GRiNS Tutorials Guide

GR*i*NS for SMIL

Editor/Player Version 1.0

Windows, Macintosh and UNIX

GR*i*NS for SMIL v1.0 Tutorial Guide for Windows, Macintosh and UNIX. August, 1999.

Important Notices

This is the *GRiNS Tutorials Guide* for Version 1.0 of GR*i*NS for SMIL. All of the information has been verified, but incremental product updates may impact part of this guide.

The *GRiNS Tutorials 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 reproducability, 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 for SMIL 1.0-win32-0 version (for Windows-95|98|NT-4). While the look of other versions of GRiNS are slightly different because of adhrence 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 for SMIL 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/SMIL 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 Tutorials Roadmap

Thank you for downloading the GR*i*NS for SMIL (GR*i*NS/SMIL) toolset for playing and (optionally) creating SMIL presentations. This publication will help you understand how GR*i*NS works, and how it can help you to create high-quality multimedia presentations for the Web easily and effectively.

Distribution Package Contents

Before you can create the presentations referenced in these tutorials, you must download and install a copy of the GR*i*NS for SMIL environment for the platform you use. You can obtain GR*i*NS via our electronic software resellers or via the GR*i*NS Web site: *http://www.oratrix.com/*GR*i*NS. A license is required to use the GR*i*NS for SMIL Editor.

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

- GR*i*NS *for SMIL Quick Start Guide*: an overview of the installation instructions for GR*i*NS and a quick tour of the basics of the GR*i*NS Environment.
- *Data*: a collection of data assets used in the *Quick Start* and the *Tutorial Guide* examples. You may use these when constructing your own presentations, or you may substitute your own favorite objects;
- *Templates*: a set of templates used in the GR*i*NS *Tutorial Guide* and which you may wish to use to build your own presentations;
- *Examples*: a collection of SMIL demonstrations;
- GR*i*NS-*Icons*: a directory containing Icons used by the GR*i*NS Editor;
- *Software:* depending on the distribution you downloaded, a GR*i*NS distribution for Windows-95/98/NT, the Apple Macintosh or UNIX.

The GR*i*NS Tutorials distribution package contains:

- GR*i*NS *for SMIL Tutorials Guide*: a step-by-step introduction to using GR*i*NS. (This is the guide you are now reading.)
- *Tutorials*: a directory containing example SMIL files used in the tutorials.
- *Assets*: a sub-directory of *Tutorials* containing media objects.

We recommend that you read the GR*i*NS *for SMIL Quick Start Guide* before going through the tutorials in this guide.

GR*i*NS Tutorials Guide

The GR*i*NS *for SMIL Tutorials Guide* will help you learn how to make SMIL presentations easily and quickly. It is divided into six tutorials:

1. *Building a Simple Presentation*: a basic introduction to the GR*i*NS Editor, showing you how to take a simple template and fill in data objects.
2. *Extending an Existing Presentation*: an overview of how slightly more complex presentations are structured and an overview of basic timing control within a GR*i*NS presentation.
3. *Working with Templates, SMIL Regions and* GR*i*NS *Resource Channels*: a description of how you can use GR*i*NS to build your own templates and on managing presentation resources (such as screen and audio spaces).
4. Using Transitions and RealMedia: an overview of the facilities in GR*i*NS for creating presentations specifically geared to RealMedia datatypes, such as RealPix, RealAudio, RealVideo, RealText, and RealFlash.
5. *Building Adaptive Presentations*: an overview of the facilities in GR*i*NS for helping you quickly build presentations that can adapt to the environments that your users may encounter.
6. *Hypermedia Support in* GR*i*NS: an overview of using the hypermedia support facilities available in GR*i*NS for links within objects, inside of a single SMIL document and across SMIL documents.

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

After you get experience working with GR*i*NS and SMIL, you should read the GR*i*NS *for SMIL User's Guide* and the GR*i*NS *for SMIL Reference Manual* for more detailed information on how SMIL and GR*i*NS can support your presentation development needs. Information on obtaining these manuals is provided with the GR*i*NS software distribution package.

