broaden the range of functions.

9.2 Object Categories

The following object categories and corresponding objects have already been defined for WireFusion.

Data: `Number`, `2D Number`, `Boolean`, `Color`, `Color Array`, `Number Array`, `Text`

Environment: `JavaScript Link`, `Keyboard`, `Mouse Area`, `System`, `Time`, `URL`

Filter: `Blur`, `Brightness`, `Bump Map`, `Emboss`, `Grayscale`, `Invert`, `Lens`, `Line Art`, `Mosaic`, `Noise`, `Plasma`, `RGB Filter`

Logic: `Boolean Operations`, `If`, `Math`, `Progressor`, `Relational`, `Sequencer`, `Tree`, `Tree Node`

Misc: `Dummy`, `Folder`, `Java`, `Scene`, `Scene Window`, `Tooltip`

Multimedia: `Image`, `Image Array`, `Sound`

Widgets: `Button`, `Checkbox`, `Label`, `Panel`, `RadioButton`, `Slider`, `Text Window`

All help files and references for the objects in WireFusion can be found in the *Help* > *Object Help* (Section 7.8).

9.3 Inserting Objects

To insert an object into your project, select the desired object in the *Object Bar*, drag it and then drop it in the *Wire View* (Figure 9.3). Or, equivalently, select the object from the *Objects* menu, drag it and then drop it in the *Wire View*.

9.4 Importing Objects

To load an already saved project or object **into** your existing project, drag the `Import a saved object` object from the *Object Bar* (Figure 9.4) and drop it in the *Wire View* where you want to place the object (or project). An Open Dialog will open, which let you browse for your object file (or project file). Objects have the extension .wob, and projects .wfp). In the Open Dialog, select the file and then click Open to load the object (or project). (To save an object, see Section 9.6.)

9.5 Renaming Objects

When you have selected an object, you can rename it from its default name (or previous name). Click on the default name. An editable area will be displayed underneath the icon. Erase the pervious name, and type the new name on the keyboard. Now click beside the icon (not on the name again) and at the keyboard Enter key. The object has been renamed (Figure 9.5).

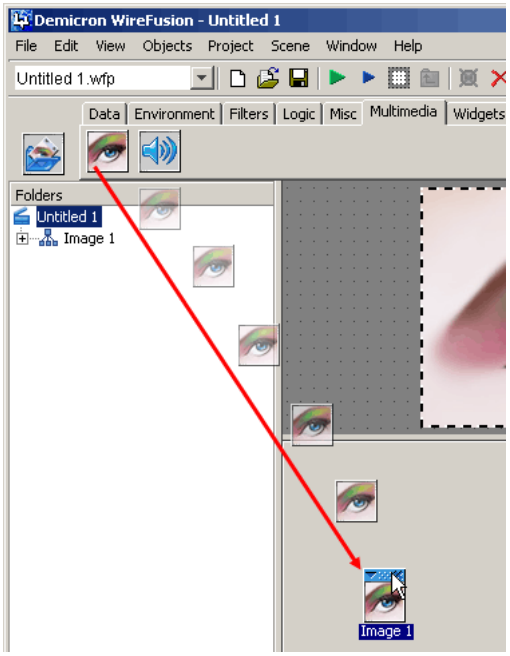

Figure 9.3: Example. Drag the `Image` object from the *Object Bar*, and then drop it in the *Wire View*

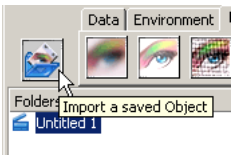

Figure 9.4: Import icon

9.6 Local Object Menu

All WireFusion objects have a drop-down menu (see Figure 9.6 and Figure 9.7). This menu can be operated by, either a click on the lefthand corner of the object, or through a right-click on the object itself, when it's in the *Wire View*, *List View*, *Layers* view or on the *Target Areas* in the *Work Area* (see Figure 9.8 and Figure 9.9). All menus are alike, exept for some objects that can contain other objects, which have options for exploring or opening up object folders.

Out-ports Contains *Out-port* options.

In-ports Contains *In-port* options.

Explorer (shortcut: E) Opens the object's folder. Alternatively, use the *Folders*

Figure 9.5: Example. Renaming the `Image` object

Figure 9.6: Open the drop-down menu

view for exploring the object structure.

Export Ports... (shortcut: P) The *Port Exporter* helps you to export in-ports and out-ports from objects placed in `Folder` and `Scene` objects. It also lets you rename and reorder already exported ports. Read more in Section 12.3.

Save... (shortcut: S) If you have configured an object in a specific way or if you have made a special function in a `Folder` or a `Scene` object, then you can save the object and load it again for later use in other projects. Read more on projects in Chapter 8.

Cut (shortcut: CTRL+X) Removes the object and places it in the WireFusion clipboard, from which you can paste. To paste the object, move your mouse pointer to a location in WireFusion where you want to place the object, then click the right mouse button to paste it.

Copy (shortcut: CTRL+C) Copies the objects to the WireFusion clipboard, from which you can paste. To paste the object, move your mouse pointer to a

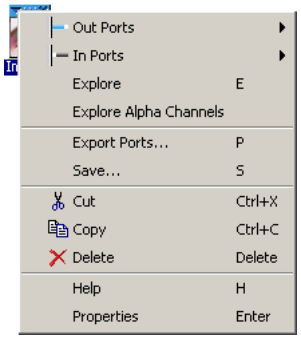

Figure 9.7: The drop-down menu opened

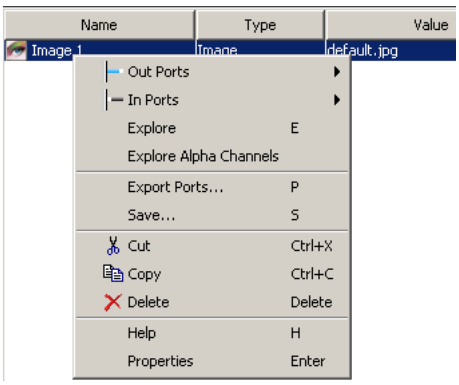

Figure 9.8: Right click an object in the *List View* to open its drop-down menu

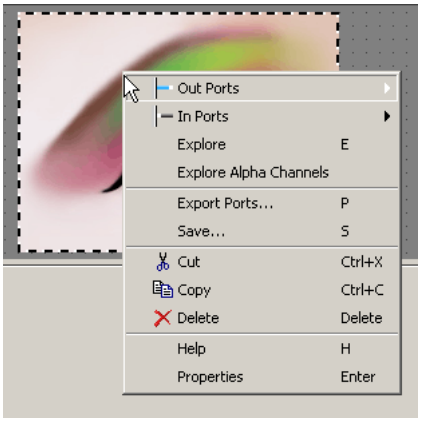

Figure 9.9: Right click a *Target Area* to open its drop-down menu

location in the *Wire View* where you want to place the object, then click the right mouse button to paste it.

Delete (shortcut: DELETE) Delete the selected object from the current project.

Help (shortcut: H) Get reference help for the selected object.

Properties (shortcut: ENTER) Opens a property dialog for the selected objects. Read more in Chapter 9.8.

9.7 Target Area objects

Certain objects have a special area that can be seen and modified in the *Stage* and *Work Area*, called the *Target Area*. The *Target Area* mainly exists for objects that have some visual appearance in your presentation. The actual area is demarked by

a dashed rectangle, when the corresponding object is selected, or the rectangle itself is selected (Figure 9.10).

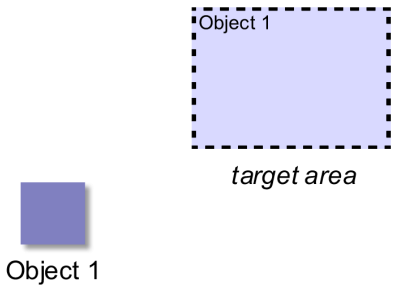

Figure 9.10: An object and its corresponding *Target Area*

Some of these objects contain graphics, as for example, the `Image` object that contains an ordinary JPEG, GIF or PNG image (Figure 9.11), or the `Text Window` object, which contains text. Other objects don't contain graphics but have some graphical result, as for example the filter objects `Blur` and `Brightness`, which both operates on images and graphics in layers beneath themselves.

Finally, there is the `Mouse Area` object, which neither contains graphics nor operates on underlying graphics, but still has a *Target Area*. For example, it can be used to set an area where special mouse events can occur. A small mouse area could, for example, function as a button, optionally with an image. The mouse area can sense when a mouse cursor is over it, and when the mouse is pressed. It can thus ignore the rest of the area of the presentation.

All objects in WireFusion having a *Target Area* are also listed in the *Layers* view, when inserted into a project. (Read more in Chapter 13).

The *Target Area* uses coordinates from the *Stage* for position and placement. The *Stage* has a coordinate system, which has its origin (origo) at the uppermost top left corner. The x and y coordinates are here at the origo; both are zero. There is a positive increment in the value of x, in the x-direction towards the right. Likewise, the value of y increments positively downwards, in the y-direction (Figure 9.12).

There are two fixed points on the *Target Area* to give relative coordinates for: in the center, and at the top left corner (Figure 9.13).

9.7.1 Moving and resizing Target Areas

The *Target Area*s properties can be changed in various ways. One way to change the basic parameters, of size and position, is by simply using the mouse in the *Work Area* (Figure 9.14 and Figure 9.15).

You can also use the numpad on your keyboard to move the *Target Area* in relative one-pixel-steps. In the *Wire View*, mark the object, which *Target Area* you

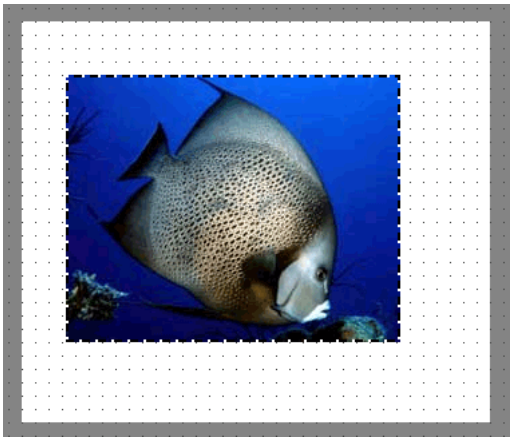

Figure 9.11: A *Target Area* of an `Image` object demarked by a dashed rectangle

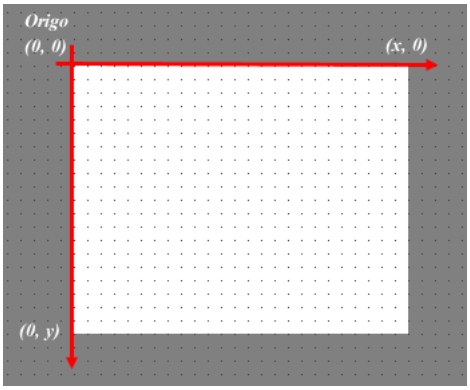

Figure 9.12: The *Stage* and its coordinate system. The origo (where x is zero and y is zero) is at the top-left corner

want to move, or mark the *Target Area* directly in the *Work Area*, with the mouse, then press, '2' for down, '4' for left, '6' for right or '8' for up (Figure 9.16). This is very useful for fine adjustments.

When a *Target Area* is selected, its positions and dimensions are shown in the information list above the *Menu Bar* (Figure 9.17).

The *Target Area* size and dimension can also be set in the object's *Properties* dialog (*Target Area* tab). (See Section 9.8.)

9.7.2 Mouse Events

All objects in WireFusion having a *Target Area* also have an *Out-port* option for mouse events at *Out-ports* → *Mouse Events* (Figure 9.18).

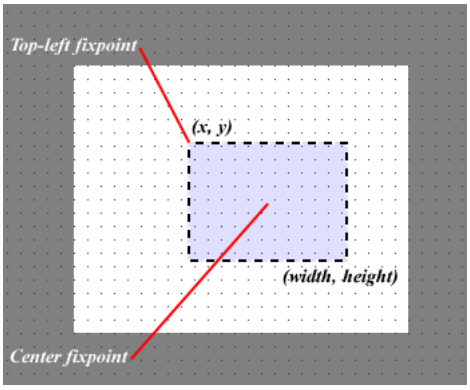

Figure 9.13: The central and top left fixpoints of a *Target Area*

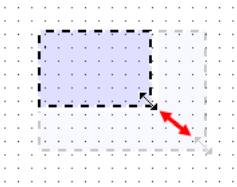

Figure 9.14: Changing the *Target Area* size by click-hold-drag in the corners of the demarcation

Mouse Move [2D Number]: sends out the mouse cursor position when the mouse has been moved (within the mouse *Target Area*).

Mouse Drag [2D Number]: sends out the mouse cursor position when the mouse has been dragged (within the mouse *Target Area*).

Mouse Roll Over [2D Number]: sends out the mouse cursor position when the mouse enters the mouse *Target Area*.

Mouse Roll Out [2D Number]: sends out the mouse cursor position when the mouse exits the mouse *Target Area*.

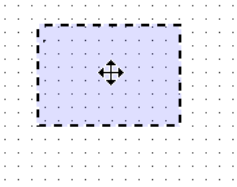

Figure 9.15: Changing the *Target Area* position by click-hold-drag the selected area

Figure 9.16: Relevant keys in keyboard numpad to move the *Target Area*

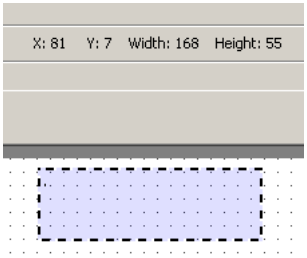

Figure 9.17: *Target Area* position and dimension information

Mouse Press [2D Number]: sends out the mouse cursor position when the mouse button is pressed (within the mouse *Target Area*).

Mouse Release [2D Number]: sends out the mouse cursor position when the mouse button is released (within the mouse *Target Area*).

These are the same ports as the `Mouse Area` object has, which means that all *Target Area* objects have the same functionality as a `Mouse Area` object, i.e. they have a "built-in" mouse area.

9.7.3 Target Area In-ports

Objects having a *Target Area* also have a special set of *In-ports*, to be located in their objects' menus. These *In-ports* help you control some *Target Area* related properties dynamically in your presentation, i.e. You can change these parameters during the presentation while it is running. To give an example, the user of your presentation can interact with it, and can re-size an image, or re-position it.

Objects with a *Target Area* can be turned on and off, during a presentation (about triggering, see Section 10.1). Two triggable *In-ports* are *Activate* and *Deactivate*. There is also a port *Stamp background* in control of applying permanency to a *Target Area*, and *Set Opacity [Number]*, which regulates transparency (example in Figure 9.19).

Activate Trigger this port in order to turn the object on. If it is a deactivated `Image`

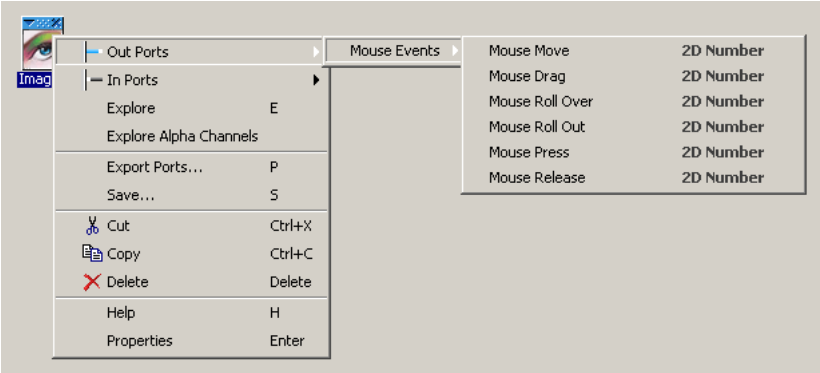

Figure 9.18: *Mouse Events Out-ports* from an `Image` object

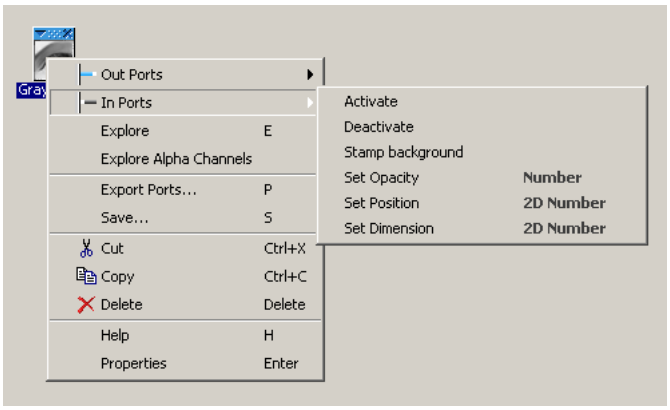

Figure 9.19: The `Grayscale` filter *In-ports*

object, then it will start to display the image when this port is triggered. If it is a filter object, as for example `Blur`, then it will start to blur all underlying graphics (See Chapter 13) when triggered. All objects are activated by default.

