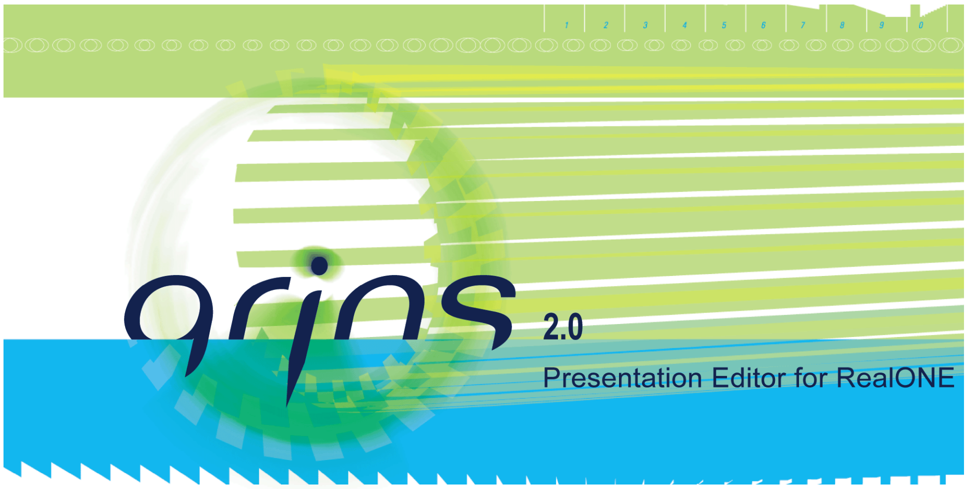


oratrix

# *Quick Start Guide*

**GR*i*NS Editor for RealONE**
Windows-98/2000/XP
November, 2001

## Important Notices

This document is the GR*i*NS Editor for RealONE *Quick Start Guide*. All of the information presented in this publication has been verified, but incremental product updates may impact part of this guide.

This version of the GR*i*NS Editor for RealONE *Quick Start Guide* has been produced for use as an off-line reference. Images and page layout have been optimized for printing on a 600-dpi (or greater) laser printer. For best reproducibility, the use of a color printer is recommended, although every effort has been made to make illustrations readable on other printers as well. If you wish to use it as an on-line reference via a PDF reader, we recommend that you increase the level of display magnification when viewing images.

The images used in this publication were taken from the GR*i*NS Editor for RealONE build 2.0-win32-93 for Windows 2000. While the look of other versions of GR*i*NS are slightly different because of adherence to common conventions on those other environments, the functionality described is similar for all versions of GR*i*NS. In order to reduce document size, only images from the Windows version have been included in this document.

We welcome your questions on GR*i*NS Editor for RealONE and comments on this documentation. Please submit all questions and comments to our support desk at *grins-support@oratrix.com*. We maintain a list server dedicated to sharing experiences among GR*i*NS Editor for RealONE users. See the on-line release notes that come with the software distribution for details of this listserver. Finally, if you wish to submit your own SMIL files as examples for other users, please send a request for submission to: *grins-examples@oratrix.com*.

# Table of Contents

# GR*i*NS Editor for RealONE Distribution Roadmap

Congratulations on selecting the GR*i*NS Editor for RealONE (GR*i*NS/RealONE) for creating SMIL 2.0 presentations. GR*i*NS/RealONE lets you harness the full power of the RealNetworks RealONE player in an easy and intuitive manner.

## Distribution Package Contents

The GR*i*NS/RealONE distribution package consists of the following components:

- GR*i*NS Editor for RealONE *Quick Start Guide*: an overview of the installation instructions for GR*i*NS and a tour of the basics of the GR*i*NS Environment. (This is this guide.)
- *Templates*: a set of templates used in GR*i*NS/RealONE, which you may use to build your own presentations;
- GR*i*NS-*Icons*: a directory containing Icons used by GR*i*NS/RealONE;
- *Software:* depending on the distribution you downloaded, a GR*i*NS distribution for Windows-95/98/2000/XP or other platform.

## Documentation

In order to reduce the size of the GR*i*NS distribution set, most documentation has been included in a separate distribution file. These manuals, available via the documents link at *http://www.oratrix.com/*, consist of:

- GR*i*NS Editor for RealONE *Reference Manual*: a comprehensive description of the GR*i*NS/RealONE features and options;
- GR*i*NS Editor for RealONE *Template Guide*: a quick and easy description of the standard GR*i*NS/RealONE template set that gets you started making RealONE presentations;
- GR*i*NS Editor for RealONE *Tutorial/User's Guide*: a set of step-by-step instructions for creating a wide range of RealONE presentations.

The documentation set also contains an assets set that you can use to build tutorial presentations.

GR*i*NS documentation is free; assets for tutorials are available to registered users.

## GR*i*NS and GR*i*NS/RealONE

GR*i*NS is available in several configurations: a *Pro* edition, intended for users who want full control over authoring and publishing to many popular SMIL-2.0 engines, and editions for particular streaming players (such as GR*i*NS/RealONE). Unless otherwise marked, all other features discussed in this Guide are available in both GR*i*NS/RealONE and GR*i*NS/Pro. (See section 6 for a full feature comparison.)

If you purchased the basic edition of GR*i*NS/RealONE and would like to upgrade to the Pro edition, please visit the GR*i*NS/RealONE upgrade page at:

*http://www.oratrix.com/*

Note that your original license number will be required for a product upgrade to GR*i*NS/Pro.

## GR*i*NS/RealONE Quick Start Guide

The GR*i*NS/RealONE *Quick Start Guide* will help you get started making SMIL presentations for the RealONE Player. It is divided into eight sections:

1. *Introduction to* GR*i*NS *and* GR*i*NS/RealONE: a description of the GR*i*NS architecture and a discussion on how you can protect your authoring investment by using the GR*i*NS approach to presentation design and maintenance.
2. *Installing and Running* GR*i*NS: an overview of how to load GR*i*NS/RealONE onto your system and what to expect when you load an existing RealPlayer presentation in the GR*i*NS/RealONE Editor.
3. *Bird's-Eye View of* GR*i*NS: a quick tour of the basics of the GR*i*NS/RealONE environment. Spending five minutes here will get you ready to create and edit complex presentations without ever having to type a single line of SMIL code.
4. *Using* GR*i*NS/RealONE: a simple tutorial that gets your fingers and mind working in GR*i*NS/RealONE compatible mode.
5. *Understanding SMIL-2.0*: an overview of the basic SMIL-2.0 concepts used by all versions of GR*i*NS.
6. GR*i*NS *Edition Comparison Information*: an overview of the differences in functionality offered in the various editions of GR*i*NS.

7. GR*i*NS *Editor Quick Reference*: an overview of the data types supported by the GR*i*NS/RealONE environment and other useful reference information for getting started with the GR*i*NS/RealONE Editor.
8. *Where Next?*: a set of pointers to GR*i*NS/RealONE resources and other hints to help you on your way to creating presentations for the RealONE Player.

Each of the sections has been written to be relatively stand-alone, but we suggest that you read them all to get a good overview of the system.

# Introduction to GR*i*NS and GR*i*NS/RealONE

The GR*i*NS/RealONE is an authoring environment for creating, editing and maintaining streaming media documents for the RealONE player. GR*i*NS/RealONE allows you to take source materials such as audio, video, image and text *assets*, and integrate them into a presentation that can be distributed via the Web. GR*i*NS/RealONE is one product in a range of tools that help you create compelling presentations for Web-based multimedia — for more details on the GR*i*NS product family, see *http://www.oratrix.com/*.

Fundamental to GR*i*NS/RealONE is that the complex task of presentation authoring is partitioned so that you have the freedom required to build and manage presentations easily. At the “assets-end,” you can work with the media assets, management tools and editors that best meet the requirements of the task at hand. At the “presentation end,” GR*i*NS allows presentations to be published to Real’s optimized datatypes and uploaded to a streaming server of your choice.

The GR*i*NS/RealONE environment has native support for the W3C SMIL-2.0 language. This means that you can open and edit existing presentations, whether or not they were made with GR*i*NS. This gives you an advantage by reducing your development time: you can update earlier versions of presentations made by others, or you can create new presentations directly with GR*i*NS/RealONE.

## Version 2.1 of GR*i*NS/RealONE

GR*i*NS/RealONE has been designed to help you produce compelling productions that are targeted directly to the RealONE player. This gives you access to the millions of users who have Real’s player installed on their systems.

## Key Features of Version 2.1 of GR*i*NS/RealONE

GR*i*NS/RealONE has been based on the feedback we have received from the thousands of GR*i*NS users world-wide. The key features of this release are:

- integrated *Preview Player* that lets you evaluate your presentation as you create it, before you publish to RealONE;
- integrated *bandwidth simulation* for multiple levels of network performance to show if your presentation will 'fit thru the pipe';
- presentation authoring using the GR*i*NS/RealONE *structured timeline*, *layout* and *hyperlink* editing views;
- simplified link use with RealONE's *context* and *browser* windows;
- *automatic conversion* with detail control parameters for nearly all media types into RealONE formats, including:
  *images*:JPG, GIF, PNG, BMP formats to streaming images,
  *video*:AVI or MPEG movies to RealVideo,
  *audio*:AIFF, MP3 or WAV to RealAudio,
- *drag and drop* editing, using both source assets and pre-converted RealMedia files (allowing you to fine-tune asset conversion with RealProducer — or any other conversion tool — before placing it in your production);
- support for SMIL's SWITCH, allowing you to build *adaptive presentations* that can optimize your production to the available bandwidth;
- the ability to *open previously created SMIL files*, even those created with Real's Slideshow, TAG Composer, or a text editor;
- support for *interaction* via the Exclusive groups;
- support for inserting *transitions* into a presentation on all types of media;
- support for embedded *animations* on objects;
- support for *layout hierarchies*
- support for *template-based editing* (both presentation and assets templates);
- easy upload to a RealSystem RealServer for hosting your presentation.

The basis of GR*i*NS/RealONE is the unique GR*i*NS *Structured Timeline View*: you create logical presentations that can be easily edited and updated without having to worry about detailed timing interactions among or within objects.

## The GR*i*NS/RealONE Work Flow

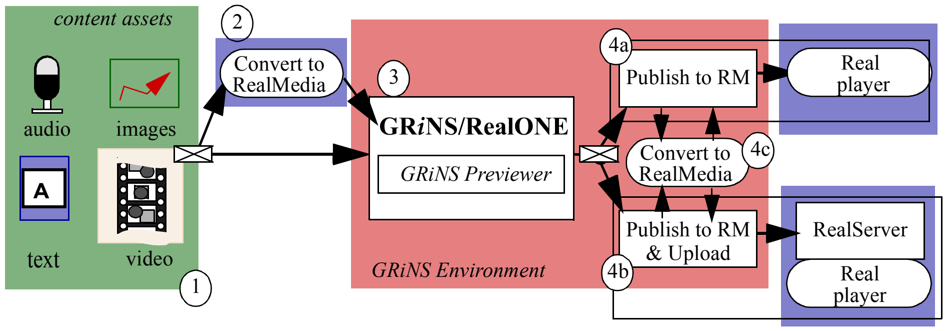
Creating compelling streaming media presentations is a complex task. The challenges that must be addressed are:

- creating or selecting appropriate media objects for the presentation
- structuring the presentation to convey the message you want
- defining a presentation that 'fits' through the network connection available to the viewer of your presentation

Designing presentations for the Web presents you with both opportunities and problems. The basic opportunity is that you can reach a world-wide audience with your message. The basic problem is that the world is a non-standard place. People speak different languages, people use different computers, people have access to the Web via connections of varying speeds.

GR*i*NS/RealONE helps you manage the complexity of the Web by giving you control over the essential parts of your presentation. Want to quickly get out Version 1? No problem: the GR*i*NS/RealONE template-based approach lets you get started in seconds. Want to make new versions that are tailored to specific groups of users, or to specific architectures or to users with high- or low-bitrate connections? Again, no problem: GR*i*NS/RealONE gives you the freedom to edit, update and maintain your presentations, and to target as many specific groups as you like.

The following illustration gives you an overview of the steps in the GR*i*NS/RealONE workflow.



1. Creating content is perhaps the most challenging part of making a compelling presentation. You should keep in mind that the Web is not TV: the Web is a delivery vehicle that can vary widely in performance, while TV is a predictable delivery vehicle that 'looks good' in spite of supporting relatively low bandwidth audio and video information. In a similar vein, be careful with image data and with text: create or select each datatype carefully, keeping your intended audience, their equipment and their patience in mind. GR*i*NS/RealONE allows you to use nearly all popular audio, video and image formats available, including AIFF, AVI, BMP, GIF, JPEG, MPEG-1&2, MP3 and PNG.
2. An optional second step in the process is the encoding of your media assets into one of RealNetworks' RealMedia formats, such as RealAudio, RealText, RealVideo or RealPix. You can also choose to have GR*i*NS/RealONE do the conversion for you, but you can sometimes exercise more control over the coding of your data by using a separate tool, such as Real's RealProducerPro. GR*i*NS/RealONE allows you to insert both converted and unconverted objects into your presentation.
3. Once your media objects have been selected, GR*i*NS/RealONE's unique structure-based approach allows you to combine, edit and maintain your presentations in a flexible and easy manner. We look at this process in detail in section 3.
   GR*i*NS/RealONE allows you to open existing presentations for editing,

whether or not initially created with GR*i*NS/RealONE. GR*i*NS/RealONE also allows you to preview your presentation incrementally, while creating it, using the GR*i*NS/RealONE Preview Engine.

4. Once your presentation is made, you can publish it for use with the RealONE player. You can do this in two ways:
   (a) publish to RealONE for presentation on your local computer. Using this option allows you to playback the entire presentation using Real's player;
   (b) publish and upload your presentation to a remote Real server. You can upload the main presentation and the associated data objects to one server and send the HTML page announcing your presentation to another location.