Introduction

The GR*i*NS Editor for SMIL

This guide will help you to learn how to create SMIL documents using GR*i*NS.

SMIL is a multimedia description language that was defined (and is maintained and further developed) by W3C: the World-Wide Web Consortium. The GR*i*NS Player was used to help W3C debug the original SMIL standard, and the GR*i*NS Editor is the only visual authoring environment available that gives you access to the full power of SMIL.

Often, SMIL documents created using GR*i*NS will be delivered to users across the Internet via a streaming multimedia player, such as the RealSystem G2 player. While you don't need streaming media to define a SMIL presentation, placing your SMIL presentation on a streaming server and having your users access it via a streaming player (such as G2) will probably provide them with the most effective presentation over the current Internet.

You can also create local presentations using GR*i*NS — that is, presentations that do not need to be accessed via a streaming server — that can be played back using any SMIL compatible player (such as the GR*i*NS for SMIL Player or RealSystem G2). Local presentations are useful if all your users are on a single high-speed network, or if you don't expect wide distribution of your work. Local presentations are also useful if you are planning to distribute your presentation on a CD-ROM.

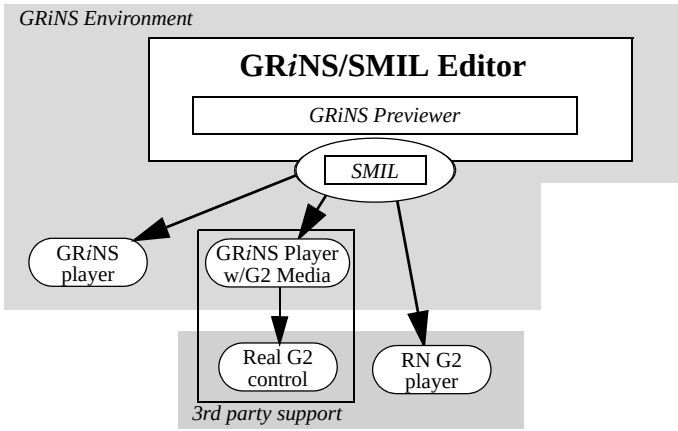
The GR*i*NS *for SMIL* Editor provides a single interface that allows you to create presentations to any SMIL compliant player. GR*i*NS uses an embedded previewer that helps you define you presentation quickly and correctly. In addition, you can edit any existing SMIL document (whether initially created with GR*i*NS or not).

While all SMIL-compliant players understand and process the SMIL-1.0 language, not all SMIL players support the same internal data types. Some SMIL players, such as RealSystem G2, have been optimized for streaming media formats like RealAudio, RealFlash, RealPix, RealText, and RealVideo. Other players, such as

the GR*i*NS/SMIL Player, allow non-streaming data types (such as HTML, MPEG video and audio, and a host of image formats).

Before you create a document, you should first develop a clear understanding of your target audience. If you audience is spread across the Internet and if you intend to use a streaming server to deliver your presentation, you should restrict the data types you use to those supported by RealNetworks. If you don't need streaming services, then you can also use other data types in your presentation.

The following illustration gives an overview of GR*i*NS/SMIL V1.0:



The GR*i*NS Editor for SMIL is a flexible interface for creating, maintaining and previewing SMIL presentations. You can target your presentation to the:

- *RealSystem G2 Player*: a full streaming SMIL-1.0 compliant player provided by RealNetworks (available for Windows and Macintosh);
- GR*i*NS/*SMIL Player with RealMedia Support*: the standard GR*i*NS/SMIL Player with extensions to render RealMedia data types using Real's G2 control (available for Windows and Macintosh);
- GR*i*NS/*SMIL Player*: a full non-streaming SMIL-1.0 compliant player (available for Windows, Macintosh and UNIX).