Deactivate Trigger this port in order to turn the object off.

The `Mouse Area` can also be activated and deactivated.

Stamp background Leaves a permanent impression of the *Target Area* in the background. An `Image` object, for instance, which is to be used as a background image, could benefit from being stamped, as the total image processing is reduced. (Not available for `Mouse Area`.)

Set Opacity [Number] Sets transparency of the *Target Area*s dynamically. Transparency is set in a number of percent, where default is one hundred percent (totally opaque). (Not available for `Mouse Area`).

Set Position [2D Number] This port receives a *[2D Number]* and sets or changes the *Target Area* position to a new position [x, y]. It depends on the *Target Area* fixpoint (See Section 9.8.3 on the *Target Area* tab.)

Set Position [2D Number] is often used in conjunction with the *Mouse Events Out-ports* in order to move, for example, an image.

Set Dimension [2D Number] This port receives a *[2D Number]* and sets or changes the *Target Area* dimension in pixels [width, height].

9.8 Object properties

All WireFusion objects have a properties dialog, which can be accessed by either clicking the *Properties* option in the local object menu, or by selecting the object and then pressing *Enter Key* (Figure 9.20).

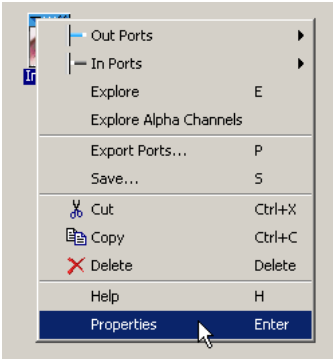

Figure 9.20: Opening the *Properties* dialog

The *Properties* dialog can contain three different tabs: *Settings*, *Target Area* and *General*.

9.8.1 General tab

All objects contain a minimum of one tab, the *General* tab, which contains general information about the specific object. (Figure 9.21).

Name: The object name.

Original name: The object name can be changed, but here you can see the original name of it.

Object ID: All objects have a unique identification number, the Object ID.

Version: Object version number.

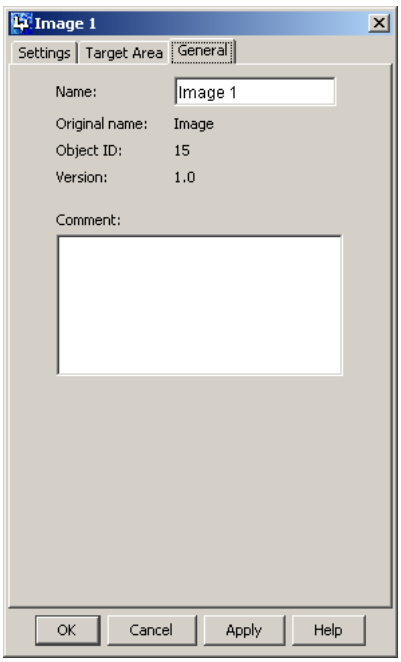

Figure 9.21: *General* tab

Comment: Add a comment to the object. It is sometimes useful to comment on an object or its function while working. The comment is shown as a tooltip when the mouse cursor is placed over the object icon in the *Wire View*. A red corner down to the right on the object icon indicates that there is a comment (Figure 9.22).

Figure 9.22: An object comment

9.8.2 Settings tab

Some objects have settings possibilities, which can be changed in the Settings tab. These settings are very individual and are custom made for each respective object. For example, the `System` object does not have any settings at all (Figure 9.23). The

`Number` object has a simple settings dialog (Figure 9.24), while the `Math` object is a bit more complex (Figure 9.25).

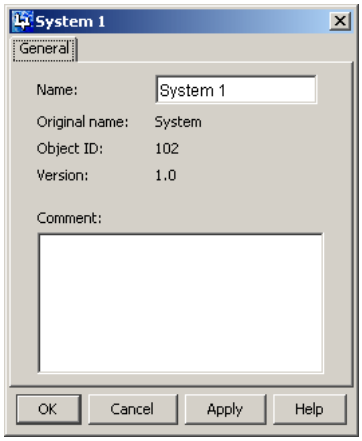

Figure 9.23: The `System` object's *Properties* dialog with no settings

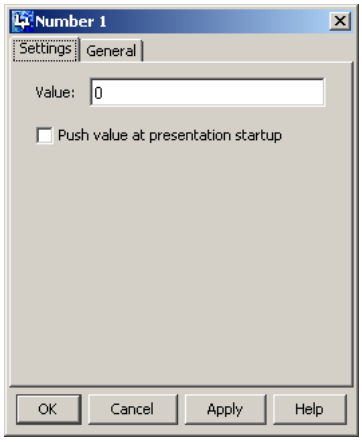

Figure 9.24: The `Number` object's settings dialog

To get help and information for each individual object settings, press the Help button in the *Properties* dialog.

9.8.3 Target Area tab

All objects having a *Target Area* also have a *Target Area* tab in its *Properties* dialog. The *Target Area* tab can be used for fine-tuning the position and size of the *Target Area*. But it can also be used for configuring other properties (Figure 9.26).

The control of options in detail:

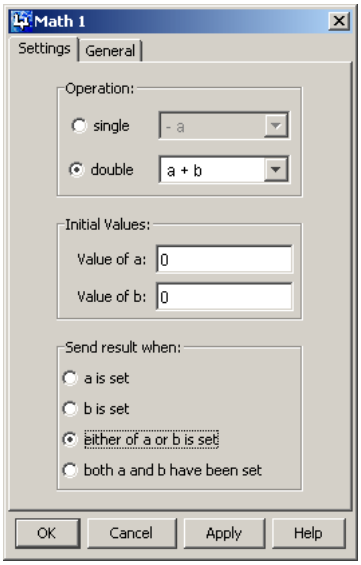

Figure 9.25: The `Math` object's settings dialog

X: (number) Sets the leftmost x-position of the *Target Area* in relation to the *Stage*.

Y: (number) Sets the uppermost y-position of the *Target Area* in relation to the *Stage*.

Width: (number) Sets the *Target Area* width in pixels.

Height: (number) Sets the *Target Area* height in pixels.

Set to stage dimension: (button) (shortcut: ALT+S) Restores the *Target Area* to the *Stage* size, and top-left position at the top-left corner of the *Stage*.

Fixpoint: (choice)

Top Left (alternative) The reference for *Target Area* will be at its top left corner.

Center (alternative) The reference for *Target Area* will be at its center. (Default choice.)

Cursor: (choice) The cursor may change its shape, if set, when it's across the *Target Area*. The change of cursors is layer sensitive, i.e. the top most layer will be predominant. (Read more about layers in Chapter 13). Default is the Unspecified Cursor cursor, which does not set the cursor to any specific shape. This is useful if you want to set a cursor for an underlying *Target Area*.

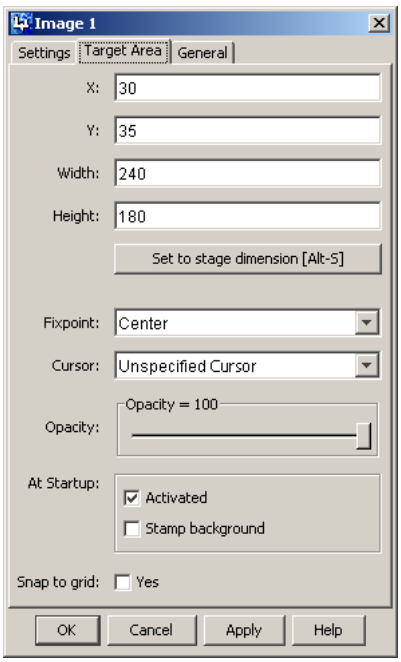

Figure 9.26: *Target Area* tab for an `Image` object

Default	Hand	Move
Crosshair	Text	Wait
SouthWest Resize	SouthEast Resize	NorthWest Resize
NorthEast Resize	North Resize	South Resize
West Resize	East Resize	Unspecified Cursor

Opacity: (slider) The object can be given a certain transparency, which is regulated through this slider. One hundred percent by default gives a totally opaque object. (Not available for `Mouse Area`.)

At Startup: (group)

Activated: (checkbox) Turns the object on at startup. By default checked.

Stamp background: (checkbox) Leaves a permanent impression of the *Target Area* in the background. An `Image` object, for instance, which is to be used as a background image, could benefit from being stamped, as the total image processing is reduced. (Not available for `Mouse Area`). (Read more on layers and *Stamp background* in Chapter 13.)

Snap to grid: (checkbox) Mark this option if you want the object's *Target Area* to snap to the grid, choose Yes. (To change the grid size, see Section 9.7.1.)

10 Connections

You do your programming by wiring objects together. The connections provide a way to exchange information and messages between the objects. Messages come into an object through an *In-port* and is sent out through an *Out-port*.

The ports also have names associated with them, such as *Result [Number]*. The name describes the information that will be sent or received. *Result [Number]* is the out-port data result from a calculation done in a `Math` object. The information found after the port, *[Number]*, indicates that the type of data sent out is a number value (Figure 10.1). Even the data types can be manipulated as they are objects, often found in the *Data* category.

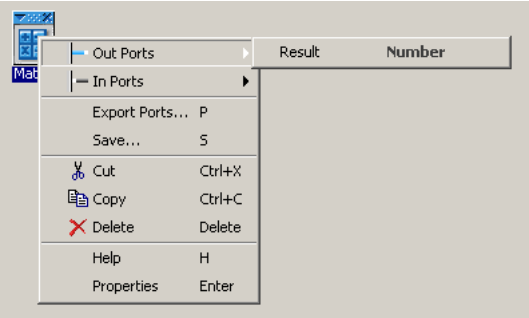

Figure 10.1: `Math` *Out-ports* → *Result [Number]*

In some cases, the information or data is a value, as for example, a `Number`, a `Color`, a `Boolean` or it can even be a `Text`. In other cases, the information is just a trigger to start some action. For example, the object `Sequencer` starts when the *In-ports* → *Run* is triggered. *Run* does not require any *In-port* data (Figure 10.2).

A simple example of sending data would be a `Progressor` which sends out a sequence of numbers, *Progress [Number]*, during a specified time. These numbers can then be received, for example, by a `Brightness` filter and its *In-ports* → *Set Brightness Level [Number]*. The numbers from the `Progressor` will then dynamically set and change the level of the brightness filter in your presentation (Figure 10.3).

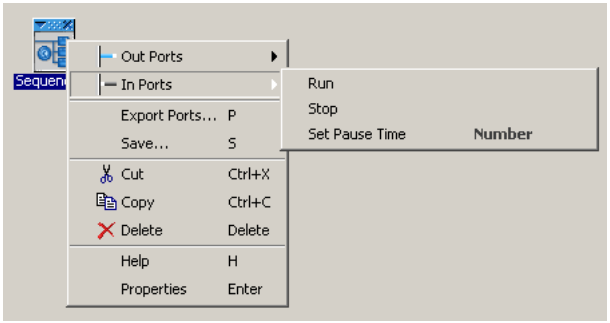

Figure 10.2: `Sequencer` *In-ports* → *Run*

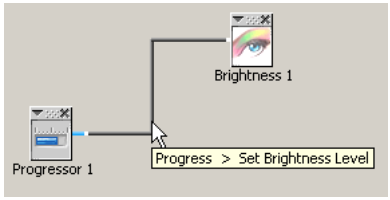

Figure 10.3: A `Progressor` connected to a `Brightness` filter

10.1 Port Constraints

It is important to notice that information containing data values **can only** be sent through ports that have identical data types. The data types can be seen in bold after the port name (Figure 10.4).

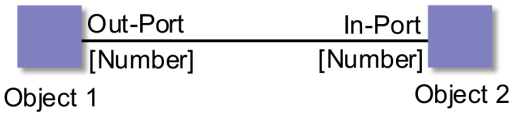

Figure 10.4: A valid connection of a *[Number]* between two objects

Some restrictions between valid *In-ports* and *Out-ports* are automatically detected, the menu options becomes inactive, and forbidden connections cannot therefore be established (Figure 10.5). *In-ports* with no data input can be **triggered** by any *Out-port* with any data (Figure 10.6).

In the prevoius example above, we sent a number *[Number]* from the `Progressor` into the `Brightness` filter. You will **not** be able to connect and send, for example,

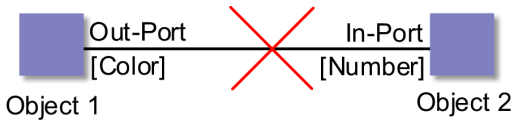

Figure 10.5: **Invalid** connection between two incompatible data types

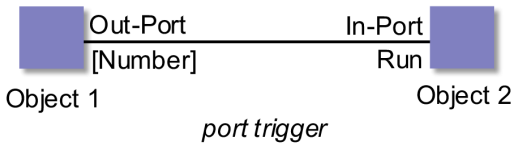

Figure 10.6: Triggering *In-ports* with no specific used data

color information *[Color]* into an object with an *In-port* that requires a number *[Number]*.

However, you can send a number *[Number]* to trigger an *In-port* which doesn't require any data. (Ports that don't require a specific data type don't have any information after the port name.) When you press the mouse button you can let a `Mouse Area` object *Out-ports* → *Mouse Events*→ *Mouse Press [2D Number]* which sends out the coordinates for the mouse pointer to trigger (start) a `Sequencer` *Run*. *Run* does not require any specific data type to start, and will hence be triggered (Figure 10.7).

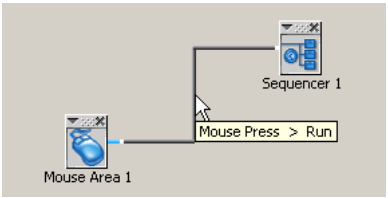

Figure 10.7: A `Mouse Area` triggering a `Sequencer`

But, any of the semantics of the presentation, naturally, cannot be guarateed to work as planned. To get where you want, you will have to carefully study descriptions of the objects and continually test parts of the presentation while developing.

10.2 Object Connections

The whole presentation and wiring you will make in WireFusion is based on the objects you choose and the connections that apply between them. Each object has a specific function and by connecting the objects together they can communicate with each other. The actual programming and connecting is done visually by using the mouse; clicking the object's drop-down menu and select one of the *Out-ports*, then clicking the other object's drop-down menu and selecting one of the *In-ports* to complete the connection.

Example

If you want a button to initiate loading a web page, you connect a `Button` object with an `URL` object. You will learn how to do this connection by the doing the following.

1. Drag and drop a `Button` into the *Wire View*.
2. The `Button` dialog opens when dropped, press OK without doing any changes.
3. Drag and drop an `URL` into the *Wire View*. Its dialog opens when dropped.
4. In the `URL` dialog, enter an URL, e.g. 'http://www.demicron.com'. Then press OK.
5. Click on the drop-down menu of the `Button` object.
6. Select an out port from the `Button`, *Out-ports* → *Button Clicked*.
7. Notice how the mouse cursor has changed.
8. Click on the drop-down menu of the `URL` object.
9. Select an *In-port* from the `URL`, *In-ports* → *Load URL*.
10. Finally test and run the presentation.

Step 1

Insert a `Button` object into your project (you will find the object in the category *Widgets*). Drag the object from the *Object Bar* and drop it in the *Wire View* window. Its dialog window will automatically open. Press OK without doing any changes (Figure 10.8).

Step 2

Insert a `URL` object into your project (you will find the object in the category *Environment*). Drag the object from the *Object Bar* and drop it in the *Wire View* window. Its dialog window will automatically open. Fill in the URL, e.g. 'http://www.demicron.com'. Click on OK to close the dialog ((Figure 10.9).