   In both cases, GR*i*NS/RealONE will convert any objects to RealMedia equivalents for you (labelled 4c in our diagram), unless you already did this conversion directly in Step 2.

Using GR*i*NS/RealONE, you can create complex presentations from scratch, or from one of the GR*i*NS/RealONE templates. You can also open any existing SMIL presentation, whether created by GR*i*NS/RealONE, another SMIL-compatible authoring tool or a simple text editor.

## Building Presentations with GR*i*NS/RealONE

This Guide will provide a quick tutorial that will help you get started with either the Pro or Real edition of GR*i*NS.

After you get your edition of GR*i*NS/RealONE installed, registered owners should download the expanded tutorials from the GR*i*NS/RealONE Web site:

*http://www.oratrix.com/*.

If you would like us to showcase your work, please send a message to

*grins-examples@oratrix.com*.

# Installing GR*i*NS Editor for RealONE

GR*i*NS/RealONE is currently available on Windows-based PCs. An Apple OS-X and a Linux port will be released as soon as a RealONE port is made to each of these platforms. Each implementation of GR*i*NS/RealONE has been developed to meet the expectations of the user community for that platform, but all versions can edit and play presentations made by any GR*i*NS/RealONE authoring environment.

## Installing GR*i*NS/RealONE

The instructions for installing GR*i*NS depend on the platform that you use. This section reviews the general instructions. See the release notes shipped with the GR*i*NS software for the latest details on installing GR*i*NS.

NOTE:
1. The first time you execute GR*i*NS/RealONE, you must provide your GR*i*NS License Key. Instructions for using this key are provided when the key is issued.

### *Installing GRiNS/RealONE on Windows 98/2000/XP systems*

To install GR*i*NS/RealONE for Windows, download and execute the file grins_real_win_2.0.exe. This program will install GR*i*NS/RealONE and related files in the location you give during the installation process. (Note that the name of the executable may vary slightly depending on how you purchased GR*i*NS.)

When complete, the GR*i*NS/RealONE icon will be added to your desktop. The default location for starting GR*i*NS/RealONE is:

```
c:\Program Files\Oratrix\GRiNS_RV2\GRiNS_RV2.exe.
```

NOTES:
1. While Administrative privileges are not required to install GR*i*NS/RealONE, some systems may restrict your access to shared disks or certain directories. Contact your local administrator if you have problems.

2. You can remove all of the GR*i*NS/RealONE Components via the 'Add/Remove Programs' program in the control panel or via the Uninstall tools provided with the distribution. Administrative rights are typically required to uninstall GR*i*NS/RealONE.
3. The GR*i*NS/RealONE components have been architected to be as unintrusive to your system as possible, but care should always be taken when removing or replacing system files.
4. In the Windows release, the file extensions *.smi* and *.smil* are claimed by Real. GR*i*NS/RealONE places its local information in a file with the *.grins* extension.

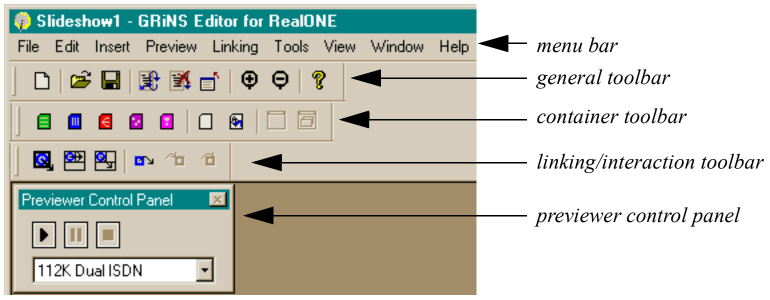
## *Checking for updated GRiNS/RealONE software*

Once GR*i*NS/RealONE has been installed on your system, you may select the Check for GRiNS update option in the Help menu to see if updated versions are available.

# A Bird's-Eye View of GR*i*NS/RealONE

## The Menu/Toolbar Structure

The following menu and toolbars are available with the GR*i*NS/RealONE Editor:


Slideshow1 - GRiNS Editor for RealONE
File Edit Insert Preview Linking Tools View Window Help
Previewer Control Panel
112K Dual ISDN
menu bar
general toolbar
container toolbar
linking/interaction toolbar
previewer control panel

The toolbars can be *docked* or *undocked*; you can also hide them using the View->Toolbars menu.

### *Pull-Down Menus*

The pull-down menus contain functions that allow you to easily create and preview presentations for the RealONE player:

- The File pull-down menu allows standard file open/save/close operations. It also lets you publish your documents to RealONE and (optionally) upload them to a RealServer. You can also set document properties via this menu.
- The Edit pull-down menu contains standard cut/copy/paste/delete operations. It also lets you access the property tabs for a selected node. If you have a content editing program installed for the datatype of the node (such as *Adobe Photoshop* for images), you can edit the content directly with that program via the Edit Content button. (Content editors are not delivered or installed with GR*i*NS/RealONE.)
- The Insert menu lets you insert new nodes into your presentation (both structure and data nodes), and lets you wrap new structure nodes around existing ones.

- The Preview pull-down menu allows you to Start, Stop and Pause the current presentation. You can preview the entire presentation, a sub-part or a single object (either a media item or a content group).
- The Linking menu allows you to create anchors and links within your presentation, so that you can jump to them from elsewhere in the presentation or define conditional activation based on user interaction. GR*i*NS/RealONE provides simplified anchoring and linking support to meet common needs.
- The Tools menu lets you perform utility tasks. These include checking bandwidth usage, converting existing RealPix objects to their SMIL 2.0 equivalents, and performing specialty tasks.
- The View menu lets you manage the complexity of the display in the various GR*i*NS views. It also lets you visualize the bandwidth constraints of your presentation and it lets you see which objects will get played at the current simulation settings.
- The Window pull-down menu allows you to open or close the active editor views (the Preview, Structured Timeline, Layout and Source views). In GR*i*NS/Pro, extra facilities are offered for manipulating the presentation.
- The Help menu gives you access to the on-line help files.

In all cases, inactive selections are 'grayed-out' in the menus. Many of the sensible alternatives (based on what you are doing in the Editor) are also available in the context-sensitive commands provided under the right mouse button.

NOTES:

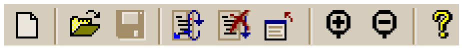
1. In GR*i*NS/RealONE, you may freely import presentations made with a text editor or with any other SMIL editor.
2. The last step in creating a presentation is typically a *Publish to Real* operation (found under the File menu); intermediate information on your presentation is saved in a file with a *.grins* type. This is a generic SMIL-compliant version of your presentation without conversions to *.rm* data types.
3. For some types of changes (like global substitutes) it is often desirable to use a text editor rather than a structure editor. This is possible with GR*i*NS/RealONE: simply open the *.grins* file with your favorite text editor and make whatever changes are necessary. You can then re-open the file from within GR*i*NS/RealONE. (Be careful when doing this, since errors put in the file by

direct text editing may make the presentation unusable from within GR*i*NS/RealONE.) You may also inspect the source at any time with the built-in source viewer within the Editor. Note that the full-featured GR*i*NS/Pro version extends this functionality with a complete source editor, adding syntax and semantic checking to the editor.

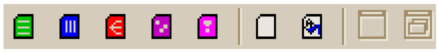
## *GRiNS/RealONE Toolbars*

GR*i*NS/RealONE provides four toolbars, each of which provides support for common editing tasks.

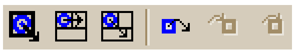
- The General toolbar provides support for opening, saving, reopening and closing *.grins* documents. It also provides controls for zooming in/out of other views.



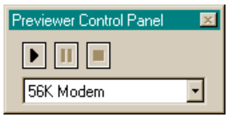
- The Containers toolbar provides support for creating various *groups* in the presentation (par, seq, switch, excl and prior), or for creating nodes or layout regions.



- The Linking and Interaction toolbar allows short-hand creation of links to the RealONE *context* and *browser* windows and begin/end *interaction events* on groups and objects.



- The previewer control allows the preview function of GR*i*NS/RealONE to be activated. Any SMIL switch settings in the document can be dynamically set by the previewer, so that the document can be previewed under a range of real-world conditions.



56K Modem

Each toolbar icon has a tool-tip documentation string. Note that the colors of the icons in the Containers toolbar match those of the respective groups in the Structured Timeline view.

The initial state of the toolbars is that all but the previewer control is docked.

## Creating Presentations Using GR*i*NS/RealONE

There are several key concepts that are common to all editions of GR*i*NS/RealONE that will help you be more productive:

- One of the GR*i*NS windows is the Previewer. This is the run-time preview of a document. While the Previewer allows a complete presentation to be viewed using conventional start/stop/pause controls, it also allows you to play sub-parts of a document — anything from a complex sub-presentation to a single media object can be activated at any time. This is a tremendous productivity enhancement tool during authoring.
- The main GR*i*NS/RealONE view is the Structured Timeline. In the Structured Timeline, you can manipulate the general structure of your presentation in a visual way. GR*i*NS/RealONE isolates you from the details of the SMIL language, leaving you with an intuitive means of putting together presentations based on a standard set of presentation building blocks. The Structured Timeline is the default view you get when a presentation is loaded into the Editor.
- Screen space and audio channels are managed using a concept known as the GR*i*NS/RealONE *Regions*.
- GR*i*NS provides powerful hypermedia and event-based interaction support: you can define anchors across media objects and then define various types of links between these anchors. GR*i*NS/Pro provides extensive features to define anchors and links at various (sub-)parts of a presentation, while GR*i*NS/RealONE gives you the ability to build simple links and anchors for navigation in and through your presentation.

GR*i*NS/RealONE also provides full support for building adaptive presentations using the SMIL switch statements. The semantics and properties of the SMIL switch are managed in the GR*i*NS structure view. Here again, GR*i*NS worries about the syntax — you simply need to define the options available.

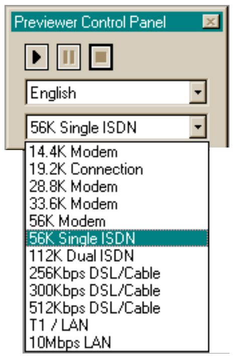
## Important Note

The SMIL 2.0 language provides many powerful primitives for creating sophisticated presentations. The more you know about SMIL 2.0, the richer your presentations will become. In order to become familiar with the basic concepts of SMIL, we suggest that you read the chapter *Understanding SMIL 2.0*, at the end of this guide. You should also read the *Real Production Guide* to understand the basic concepts of both SMIL and streaming presentations with RealONE.

## An Overview of the GR*i*NS Authoring Views

This section will look at each of the views provided by GR*i*NS/RealONE and gives you an overview of how GR*i*NS helps you to manage the workflow during document creation and maintenance. The GR*i*NS/RealONE Editor was designed to let you not only build, but also edit, tailor and refine presentations — even if they weren't made with the GR*i*NS Editor initially!

### *The Previewer: The Embedded Presentation Player*



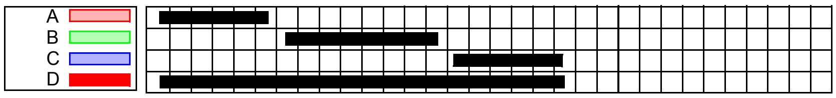
One of the unique features of GR*i*NS is the ability to incrementally preview a presentation as it is being designed. To do this, select Preview from the Preview menu or click on the VCR-like start button on the Preview control window. If you have a switch in your presentation, the *preview control* lets you set the environment values used during previewing.

### *The Structured Timeline: The Logical View of a Presentation*

When you open a presentation, the Editor gives you a picture of the relationship among the objects in that presentation. We call this the Structured Timeline. The Structured Timeline can be very simple for simple presentations, or it can be a complex presentation roadmap for large, complex presentations.

> **GR*i*NS takes a different approach to creating presentations than most timeline or text-based systems, so getting to know the nature of GR*i*NS now will help you to become more productive in the future.**

The following illustration is typical of most timeline-based editors:



In this approach, a set of media items is listed on the left and their activation times are shown on the time graph. Pretty simple. Unfortunately, the time graph doesn't give us essential information. First, the length of the bottom line (node D): is it 20 time units long because this is the sum of the times of A, B and C — or are A, B and C 20 seconds in total because of D? What happens if we substitute B with an item that is 12 units long — should C shorten its duration (because of its relationship to D), should B and C overlap, and should the duration of D be adjusted? Also: what if we want to show a video on a high-bandwidth connection and a sequence of four images on a low-bandwidth connection — how do you draw *that*?

Timeline systems actually hide the logical relationships among items. This means that editing, updating and maintaining a presentation is a pain in the fingers of the designer. You can't reuse combinations of media, and rather than having the timeline automatically created for you, you have to do it all by hand.

The editing philosophy behind GR*i*NS is that of *structured authoring*. As with structured programming and structured design, it is often useful to create documents based on the structural relationships among components rather than getting lost in the details of a single object's activation. This not only makes initial creation easy, it also makes editing and updating a document much easier to accomplish.