The choice of target player will depend on your application. To reach the widest possible audience, tailor your presentation to use only RealMedia data types and

the RealSystem G2 player. To be able to integrate HTML into your SMIL presentation, or to make a SMIL presentation that can also be viewed on a UNIX platform, use the GR*i*NS/SMIL Player. If the G2 player is installed, you can also use all of the RealMedia data types in your presentation — but only on platforms for which RealMedia is supported.

Notes:

1. In order to render Real's proprietary RealMedia data types (RealAudio, RealVideo, RealFlash, RealText and RealPix) — either via the GR*i*NS Player or via Real's G2 player — a version of G2 must be installed on the user's system. The RealSystem G2 player is not delivered with the GR*i*NS distribution. It must be obtained separately from Real Networks at: *www.real.com*.
2. Presentations created with the GR*i*NS/SMIL Editor can be rendered using any other SMIL-1.0 compliant player, but check the player's documentation to determine the data types it supports.

A Note on the Illustrations and Conventions Used in this Guide

GR*i*NS is available for Windows, Macintosh and UNIX platforms. This section discusses the impact that platform dependencies have on the content of the tutorials. Most of the dialogs and menus are identical on all three platforms, so this guide can be used for all three platforms. However, users of the Mac and UNIX versions should keep the differences summarized below in mind.

Microsoft Windows

In order to provide a readable tutorial set, this Tutorials Guide provides illustrations and examples drawn only from the Windows version. GR*i*NS adheres to most of the standard conventions for Windows software, except as noted in the text below. Please consult the GR*i*NS *Release Notes* for the most recent updates on product performance and conformance.

Apple Macintosh

Since the Mac has only a single-button mouse, you should substitute a CTRL-click selection where the text refers to a selection using the right mouse button. Contextual menus (selected with right-mouse on windows) will pop up on the Macintosh if you keep the mouse button depressed for about half a second.

The Macintosh does not have the concept of windows-within-windows, hence each open view has its own window. There is one point where this can be slightly confusing: it is possible to have a document open, but no views (open windows) on this document. The Window->Close directive closes a single view, and File->Close closes the document (and all views). The File->Open Documents menu shows the open documents and allows you to activate one of them.

The player toolbar is shown when you open the player view; there is no Editor toolbar. The keyboard shortcuts follow standard Macintosh practices.

SGI and SUN UNIX notes

Each view in the Editor under UNIX has its own window. Moreover, each window has a private menubar. The order of menus and commands within menus is the same as on Windows and Macintosh, but menus and commands that have no meaning within a certain view are omitted. Unix keyboard shortcuts follow standard UNIX conventions, and are listed in the menus.

Under UNIX, GR*i*NS uses a small separate window where you can open documents and quit the editor, and another small window per open document where you select which views to open, save the document, etc.

Tutorial 1: Creating a Simple Presentation

Overview and Goals

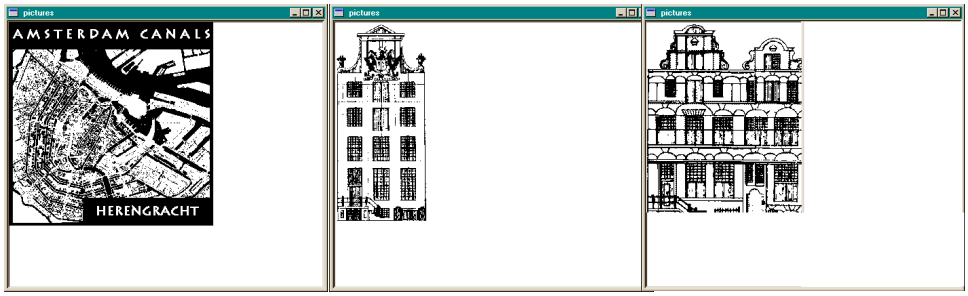
The purpose of this tutorial is to create a very simple SMIL file that consists of a sequence of objects that get rendered onto a default area of the screen. This presentation makes use of a GR*i*NS Template document.

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

1. Start the editor and select the *Basic-Slideshow* template;
2. Go into the Structure View;
3. Select each object in the template and define a image to be associated with each block;
4. Play the presentation

We do this step-by-step so that you can follow the process in great detail.