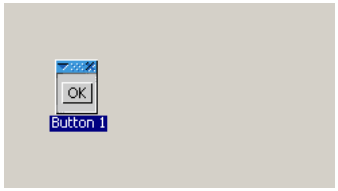

Figure 10.8: `Button` object inserted

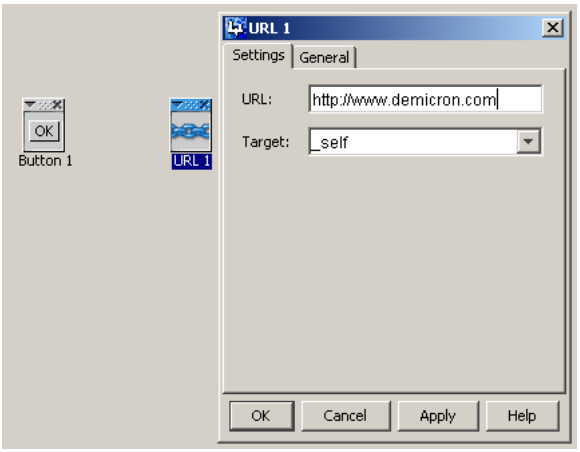

Figure 10.9: The `URL` dialog. Fill in a URL and press OK.

Step 3

Click the drop-down menu of the object that you want to send out information **from**, i.e. the `Button` object (Figure 10.10). You will find the drop-down menu in the upper left corner of the object. Alternatively, by right clicking on the object, you will reach the same menu.

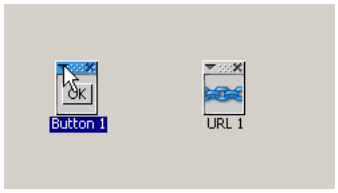

Figure 10.10: Opening the `Button` menu

Step 4

Choose *Out-ports* of the `Button` object, and then *Button Clicked*. You can now release the mouse button (Figure 10.11).

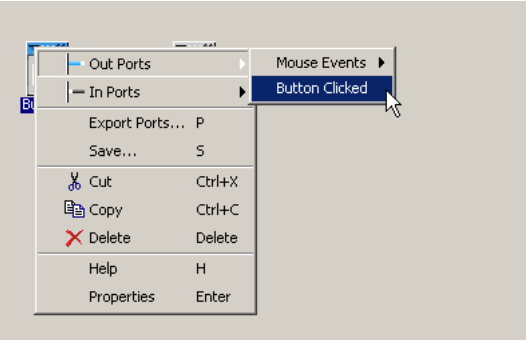

Figure 10.11: Choosing *Out-ports* → *Button Clicked*

Step 5

Notice that the mouse cursor now has changed into a connecting socket plug (Figure 10.12). If you accidentally click somewhere in the *Wire View* beside the `Button` object, the connection will be aborted, and you will have to begin again from Step 3.

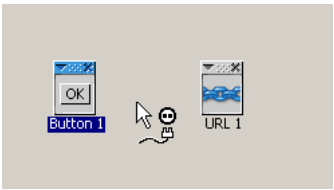

Figure 10.12: In connection mode

Step 6

Click the drop-down menu of the object to which you want to send information **to**, that is `URL` object (Figure 10.13). Or, alternatively, right click on the `URL` object.

Step 7

Choose the *In-ports* of the `URL` object, then *Load URL* (Figure 10.14).

Figure 10.13: Opening the URL menu

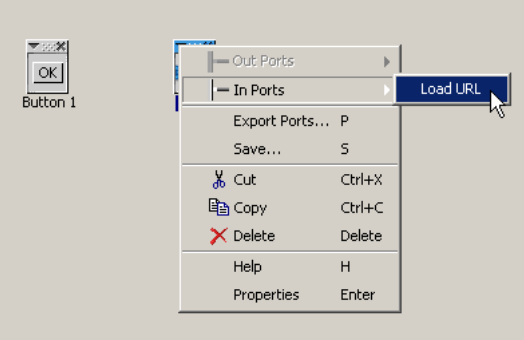

Figure 10.14: Choosing URL object, *In-ports* → *Load URL*

Step 8

The connection is now completed (Figure 10.15).

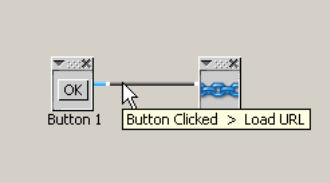

Figure 10.15: Connection completed

Step 9

Test and run the presentation by choosing the *Preview Presentation in Browser* button from the *Menu Bar*or press Ctrl+F9 on your keyboard.

10.3 Colored Connections

The ends of a connection are colored in order to know which end is associated with an *Out-port* and which end is associated with an *In-port*. *Out-port* ends are **blue** with a small dot, while *In-port* ends just have a small white dot as seen in Figure 10.16.

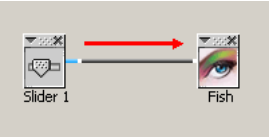

Figure 10.16: Colored connection ends denote *Out-ports* and *In-ports*

10.3.1 Changing Connection Color

Sometimes there are a lot of connections and it can be confusing with all the connection wires. A excellent feature which facilitates and makes it easier to survey your wiring is to change the wire colors. In some cases, when connecting distant objects, it can also be useful to hide the connections by invisible wires (Figure 10.17).

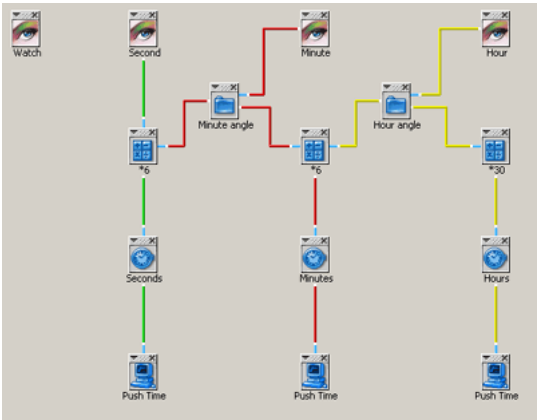

Figure 10.17: Colored wires

To change the color, place your mouse pointer over the wire and right click to have the wire menu opened (Figure 10.18). Possible colors are: *Default*, *Invisible*, *Discreet*, *Red*, *Green*, *Blue* or *Yellow*.

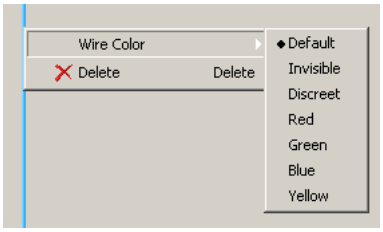

Figure 10.18: Wire menu opened

10.4 Connection Information

In the *List View* window, objects and connections are listed in an ordered way. When you select an object in the *Object List*, all connections to and from this object will be listed in the *Wire List* to the right. *Out-port* connections from the object are listed in the upper part and *In-port* connections in the lower part (Figure 10.19).

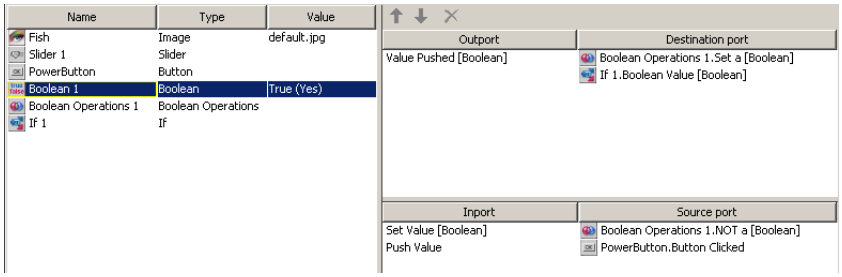

Figure 10.19: *List View*

In (Figure 10.20), we see that a `Slider` object sends its value through the *Out-port* (Outport) *Slider Value [Number]* and enters (Destination port) the `Image` object's *In-port Set Opacity [Number]*.

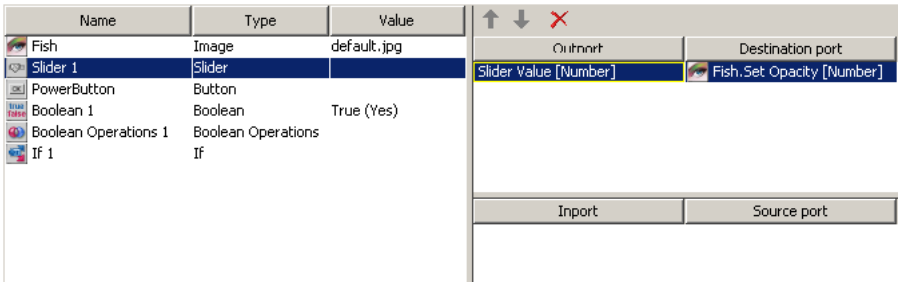

Figure 10.20: `Slider 1` value enters the destination port in `Fish`

We get the reverse information if we instead select the `Image` object. We see

that the *Set Opacity [Number] In-port* (Inport) receives the value from the `Slider` object's port *Slider Value [Number]* (Source port) (Figure 10.21).

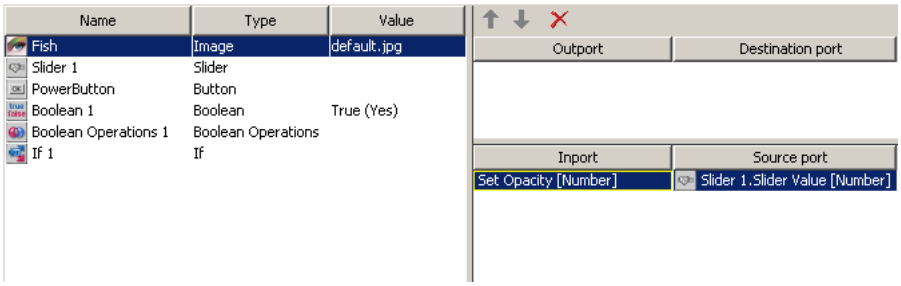

Figure 10.21: `Fish` receives value from the source port *Slider Value [Number]* in `Slider 1`

To get quick and short information about a certain connection in the *Wire View*, just place the mouse over the connection and a tooltip with the connection information becomes visible (Figure 10.22).

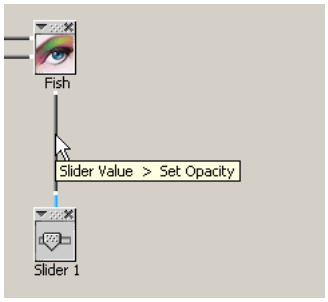

Figure 10.22: Connection tooltip

Alternatively you can select the connection with the mouse and some associated extended information becomes visible in the *Information Bar* (Figure 10.23).

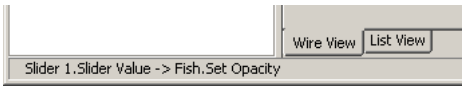

Figure 10.23: Connection information

The connection information '*Slider1.Slider Value − > Fish.Set Opacity*' (Figure 10.24) has the following meaning.

1. **From:** the `Slider` object here termed '`Slider 1`', the *Out-ports→ Slider Value [Number]* sends out a *[Number]* with the slider value stored in it.

2. **To:** the slider value, stored in the *[Number]*, enters the `Image` object here termed '`Fish`' and its *In-ports* → *Set Opacity [Number]*, which sets the opacity level.

When connecting objects by invisible wires, the *Connection Information* tooltip becomes more extensive and also tells you which objects the information is coming from and going to (Figure 10.24).

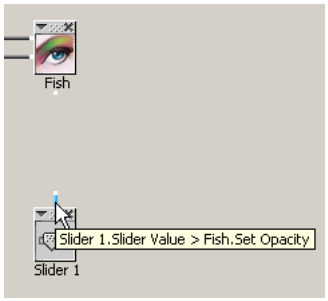

Figure 10.24: Tooltip for invisible connections

10.5 Connection Order

The order of how connections are sent out from an object in one specific frame can be controlled in the *Wire List*. (Figure 10.25) shows one `Button` object, three `Text` objects and one `Text Window` object. They are connected so that when then button is clicked, it will push the text contents from the three individual `Text` objects, named Start, Middle and End. Start contains the text 'Start', Middle the text 'Middle' and End the text 'End'. The three texts will then be appended to the `Text Window` object, which will display the text in the presentation.

The following connections were made:

1. `Button 1` *Button Clicked* → `Middle` *Push Text*
2. `Button 1` *Button Clicked* → `End` *Push Text*
3. `Button 1` *Button Clicked* → `Start` *Push Text*
4. `Middle` *Text Pushed [Text]* → `Text Window 1` *Append [Text]*
5. `End` *Text Pushed [Text]* → `Text Window 1` *Append [Text]*
6. `Start` *Text Pushed [Text]* → `Text Window 1` *Append [Text]*

If we select the `Button` object and look in the *Wire List* we see the order of how the three `Text` objects will be pushed. The current append order is, first the

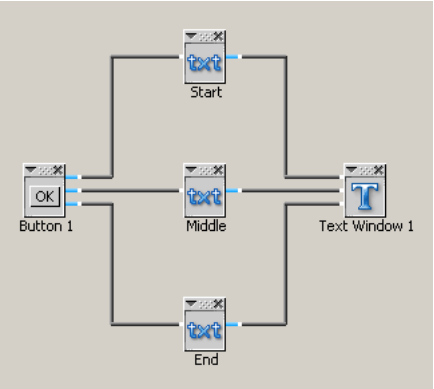

Figure 10.25: Three texts appended in the `Text Window` object

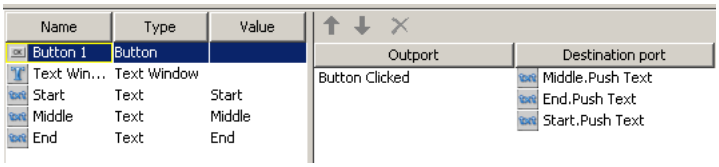

Figure 10.26: *Wire List* with the current connection order

'Middle' text, second the 'End' text and last the 'Start' text (Figure 10.26). The current order is based on the actual order we made the connections.

A preview of appending the text in this connection order is shown in (Figure 10.27).

These three events will be triggered from the same *Out-port* (*Button Clicked*), and in one and the same frame (simultaneously), we can therefore sort them using the up and down arrows found in the *Wire List* (Figure 10.28, Figure 10.29 and Figure 10.30).

A preview now shows how that the append text currently is in the desired order (Figure 10.31).

In-port connections cannot be re-ordered. Only connections wired to one and the same out-port can be re-ordered.

If you want to trigger events in a specific order but with a time delay between the events, then you will have to use a `Sequencer` object instead.

10.6 Inhibit Connections

You can control or inhibit connections by deleting or aborting them.

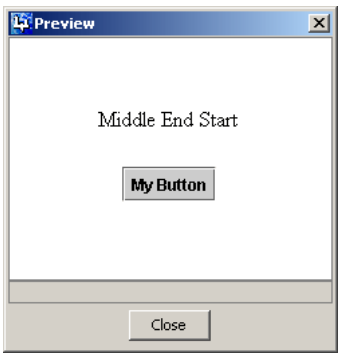

Figure 10.27: Preview. Text appended as 'Middle End Start'.

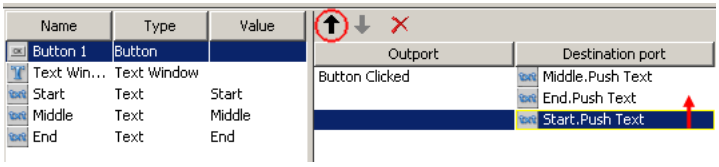

Figure 10.28: Start-connection triggered last, moving it up

Delete a connection

To delete a connection, select a connection with the mouse and click the Delete in the *Menu Bar*, or pressing the *Delete Key* on the keyboard.

Abort connection mode

If you have selected, for example, one of the *Out-ports* in an object, the program expects you to choose one of the *In-ports* in another object, in order to finish the connection. After choosing a port (out or in) the Cancel connection mode connection mode and the connection mode button will be visible in the *Menu Bar*. Click it or click anywhere in the *Wire View* to discontinue the connection.