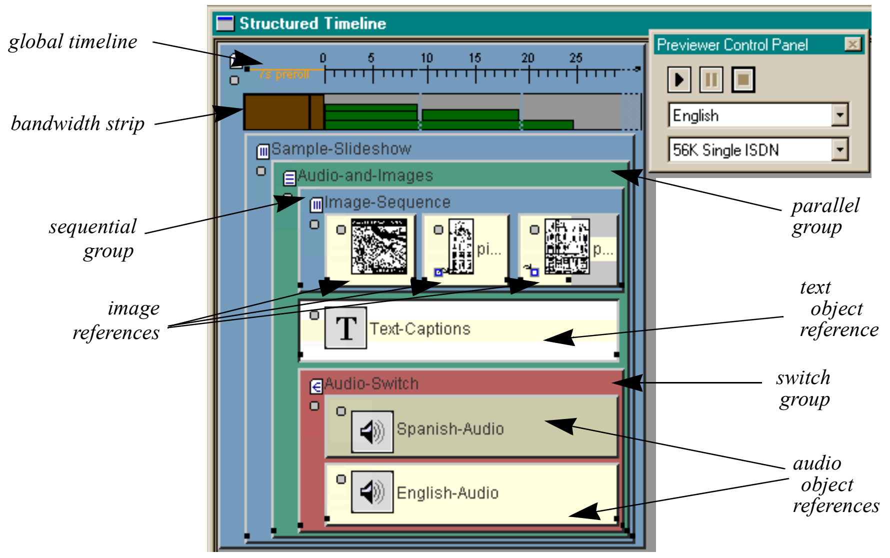
For example, if a single presentation contains an audio node and an image node that are to be presented together, GR*i*NS puts these inside a *parallel structure group*. If, on the other hand, a presentation contains two audio nodes that get played one after the other, GR*i*NS puts these inside a *sequential structure group*. If you want to define a set of alternatives (such as having both Spanish and English audio tracks available in a presentation), GR*i*NS puts these inside of a *switch group*. All of these notions reflect the way that SMIL works: the parallel group is equivalent to

SMIL's PAR, the sequential group to SMIL's SEQ and the switch group to — you guessed it! — SMIL's Switch. (This is no accident: the designers of GR*i*NS/RealONE were also the co-designers of SMIL.)

Consider the following SMIL code fragment:

```
<par id="Audio and Images">
   <seq id="ImageSequence">
      <img src="a.jpg" dur="9s" />
      <img src="b.jpg" dur="9s" />
      <img src="c.jpg" dur="4s" fill="freeze" />
   </seq>
   <text id""Text-Captions" src="d.txt" fill="freeze"/>
   <switch id="AudioSwitch">
      <audio id="SpanishAudio" src="esp.wav" systemLanguage="Spanish"/>
      <audio id="EnglishAudio" src"eng.wav"/>
   </switch>
</par>
```

The following annotated view shows the basic GR*i*NS version of the same file.



This picture describes a presentation named *Sample Slideshow*. It contains three images played in sequence, which are presented in parallel with one Text object containing text comments and either an English-language or Spanish-language audio object (the selection of which will be determined by a user preference in the Player).

*Please re-read the last paragraph and look closely at the image: if you understand the correspondence, then you already understand the basics of GRiNS.*

Unlike a standard timeline, the GR*i*NS Structured Timeline shows the structure containers that define the logical relationships around the media. With a bit of practice, this gives you the full logical and temporal story in one view.

The purpose of the Structured Timeline is to give a logical projection of the contents in a presentation to the author. The Structured Timeline contains the following types of information:

1. It shows the *media objects* used in the presentation; these are pointers to a particular object located either in a local file or as a remote asset on the Web or in a database, and its associated properties for this particular use in the presentation.
2. It shows what kind of *structure groups* are used to compose the presentation. These structure groups can contain a nested combination of data objects or other structure groups.
3. It shows a general presentation timeline. For this simple presentation, in which all times are resolved (that is, there is no interaction), the timeline matches the passage of time as seen by the viewer. (This is not always the case.)
4. It shows a *bandwidth usage strip*. The strip shows when objects are loaded, and if enough bandwidth exists to execute the presentation. It also shows preload time and stalls. Note that **bandwidth is not an abstract concept**. The *player control* allows you to set a particular presentation bandwidth and see how your presentation will behave for that class of connection.

Each of the basic types of container objects used in the Structured Timeline are color coded for easy recognition. With a bit of practice, you can build a mental picture of your presentation without ever hitting Preview — but since GR*i*NS lets you play anything from a single node to an entire presentation during editing, save those brain cells for other activities!

You can perform nearly all of the editing operations required to create most basic presentations from within the Structured Timeline. You can add, move, cut, copy, and paste objects (both data objects and structure containers) using the menu options or right-mouse short-cuts. You can define SMIL switch groups and identify the alternatives associated with the switch. You can create basic hyperlinks between nodes in the presentation. You can zoom in/out of parts of the Structured Timeline. You can also access all of the properties of each of the objects in the Structured Timeline for detailed control of the presentation.

When possible, a thumbnail image of either the contents or the type of data objects is shown. If the view becomes too crowded, these thumbnails are scaled or removed. (You may influence this behavior in GR*i*NS/Pro.)

You can select any object in the Structured Timeline and then select the Preview single object function from the Preview menu to preview that object (or structure container). If you select Play from object instead, the presentation will start at that point and continue playing through the rest of the presentation's objects. To view the entire presentation, select the Preview option from the Preview menu.

The Structured Timeline is the default Editor view. It can be activated by selecting Structure view from the Window menu. The Structured Timeline can be closed by using standard windows techniques for your platform.

NOTES:

1. There are often several ways of selecting GR*i*NS editing functions: you can use the pull-down menus across the top of the Editor, you often can use the context menus under the right mouse button or you can make use of the keyboard equivalents for most commands.
2. The keyboard equivalents have been customized to meet the expectations of experienced users on each of the platforms upon which GR*i*NS is implemented. The structure of the menus is constant across all versions of GR*i*NS in as far as they are relevant for those platforms.

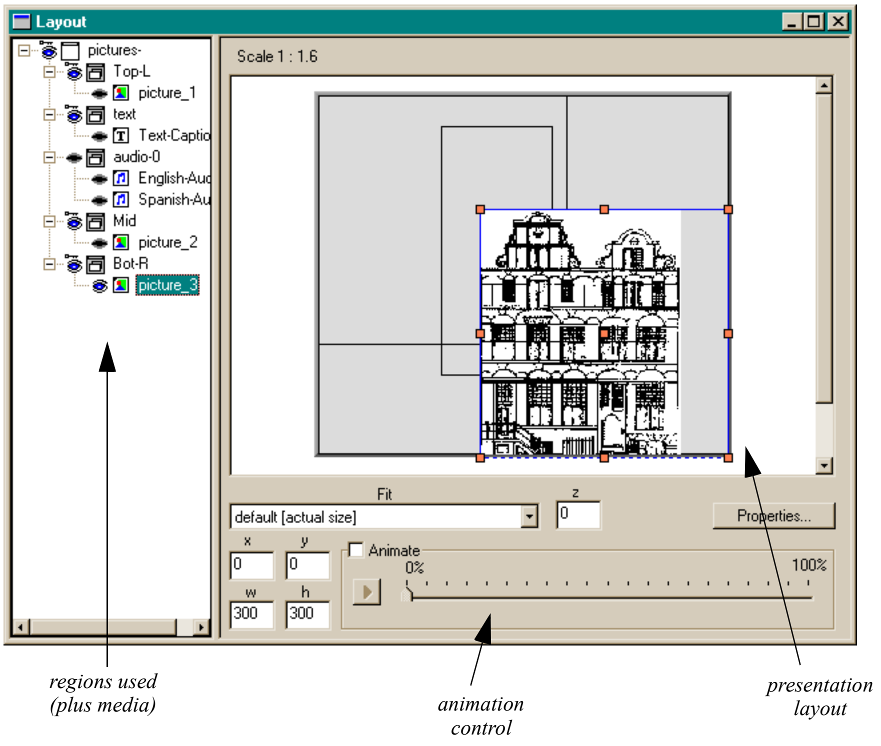
***GRiNS Use Tip:***

*If you select Preview->Preview Single Node for an image or text file, the image/text will only appear if its duration is explicitly defined in the node.*

*The Layout View: Managing Visual and Auditory Resources*

When working with template-based presentations, the template contains a number of Regions to organize screen and other resources in the presentation. If you use these regions, you will typically not need to visit the Layout view. For more complex presentations, or when creating your own templates, the Layout view provides a means of creating, sizing, editing and placing new regions.

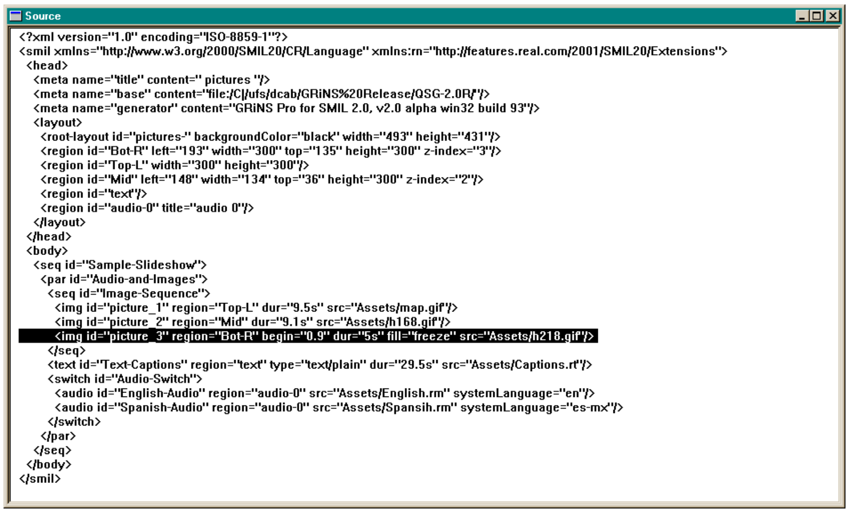
The following illustration shows an annotated Layout view.



When activated, the Layout view shows a window similar to the GR*i*NS Player window and a separate editing window for selecting layout regions. You may select media objects for layout previewing, and you may animate objects.

*The Source View: Seeing the Translated Document*

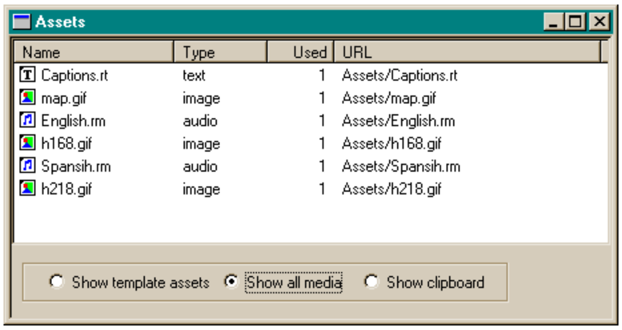
In GR*i*NS/RealONE, you can view the SMIL source for you presentation at any time by selecting Window->Source. You can search for particular strings or look at the translated view of the document.