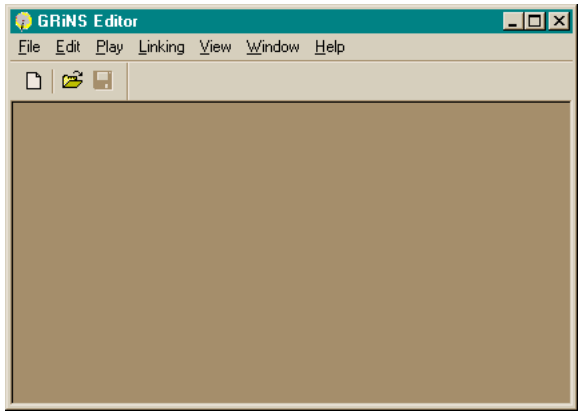
The result of building the first part of this tutorial is a series of three pictures displayed in the (player-dependent) default location on the screen. A thumbnail presentation preview is:



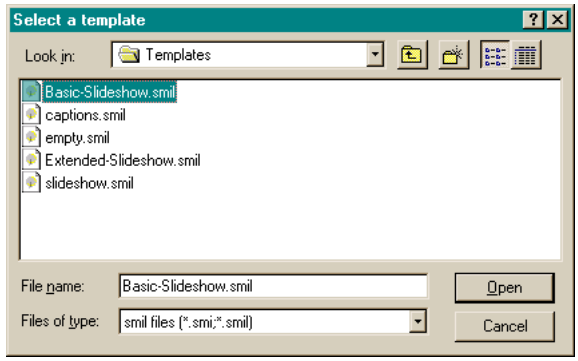
We close this tutorial with background information on GR*i*NS that will make your creation of template-based more effective.

Opening a Template File

If you have not already done so, start-up the GR*i*NS Editor; this will give you the following window:



Next, go to the File menu and select New, which allows you to select a GR*i*NS Template file. In this tutorial, select the *Basic-Slideshow.smil* file, as shown below:[1]

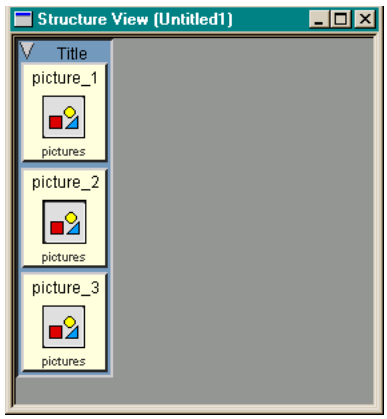


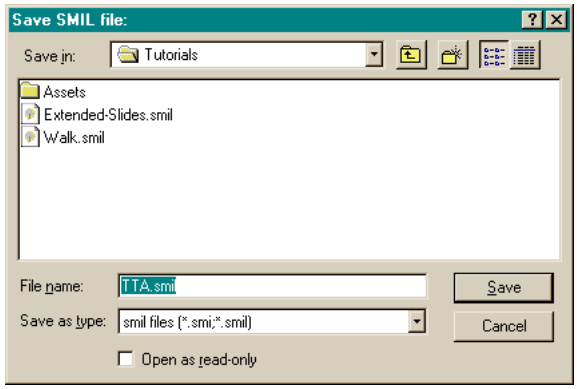
This will create a new document called *Untitled1.smil*.

1. On Macintosh versions of GR*i*NS, you may need to navigate to the Templates folder located in the GR*i*NS application folder.

Whenever you create or open a file, you see the Editor's basic view, shown at right. In this view, we see you blue outer container labelled *Title* and a series of three light-colored nodes called *picture_1*, *picture_2*, and *picture_3*. At the top-left of the structure container is a small triangle, pointed downward. Give the triangle a click to see one of GR*i*NS/SMIL's information hiding features.

(For SMIL fans: the blue structure container is a SEQ — or sequential — container.)





Before continuing with the document, save your work in the Tutorials directory. Do this by selecting Save as ... from the File menu and name your file *T1.smil*.