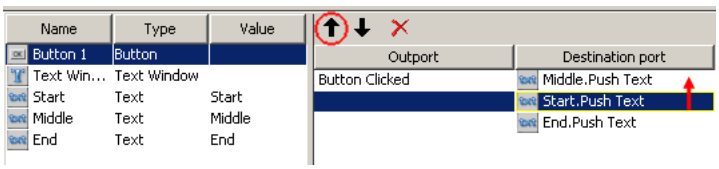

Figure 10.29: Start-connection triggered secondly, moving it up

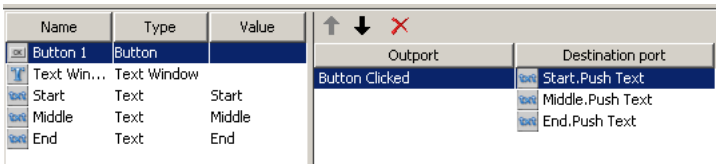

Figure 10.30: Start-connection triggered first

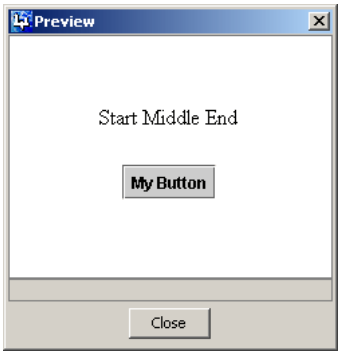

Figure 10.31: Preview. Text appended correctly, 'Start Middle End'.

11 Forwarding and Storing Data

11.1 Parameters

Some *In-ports* expect to receive data in order to function properly. This data is called a parameter. For example, if you connect a `Mouse Area` object to move an `Image` object across the screen, the `Image` object expects to receive a parameter specifying the position to move to (a *[2D Number]*, in this case). A parameter can for example be a number, a text, a color, a boolean value (true or false) or any of the other WireFusion data objects, which can be found in the *Data* category. Some *Out-ports* also generate parameters and include them with the messages they send.

11.2 Push parameters

Data parameters, as for example, numbers, colors, texts and boolean values are stored in separate objects. The numbers, for example, can be a position, a dimension or a parameter value, such as the brightness level in the `Brightness` object. Other objects sometimes need this data. To foward data, on a certain demand, from one object to another the *Push* and *Pushed* ports are used. The *In-port* → *Push*, works as a trigger and when it is triggered, it tells the object to send the stored data through the *Out-port* → *Pushed* (Figure 11.1).

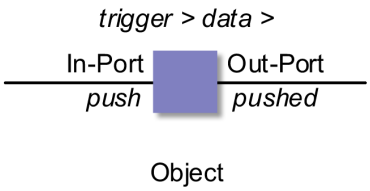

Figure 11.1: Forwarding data through an object

Example

Say that we want to display a number in the *Console* window, when the user clicks a button (Figure 11.2).

1. Drag and drop a `Button` into the *Wire View*.
2. The `Button` dialog opens when dropped, press OK without doing any changes.
3. Drag and drop a `Number` into the *Wire View*. In its dialog, replace the default number 0 with number 10. Then press OK.
4. Drag and drop a `System` into the *Wire View*.
5. Connect `Button` *Out-ports* → *Button Clicked* with `Number` *In-ports* → *Push Value*
6. Connect `Number` *Out-ports* → *Value Pushed* with `System` *In-ports* → *Console* → *Print Number [Number]*
7. Finally test and run the presentation using the *Console* window

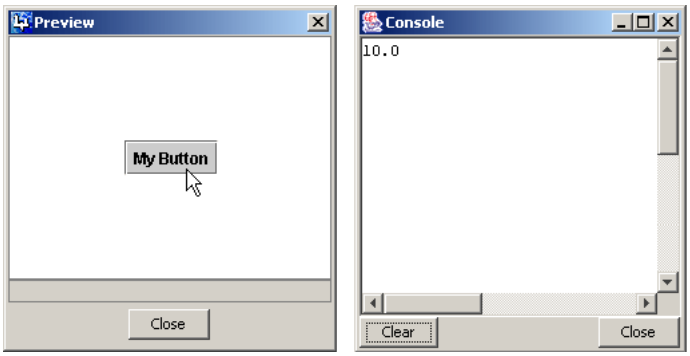

Figure 11.2: Preview of number shown when the user clicks the button

Step 1

Open a new project and place a `Button` object, found in the *Widgets* category. The button *Target Area* should be placed in the middle of the *Stage*. Leave it there.

Step 2

The `Button` dialog opens when dropped, press OK without doing any changes.

Step 3

Drag and drop a `Number` object, found in the *Data* category, into your project. When its dialog window opens, replace the default number 0 with number 10. Then press OK.

Step 4

Drag and drop a `System` object, found in the *Environment* category, into your project.

Step 5

To trigger the `Number` object and to make it send out the stored number 10 when the button is clicked, connect the `Button` *Out-ports* → *Button Clicked* with the `Number` *In-ports* → *Push Value* (Figure 11.3 and Figure 11.4).

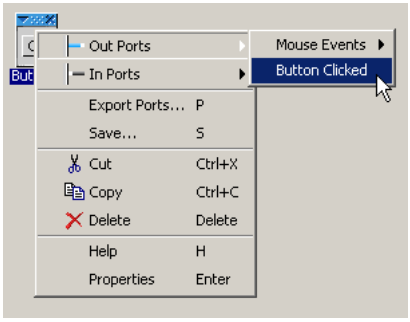

Figure 11.3: `Button` *Out-ports* → *Button Clicked*

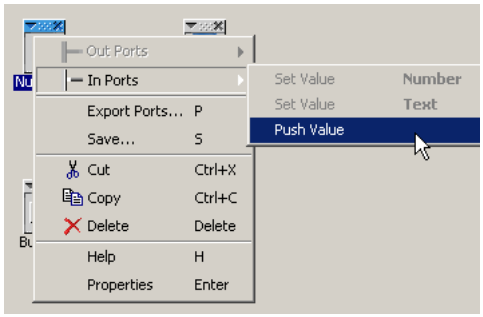

Figure 11.4: `Number` *In-ports* → *Push Value*

Step 6

Now, the stored number has been triggered and will be sent out through the `Number` *Out-ports* → *Value Pushed [Number]*. We will connect this port with the `System` *In-ports* → *Console* → *Print Number [Number]*, in order to have the number displayed in the *Console* window (Figure 11.5 and Figure 11.6).

In this step, the *Out-port* sends out the data type *[Number]* to an *In-port*, which requires a data of the same type, i.e. *[Number]*.

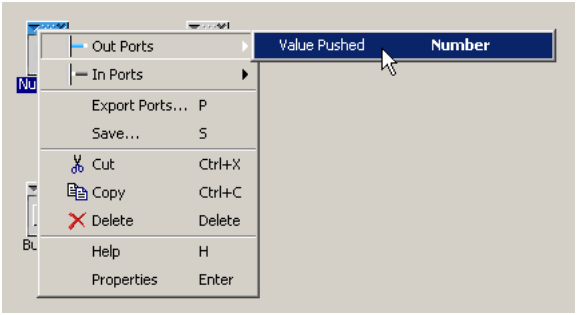

Figure 11.5: `Number` *Out-ports → Value Pushed [Number]*

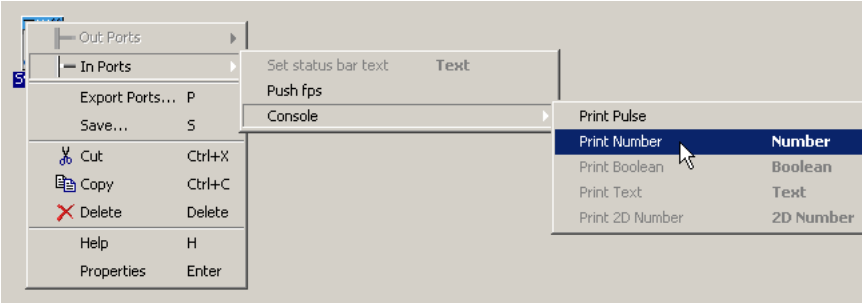

Figure 11.6: `System` *In-ports → Console → Print Number [Number]*

Step 7

The connections are done, you are now ready to test and run the presentation. First open the *Console* window by menu *View* > *Console*, or press Ctrl+Shift+C, then preview the presentation by pressing F9 (Figure 11.7 and Figure 11.2).

11.3 Set parameters

Sometimes it is useful to store parameters in a data object, and a method to do this is to use the *In-port Set*. Let us illustrate how it works with a step-by-step example.

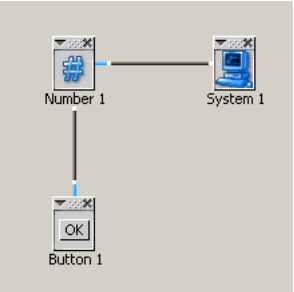

Figure 11.7: Final connections between `System`, `Text` and `Button`

Example

We want to store an animated number in a `Number` object and when a button is clicked the stored number will be displayed in the *Console* window.

1. Follow step 1 to 7 above in the 'Push parameters' example, Subsection 11.2.
2. Drag and drop a `Progressor` object into your project. In its *Properties* dialog, change At startup from Run to Loop. Then press OK.
3. Connect `Progressor` *Out-ports* → *Progress [Number]* with `Number` *In-ports* → *Set Value*.
4. Finally test and run and the presentation using the *Console* window.

Step 1

Follow step 1 to 7 above in the 'Push parameters' example (Subsection 11.2) and make sure it works.

Step 2

Drag and drop a `Progressor` object, found in the *Logic* category, into your project. In its dialog, at At startup, switch from Run to Loop. Leave the rest as it is. Then press OK. The `Progressor` will automatically start and loop from 0 to 100 in 1 second, when the presentation is started.

Step 3

The `Progressor` numbers are sent out through the *Out-port Progress [Number]*. We will use them to set the `Number` object, connect the `Progressor` *Out-ports* → *Progress [Number]* with `Number` *In-ports* → *Set Value [Number]* (Figure 11.8).

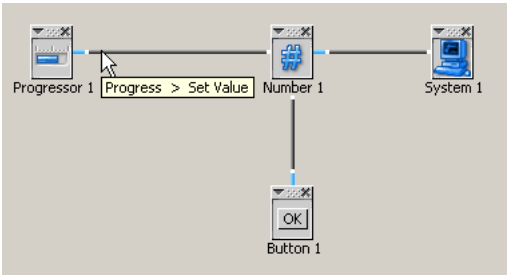

Figure 11.8: `Progressor` sets the `Number`

Step 4

The connections are done, you are now ready to test and run the presentation. First open the *Console* window, choose *View* > *Console*, or press Ctrl+Shift+C, then preview the presentation by pressing F9. The `Progressor` updates and sets the `Number` continuously with new numbers. When you press the button, the current stored number will be displayed in the *Console* window (Figure 11.9).

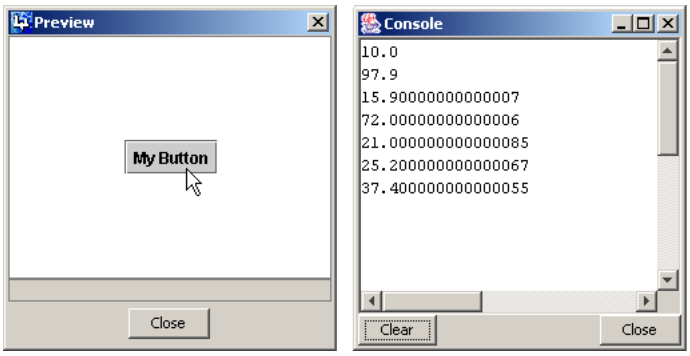

Figure 11.9: *Console* window with numbers displayed when button is clicked

12 Grouping Objects

To help you organize your project and eliminate visual complexity, you can group objects into grouping objects. In WireFusion there are two types of grouping objects, i.e. objects that can contain other objects; the `Folder` object and the `Scene` object. They are very much alike, with the difference that the `Scene` object has a *Stage* while the `Folder` object does not.

Generally, you group together a set of objects that perform a certain task. For example, if a part of your project is a visual display, like a mobile phone display, then you could group the set of objects that accomplishes this into a `Scene` object. Else, if you have a logic or mathematical function, something that doesn't have any visual contents, then you could group the objects into a `Folder` object instead.

A group of objects could be considered as stored inside a grouping object (a `Scene` object or a `Folder` object). The grouping implies exporting relevant ports to the outside of the group. This is because the ports will not be visible unless exported to objects at the same level as the grouping object (Figure 12.1). Read more about exporting ports in the Port Exporter section below (Section 12.3).

A grouping object can be saved and reused in later presentations or just cloned within the same presentation. As a grouping object is in itself an object, grouping objects can nest. That is, they can be 'inside' themselves. In fact a whole presentation can be inside itself, because the `Scene` object can nest.

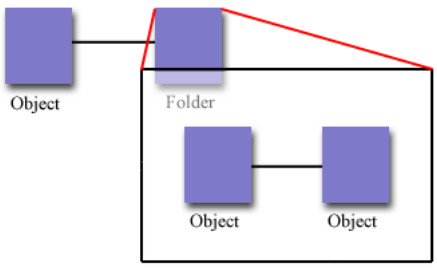

Figure 12.1: A group of objects gathered in a `Folder`

12.1 The `Scene` Object

The `Scene` object has a special function in WireFusion, as it can be seen as the foundational object. When starting WireFusion, a new and empty project opens by default. It is in fact an empty `Scene` object that is opened. The root object of all presentations is always a `Scene` object and it contains all other objects in your presentations, and your final presentation will be displayed on its *Stage*.

You develop your presentation by connecting objects together in the *Wire View*. They are all related to the root `Scene` object. The `Scene` object also has its *Work Area*, which contains the *Stage*, where the final presentation will be displayed. In here you can arrange, move or resize the *Target Area*s of your images, mouse areas etc (Figure 12.2). (Read more on *Target Area*s in Section 9.7).

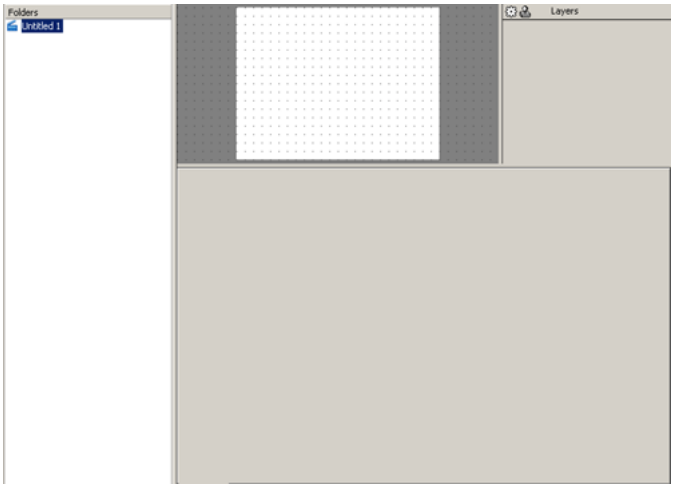

Figure 12.2: The *Wire View* and *Work Area* of the root `Scene` object

Saved `Scene` objects can be imported into your present project and work as local scenes. Normally you use a `Scene` object in your presentation, as a local scene, if you want to group a visual function. For example it could be a product display, which contains all the graphics and all the functionality of the display (Figure 12.3). You can then easily turn the display on and off, or re-use it in other presentations.

Interactivity (mouse events) that you create in a `Scene` works even if you import the `Scene` into a project and uses it as a local scene. For example, this means that you can have the complete function of a touch sensor screen stored in a `Scene` and when you use it as a local scene, the touch sensor functionality is still working (Figure 12.4).

Figure 12.3: A `Scene` object used as a display

Figure 12.4: A working touch sensor screen used as a local scene (marked with a red border)

12.2 The `Folder` Object

The `Folder` object is very much the same as the `Scene` object, except that it lacks the *Work Area* and the *Stage*. It is normally used when creating non-visual functions, like a logical function for example (Figure 12.5).

A function created and stored in a `Folder` object can be saved, just like any other WireFusion object, and reused in other projects (Figure 12.6) or just cloned within the same project (Figure 12.7 and Figure 12.8).

Objects having a *Target Area* (such as `Image`, `Blur` or `Mouse Area`) cannot function inside a `Folder` object (as it has no *Stage*) and their icons are therefore disabled both in the *Object Bar*and in the *Objects* menu when you are located in a `Folder`.