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<smil xmlns="http://www.w3.org/2000/SMIL20/CR/Language" xmlns:rn="http://features.real.com/2001/SMIL20/Extensions">
 <head>
  <meta name="title" content=" pictures "/>
  <meta name="base" content="file:/C|/ufs/dcab/GRiNS%20Release/QSG-2.0R/"/>
  <meta name="generator" content="GRiNS Pro for SMIL 2.0, v2.0 alpha win32 build 93"/>
  <layout>
   <root-layout id="pictures-" backgroundColor="black" width="493" height="431"/>
   <region id="Bot-R" left="193" width="300" top="135" height="300" z-index="3"/>
   <region id="Top-L" width="300" height="300"/>
   <region id="Mid" left="148" width="134" top="36" height="300" z-index="2"/>
   <region id="text"/>
   <region id="audio-0" title="audio 0"/>
  </layout>
 </head>
 <body>
  <seq id="Sample-Slideshow">
   <par id="Audio-and-Images">
    <seq id="Image-Sequence">
     <img id="picture_1" region="Top-L" dur="9.5s" src="Assets/map.gif"/>
     <img id="picture_2" region="Mid" dur="9.1s" src="Assets/h168.gif"/>
     <img id="picture_3" region="Bot-R" begin="0.9" dur="5s" fill="freeze" src="Assets/h218.gif"/>
    </seq>
    <text id="Text-Captions" region="text" type="text/plain" dur="29.5s" src="Assets/Captions.rt"/>
    <switch id="Audio-Switch">
     <audio id="English-Audio" region="audio-0" src="Assets/English.rm" systemLanguage="en"/>
     <audio id="Spanish-Audio" region="audio-0" src="Assets/Spansih.rm" systemLanguage="es-mx"/>
    </switch>
   </par>
  </seq>
 </body>
</smil>
```

*(In the GRiNS/Pro version of the editor, the source view is also fully editable. Any changes you make to the document will be syntax-checked and applied against the SMIL 2.0 DTD for correctness. Integrated error feedback is also given in GRiNS/Pro.)*

## *The Assets View: Managing the Objects in the Presentation*



| Name | Type | Used | URL |
|---|---|---|---|
| Captions.rt | text | 1 | Assets/Captions.rt |
| map.gif | image | 1 | Assets/map.gif |
| English.rm | audio | 1 | Assets/English.rm |
| h168.gif | image | 1 | Assets/h168.gif |
| Spansih.rm | audio | 1 | Assets/Spansih.rm |
| h218.gif | image | 1 | Assets/h218.gif |

○ Show template assets [✓ Show all media] ○ Show clipboard

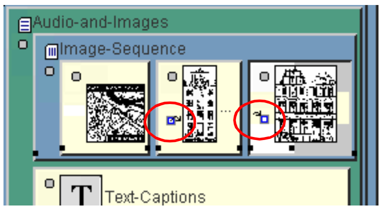
The Assets Window allows you to track use of individual media objects. It also allows you to manage the presentation clipboard, and it allows a template author to create structure templates that contain all types of predefined assets for use in a presentation. (We look at this in detail in the *Tutorial Guide*.)

## *The Hyperlink View: Creating Order within the Chaos of Links*

The GR*i*NS/RealONE Editor provides you with full control over the RealONE *Browser* and *Context* windows. Using simple short-cut icons from the Linking toolbar, you can take any media item and specify that an HTML page gets synchronized in other RealONE windows. (We explain how to do this in the next chapter.)

You can also use GR*i*NS to make links to other SMIL documents via *external links*. The GRiNS/Pro upgrade provides further facilities to create internal links. In GR*i*NS/RealONE, you can create anchors that cover an entire media object for its full activation time: these are *whole node* anchors. In GR*i*NS/Pro, you can create whole node anchors, plus *partial node* anchors that are active on only a part of a object (that is, over a hot spot somewhere over an image or video). You can also create *timed anchors*, that are active for only a particular part of the activation time of an object.

GR*i*NS/RealONE also provides controls to make simple interactive presentations using SMIL 2.0 events. Two symbols identify the presence of events:

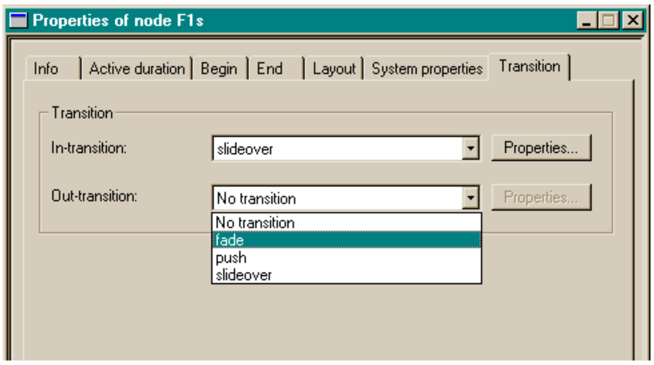


- the *event source* icon with the arrow pointing out from the center of an event box, which indicates that this node contains an event that is the source of interactive behavior in another node.
- the *event destination* icon with the arrow pointing into the center, which indicates that this node contains an anchor that is the source of a link.

You can select either icon to see the *event arc* relationship.

## *The Transitions Window: Input and Output Visual Control*

Each object in a presentation may have one or more visual transitions associated with it. These transitions are defined in the presentation template, and can be selected by going to the properties tab for the object on which you wish to define the transition. If you select the transitions tab, you can select and set either the input or output transition for the object.


Properties of node F1s

Info | Active duration | Begin | End | Layout | System properties | Transition

Transition

In-transition: slideover [Properties...]

Out-transition: No transition [Properties...]

No transition
fade
push
slideover

For each of the selected transitions, you can select further properties of that type (such as direction or duration).

# Using GR*i*NS Editor for RealONE

This section provides a short tutorial to get you started with GR*i*NS/RealONE.

## The GR*i*NS Workflow Model

To understand how GR*i*NS/RealONE works, consider this:

- The Structured Timeline is the main editing view for a presentation created with the GR*i*NS Editor. It helps you to quickly and easily define relationships among data objects, and allows you to define structure containers that help you organize, edit and maintain your presentation.
- The Layout window provides facilities to exactly position media objects to meet the needs of your design. (You can position objects relative to region boundaries or relative to each other.) You can also make dazzling animations via the built-in animator.
- You can add input and output transitions to objects or groups of objects. (In GR*i*NS/RealONE, the template you use may constrain transition options.)
- The Previewer allows you to preview your work-in-progress. Once you are satisfied, you can publish to the RealONE Player.
- You may choose to 'pre-convert' your image, text, audio and video information into RealMedia formats before you add them to a presentation, or you may elect to have GR*i*NS/RealONE do the conversion for you automatically. If you let GR*i*NS/RealONE do the conversion, you can influence how conversion and compression takes place from within the Properties sheet attached to each object (found under the Edit menu).

GR*i*NS/RealONE has been designed to be a template-based editor. GR*i*NS/Pro gives you more control over all aspects of the document creation process, including the construction of full presentations from scratch. The tutorial in this section provides an overview of basic GR*i*NS/RealONE functionality.

## Tutorial: Creating a Simple Presentation

The purpose of this tutorial is to create a presentation that consists of a sequence of objects that get rendered against a background image. We'll add some transitions and animations to spice up the presentation and we'll show how you can use some special GR*i*NS/RealONE facilities to make editing more efficient.

### *Overview and Goals*

In order to create this presentation, we will perform the following steps:

1. Start the Editor and select the *Slideshow with Background Music* template;
2. Insert media objects into the presentation;
3. Add transitions on the media objects, and animate their positions;
4. Add a link to the RealONE Browser window;
5. Preview the presentation; and
6. Publish the presentation for use with RealONE.

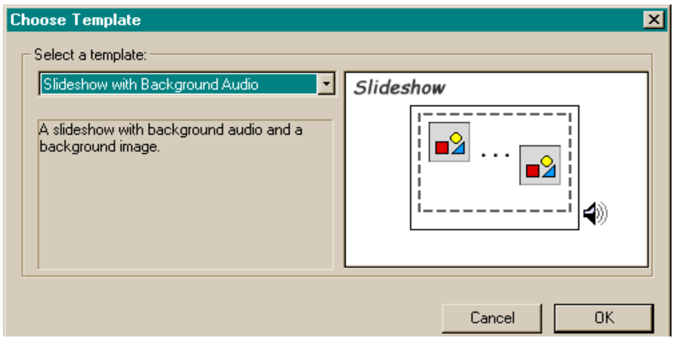
To see what we are aiming for in this example, open the presentation *QSG-T1.smil* from within the RealONE player. You can find the presentation in the GR*i*NS/RealONE installation directory tree; the default location is:

*c:*\*Program Files*\*Oratrix*\\GR*i*NS_*RV2*\*DocCommon*\*QSG-T1.smil*

(the exact location will depend on where GR*i*NS/RealONE was installed). You also may want to view the SMIL source — if you like that sort of thing — but it isn't required to understand the basic presentation. (Don't open the *.grins* file yet.)

### *Opening a Template*

If you have not already done so, start-up the GR*i*NS Editor; if you get a pop-up box asking if you want to create a new presentation or edit an existing one, select New Presentation. Otherwise, if no pop-


Choose Template

Select a template:

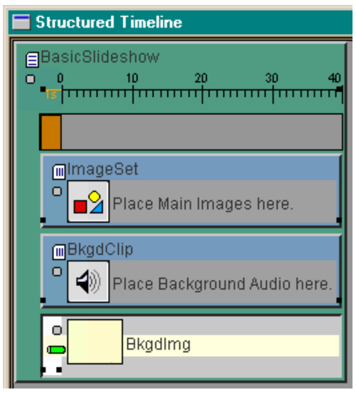
Slideshow with Background Audio

A slideshow with background audio and a background image.

up appears, go to the File menu and select New, which allows you to select a GR*i*NS Template file. Select the *Slideshow* template shown above. Once you open the template, you will be prompted for a location/name to save the file. GR*i*NS will remember the last folder you used for saving presentations; if there is no last folder, GR*i*NS uses the Desktop.

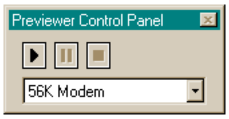
For this template, use the name QSG-T and use a non-GR*i*NS folder.

Whenever you create or open a file, you see the Editor's Structured Timeline. In this view, we see the green outer container labelled *BasicSlideshow.* This is a *parallel* (PAR) group, containing three objects plus a timeline and a bandwidth strip.



A parallel group defines a local timeline upon which its contents get scheduled. In this example, the "contents" are a background image and two container objects (*ImageSet* and *BkgdClip*). We will place the media objects into these containers.

Before we populate the template, look at the bandwidth strip. It says that there will be a one-second delay at the start of the presentation — this delay is used to load the background image. If you see a different delay time, this may be because the *previewer bitrate* is not set to 56K Modem, as is shown in the figure below.



In order to get a useful idea of presentation performance, you always need to define the assumptions about the client's network connection. (You should try setting the bit rate in the Previewer Control to several values — higher and lower than 56k Modem — and see the result on the bandwidth strip. Once you are done, set the value back to 56K Modem.)

NOTES:

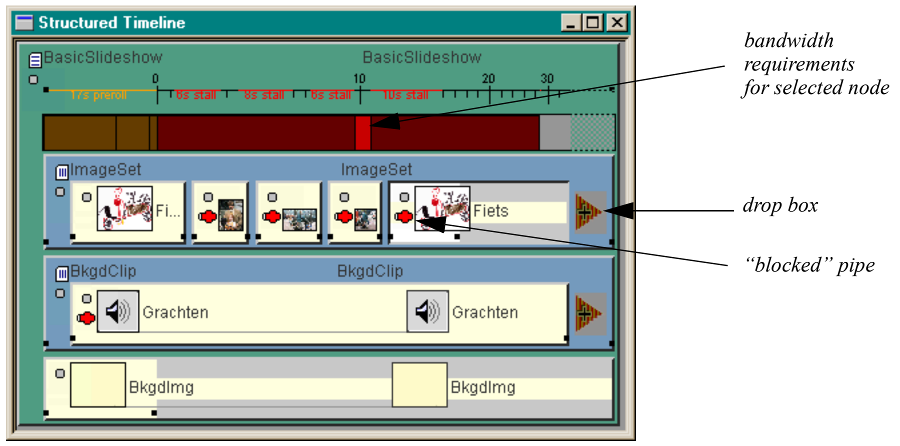
1. The colors used for the various GR*i*NS windows depend on the Windows color scheme installed for your system. This tutorial uses the Desert color scheme.

*Inserting the Media Objects*

The *SlideShow* template contains all of the information required to create a presentation *except* the actual images used and the background audio. To add these, do the following:

1. Using the Windows file navigation, open the *Assets* folder in the *DocCommon* folder (in the GR*i*NS root directory *c:/Program Files/Oratrix/*GR*i*NS*_RV2*);
2. Select and drag the *Grachten.mp3* file to the container with the text annotation: *Place Background Audio Here*;
3. Select and drag the image *Fiets.gif* onto the container with the text annotation: *Place Images here*;
4. Note that the text goes away and a triangular drop box appears on the ImageSet container. Select and drag images *Pict1.jpg, Pict2.jpg* and *Pict3.jpg* onto this drop box to add them to the container;
5. Finish the show by selecting and dragging a second copy of *Fiets.gif* to the end of the ImageSet container.

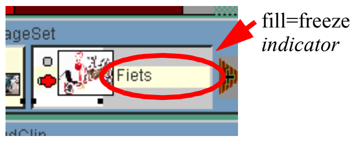
The illustration shows a 'filled' version of the Structured Timeline, containing thumbnail icons for each of the images you have placed on the presentation. It also contains a bunch of warning symbols, because we are asking a lot of our 56K modem! (This view also contains a lot of other information useful to an author, but we'll get back to those in a page or two.)



## *Previewing the Entire Presentation*

Hit the Play icon in the Preview Control. You will see the background image, hear a charming tune, and see five images, each lasting 3 seconds. Actually, you will see four images lasting 3 seconds and one image lasting as long as the rest of the music. (This is about 20 seconds.) In the *Slideshow* template, each image object has a default duration of 3 seconds, but the last image stays on the screen until its *time context* ends. In SMIL terms, it has its fit attribute set to freeze. (The template author took care of this for you.)

If you look in the structure view, you can see a visual representation of fill=freeze by the colored bar that extends to the end of the time container. Note that the background image (at the bottom of the green PAR) has this behavior, too.



The distinction between an object's duration and its visual rendering time is very important in SMIL and in RealONE. We'll see that it has a lot of obvious and subtle side-effect in defining a presentation.

## *Previewing Part of a Presentation*

GR*i*NS allows you to selectively play parts of a presentation. To get a quick taste of how this works, select the container labelled *ImageSet* (it contains the five bicycle images). Go to the Preview menu and select Preview Single Object. This plays all five images, but not the audio and not the background image. Now, select the first image in the sequence (*Fiets.gif*), and again select Preview Single Object. You now see only that one image. You can preview any object at any level (at any time!) using this very handy — and uniquely GR*i*NS — feature.

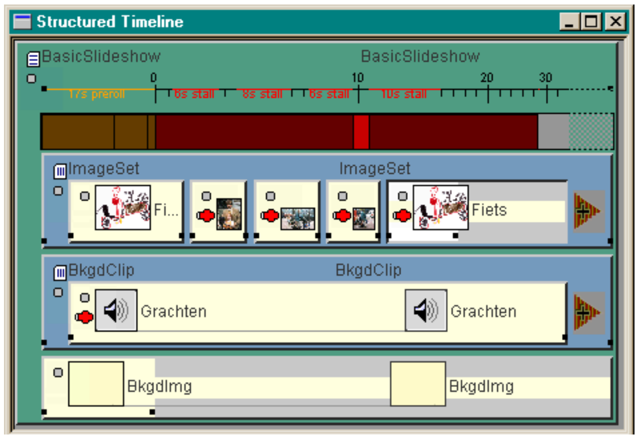
You can also select Preview From Object in the Preview menu. For example, select the second image in *ImageSet*. Preview From Object begins previewing the presentation from the point you selected. Most objects from the context are also started, unless an unacceptable performance burden results.

*Making the Presentation More Attractive*

There are several things you can do to make the presentation more attractive. In the text below, we will discuss the following:

- fix presentation bandwidth/performance problems
- add transitions to the images
- reposition some images to be centered in the viewing area, and
- add a simple animation

<u>Fixing Bandwidth-Related Problems</u>



Take another look at the Structured Timeline window. The view shows several red pipes, each indicating a potential performance problem. Click once on the red pipe on the second image in the *ImageSet* container; it says that we need another 6.5 seconds to load this image. In other words, if we delay the start of this image, the presentation should behave somewhat better (at least at 56K, which is our current analysis bitrate). Before we do this, click on the red pipe on the audio object. It says we need another 416 bps to play this music.

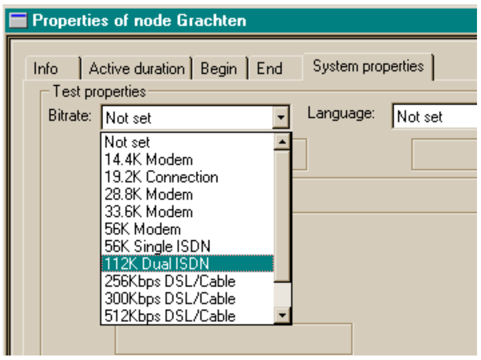
Now, set the bitrate in the Preview Control to 112K. At this rate, we have plenty of headroom for the audio, but we are still a bit tight on bandwidth for the images.

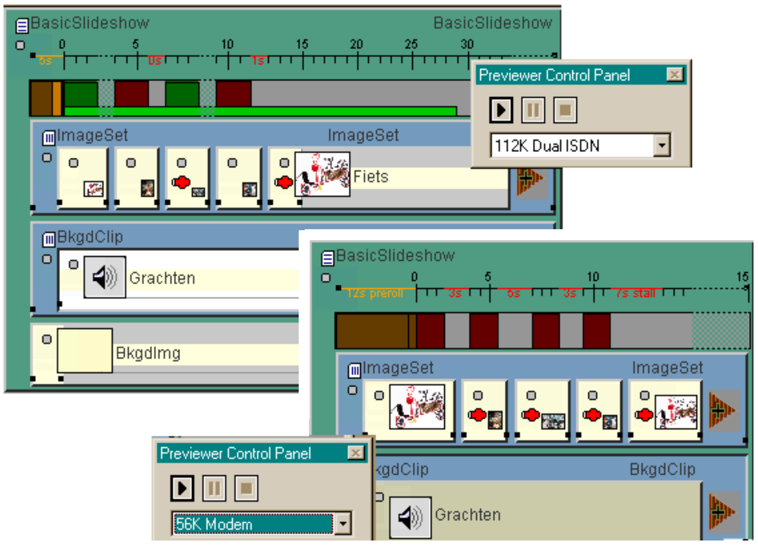
In fixing these problems, we can take one of two approaches:

1. we can design for the low-end, making it less interesting for the high-end, or
2. we can adapt the presentation, so that it does something sane on a low speed connection, but also has some more pizazz at the higher end.

This is easily done in GR*i*NS. Go to the audio object, and double-click. This brings up the Properties box. Go to the System Properties tab, and set the *System Bitrate* selector to 112K Dual ISDN. Then select OK. What you have done is to say:

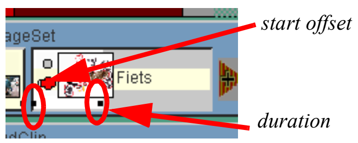
> *Play this node ONLY if the system bitrate has been determined to be 112K or higher. If it is lower, skip this node in the presentation.*





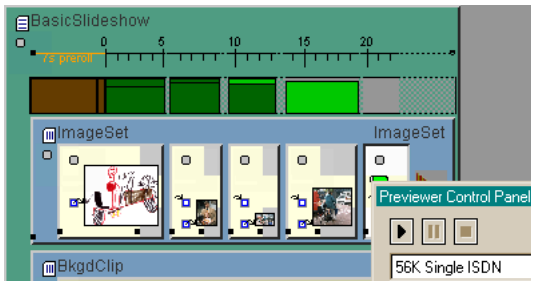
Set the Preview bitrate to 112K and then 56K, as shown above. In the 112K case, all of the objects are in the presentation. In the 56K case, the audio is selectively excluded (as is indicated by its darker, inactive color.) By selectively excluding objects, you can create compelling content for the high-end without totally disenfranchising the low-end.

Setting Object Start Times and Durations



If we set the previewer bitrate to 56K Modem, and look at the *ImageSet* container, we see that we still have some problems with images that need more time to load at the given bitrate. To correct this, we can use two GR*i*NS controls: the *start offset* control and the *duration* control. These controls are located at the bottom of each object in the Structured Timeline.

While it is tempting to think of the start offset control as defining the *start time* of an object, it really defines *a delay from the end of a previous object*. The duration control defines the simple duration of the object; the object's fill behavior will determine how long it is on the screen.



Working left to right, drag the start offset control to the right to adjust the amount of time available to bring in the image. Use the "time needed" figure in the red pipe as a guide. (Note the space-saving convention of adjusting the timeline and the icons when you do this — this is GR*i*NS's unique method for saving your elbows!) When you are done, your structured timeline window should look like the image above. In general, the more of the bandwidth strip consumed with image blocks, the less headroom you have for unpredictable network performance, but since GR*i*NS tried to be conservative, a picture as above is a good place to start.