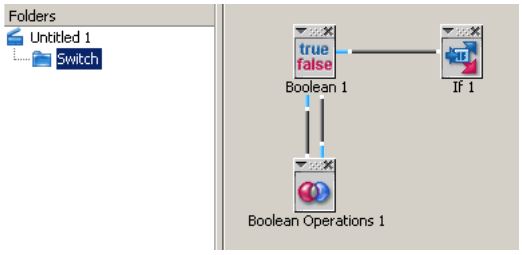

Figure 12.5: A switch function stored in a `Folder` object

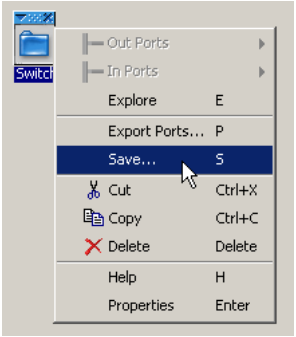

Figure 12.6: Saving an object

If you have connected a set of objects together that perform a certain task (let's say that they are placed and connected in the *Wire View* of your main `Scene` object), and then would like to store them in a `Folder` object, you could then simply select the objects using your mouse and then cut/copy and paste them into a `Folder` object (or `Scene` object if any of the objects has a *Target Area*). Connections between the copied objects will be preserved, while connections to objects that aren't copied will be broken (Figure 12.9). To connect or re-connect objects from inside a `Folder` (or `Scene`) to objects outside the `Folder` (or `Scene`), you need to use the Port Exporter. Read about the Port Exporter below.

12.3 The Port Exporter

When creating a function that should perform a certain task, you place and connection objects in a grouping object. It could for example be in a `Folder` object. The objects inside the `Folder` are hereby isolated from other objects, outside the `Folder` itself. They are not at the same hierarchical level as the other objects (outside the `Folder`), and can hence not be connected to them directly. To solve this problem of isolation from objects outside the `Folder`, the *Port Exporter* is used. The *Port Exporter* (as its name implies) exports objects' in- and out-ports up one level in the

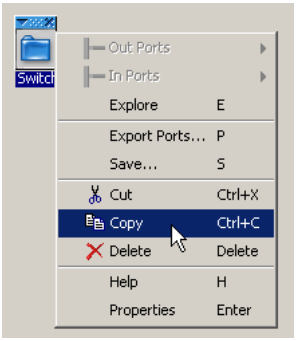

Figure 12.7: Copying an object

Figure 12.8: Pasting an object

object hierarchy.

We illuminate this with an example. An object (`Object 1`), placed for example in a `Folder` object, is one level below another object that is placed beside the `Folder` object (`Object 2`). See Figure 12.10. In order to connect the two objects together (i.e. `Object 1` and `Object 2`), let's say we want to connect port x from `Object 1` to `Object 2`, In order to achieve this we then need to export port x up one level, so that it becomes visible in the `Folder` object's local menu. We could then connect it with `Object 2` (Figure 12.11).

When the *Port Exporter* dialog is opened, you can choose to export *In-ports* by clicking the *In-port* tab or *Out-ports* by clicking the *Out-port* tab (Figure 12.12).

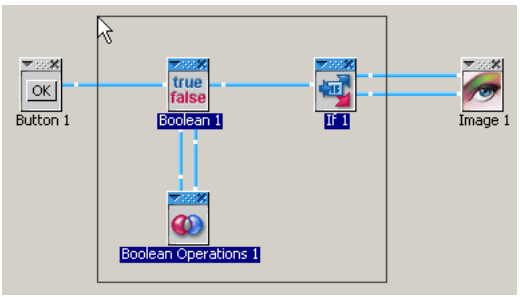

Figure 12.9: Selecting several objects

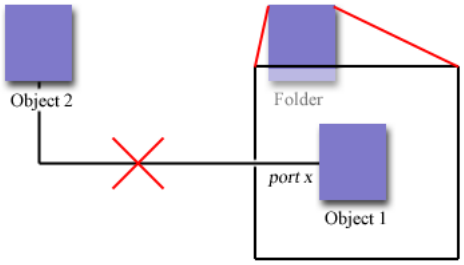

Figure 12.10: Not possible to connect `Object 1` to `Object 2` directly, without exporting port x up one level in the object hierarchy

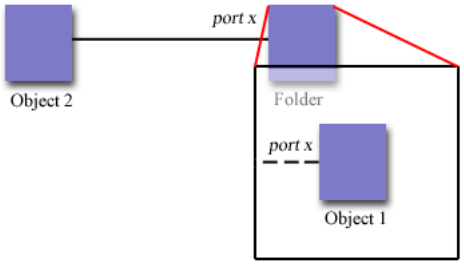

Figure 12.11: Possible to connect `Object 1` and `Object 2`, as port x is exported and visible in the `Folder` local menu

12.3.1 Exporting ports

The procedure of exporting ports is explained by the following example.

Example

We want to create a simple function inside a `Folder`. Then we want information from outside the `Folder` to enter an object located within the `Folder`. This trivial example will play a sound when the mouse button is pressed.

1. Start a new project. Place a `Folder` and a `Button` in the *Wire View*.
2. Open the `Folder` object by double clicking, and place a `Sound` object inside it.
3. Select the *Export Ports...* from the `Sound` object's menu. The *Port Exporter* dialog is opened.
4. In the *Port Exporter*, tab *In-ports*, mark the *Exported* check box for the *Start In-port*.

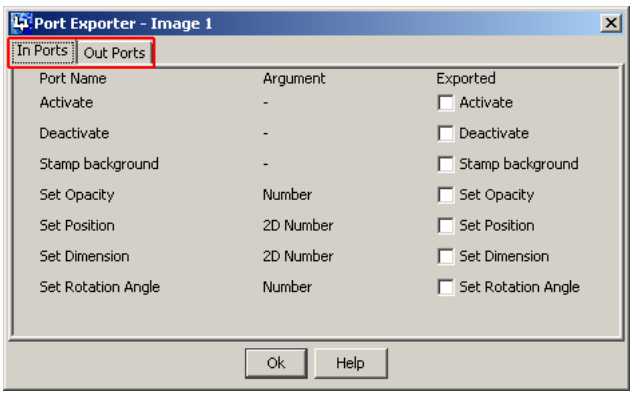

Figure 12.12: *Port Exporter*for an `Image` object

5. When prompted, rename the default name *Start* to *Start Sound* instead.
6. Click the `Up` button to move up in the hierarchy.
7. Connect `Button` *Out-ports* → *Button Clicked*, to `Folder` *In-ports* → *Start Sound*.
8. Test and run the presentation.

Step 1

Start a new project. Place a `Folder` (found in the *Misc* category) and an `Button` (found in the *Widgets* category) into the *Wire View* (Figure 12.13). If you open the `Folder` menu, you will see that there are no *Out-ports* or *In-ports* available (Figure 12.14).

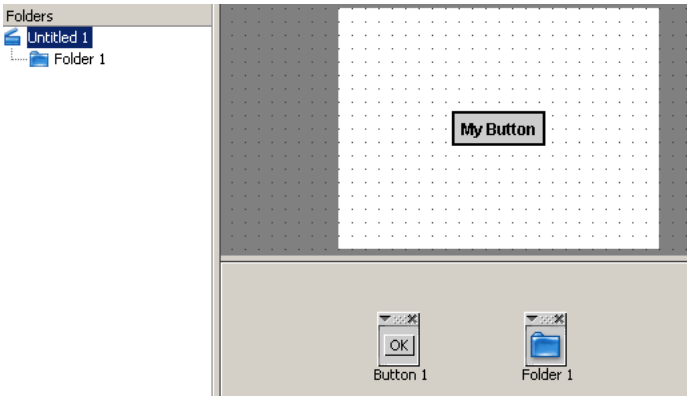

Figure 12.13: A `Folder` and a `Button` in a new project

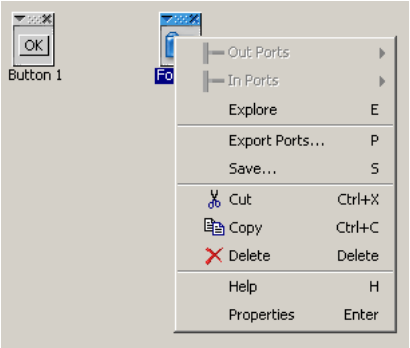

Figure 12.14: A `Folder` with **neither** *Out-ports* nor *In-ports*

Step 2

Double click the `Folder` to open it up (or select it and press E on your keyboard), then place a `Sound` object in the `Folder`. The `Sound` object's dialog opens when dropped and by default it contains a sound (a ring signal), so there is no need to browse for a new sound file. Press OK (Figure 12.15).

Figure 12.15: A `Sound` object inside the `Folder`

Step 3

Open the `Sound` object menu and select *Export Ports...* (Figure 12.16). The *Port Exporter* dialog will now be opened.

Step 4

In order to make the `Sound` *In-port* → *Start* visible in the `Folder` menu, we need to mark the *Exported* check box for the *Start In-port*(Figure 12.17).

Step 5

When the check box has been marked, you will be prompted to rename the port if you would like. Let's change the default *Start* to *Start Sound*. When you are done, press OK (Figure 12.18).

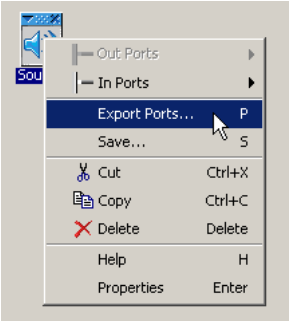

Figure 12.16: Opening the *Port Exporter*

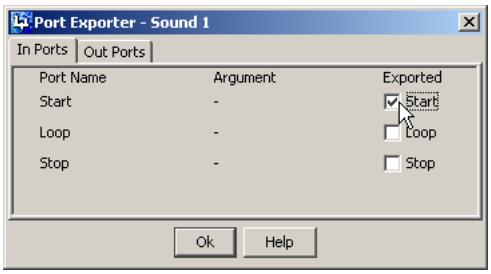

Figure 12.17: Exporting the *Start* port

Step 6

The `Sound` *In-port Start* (renamed *Start Sound*) has now been exported up **one level**, and is visible in the `Folder` menu. Click the `Up` button (found in the program *Menu Bar*) to move up in the object hierarchy. Now, the `Folder` and `Button` objects should be visible in the *Wire View* again. Open the object menu of the `Folder` to see that the *Start Sound* port is visible (Figure 12.21).

Step 7

Now you can let the `Button` object trigger the `Sound`. Connect `Button` *Out-ports* → *Button Clicked*, to `Folder` *In-ports* → *Start Sound*.

Figure 12.18: Changing the port name to *Start Sound*

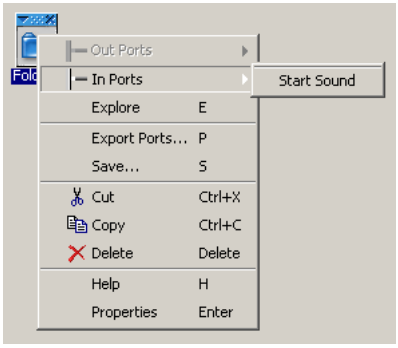

Figure 12.19: The *Start Sound In-port* visible in the `Folder` menu

Step 8

The simple project is done. A `Button` triggers a `Sound`, which resides in a `Folder`. Press F9 to preview your presentation.

12.3.2 Exporting Ports more than one level

In the example above we exported a port **one level**, from the `Sound` object placed in a `Folder` so that it became visible in the `Folder` menu. However, sometimes it's necessary to export several levels. For example, let's say that you have placed a `Sound` object (`Sound 1`) in a `Folder` (`Folder 2`), which in turn is placed in yet another `Folder` (`Folder 1`) (Figure 12.20).

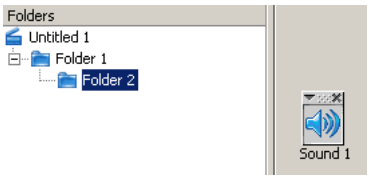

Figure 12.20: A `Sound` in a `Folder`, which is placed in yet another `Folder`

Analogously with the example above, you just use the *Port Exporter* to export the object port so that it becomes visible, first in the `Folder 2` menu, then you enter the *Port Exporter* from the `Folder 2` menu and export it again (Figure 12.21). Now it's visible in the `Folder 1` menu too (Figure 12.22).

12.3.3 Sorting Ports

The *Port Exporter* also helps you to sort the ports you have exported. Simply open the *Port Exporter* from the object you have exported the ports to (this means the `Folder` object for the above example), and then use the *Move Up* and *Move Down*

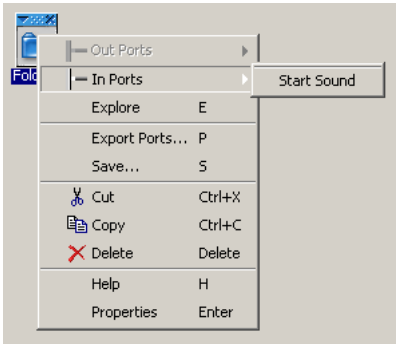

Figure 12.21: The *Start Sound In-port* visible in the `Folder 2` menu...

buttons to move the ports (Figure 12.23). You can also rename already exported ports here (Figure 12.24). The order in which the ports appear in the *Local Object Menu* does **not** affect anything, because of its internal order.

You can also create submenus when naming the exported ports. This is done by using a pipe '|' (Figure 12.25 and Figure 12.26).

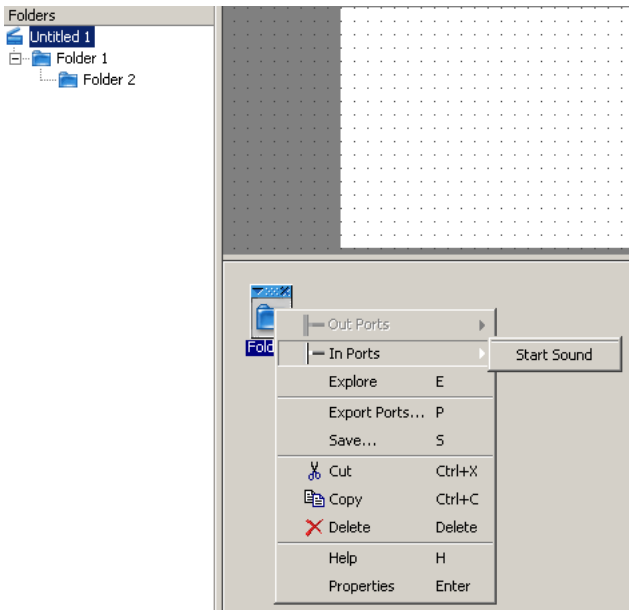

Figure 12.22: . . . and then also in the `Folder 1` menu

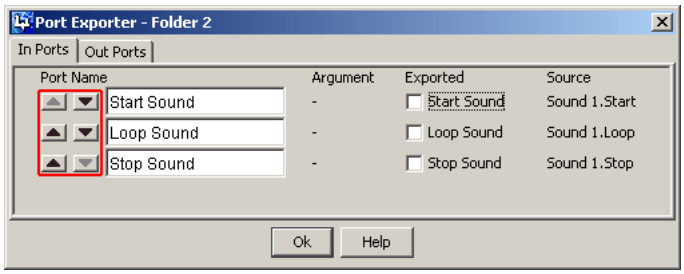

Figure 12.23: Changing the port order

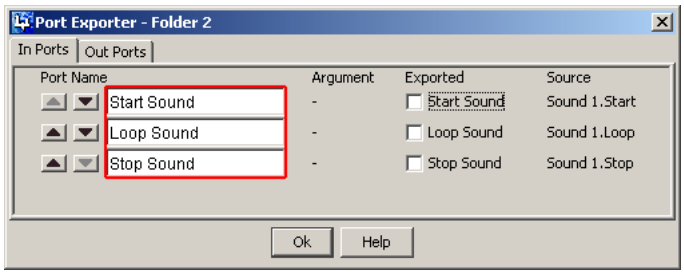

Figure 12.24: Changing the port name

Figure 12.25: Creating submenus with '|'

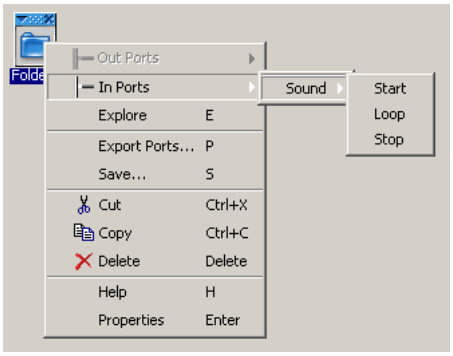

Figure 12.26: A submenu named *Sound* in the `Folder` menu

13 Layers

Layers are similar to acetate or transparent film, stacked on top of each other (Figure 13.1). You probably know how Adobe Photoshop®or Macromedia Flash®work with their layers. The layers in WireFusion do not work exactly the same, but the concept is similar.