## Adding Transitions to Media Objects

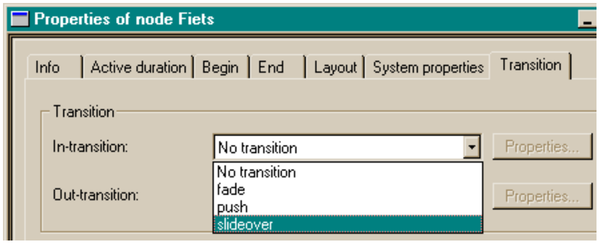
The RealONE player works hard to deliver your streaming media on schedule. Once that media has arrived, however, the Player can also help you make things look more exciting. Three classes of media handling can be used: transitions, animations and layout positioning. The interesting thing about all of these features is that they consume zero bandwidth. They do, of course, make some additional demands on the client's platform, so don't overdo the use!

Adding transitions involves:
1. selecting the node on which you want the transition,
2. opening the Properties->Transitions tab on that node, and
3. selecting either the input or output transition.

We will look at all of these in turn.

In GR*i*NS/RealONE, you can select the transitions that have been defined in the presentation template. GR*i*NS/Pro lets you define custom templates.

Start by selecting the first image object in the *ImageSet* container. Open its Properties tab by double-clicking or via Edit->Properties, and select the Transitions tab. Select an input transition of type: *slideover* and select Apply.



You can preview this node by selecting Preview->Preview Single Object (also available under the right mouse key).

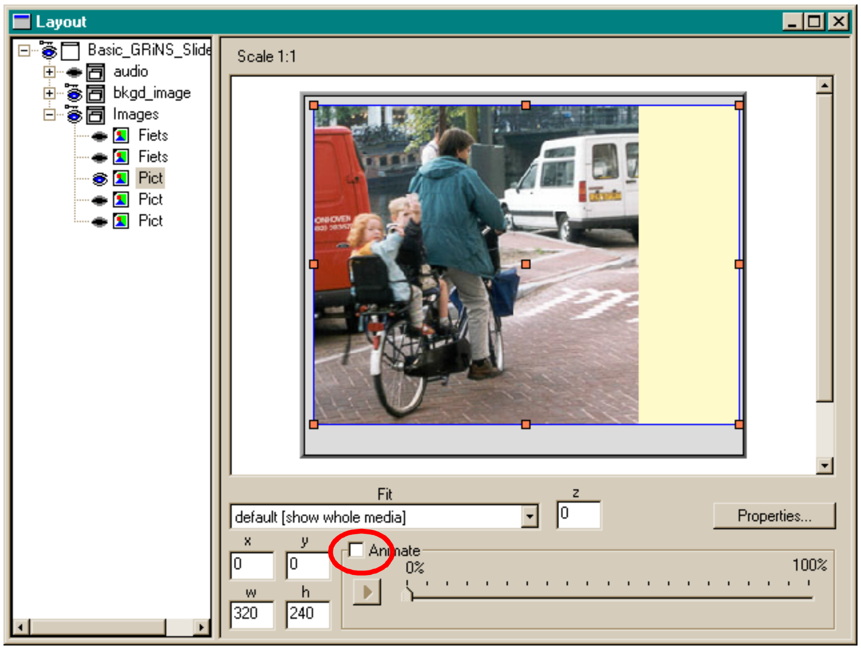
Now, add input transitions of various other types to the other four nodes in the ImageSet object. You can preview each node individually, or you can preview the entire presentation. For purposes of creating a predictable tutorial, set the input transition on the second image (*Pict1.jpg*) to Fade.

Adding output transitions in sequences often requires that you change the value of the fill attribute from *freeze* to *transition*, done via Properties->ActiveDuration.

## Adding Animation

You can easily add object-based animations to your presentation. What you animate is the *rendering position and size*, not the content itself. (If you want animated content, you should add Flash or SVG objects to your presentation.) Since position and size are specified via the Layout window in GR*i*NS/RealONE, this is also where the animation control is found.

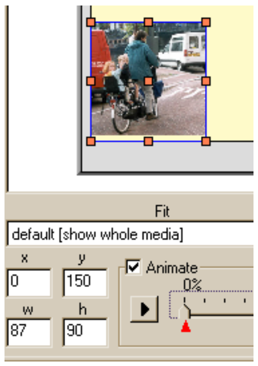
Select a media node that you would like to animate. For example, select the fourth image in *ImageSet*, the one named *Pict3.jpg*. Next, open the Layout window (via Windows->Layout). The view you get is:



On the left side of the Layout window is the region/media selection tab. At top right is the placement window, and at bottom right is the Animate control. Enable animation via the Animate check box.

The scale on the Animate bar defines the duration of the animation in terms of the active duration of the media object. (If you make the duration longer in the Structured Timeline, the animation will occur more slowly.)

Set the pointer on the Animate Timeline to 0%, and using the drag handles, make the image small, and place it in the lower left hand corner. It should look something like the image at right. (Make sure you turn on Animate before you resize the image.) Note the value of the Fit attribute — this will determine how an image behaves during resizing. Show whole image usually gives a desired result, but you can try others.



Fit: default [show whole media]

x: 0
y: 150
w: 87
h: 90

☑ Animate
0%

Now, hit the Play icon next to the Animate timeline. You'll see a preview of the animation. The preview repeats continuously; you can stop it via the same button (which now is a Stop icon.)

SMIL and GR*i*NS/RealONE use a 'return to rest' animation model. That is, you define a non-standard position, and the animation will return the object to its original position and size. You can define intermediate points by positioning the slider. If you select an arrow under the Animate timeline and then shift-drag the value, you can copy existing size and position parameters.

You can delete any animation point by double-clicking its timeline triangle.

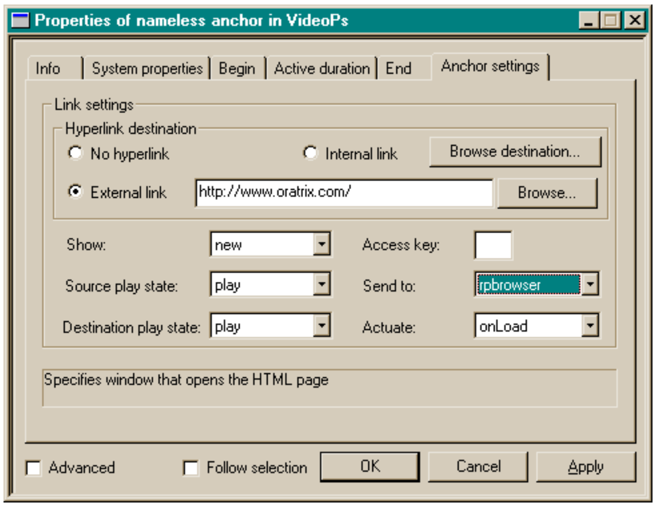
## Object Positioning

When you created the animation, you did two things: defining a position/size pair on the region holding a media object, and associating a time on the Animate timeline. You can also change the places where objects get rendered by simply defining a size/place pair. You can also define relative positioning of objects by selecting them in 'eyes' mode in the left pane, and then choosing one to reposition and scale. Not having every image appear at top left makes your presentation much more interesting to watch.

## Synchronizing Content with the RealONE Browser Window

To create a self-firing link to the RealONE Browser window, do the following:

1. Select the node in the presentation that will serve as the source node for the link. (This is the node that, when played, causes the page to appear.)
2. See if the linking toolbar in the GR*i*NS/RealONE Editor has been activated; if not, turn it on via the View->Toolbars->Timing and Linking menu entry. The toolbar is shown above.
3. Click on the icon with an arrow pointing to the context window, as shown at right. ← *create Browser Window link*
4. Enter the URL (either as a full Web path or as a local name) into the property dialog box for the newly created anchor. The property box and an example URL are shown below:


Properties of nameless anchor in VideoPs

Link settings — Hyperlink destination: ( ) No hyperlink ( ) Internal link (•) External link: http://www.oratrix.com/

| | | | |
|---|---|---|---|
| Show: | new | Access key: | |
| Source play state: | play | Send to: | rpbrowser |
| Destination play state: | play | Actuate: | onLoad |

Specifies window that opens the HTML page

That's it. Now, when you play the presentation, the Web page appears in the RealONE browser window.

(Creating a self-firing, proactive link to the context window uses the same process as for the browser window, except that the context window icon is used.)

## *Publishing the Presentation for RealONE*

You can preview the presentation from within the GR*i*NS/RealONE editor, but if you want others to see your show — which is the whole point! — you will need to publish it for use with the RealONE player. Publishing your presentation for RealONE consists of several steps, some of which are optional.

### Generate the RealONE Presentation

Generating the presentation consists of:
1. converting any non-RealMedia components to RealMedia,
2. creating the various RealONE control files, and
3. creating the HTML files for your presentation

All of these steps are automated; they are triggered by the File->Publish for RealONE command.

Once you start this process, a final bandwidth check is made for the bitrate specified in the previewer. All of the non-RealMedia objects (such as *Grachten.mp3*) are converted to *.rm* files. (You can influence this conversion via the Properties->Conversion tab on each media object. Make sure the "Advanced" button is selected at the bottom of the tab to view all of the options.) The files required for RealONE are placed in the directory containing the *.grins* file — you will see a QSG-T.smil file, a folder with data assets and various RealONE control files.

### Generate the G2 Presentation and Upload

Once the presentation has been created, you can also upload it automatically to a RealServer. Select Edit->Publish for RealONE and Upload to do this from GR*i*NS. Please check with your system or network administrators to set the various upload parameters required for the RealServer you plan to use for hosting your presentation.

## A Note on Designing Adaptive Presentations

As a content author, you can either create a simple presentation that assumes your users/clients will be able to view all of the data objects that you send them. You can also take a broader view by specifying some alternatives to either individual objects or to entire sub-parts of the presentation. Such alternatives can be based on presentation resources (for example, during peak hours on the Web, it may be wise to substitute a slide-show presentation for a piece of video), or the choice may be based on natural language (for example, having both English and Spanish versions of an audio and/or a set of text messages available within the same document).

One of the primary reasons to support alternate content is to help users who, for whatever reason, cannot accept one format of data—such as audio for a deaf user, or video for a blind user, or high-resolution information for a user attached to a low-resolution computer —by providing a more convenient type (such as text instead of audio). It is important to realize that this is not an "us vs. them" issue: all of us are unable to accept certain types of media at certain times, depending on our work or use situations. For example, if you are working without headphones in a shared office space, you may be unable to accept audio information, even if there is nothing wrong with your hearing — the circumstances simply don't allow you to disturb others around you!)

GR*i*NS/RealONE was designed to make it easy to set Bitrate and Language properties of individual nodes. If you want more control over your presentation — including defining content for deaf or blind users (or general Accessibility needs), you may want to upgrade to the GR*i*NS/Pro editor.

# Understanding SMIL 2.0

The W3C's SMIL format for encoding multimedia presentations for delivery over the Web is a little-known, but widely used standard. First released in mid-1998, SMIL has been installed on approximately 200,000,000 desktops world-wide, primarily because of its adoption in RealPlayer G2/7/8, Quicktime 4.1, and Internet Explorer 6. In August, 2001, the W3C released a significant update to SMIL: version 2.0.

## Design Goals

The goals for SMIL 2.0's design can be partitioned across three broad categories:

- *extend the functionality of SMIL 1.0*: the designers of the first version of SMIL purposefully kept the language simple and relatively frills-free. SMIL 2.0 provides some desirable additions, including: support for increased interaction, enhanced timing semantics, extended content control and layout facilities, plus new animation and transitions primitives.
- *maintain a declarative, XML format*: while the integration of multimedia content is often the province of scripted or other programmed definition, SMIL 2.0 was developed to remain a fully declarative rather than a procedural format. In this way, a SMIL description doesn't define *how* a presentation is implemented, but rather *what* it is that the author wants — the implementation of the specification left up to the SMIL player. Also, in keeping with the first version of SMIL, SMIL 2.0 is fully XML compliant.[1]
- *introduce a module-based structure*: where SMIL 1.0 was a simple, single 29-page specification, SMIL 2.0 has been defined in terms of over 50 modules that can be partitioned into 13 functional groups, described in over 500 pages. By using a module structure, key aspects of SMIL 2.0 can be integrated into other XML-based languages without requiring support for the entire SMIL 2.0 specification. Even before the release of the SMIL 2.0 Language specification, the modularization policy bore fruit: parts of SMIL have been integrated into

1. SMIL 1.0 was not only an XML-compliant language, it was the *first* XML language released by the W3C when it was published in 1998.

several other XML languages — such as SVG — and more examples of module reuse are expected.

As of the summer of 2001, several commercial versions of SMIL 2.0 were available (the first of which, Oratrix's GR*i*NS/SMIL 2.0 player, was released in September, 2000), and major mass-market media companies such as RealNetworks and Microsoft had announced support for all or some of SMIL 2.0's features. The 3GPP consortium, the standardization body coordinating the deployment for next-generation wireless devices, has also recommended that SMIL 2.0 be used as the basis for wireless multimedia devices.

## SMIL: what it isn't (and what it is!)

Often, multimedia presentations are characterized by their content, rather than their composition. In this sense, there has been some confusion as to what a SMIL presentation is, and what it isn't.

To begin with the latter, it is fairly easy to summarize what SMIL 2.0 is *not*:

- *SMIL 2.0 is not a Flash substitute*: Flash is a proprietary content media type that is primarily used for small animations. SMIL 2.0 is not a content media type, in the sense that it does not define any particular type of media (such as vector or raster images, videos, text or audio data). Instead of media content, SMIL defines XML-based media composition. A SMIL presentation can include Flash objects.
- *SMIL 2.0 is not an MPEG-4 substitute*: MPEG-4 is a highly touted (but lightly implemented and deployed) format for describing the contents and interactions of media objects. More precisely, MPEG-4 is a family of protocols that covers a wide range of media-related concerns, but not a specific solution to any one class of media presentation. As we will see, there has been considerable work done to coordinate the development of SMIL 2.0 and MPEG-4 through the XMT specification.
- *SMIL 2.0 is not a D-HTML substitute*: Dynamic HTML (D-HTML) was introduced as a way of introducing local time- and animation effects into a static HTML web page. While some of the animation primitives in SMIL 2.0

resemble the functionality of some uses of D-HTML, SMIL's scope is much broader than the local nature of D-HTML.

What, then, is SMIL 2.0? It is a collection of XML elements and attributes that can be used to describe the temporal and spatial coordination of one or more media objects. SMIL 2.0 allows a user to define how independent media objects are to be integrated during the lifetime of a presentation. That presentation may be delivered via a streaming server, or it may be played locally.

SMIL 2.0 defines 10 major functional groupings of elements and attributes. (See Figure 1.) Of these, *Timing and Synchronization* is the core of the SMIL specification. The functional groupings represent collections of individual SMIL 2.0 modules, each of which defines elements and attributes intended to address specific multimedia issues in a reusable manner. The number of modules per functional grouping varies from 2 to about 20. In general, the more modules per grouping, the finer the granularity of each module.

The modularization of SMIL 2.0 was intended to not only facilitate the reuse of SMIL functionality in other XML languages, but to also help define a number of SMIL *profiles*, each of which would provide a collection of modules that could be customized to the primary goal of the profile.

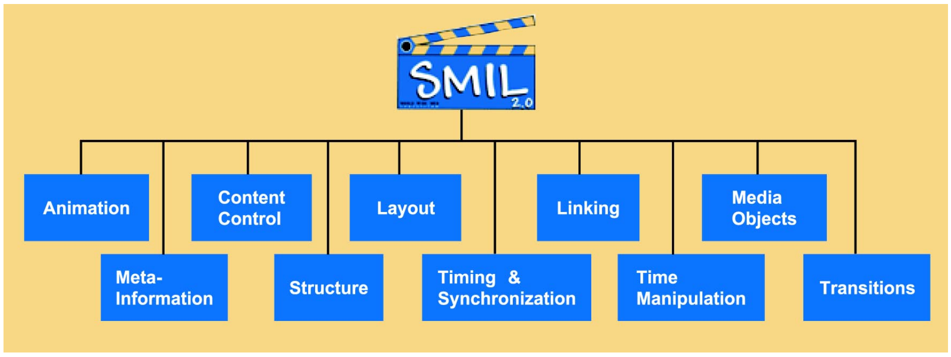


Figure 1: *SMIL 2.0 Functional grouping of module sets*.

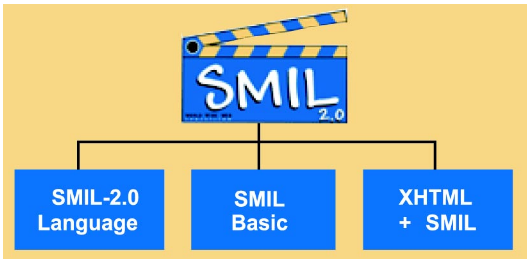


Figure 2: *SMIL 2.0 Profiles.*

Figure 2 shows the initial partitioning of SMIL 2.0 profiles. Of these, the SMIL 2.0 Language and Basic profiles completed the required review and implementation requirements of the W3C for the summer 2001 release of SMIL 2.0. The XHTML+SMIL profile, which seeks to integrate SMIL timing, animation and transitions modules (among others) into XHTML, was not completed and remains under development by the W3C. In addition to the three SMIL profiles, a version of SMIL Animation was also released by W3C in July 2001 for final review by W3C members.

## Structure, Timelines and SMIL

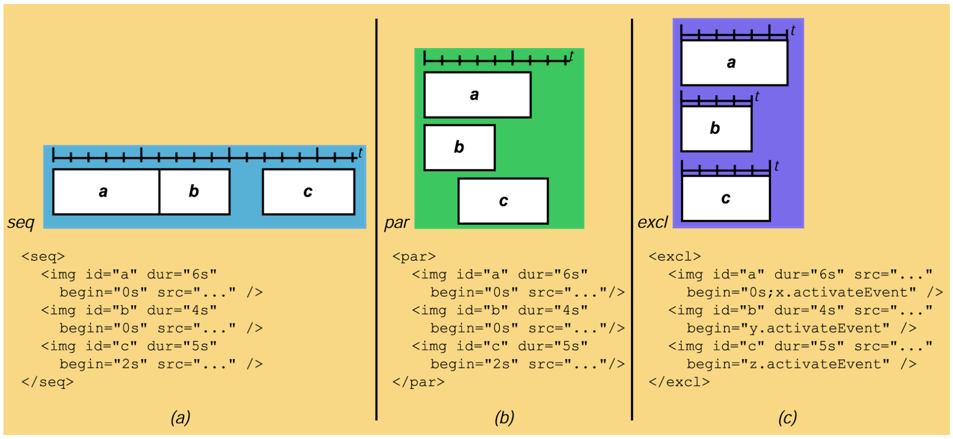
A SMIL presentation is a *structured* composition of autonomous media objects. As illustrated in Figure 3, there are three basic timing containers that can be used in a SMIL presentation:

- *seq* — *sequential time container*: the children of a *seq* container are rendered in such a way, that a successor child never can begin before its predecessor child completes. A successor child may have an additional start delay, but this delay cannot resolve to be negative in relation to the end time of its predecessor.
- *par* — *parallel time container*: the children of a *par* container are all rendered in "parallel". In terms of SMIL's timing model, this does not mean that they get rendered at the same time, but rather that they share a common timebase defined by the *par* container. It also means that any or all of the children can be active at any time that the parent *par* is active. The *par* is the most general SMIL time container.
- *excl* — *exclusive time container*: only one of the children of an *excl* can be active at a time. The order of activation of the children depends on the *begin* attribute of the *excl*'s children. Typically, each will have an event-based start condition such as:

```
begin="button1.activateEvent"
```

that allows one of the children to start on demand. The *excl* container is new in SMIL 2.0.

The three basic time container types can be nested in a presentation hierarchy. That is, any child of either a *par*, *seq* or *excl* can be a simple media object or an embedded *par*, *seq* or *excl* container. As in SMIL 1.0, the hierarchy can represent relatively static presentation timing. The introduction of event-based activation/termination in SMIL 2.0 also allows a dynamic activation path to be defined. Note that most other multimedia formats only support a *par*-like semantic, not the *seq* or *excl* semantics.



```
<seq>
  <img id="a" dur="6s"
    begin="0s" src="..." />
  <img id="b" dur="4s"
    begin="0s" src="..." />
  <img id="c" dur="5s"
    begin="2s" src="..." />
</seq>
```

```
<par>
  <img id="a" dur="6s"
    begin="0s" src="..."/>
  <img id="b" dur="4s"
    begin="0s" src="..."/>
  <img id="c" dur="5s"
    begin="2s" src="..."/>
</par>
```

```
<excl>
  <img id="a" dur="6s" src="..."
    begin="0s;x.activateEvent" />
  <img id="b" dur="4s" src="..."
    begin="y.activateEvent" />
  <img id="c" dur="5s" src="..."
    begin="z.activateEvent" />
</excl>
```

Figure 3: *SMIL 2.0 Temporal Groups*.

(a) shows a SMIL *seq* container, plus a SMIL 2 code fragment. Note that *begin* times are relative to the predecessor end.

(b) shows a *par* container, with the same timing offsets. Note that *begin* times are relative to the containing *par*.

(c) shows an *excl* container. Object *a* starts at time 0 of the excl *and* whenever object *x* (not shown) is activated.

Objects *b* and *c* are started only when objects *y* and *z* (not shown) are activated.

Since the actual activation times depend on event activity, a common timeline cannot be used to model the relationships among *a*, *b* and *c*.

## *Timeline Definition*

A fundamental property of multimedia presentations is that they contain media objects requiring some notion of *time* for correct presentation. In order to model presentations, a timeline metaphor is often used. To understand SMIL 2.0, the *timeline* metaphor can be applied to containers that hold only static objects for which all timing (including activation time, termination time and object duration) are known at the time of authoring. For containers like the *excl*, where the activation or termination time is unknown until the object actually starts, or for *par*'s containing objects with unresolved *begin* or *end* times, the timeline metaphor is essentially useless. While this presents a challenge to authoring software designers, it provides a SMIL 2.0 document with unprecedented temporal flexibility and adaptability because the effective presentation timeline is determined at runtime, based on the individual media objects activated and their activation order.

Unlike media formats which force the user to explicitly define when each object begins and ends relative to a single timeline, SMIL provides a logical timing framework in which the structured relationship of objects can be used to define most timing relationships among objects. When writing a SMIL file, the author does not have to worry about the exact starting or ending times, since these can be gleaned from the presentation's *par/seq/excl* structure. In a SMIL specification, the timeline is determined by the structured composition of objects, rather than the timeline being the basic compositional unit.

The decoupling of timing resolution from presentation specification allows, for example, the entire timing of a presentation to be changed based on the dynamic associations of content. Consider the following SMIL 2.0 code fragment:

```
<par endsync=”select”>
 <img id=”btn_a” src=”...” dur=”10s” />
 <img id=”btn_b” src=”...” dur=”5s” />
 <excl id=”select”>
   <text  src=".../todays_txt.html" begin=”btn_a.activeEvent” dur=”25s”/>
   <video src=".../todays_video.mpg” begin=”btn_b.activeEvent” />
 </excl>
 <audio src=".../todays_tune.mp3" repeat=”indefinite”/>
</par>
```

The outer *par* contains two button images (*btn_a* and *btn_b*), an audio object reference, and an *excl* container (named select). The *excl* contains a text object reference and a video object reference. Note that each of the media object references is indirect: they point to external data that can change on a hourly, daily or weekly basis.

The button images are displayed for 10 and 5 seconds, respectively, when the *par* starts. The background music also begins, and repeats indefinitely — that is, it repeats until the containing *par* ends. Since the *par* contains the directive:

```
        endsync=”select”
```

it will end when the *select* object ends. The *select* object ends when either the text or the video object ends, depending on which one is activated by the presentation viewer. (If the viewer selects *btn_a*, the text object will be shown for 25 seconds; if the viewer selects *btn_b*, the video object will play for the intrinsic duration that object, whatever that is.) All the while, the background music keeps repeating.

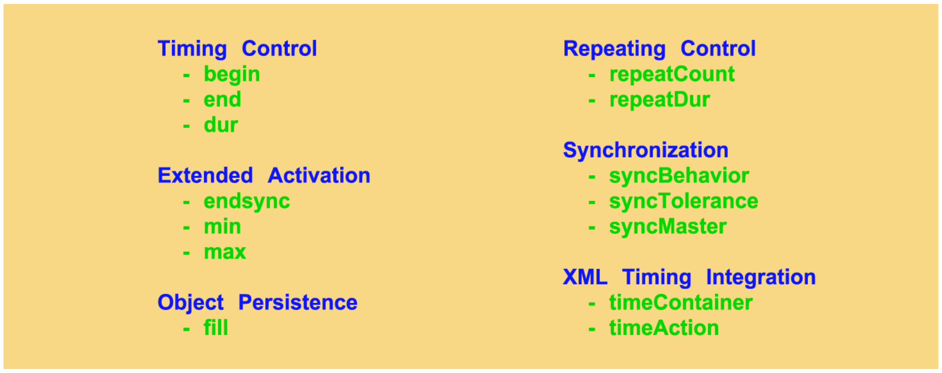
Drawing the presentation timeline is left as an exercise to the reader.

## SMIL 2.0 Timing and Synchronization

The timing and synchronization functional group represents the core of SMIL 2.0 functionality. The group is divided into 19 modules, each of which defines a collection of XML elements and attributes that control some aspects of timing.

The three basic SMIL 2.0 timing elements *seq*, *par* and *excl* were introduced above. Each of these elements form parent timing containers in which media objects or other timing containers can be placed. For any content within a container (whether a media object or structure container), the primary issues to be addressed are:

- when does the element begin?
- how long is it active?
- what happens to it when it is no longer active?
- other than strict temporal, are there other conditions that cause an element to end?



**Timing Control**
- begin
- end
- dur

**Extended Activation**
- endsync
- min
- max

**Object Persistence**
- fill

**Repeating Control**
- repeatCount
- repeatDur

**Synchronization**
- syncBehavior
- syncTolerance
- syncMaster

**XML Timing Integration**
- timeContainer
- timeAction

Figure 4: *Primary SMIL 2.0 timing and synchronization attributes*.

These questions are 'answered' by specifying a set of attribute values on either the parent time container or any of its children. SMIL 2.0 has an extensive set of attributes to control timing, most of which carry sane defaults so that basic timing and synchronization operations can be accomplished easily.

The basic collection of timing and synchronization attributes are shown in Figure 4. Note that not all of the SMIL 2.0 profiles support all of the attributes, and the syntax of defining and setting the attribute values may vary, but the core functionality of timing, extended activation control, object persistence and repeating element control are reasonably universal. A design objective of SMIL 2.0 was that each attribute should have a well-defined semantic that remains constant across profiles.

## *Begin, End and Dur*

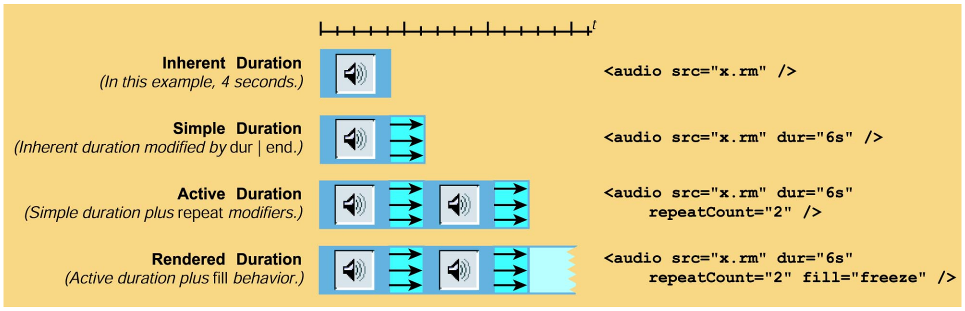
The *begin* and *end* attributes are very similar in terms of syntax and semantics. The primary differences between these attributes in SMIL 1 and SMIL 2.0 is that SMIL 2.0 allows multiple values to be specified for begin and end. (An example of this was shown in Fig. 3.) The first begin/end value that is satisfied will cause a corresponding element to start or end. It is possible to mix both scheduled and event-based activation/termination in one attribute. For example:

```
begin="3s;button.activateEvent"
```

will cause the associated element to start either at 3 seconds after the default time at which the element would otherwise be able to start, or when the *activateEvent* event associated with another element with an ID of button occurs (typically via a mouse click). The 'default' begin time varies with the parent time container: for a *par* and *excl*, it is when the container starts; for a *seq*, it is when the previous element in the *seq* ends (or at the start of the *seq* for the first child element).

Once started, an element will have a certain duration, which may be determined in several ways (see Figure 5). In general, so-called discrete media — media without an inherent notion of time, such as images or a page of text—have a default duration of 0 seconds, while continuous media—media with an inherent notion of duration, such as an audio or video object— use that inherent duration as default. (Some implementations of SMIL 1.0 used 5 seconds for the default duration of a discrete media item. All SMIL 2.0 implementations should use the "real" default value of 0.)

An object with a duration of 0 seconds would ordinarily not be visible during a presentation. Such objects can be displayed if the *fill* attribute is set to a value of *freeze*. A frozen object is displayed after the end of its active duration until its


t
Inherent Duration
(In this example, 4 seconds.)
Simple Duration
(Inherent duration modified by dur | end.)
Active Duration
(Simple duration plus repeat modifiers.)
Rendered Duration
(Active duration plus fill behavior.)

```
<audio src="x.rm" />

<audio src="x.rm" dur="6s" />

<audio src="x.rm" dur="6s"
     repeatCount="2" />

<audio src="x.rm" dur="6s"
     repeatCount="2" fill="freeze" />
```

Figure 5: Various SMIL 2.0 duration concepts.
The *inherent duration* is the duration of the media object, if any. The *simple duration* is the *inherent duration* modified by the *dur* attribute. The *active duration* is the *simple duration* modified by the *repeatCount* (and *repeatDur*) attributes. The *perceived duration* is the visual behavior of an element after its *active duration*, before its parent time container ends.

parent time container ends — for discrete media, this will be the image or text, for continuous media, it will be the last frame/sample. The following:

```
  <par dur="10s">
    <img begin="3.5s" fill="freeze" src="..." />
  </par>
```

will display an image 3.5 seconds after the parent *par* begins. The active duration of the object will be 0 seconds, but the object will remain visible for 6.5 seconds (until the parent *par* ends). Two values of *fill* define a rendering duration that extends beyond the parent: *transition* and *hold*.

As in SMIL 1.0, if a *par* has multiple children active, it can specify that the entire *par* ends when the first child ends, when the last child ends or when a named child ends. This is done via the *endsync* attribute. (The default is: `endsync="last"`.)

SMIL 2.0 has two new attributes that provide extra duration control: *min* and *max*. These attributes, which developed out of the SMIL 2.0/MPEG-4 integration work, can be used to define a lower or upper bound on the active duration, regardless of that element's timing characteristics.

## *What goes around, comes around...*

The default behavior of all elements is that they play once, for either an implicit or explicit duration. The SMIL 2.0 *repeatCount* attribute allows an iteration factor to be defined that acts as a multiplier of the object's simple duration. A special value *indefinite* is used to specify that an element is to be repeated continually until the parent time container ends. Although it is tempting to define this as "forever," this is not correct: the parent time container defines the context of *indefinite*. The *repeatDur* attribute defines a duration for all of the repeated iterations.

## *Synchronization behavior*

In a perfect world, all of the defined timing in a specification would be implemented perfectly by a SMIL 2.0 player. Unfortunately, the world is not only imperfect — it is also unpredictable. In order to provide some measure of control in the face of unpredictably, SMIL 2.0 provides three high-level synchronization control attributes: *syncBehavior* allows a presentation to define whether there can be slippage in implementing the composite timeline of the presentation; *syncTolerance* defines how much slip is allowed; *syncMaster* allows a particular element to become the "master" timebase against which all others are measured.

### *XML integration*

When used in a native SMIL 2.0 document (one in which the outer XML tag is `<smil>`), the nature and meaning of various timing elements is clear. When integrating SMIL timing into other XML languages, a mechanism is required to identify timing containers. The SMIL 2.0 specification does this using the *timeContainer* and *timeAction* attributes.

### *Events and Hyperlinking*

In the normal course of processing, the activation hierarchy of a SMIL 2.0 document determines the rendering of document elements. The user can influence the elements selected by using SMIL 2.0 *events*. The event architecture allows document components that are waiting to be activated or terminated to actually start or stop. There are several uses of events allowed in SMIL 2.0, but perhaps the most important new semantic in the language is the combination of events and the *begin*/*end* attributes. In further combination with the *excl* element, events provide a very powerful mechanism for conditional content activation.

SMIL 2.0 also supports a rich hyperlinking architecture. Unlike links in (X)HTML, the fundamental concept of the SMIL link is that it models a *temporal seek* in a presentation. Rather than simply activating a target element, the target play state that is activated is identical to the state that the presentation would have been in if the target point had been arrived at 'naturally'. (One exception is that all event-based activation is ignored.) This means that all nodes temporally between the source and destination of the link need to be evaluated to see if they would have contributed to the final target play state. The temporal seeking and activation facility allows very polished presentation construction — but its implementation in the player is not for the faint-hearted!

## Content Control Elements and Attributes

One of the major innovations of SMIL was support for conditional content via the *switch* element.

```
<switch>
  <video src="..." systemBitrate="115200" />
  <seq systemBitrate="57344">
      <img src="img1.png" dur="5s" />
      ...
      <img src="img12.png" dur="9s" />
  </seq>
  <text src="desc.html" dur="30s"/>
</switch>
```

In this fragment, a video object is rendered if the system bitrate (actual or configured) is set to 112K or above. If the bitrate is 56K or above — but below 112K — a sequence of images is shown instead. If no other element had been selected in the *switch*, the text object is shown.

One of the limitations of the SMIL 1.0 switch element was that it only allowed selection based on pre-defined list of "system" attributes. SMIL 2.0 extends this notion with a user-defined set of test attributes: *custom test attributes*. The custom test attributes can be defined by the author and selected by the user directly or indirectly via the player. SMIL 2.0 also allows test attributes to be used "in line": that is, outside the switch element. Any media object reference containing a system or custom test attribute will be evaluated as if it were wrapped into a single-element switch.

To a first approximation, both the event mechanism and the content control facilities provide a means for dynamic selection of objects in a presentation. The basic difference between these facilities is that the event mechanism works on objects that have been recognized by the SMIL 2.0 scheduler, while the content control facility determines which object the scheduler gets to evaluate. The actual selection process associated with the conditional control primitives may be static or dynamic. That is, the selection may be done at parse time (when the document is loaded) or at each iteration through the document. The specific selection policy is a property of the SMIL player.

## Transitions and Animation

SMIL 2.0 significantly extends the facilities available for performing local operations on media objects in a document. Two of the most visible of these facilities are support for *transitions* and *animation*.

SMIL’s transition support allows a set of basic transitions to be defined as part of the SMIL *head* element, and then to instance one of the available transition types as an input or output transition on a media object. For example, consider the following SMIL 2.0 fragment in Figure 6.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<smil xmlns="http://www.w3.org/SMIL20/Language">
  <head>
    <layout>
    ...
    </layout>
    <transition id="fade" type="fade" dur=”1s”/>
    <transition id="push" type="pushWipe" dur=”0.5s”/>
  </head>
  <body>
    ...
    <img src=”...” transIn="fade"/>
    ...
    <video src=”...” transOut="push"/>
    ...
  </body>
</smil>
```

Figure 6: *Transitions architecture in SMIL 2.0.*
The transitions are defined in the head section and referenced in the body section as either input or output transitions.

Transitions can be applied to all visual media, or collections of media. Each transition can have certain timing properties (duration) and other transition-specific properties (direction).

Animation in SMIL 2.0 comes in two flavors. First, there are animations that apply to attributes and elements in the SMIL presentation. This includes animating the position of a rendering region or a background color. A second type of animation support is the SMIL 2.0 Animation Specification, which gives generalized temporal animation support for integration into XML languages.