In Photoshop, for instance, when a filter is placed in a certain layer it will only have an impact on the graphics in that specific layer, but in WireFusion a filter will affect all layers below itself (Figure 13.2).

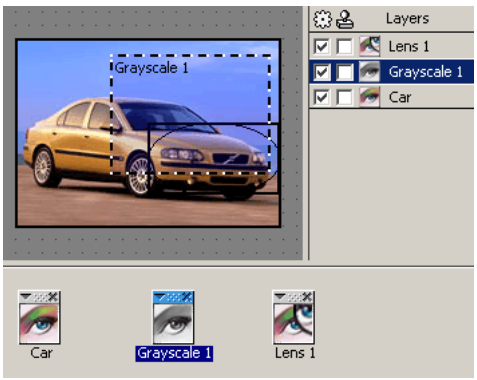

Figure 13.1: The *Layers* in WireFusion

Figure 13.2: The result

All objects in WireFusion having a *Target Area* will be listed in the *Layers* view, these are usually objects with some kind of visual contents (such as an image, button or slider) or a visual effect (such as a filter function). Each layer in WireFusion can only hold one and only one object, and layers cannot be merged.

When you mark a layer in the *Layers* view, it will temporarily pop up in the front in the *Work Area* so that you can re-position and re-size it with greater ease.

There is no predefined limit of how many layers you can have in WireFusion.

As each layer contains only one object, you can also use the *Layers* view for making connections between objects or entering their properties dialog. Right click on a layer to open its object menu (Figure 13.3). (Section 9.7).

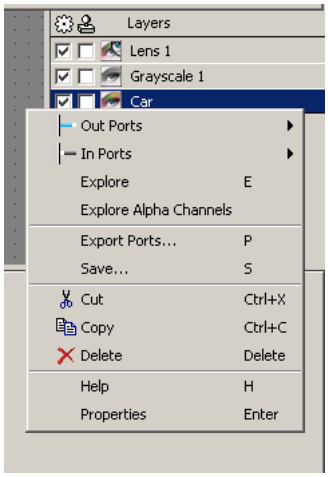

Figure 13.3: Right click to open object menu

13.1 Reordering layers

When you create a new WireFusion project, it contains no layers. Layers are created automatically in WireFusion. Whenever an object, having a *Target Area*, is dragged into the project, a new layer will automatically pop up in the *Layers* view. New layers are always placed on top.

To change the order of layers:

1. In the Layers view, select the layer that you want to move (Figure 13.4).
2. Drag the layer up or down in the *Layers* view. When the red highlighted line appears in the desired position, release the mouse button (Figure 13.5).

13.2 Activated and deactivated objects

Two checkboxes appear to the left side of the name of the layer, *Activate* and *Stamp background*. (Duplicate commands reside in each objects' *Properties* dialog, see Section 8.5).

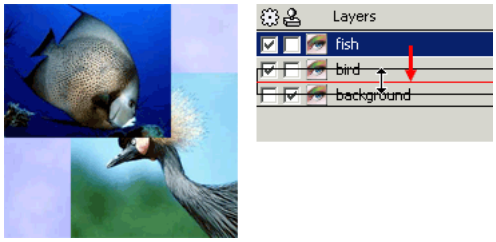

Figure 13.4: Dragging of Fish layer

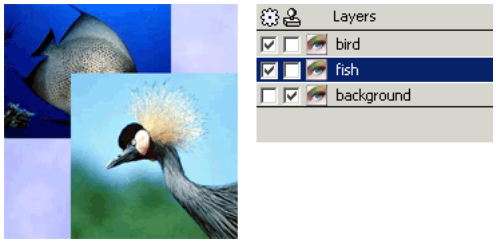

Figure 13.5: Fish layer dragged below Bird layer

When the left most checkbox is marked, the object is automatically turned on at startup. It is now activated. When it is unmarked, it is deactivated (Figure 13.6).

Figure 13.6: Bird image activated

Activate implies that the object is updated or redrawn every time in each new frame. The object remains floating, i.e. it can be repositioned and resized, and it will have an impact on the background, but not permanently. It depends on the object, and on the size of the object's *Target Area*, as to how much the presentation will consume of the computer's CPU. E.g. `Blur` demands more of the computer's computational resources, than `Invert`.

Deactivate implies that the *Target Area* is not updated at all and will hence be invisible. However, invisible objects will not consume any CPU usage, no matter which object or *Target Area* size.

An object can be activated and deactivated during a presentation through the object's in ports *Activate* and *Deactivate*. See Section 9.7.3.

The right most checkbox, *Stamp background*, lets everything you apply stick in

the visual background of your presentation permanently. By marking the right most checkbox, the object will stamp the background once at startup (Figure 13.7).

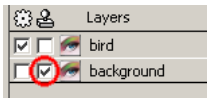

Figure 13.7: Background image stamped in the background

An object can leave a stamp, at any time, in the background during a presentation through the object's *In-port Stamp background*. See Section 9.7.3.

Unmark the *Activate* checkbox and mark the *Stamp background* checkbox if you want an image to work as a permanent background image without consuming any extra CPU.

13.3 Mouse Events

All objects having a *Target Area* also have an out-port called *Mouse Events* (see Section 9.7.2). These ports control the mouse events when the mouse cursor is within the object *Target Area*. However, mouse events are only turned on when the object is activated, i.e. They will be turned off when the object is deactivated or stamped in the background.

Mouse events are sensitive to layer orders. An object placed in a layer above other objects will block any mouse events to the objects below itself. However, if there are no connections to the object's mouse events ports, then it will not block mouse events to other underlying objects.

14 Alpha channels

Alpha channels is a graphic term for a quite simple concept. Normally, image files store the visibility components of color in three channels, the RGB channels (Red, Green, Blue). Every dot in a picture is then composed of a blend of varying values — these values signify 256 shades each of red, blue and green.

Some bitmaps can also store 256 levels of transparency — they have the ability to incorporate an `Alpha Channel`. This channel is basically a selection or mask represented in 256 colors, normally of the grayscale spectrum. White stands for 100 percent opaque, black represents 100 percent transparent and the shades of gray in between, represent varying degrees of transparency.

14.1 Alpha channels in WireFusion

The `Alpha Channels` in WireFusion essentially work as described above, but with the difference that a bluescale is used to represent the opacity, instead of a grayscale spectrum. The alpha channel has 256 levels (or 8 bits) of transparency. Parts or components of the alpha channel graphics having blue values equal to 0 are completely transparent, blue values equal to 255 are totally opaque and blue values between 0 and 255 represent shades of transparency.

Generally, all objects in WireFusion having a *Target Area*, and a visual effect displayed in it, also have an alpha channel that can be used to shape the otherwise rectangular *Target Area*. You can easily see if an object has an alpha channel, it then has an `Alpha Channels` icon below its icon in the *Folders* view (Figure 14.1).

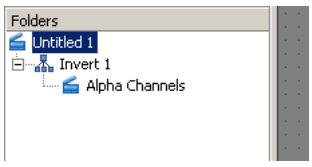

Figure 14.1: An `Invert` object with its `Alpha Channels`

As you can see, the `Alpha Channels` icon has the same icon as a `Scene` object, and in fact, it is a `Scene` object. Everything you do in this `Scene`, and on its *Stage*, will work as your alpha channel. This means that you can have animated and interactive alpha channels in WireFusion.

The best way to learn and understand the alpha channels is by a step-by-step example.

Example

We want an invert filter to be shaped like a star and applied on an image.

1. Place an `Image` in the *Wire View*.
2. Place and `Invert` in the *Wire View*. Change its dimensions to 50×50 pixels.
3. Open the `Invert Alpha Channel`.
4. Change the `Alpha Channel` *Stage* dimension to 50×50 pixels.
5. Place an `Image` on the `Alpha Channel` *Stage*. Select a 50×50 pixels alpha map shaped as a star.
6. Preview the presentation.
7. Make the `Invert` moveable. Connect the built-in mouse events in the `Invert` object to its *Set Position [2D Number] In-port*.

Step 1

Place an `Image` object in a new and empty project. When its dialog opens, click OK to close it without changing anything.

Step 2

Place an `Invert` object and when its dialog opens, change the *Target Area* dimensions (width and height) to 50×50 pixels. Then click OK to close the dialog. Make sure the `Invert` is placed in a layer above the `Image`, as you want the invert filter to operate **on** the image (Figure 14.2).

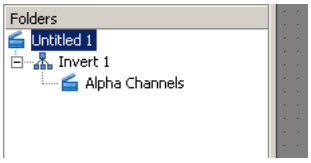

Figure 14.2: The `Invert` object in a layer above the `Image`

Step 3

Now we want to use an alpha channel to shape the `Invert` *Target Area*. Navigate to the `Invert Alpha Channels` in the *Folders* view, or alternatively, right-click the `Invert` object to open its local menu and select *Explore Alpha Channels* to jump there directly (Figure 14.3).

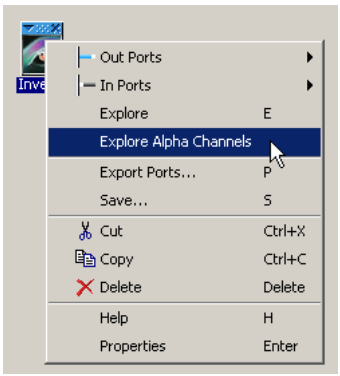

Figure 14.3: Jump to the `Alpha Channels`

Step 4

You now see the `Alpha Channel` stage. Resize the stage by selecting *Scene*, then *Scene properties*. Set width to 50 pixel and height to 50 pixel (50×50 pixels).

Step 5

Place an `Image` object in the `Alpha Channel` *Stage*. When its dialog opens, change the image to a bluescale image, or an alpha channel image (Figure 14.4). Then click OK to close the dialog. Place the image so that it covers the *Stage* (Figure 14.5).

Step 6

You can now preview the presentation, press F9. A resulting inverting is now shaped after the alpha channel image (Figure 14.6).

Step 7

Use the built in mouse events in the `Invert` object to move the object itself, connect (Figure 14.7):

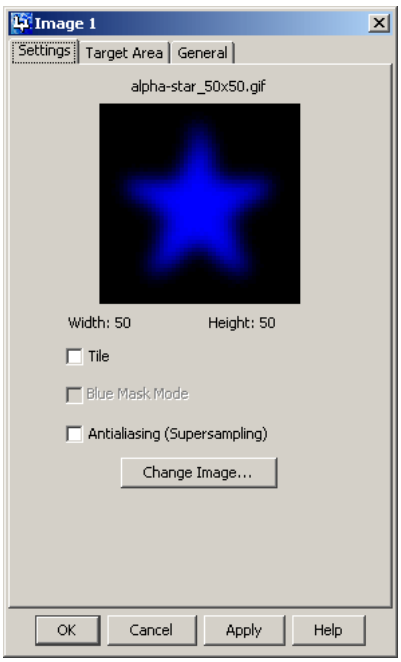

Figure 14.4: The alpha channel image shaped like a star

- From `Invert`, *Out-ports* → *Mouse Events* → *Mouse Move [2D Number]*, to `Invert`, *In-ports* → *Target Area* → *Set Position [2D Number]*.

The slightly shaded edges of the star and the inverting filter with the alpha channel, will follow the cursor movements (Figure 14.8).

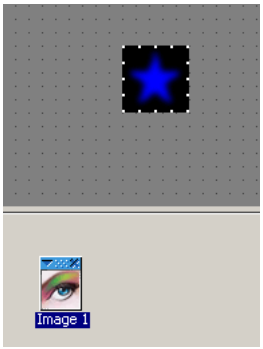

Figure 14.5: The alpha channel image on the alpha channel stage

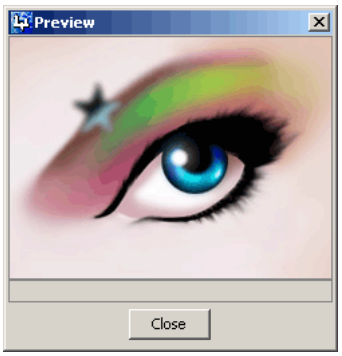

Figure 14.6: The result of the invert filter shaped after the alpha channel image

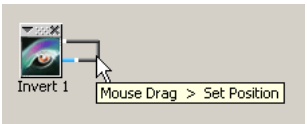

Figure 14.7: Invert object connected to itself

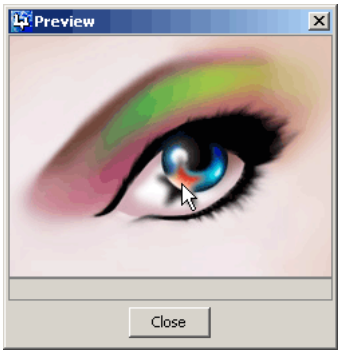

Figure 14.8: Inverting applied, following the cursor

14.2 Mouse Map

The Mouse Map is essentially an image with a specific color value that represents the shape of the Mouse Area's *Target Area*. It works more or less as the alpha channels, but for the Mouse Area object instead. Parts or components of the Mouse Map image having blue values equal to 0 will form the shape of the Mouse Area object's *Target Area*.

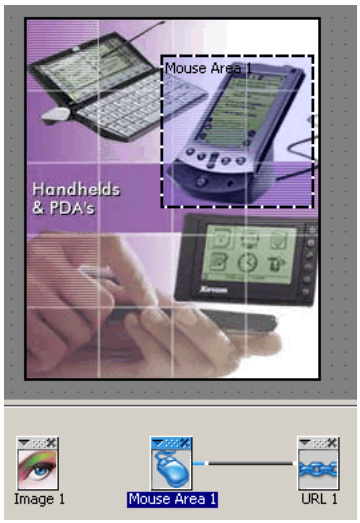

Figure 14.9: A Mouse Area connected to an URL

In the above figure (Figure 14.9) the Mouse Area *Target Area* is a rectangular area with 115×115 pixels. If you want the area to be active precisely over the handheld computer, then you will need to map it. This is done with the Mouse Map function.

Navigate to the Mouse Map by clicking its icon in the *Folders* view, or alternatively, right-click the Mouse Area object to open its local menu and select *Explore Mouse Map* to jump there directly (Figure 14.10). Then place your mouse map image here (Figure 14.11).

As with the alpha channel, the Mouse Map is a Scene object. Everything you do in this Scene, and on its stage, will work as your mouse map. This means that you can have animated and interactive mouse maps in WireFusion (Figure 14.12).

You don't have to have the same size on your Mouse Area *Target Area* as your Mouse Map *Stage*. They will fit automatically.

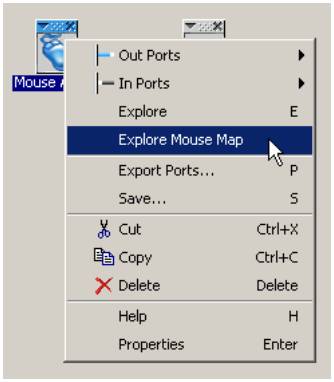

Figure 14.10: Jump to the `Mouse Area`

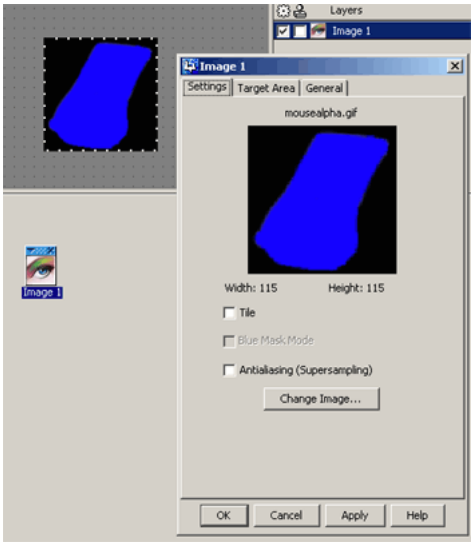

Figure 14.11: The `Mouse Map` image, which will shape the `Mouse Area` *Target Area*

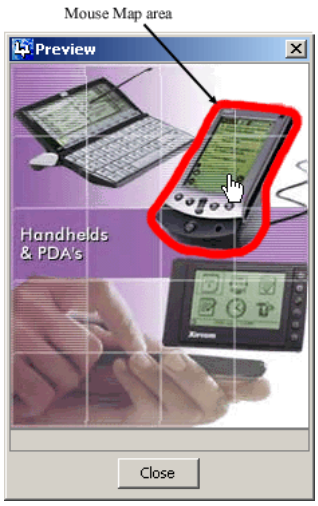

Figure 14.12: The `Mouse Area` is now only active within the `Mouse Map` area instead of the 115×115 pixel large *Target Area*

15 Optimizing your presentations

The performance of your WireFusion presentations can differ a lot, depending of how you optimize them. There are several aspects to think of while developing and we will enlighten the most important issues here.

15.1 Frames per second (fps)

Traditional animation involves a series of still images. As with movies, each image is called a **frame**. Other web authoring tools, like Macromedia Flash®, are very well suited for creating animations, i.e. they normally create a sequence of still images (frames) displayed in a player (Figure 15.1).

Figure 15.1: A frame based animation

With WireFusion, however, you generally don't create animations. You create real-time interactivity. And when you create true interactivity, you cannot have already made frames, since you don't know what the user wants to do. Therefore, the frames have to be created "on the fly". WireFusion does this creation in real-time simultaneously as one interacts with the presentation.

When viewing a presentation created with WireFusion it will not display the next frame until it is finished creating (calculating) the current frame. This means that the presentation is dependent on the user's computer speed. This is something you, as developer, should have in mind when creating presentations.

The rate at which each frame is displayed is measured in *frames per second (fps)*. The movies you see at a theater displays 24 frames per second. Higher frame rates results in smoother animations, but sometimes 10 frames per second can be sufficient. Movies have a fixed frame rate, but with WireFusion, you can set the maximum number of frames per second that should be displayed.

To set the *fps* for your presentation, choose *Project* > *Properties...* (or click the *Scene properties* button in the *Menu Bar*, while located in the root `Scene`). The *Scene properties* dialog opens (Figure 15.2). In here you can set the *fps*. This value is the maximum number of frames that will be displayed in your presentation, and

the user's computer will try to show your presentation at this frame rate. Default is 25 frames per second.

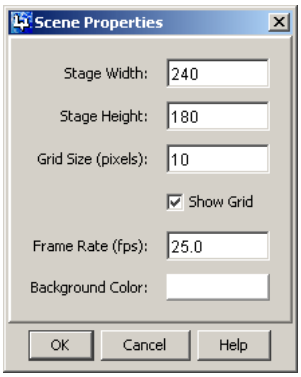

Figure 15.2: *Scene properties* dialog

Optimization tips

Suppose that you have made a pretty CPU demanding presentation, but that 10 fps is enough to have it running smooth and good looking. Using a higher fps would then only slow down the user's computer.

Always try to set the Frame Rate (*fps*) value to the lowest value possible in order to not slow down the user's computer unnecessarily.

While developing and testing your presentations, you can test the frame rate. Click anywhere in the presentation window (to give focus to the presentation) and then write the invisible code word '**showf**' (which stands for '*show fps*') on your keyboard. The present fps value of your presentation will be displayed in the top-left corner (Figure 15.3).

You can also benchmark your presentations. Click anywhere in the presentation and then write the invisible code word '**showb**' (which stands for '*show benchmark*') on your keyboard. Full CPU power will be given to your presentation and the fps will be displayed in the top-left corner (Figure 15.4).

15.2 Image processing

One of WireFusion's strengths is its ability to handle real-time image processing in your presentations. So, what does this mean? Basically, image processing means

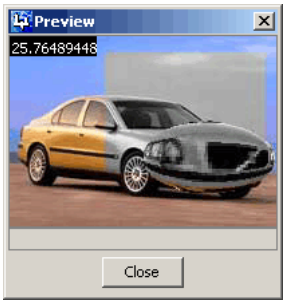

Figure 15.3: Displaying the present fps, which is set to display a maxumim of 25 fps

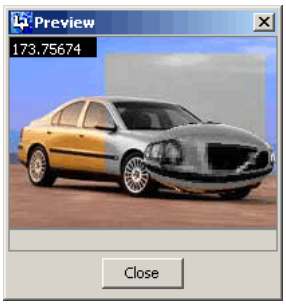

Figure 15.4: Benchmarking the presentation giving it full CPU power

that the pixels in a bitmap image is manipulated in some way so the resulting image alters, and normally this involves consuming of, more or less, CPU power. This, of course, depends of the CPU speed, the complexity of the manipulation and the required quality (Figure 15.5).

A very simple image processing filter is the grayscale filter. It just converts the pixel color values into a corresponding grayscale value (Figure 15.6).

Several off-line softwares, such as Adobe Photoshop®, has filter plug-ins, that can manipulate the image in a lot of different ways. What you do in Adobe Photoshop® is that you make a selection and then you apply the filter function in the specified selection. You normally also have the possibility of changing some filter parameters, which helps to manipulate the image differently. All of this can be done in WireFusion too. WireFusion comes delivered with twelve basic filter objects, which can be found in *Objects* > *Filter* or in the *Filter* tab in the *Object Bar*. In WireFusion you add your filter object to you project, specify its *Target Area* and then activate it. This will start the filter and it will manipulate all underlying pixels (graphics).

As your presentation probably will be online (on the web), you might want the users to interact with it. WireFusion was therefore created to handle image processing filters in real-time. So, what do we mean by real-time? With real-time we mean that the user can move a filter *Target Area* (a selection in Adobe

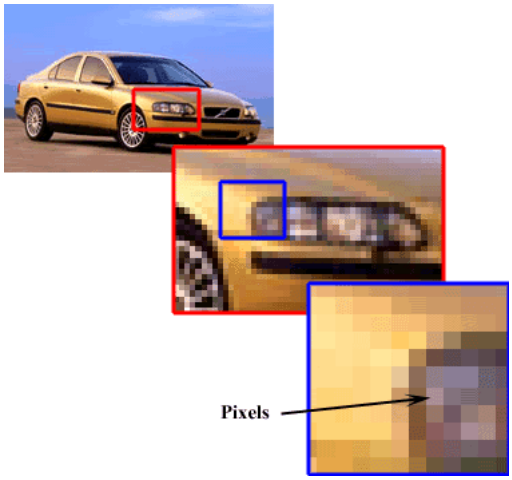

Figure 15.5: Close-up of of a bitmap image

Figure 15.6: A grayscale filter using a alpha channel

Photoshop®), resize it and even change its parameters in your presentation and it will be updated directly and automatically.

For example, let's say that you have applied a brightness filter to your presentation, symbolizing a shadow. Then you can have the size of the shadow, the position as well as the brightness level changed, depending on where the mouse pointer is.

Optimization tips

Image processing consumes CPU and some objects consume more than others.

Remember not to have too many image processing filters in your presentation at the same time.

The least CPU intensive filters are `Brightness`, `Grayscale`, `Invert`, `Mosaic` and `Noise`.

15.3 Target Area

All objects having a *Target Area* with some visual effect (`Image`, `Image Array`, `Scene`, `Scene Window`, all widget objects and all filter objects) consume CPU when they are activated and not stamped in the background. Some objects consume more than others, as explained in the Image processing section above. But another important cause of consuming CPU is the size of the *Target Area*. The larger area, the more CPU is consumed.

Optimization tips

Try not to have too large of an area(s) for your *Target Area* objects. It will slow things down considerably. Larger *Target Area* means more pixels to calculate. For example, a *Target Area* of 100×100 pixels ends in a total of 10.000 pixels. A 200×200 pixel area ends in a total of 40.000 pixels, i.e. four times more pixels to calculate than the 100×100 area.

All objects with a *Target Area* have in-ports for activating and deactivating. Try to deactivate a *Target Area* object when it's not needed, and then re-activate it again if needed again.

Image objects displaying static backgrounds, make sure to use the *Stamp background* option instead of having them activated. This will save a lot of CPU.

15.4 Scenes

All active stages consume CPU. All presentations have at least one active stage, the root `Scene` object's stage. That is your presentation display. The larger display your presentation has, the more CPU it will consume (Figure 15.7).

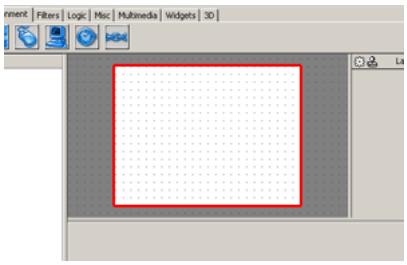

Figure 15.7: Presentation display size

To change the *Stage* size of your presentation, choose *Project* > *Properties...* (or click the *Scene properties* button in the *Menu Bar*, while located in the root `Scene`). To change the size of a local *Stage*, choose *Scene* > *Scene properties* (or press the *Scene properties* button in the *Menu Bar*) while located in a local `Scene`.

Optimization tips

Try not to make your presentation display too large.

15.5 Resources

WireFusion presentations use many externally made resources, such as images, sounds, 3D objects etc. These resources are made with third party programs and they need to be optimized in their respective program. This is important in order to receive small (in size) and bandwidth friendly presentations suited for the web.

Optimization tips

If you intend to use an image in your presentation with the dimension 300×200 pixels. Do not save it as 600×400 in your image software and resize it to 300×200 in WireFusion. This will only cost you unnecessary kilobytes. Try to save it directly in the dimension you will need it in WireFusion.

Always try to optimize the size when saving your images in your image software. Use compression for JPEG images and minimize the colors for GIF and PNG images. Being careful at this stage will save you many kilobytes.

15.6 Reuse of objects

WireFusion reuses the objects' code whenever it can. This means that you, for example, can have one or twenty `Image Array` objects in a presentation, the final code will hardly increase. However, the size of your presentation depends on how many different types of objects you use in your presentation. For example, using twenty `Image Array` objects costs less in file size than if you use one `Image Array` object and one `Sound`object.

Optimization tips

Try to use as few object types as possible, in order to obtain as small a final presentation(s) as possible.

Check your project before publishing it to ensure that you haven't kept any objects that won't be needed, and especially an object type that isn't used elsewhere in your project, as it will increase the file size of your final presentation unnecessarily.

15.7 Summary of optimization tips

Here follows a summary of the most important issues to think of when optimizing your WireFusion presentations.

- Always try to set the Frame Rate (*fps*) value to the lowest possible value
- Test the frame rate of your presentations with '**showf**' and '**showb**'
- Don't use too many filter objects in one and the same presentation
- Don't use too large *Target Area*s
- Deactivate non-needed *Target Area* objects
- Use *Stamp background* option for background images
- Don't use *Stage* dimensions that are too large (presentations display)
- Save your images to the size dimension you will need in WireFusion
- Compress and optimize colors of images in your image software
- Remove unused objects in your project before publishing
- Always test your presentations on the requested minimum required target computer systems

16 Testing your presentations

It is very important that you test and preview your presentation along with the development of new functions, in order to achieve a well working presentation. It can be quite difficult to debug your presentation for errors if you haven't tested every step of it in the creation process. There are two methods to preview your presentation while developing without leaving WireFusion; in the internal viewer and in your favourite browser.

16.1 Hierarchy

There is a hierarchy of objects in your WireFusion projects similar to the schematic view seen in (Figure 16.1).

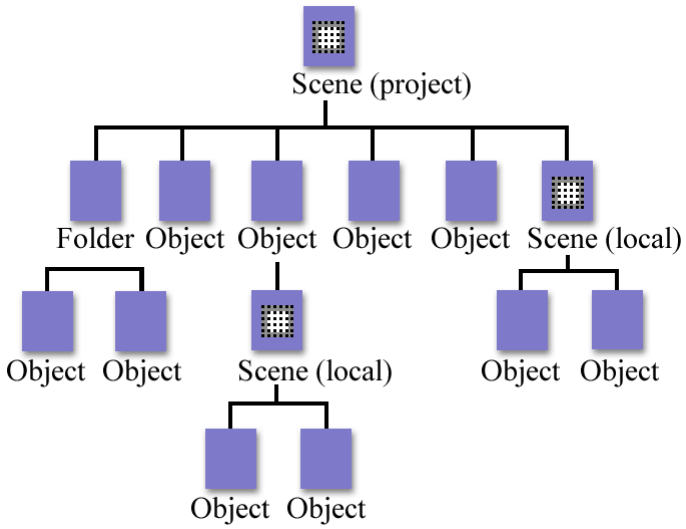

Figure 16.1: A schematic view of an object hierarchy in a project

At the top there is the main `Scene` object of the project, which will display your presentation. No matter where you are located in the object hierarchy, you can always preview your project, i.e. the main `Scene`, by pressing F9 (internal viewer) or Ctrl+F9 (browser) (Figure 16.2).

Figure 16.2: The Preview Presentation buttons

All other objects in your project are grouped in the main `Scene` object. In some projects you might have local `Scene` objects, i.e. you will display another `Scene` object's stage on your main stage (Figure 16.4). In order to preview a local `Scene` you first have to be located in it (Figure 16.3), then press F8 (internal viewer) or CTRL+F8 (browser). You can navigate your way down in the hierarchy using the *Folders* view or by exploring an object. (Read more about navigating projects in Chapter 8.)

Figure 16.3: The *Preview Scene* buttons

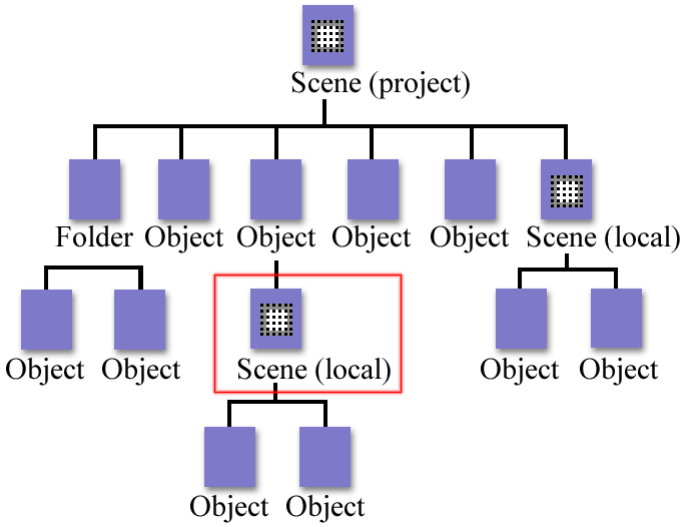

Figure 16.4: Located in a local `Scene` object down in the hierarchy

16.2 Viewer

The internal viewer allows you to test your presentations directly inside WireFusion. It starts quickly and is normally used when you make small changes in your presentations, like changing a position or a value. For larger and more detailed tests, it is

recommended that you preview it in the browser (Figure 16.5).

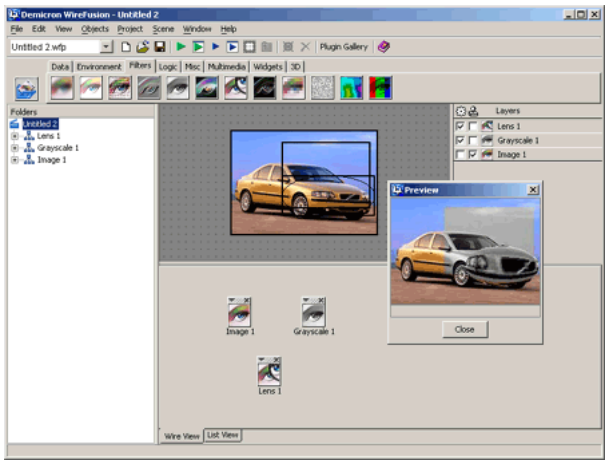

Figure 16.5: Preview in the internal viewer

To preview the project in the viewer, choose *Project* > *Preview Presentation* or press F9. Alternatively, click the *Preview Presentation* button in the *Menu Bar* (Figure 16.6).