## Layout

Unlike HTML, which uses an indirect layout model via CSS, SMIL also supports a direct layout model for managing the visual and audio rendering space associated with a presentation. In SMIL 2.0, the SMIL basic layout mechanism is supported with elements and attributes that allow layout to be specified as a hierarchy of rendering regions, and to be able to support multiple top-level presentation windows. The support for hierarchical layout is especially important when integrating animation into a presentation: content that is logically grouped together can be moved in concert by animating the position of the common parent region. Multiple top-level windows allow a single presentation to segment control and rendering operations in different parts of the screen in a coordinated manner.

Another enhancement to SMIL 2.0 layout is the support for sub-region positioning.This facility allows an object to be placed at a particular X,Y offset within a region or aligned to a *registration point*. (In SMIL 1.0, all objects were placed at the top left corner of a region.) The placement occurs in-line, as part of the media object reference. SMIL 2.0 also supports an alignment mechanism for content in regions; this allows a set of images of varying size to be centered at a specific point in a region. SMIL 2.0 also allows multiple objects to be active in a region simultaneously, relaxing a restriction from SMIL 1.0.

*(This chapter was adapted from the book* SMIL: Interactive Multimedia on the Web*, by Lloyd Rutledge and Dick Bulterman. A version of this chapter also appeared in an article in IEEE Multimedia Magazine, by Dick Bulterman, founder of Oratrix.)*

# GR*i*NS Edition Comparison Information

## GR*i*NS Packaging Information

The GR*i*NS Editor for RealONE is one member of a family of editing products for the SMIL 2.0 standard. This section provides a feature differentiation and compatibility matrix for the GR*i*NS/RealONE and GR*i*NS/Pro editions.

The features of each product are subject to change without notice.

| ***GRiNS Feature Table*** | **GRiNS / Pro** | **GRiNS / RealONE** |
|---|---|---|
| **Standard Features** | | |
| Structured Timeline Editing | ☑ | ☑ |
| Layout Editor | ☑ | ☑ |
| Animation Editor | ☑ | ☑ |
| Transition Selector/Editor | ☑ | ☑ |
| Source View/Editor (see below) | ☑ | ☑ |
| | | |
| Full GUI-based editing | ☑ | ☑ |
| Design from Templates | ☑ | ☑ |
| Create custom presentations | ☑ | ☑ |
| Import existing presentation | ☑ | ☑ |
| Publish to popular players | ☑ | ☑ |
| | | |
| Create par/seq/excl/switch groups | ☑ | ☑ |
| Integrate audio/video/text/images | ☑ | ☑ |
| Drag&Drop editing | ☑ | ☑ |
| Infinite Undo Stack | ☑ | ☑ |
| Create (timed) links | ☑ | ☑ |
| Insert events | ☑ | ☑ |
| | | |
| Preview Presentation | ☑ | ☑ |
| Preview Individual Node/Group | ☑ | ☑ |
| Preview from any Node/Group | ☑ | ☑ |
| Model Network performance | ☑ | ☑ |
| Preview for different bitrates | ☑ | ☑ |

| *GRiNS Feature Table* | GRiNS / Pro | GRiNS / RealONE |
|---|---|---|
| **New Presentations:** | | |
| From Templates | ☑ | ☑ |
| Custom Document | ☑ | ☑ |
| Create New Template | ☑ | -- |
| **Import:** | | |
| From Existing SMIL-2 | ☑ | ☑ |
| From Existing SMIL-1 | ☑ | -- |
| From Existing RP-6/7/8 | ☑ | ☑ |
| **Publish:** | | |
| To SMIL-2 (Generic SMIL-2) | ☑ | -- |
| To RealONE | ☑ | ☑ |
| To SMIL-1 | ☑ | -- |
| To RP-6/7/8 | ☑ | -- |
| To HTML+Time | ☑ | -- |
| **Publish & Prune to Profile** | | |
| Based on Bitrate | ☑ | -- |
| Based on Language | ☑ | -- |
| Based on Captions | ☑ | -- |
| Based on Any Test Attribute | ☑ | -- |
| **Convert** | | |
| RealPix to SMIL 2.0 | ☑ | ☑ |
| SMIL 2.0 to RealPix | ☑ | -- |
| Media Objects to RM | ☑ | ☑ |
| Media Objects to WMP | ☑ | -- |
| **Content** | | |
| Link to Installed 3rd Party Editors | ☑ | ☑ |
| Embedded image trimmer | ☑ | -- |
| TWAIN object import | ☑ | -- |
| **Toolbars** | | |
| Standard Configuration | ☑ | ☑ |
| Customize Configuration | ☑ | -- |

| ***GRiNS Feature Table*** | **GRiNS / Pro** | **GRiNS / RealONE** |
|---|---|---|
| **UI Features** | | |
| Show Asset Thumbnails | ☑ | ☑ |
| Customize Asset Thumbnails | ☑ | -- |
| Show Playable in Switch | ☑ | ☑ |
| Customize ‘Show Playable’ | ☑ | -- |
| **Structured Timeline View** | | |
| Standard Timeline View | ☑ | ☑ |
| Customized Timeline View | ☑ | -- |
| Optimized Structured Timeline | ☑ | -- |
| **Source View** | | |
| View Source | ☑ | ☑ |
| Edit Source | ☑ | -- |
| **Animation Editor** | | |
| Child Animations on Nodes | ☑ | ☑ |
| Animations on Regions | ☑ | ☑ |
| Full Animation Elements | ☑ | -- |
| **Exclusive Nodes** | | |
| Create EXCL | ☑ | ☑ |
| Create PRIORITY CLASS | ☑ | ☑ |
| **Transitions** | | |
| Use Standard GR*i*NS set | ☑ | ☑ |
| Edit Standard GR*i*NS set | ☑ | ☑ |
| Define new transitions | ☑ | -- |

| ***GRiNS Feature Table*** | **GRiNS / Pro** | **GRiNS / RealONE** |
|---|---|---|
| **Switch Support** | | |
| Specify Bitrate | ☑ | ☑ |
| Specify Language | ☑ | ☑ |
| Specify Captions | ☑ | -- |
| Specify Audio Desc | ☑ | -- |
| Specify Subtitle/Overdub | ☑ | -- |
| Specify Screen/Depth | ☑ | -- |
| Specify CPU / OS | ☑ | -- |
| Specify Player/Req/Comp | ☑ | -- |
| **Custom Tests** | | |
| Play Custom Tests | ☑ | -- |
| Define Custom Tests | ☑ | -- |
| **Tools** | | |
| Summary Bandwidth Usage | ☑ | ☑ |
| Detailed Bandwidth Strip | ☑ | ☑ |
| Align/Distribute Nodes in Layout | ☑ | -- |
| **Layout Support** | | |
| Define root-layout via UI | ☑ | ☑ |
| Edit root-layout numerically | ☑ | -- |
| Define multiple topLayout via UI | ☑ | -- |
| Edit topLayout numerically | ☑ | -- |
| Define regions via UI | ☑ | ☑ |
| Edit regions numerically | ☑ | -- |
| Define reg hierarchy via UI | ☑ | ☑ |
| Edit reg hier. numerically | ☑ | -- |
| Define sub-regions via UI | ☑ | ☑ |
| Edit sub-region numerically | ☑ | -- |
| Select regPoints via UI | ☑ | ☑ |
| Define regPoint numerically | ☑ | -- |

| ***GR*i*NS Feature Table*** | **GR*i*NS / Pro** | **GR*i*NS / RealONE** |
|---|---|---|
| **Timing Specification** | | |
| Specify Duration | ☑ | ☑ |
| Specify EndSync | ☑ | ☑ |
| Specify Fill | ☑ | ☑ |
| Specify repeatCount | ☑ | ☑ |
| Specify repeatDuration | ☑ | ☑ |
| Specify Min | ☑ | -- |
| Specify Max | ☑ | -- |
| **Begin/End Specification** | | |
| Delay / NodeID / Indefinite | ☑ | ☑ |
| AccessKey / WallClock | ☑ | -- |
| Event | ☑ | ☑ |
| X-Path Based Events | ☑ | ☑ |
| Restart Semantics | ☑ | -- |
| **Media** | | |
| via Drag&Drop | ☑ | ☑ |
| via local name | ☑ | ☑ |
| via URL | ☑ | ☑ |
| ClipBegin / ClipEnd | ☑ | ☑ |
| Event Sensitivity | ☑ | ☑ |
| Erase | ☑ | -- |
| **Synchronization Behavior** | | |
| Specify Sync Default | ☑ | -- |
| Specify Sync Behavior | ☑ | -- |
| Specify Sync Master | -- | -- |
| **Productivity Enhancements** | | |
| Specify Default Layouts for Groups | ☑ | ☑ |
| Specify Default Timing for Groups | ☑ | -- |
| Specify Default Fill for Groups | ☑ | -- |
| Specify Default Transitions | ☑ | -- |
| Merge Nodes with Parents | ☑ | -- |

| ***GRiNS Feature Table*** | **GR*i*NS / Pro** | **GR*i*NS / RealONE** |
|---|---|---|
| **Linking** | | |
| Create whole-node anchors | ☑ | ☑ |
| Create partial-node anchors (X/Y) | ☑ | -- |
| Create timed anchors | ☑ | -- |
| Create external anchors | ☑ | ☑ |
| Create internal anchors | ☑ | -- |
| Create Link to RealONE Browser Window | ☑ | ☑ |
| Create Link to RealONE Context Window | ☑ | ☑ |
| **Events** | | |
| Create short-cut events | ☑ | ☑ |
| Create/edit begin events | ☑ | ☑ |
| Create/edit end events | ☑ | ☑ |

NOTE:

1. The available versions of the GR*i*NS/RealONE products are updated regularly. Please consult the following Web URL for the most recent version of the GR*i*NS product matrix at:
   *http://www.oratrix.com/*

# GR*i*NS/RealONE Quick Reference Information

## SMIL Compliance Information

GR*i*NS/RealONE supports the entire SMIL 2.0 specification, although not all features are previewed during editing. Documents that make use of these constructs are parsed correctly, but the features are ignored during rendering.

## Supported Media Table

The following chart gives a listing of the media types supported by the Win32 versions of GR*i*NS Editor for RealONE:

| MIME type | Extensions | Windows 98SE/2K/XP |
|---|---|---|
| audio/basic | au | yes |
| audio/x-aiff | aiff, aifc, aif | yes |
| audio/x-wav | wav | yes |
| image/jpeg | jpeg, jpg | yes |
| image/png | png | yes (2) |
| image/tiff | tiff, tif | yes |
| image/x-portable-anymap | pnm | no |
| image/x-portable-bitmap | pbm | no |
| image/x-portable-graymap | pgm | no |
| image/x-portable-pixmap | ppm | no |
| image/x-rgb | rgb | yes |
| image/x-xbitmap | xbm | no |

| MIME type | Extensions | Windows 98SE/2K/XP |
|---|---|---|
| | bmp | yes |
| | ras | yes |
| | tga | yes |
| video/mpeg | mpeg, mpg | yes |
| video/quicktime | qt | yes |
| video/x-msvideo | avi | yes |
| video/x-sgi-movie | mov | no |
| text/html (5) | html, htm | no |
| text/plain | txt | yes |

Notes
1. Uncompressed WAV only.
2. Support seems to be somewhat buggy.
3. Not all encodings supported.

Each of these formats is converted to the appropriate RealSystem datatype. For highest quality rending of a final presentation, we recommend converting some datatypes to RealMedia before inserting them in a presentation, if possible.

## RealONE Media Conversion

The following chart describes the levels of support provided in the GR*i*NS/RealONE version for the listed RealMedia data types used in the RealNetworks RealONE Player:

| RealMedia | Extension | Importable | Auto-Generated |
|---|---|---|---|
| RealAudio | ra, rm | yes | yes |
| RealVideo | rm | yes | yes |
| RealText | rt | yes | no(1) |
| RealPix | rp | yes | no |

Notes

1. GR*i*NS/RealONE provides support for the automatic generation of simple RealText documents from immediate strings in the Editor, but it does not at present provide full RealText editing facilities. This is expected in a future release.
2. GR*i*NS/RealONE can import and convert most existing RealPix files. Limited support is available for creating RealPix from SMIL 2.0 constructs for users wishing backward compatibility.

## References and Links

Please see the Links section of the GR*i*NS/RealONE Web site (*www.oratrix.com/GRiNS*).

# Where Next?

The GR*i*NS Editor for RealONE *Quick Start Guide* provides you with a general overview of GR*i*NS and a particular introduction to GR*i*NS/RealONE.

Once you have installed GR*i*NS you can either set out on your own in the process of creating multimedia presentations, or you can save your time and energy by using a set of simple and straight-forward GR*i*NS/RealONE templates. Instructions for using these can be found in the GR*i*NS/RealONE *Template Guide*, which is available in on-line (HTML) or off-line (PDF) formats. Consult the documentation provided with your GR*i*NS/RealONE Editor access key to find the current set of templates for your version of GR*i*NS.

You should also consult the RealONE IQ Production Guide, available from RealNetworks, Inc. This guide provides an overview of the facilities supported by the RealONE player. It also provides a comprehensive introduction to the concepts of streaming media and details on the SMIL 2.0 language.

The GR*i*NS Web site (*www.oratrix.com/*) provides a set of *release notes* and *issues reports* for each version of the GR*i*NS Editor and Player releases. This site also contains a publicly-available version of the GR*i*NS Frequently-Asked Questions list and errata to this and other GR*i*NS publications. If you purchased GR*i*NS, you will have had the opportunity to enroll for our automatic notification service for the version(s) of GR*i*NS you acquired.

We recommend that you use the Check for GRiNS update button in the Help menu. This version will poll the GR*i*NS Web site to see if an updated version of GR*i*NS exists.

We welcome your comments, criticisms, compliments and suggestions. You can reach us at: *grins-documentation@oratrix.com*.

We hope you find working with GR*i*NS a productive experience!