Figure 16.6: *Preview Presentation* button in the *Menu Bar*

To preview a local `Scene`, choose *Scene* > *Preview Scene* or press F8. Alternatively, click the *Preview Scene* button in the *Menu Bar* (Figure 16.7).

Figure 16.7: *Preview Scene* button in the *Menu Bar*

If you leave the viewer up and running (without closing it) while developing, and if your presentation is CPU intensive, then it will affect the WireFusion program performance. Close the viewer to ensure better performance.

16.3 Browser

To preview your presentation in the browser will give you a more accurate testing and closer to "real life" and final experience. You can also test certain features in

the browser, which you can't test in the internal viewer. For example, you cannot test the JavaScript functionality in the internal viewer, only in the browser.

You choose your favourite browser in *File* > *Preferences* (Figure 16.8).

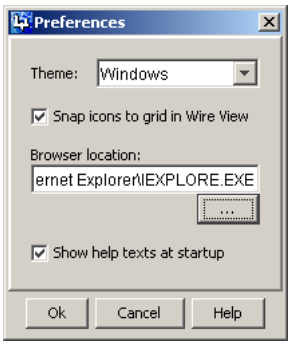

Figure 16.8: Selecting your favourite browser in *File* > *Preferences*

To preview the project in the browser, choose *Project* > *Preview Presentation* in browser or press Ctrl+F9. Alternatively, click the *Preview Presentation* in browser button in the *Menu Bar* (Figure 16.9 and Figure 16.10).

Figure 16.9: *Preview Presentation* in browser button in the *Menu Bar*

To preview a local `Scene` in the browser, choose *Scene* > *Preview Scene* in browser or press Ctrl+F8. Alternatively, click the *Preview Scene* in browser button in the *Menu Bar* (Figure 16.11).

After deployment, it is important to test your presentation in all target browsers and on all target platforms before publishing it to your web site.

If you leave the browser up and running (without closing it) while developing, and if you presentation is CPU intensive, then it will affect the WireFusion program performance. Close the browser to ensure better performance.

16.4 Debugging

Sometimes when developing you will run into situations when your presentation, or function, doesn't perform what you would like it to perform. Then you need methods to debug it, i.e. to know and understand what's happening in your code.

The *Debugger* in WireFusion lets you debug a presentation in the internal viewer only (i.e. you can not debug the presentation in the browser).

The *Debugger* can be used to:

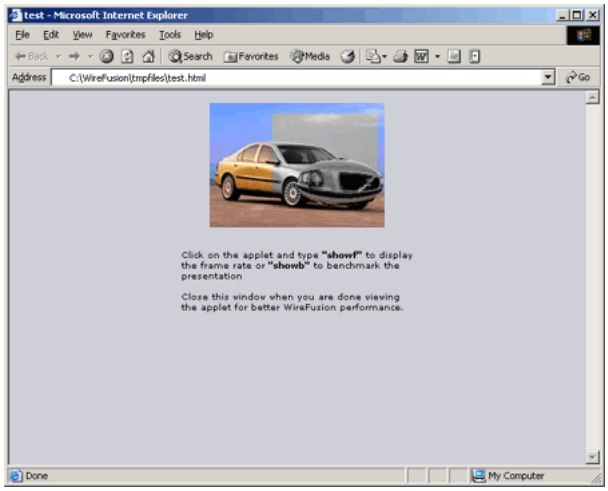

Figure 16.10: Preview in browser

Figure 16.11: *Preview Scene* in browser button in the *Menu Bar*

- Step through the presentation
- Inspect parameter values
- Set breakpoints

After you have successfully built your presentation, you can start using the *Debugger*.

16.5 Preparing to use the Debugger

Before running your presentation under the *Debugger*, you may want to add a breakpoint to your code, or a certain connection wire. The breakpoint will suspend the presentation and display the connection information in the *Debugger* window.

To add a breakpoint, do this:

Open the *Wire View* to view the connection where you want the *Debugger* to stop. Right-click on the connection and select *Toggle Breakpoint* (Figure 16.12) from the wire menu.

If your running presentation hits a breakpoint, the *Debugger* will stop automatically.

To remove a connection breakpoint, simply deselect the *Toggle Breakpoint* again in the wire menu.

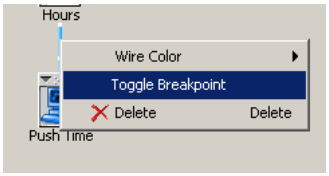

Figure 16.12: Adding a breakpoint in the wire menu

16.6 Debugger Window

To run the presentation through the *Debugger*, select *Project* > *Debug Presentation* from the menu bar (Figure 16.13) or press ALT+F9. The *Debugger* window and the preview window will open.

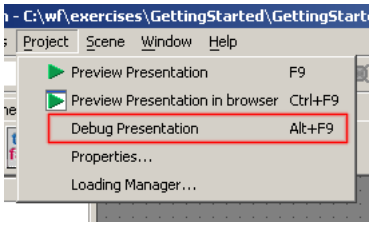

Figure 16.13: Starting the *Debugger*

For each breakpoint or stepped connection wire, the *Debugger* window displays:

- Parameter value (or pulse) sent between objects
- Object names

Figure 16.14 shows how a line in the *Debugger* looks like when a breakpoint is hit.

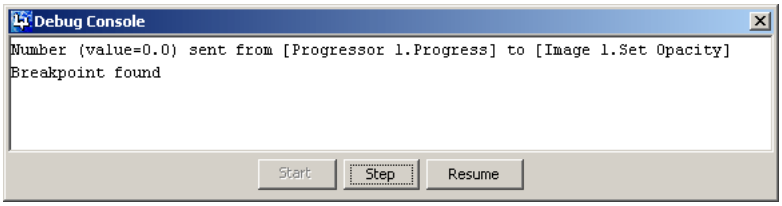

Figure 16.14: The *Debugger* window

The information has the following meaning:

1. Number (value= 0.0) is the parameter sent through the connection wire.

2. The above value is sent out from the *Out-port* `Progressor` found in the '`Progressor 1`' object, [Progress 1. Progress].

3. And then sent in to the *In-port Set Opacity [Number]* found in the '`Image 1`' object, [Image 1, Set Opacity].

It is also possible to debug a local scene (instead of the complete project) by selecting *Scene* > *Debug Scene* from the menu bar.

16.7 Stepping through connections

In the *Debugger* window you can walk through your connections using the following navigation buttons.

Start (button) The *Debugger* is stopped at startup by default. If a breakpoint is used, click the Start button to start the *Debugger*.

Step (button) Click the Step button to execute one connection at the time. You can click the Step button at any time, it will pause the *Debugger* if running and display the next connection to be executed. The connection at the breakpoint is not executed until stepped or resumed.

Resume (button) Click the Resume button to continue debugging from a stopped position or a breakpoint. The *Debugger*will stop automatically if it hits a breakpoint.

When a breakpoint is hit or when stepping through the code, the connection wires in the *Wire View* get highlighted with blue color. When stepping through the code, the *Debugger* automatically navigates through your project, in order to display the highlighted connection wires.

16.7.1 Console window

An alternative method for probing an object or a port, i.e. getting a pulse or a value, is to use the *Console* window. To start the *Console* window, choose *View* > *Console* or press CTRL+SHIFT+C or mark the *Show Console* checkbox found in the internal viewer. The *Console* window will only display messages when you preview your presentation in the internal viewer, and not in the browser. Most browsers have their own Java Console, which can be activated in the browser preferences, or wherever the browser Java is activated.

In order to probe you need to have a `System` object in your project (found in the *Environment* category). The `System` has an *In-port* submenu called *Console* which has five *In-port* options, which all display in the *Console* window when receiving incoming messages.

- *Print Pulse*: Prints 'Pulse' when triggered.

- *Print Number*: Prints the incoming number value.
- *Print Boolean*: Prints the incoming Boolean value.
- *Print Text*: Prints the incoming text strings.
- *Print 2D Number*: Prints the incoming two-dimensional number value (x ,y).

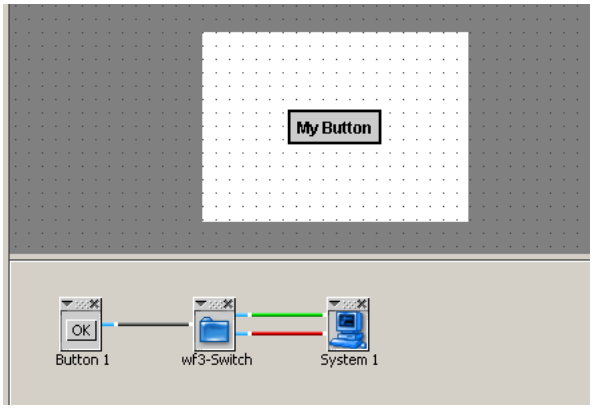

Figure 16.15: Probing the boolean value of a switch function

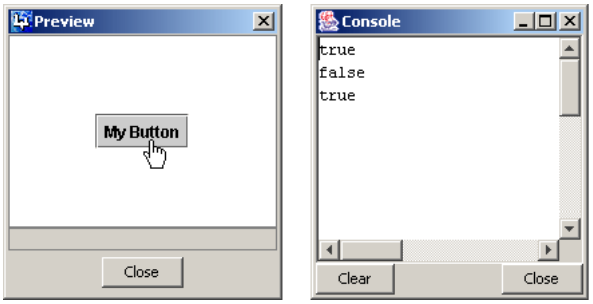

Figure 16.16: The result of the probing displayed in the *Console* window

17 Deployment

The final stage of your WireFusion production is publishing to the Internet. Before you publish your presentation, it is often a good idea to use the Loading Manager, which helps you to control the streaming order of your presentation while loading it.

17.1 Loading Manager

By using the *Loading Manager...*, found in the *Project* menu, media resources such as images and sounds can be streamed into your WireFusion presentation after it has been started. This is very useful in order to decrease the loading time and hence increase the user experience (Figure 17.1).

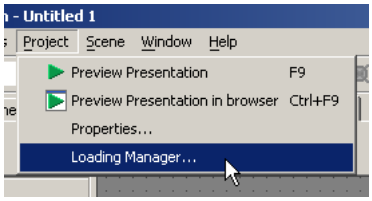

Figure 17.1: Starting Loading Manager

By default, all resources are located in the right field of the Loading Manager, called Preloaded files. All resources placed here will be loaded before the presentation is started. To have a resource streamed into the presentation after the startup, mark the resource and click the transfer button (left arrow) to place it in the Streamed files field. Resources placed in Streamed files will be streamed, one by one, in the order they are listed. To change the order, use the up and down arrows. If a resource in the streaming list has not been loaded yet, but is requested by the presentation, then it will immediately pass the line and load simultaneously with the present streamed resource (Figure 17.2).

Images or sounds that are activated or started directly at the presentation startup will be preloaded, even if they are placed in the Streaming files list.

Figure 17.2: Loading Manager

17.2 Publish Dialog

When you publish a WireFusion presentation to HTML, you publish it as a Java applet. In order for the users to see your presentation on the web, it has to be placed in a web page and the users must have a Java enabled browser.

To publish a presentation, choose *File* > Publish... or press Ctrl+P. The publishing dialog opens (Figure 17.3).

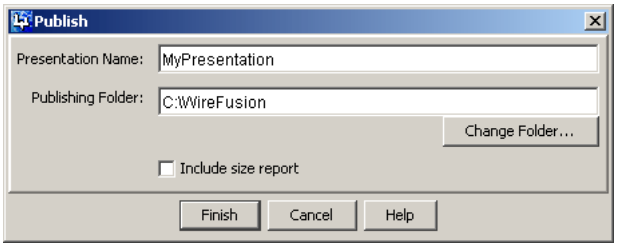

Figure 17.3: The Publish dialog

Choose a name for you presentation (by default your project name is entered), select a folder where to publish it and then press Finish.

Your presentation is now published and saved to *C:\WireFusion* with the name *MyPresentation.html* (Figure 17.4). The directory called '*resources*' contains your published presentation and must be located in the same directory as the HTML file.

The '*resources*' directory contains two sub-directories: '*shared*' which contains certain files that will always be the same (no matter which presentation you create) and then '*MyPresentation*' which contains specific files needed for your presentation

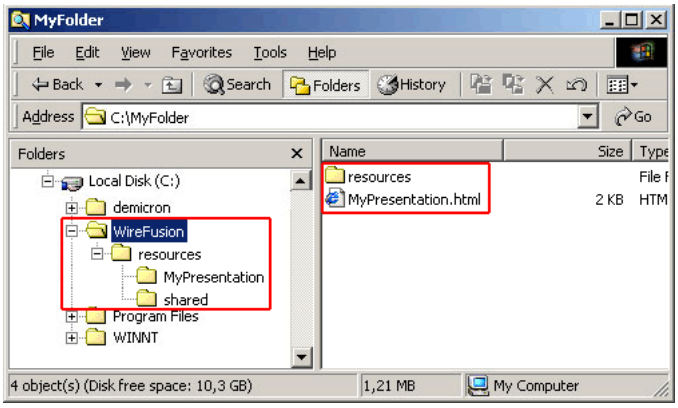

Figure 17.4: *MyPresentation* published to *C:\WireFusion *

(images, sounds, special objects etc).

If you intend to publish several presentations to one and the same web site, then you can beneficially publish them into the same directory. All presentations on this site will then use the same '*shared*' directory, which will be cached by some browsers and systems, and hence speed up the loading (and re-loading) time (Figure 17.5).

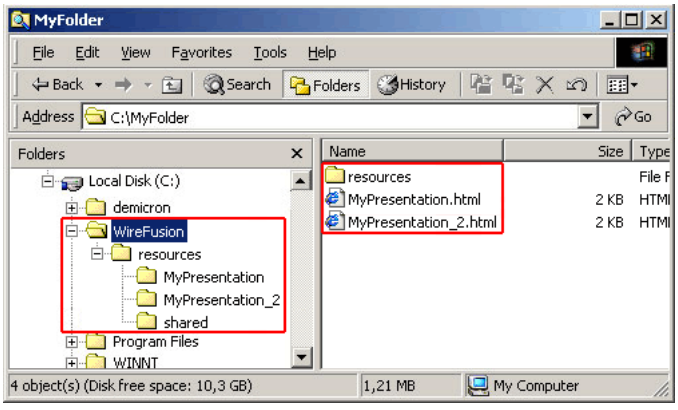

Figure 17.5: *MyPresentation_2* also published to *C:\WireFusion *

WireFusion publishes your presentation to an HTML page automatically, which can be edited later with a third party HTML authoring program, like for example Macromedia Dreamweaver® or Adobe GoLive®.

Simply copy the HTML code between the line:

`<!-- BEGIN WIREFUSION PRESENTATION CODE -->`

and

```
<!-- END WIREFUSION PRESENTATION CODE -->
```

17.3 Size report

Mark the Include size report checkbox (Figure 17.3). to have a size report displayed in the published HTML page (Figure 17.6).

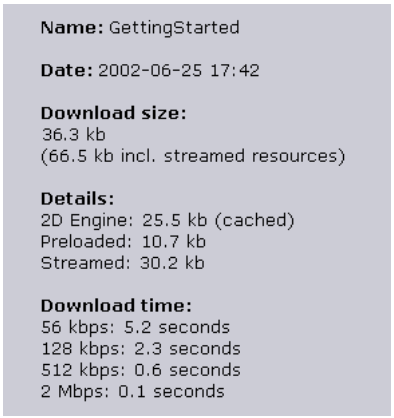

Figure 17.6: Size report

The total size is reported as well as the size of the individual parts (2D Engine, preloaded files and streamed files). A theoretical download time is also calculated.

18 Summary of Working with WireFusion

In these chapters you have learned about:

- The principles of working with WireFusion
- The main locations of menues, views and program controls
- How to build and control projects
- The WireFusion ready-made objects
- How to connect the objects
- The meaning of in- and out-ports
- The idea of target areas
- Forwarding and storing data
- How to group objects
- Some elementary information on image processing
- The basics on time-sequences in frames
- How to use layers
- The possibilities of alpha channels
- The mouse map
- Testing the projects
- Deploying and publishing projects