You can save your file anywhere, but it should be in the same directory as the root of all of your media assets.

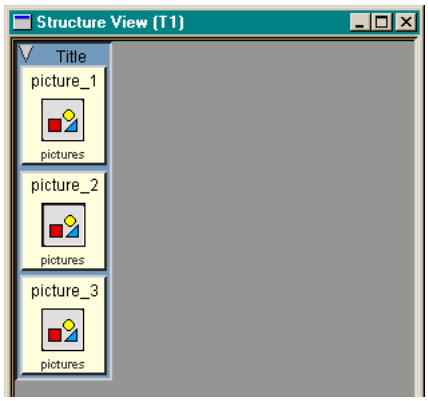
NOTES:

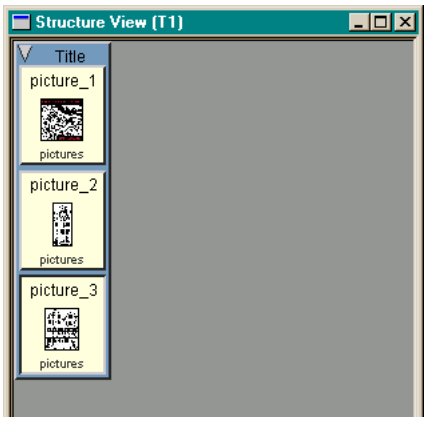
1. The opening screen assumes that you use the default configuration for GR*i*NS as distributed;
2. Some of the default dimensions of various GR*i*NS windows have been changed in the illustrations in these tutorials to make for a space-efficient presentation of GR*i*NS materials;
3. The colors used for the various GR*i*NS windows depend on the Windows color scheme installed for your system. This tutorial uses the Desert color scheme;
4. If you work on a multi-user system, make sure you don't overwrite files created by other users of this tutorial. On a multi-user system, each user should create their own copy of the Tutorials directory.

Inserting the Media Objects

The *Basic-SlideShow* template contains all of the information required to create a simple presentation *except* the actual images used. To get started quickly with GR*i*NS, do the following:

1. Open the *Assets* folder in the *Tutorials* directory (typically located in the GR*i*NS root directory; this is usually *c:/Program Files/Oratrix/*GR*i*NS *1.0* on Windows platforms);
2. Select the image *map.jpg* from the folder and drag it on to the light-colored box labelled *Picture_1*;
3. Select the image *h168.jpg* and drag it on to the box labelled *Picture_2*;
4. Select the image *h218.jpg* and drag it on to the box labelled *Picture_3*;





The illustration at left shows a 'filled' version of the structure view, containing icons for each of the images you have placed on the presentation.

You can now play the presentation by selecting Play from the Play menu.

That's it: you've created your first GR*i*NS/SMIL presentation! GR*i*NS allows you to perform some special functions that help make editing easier and more productive. Many of them are introduced in this tutorials guide.

NOTES:

1. On Windows versions of GR*i*NS, you need to close and re-open the Structure view after saving the file for the new name to appear as part of the Structure View's title bar.

Playing the Presentation

Playing the entire presentation

In the *Basic-Slideshow* template, each object has a default duration of 5 seconds. As a result, when you play the presentation, you see each object for five seconds, with each new object replacing the old.

To see this behavior, select Play. You can do this in three ways:
1. using the triangular Play button on the shortcut bar;
2. by selecting Play -> Play command from the menu bar
3. by selecting the Ctrl+P key combination.

Playing part of a presentation

GR*i*NS allows you to selectively play parts of a presentation.

To get a quick taste of what GR*i*NS can do, select the box labelled *Picture_1* (it contains the icon of the *map.jpg* image). Go to the Play menu and select Play Node. Once you do this, you get to see the Map image. Only that single node is played. You can use Play Node on any GR*i*NS node (data nodes or structure nodes).

You can also select Play From Node in the Play menu. For example, select the box labelled *Picture_2* in the Structure View. You now see both the second and third pictures — that is, all nodes are played starting at the point that you selected Play From Node.

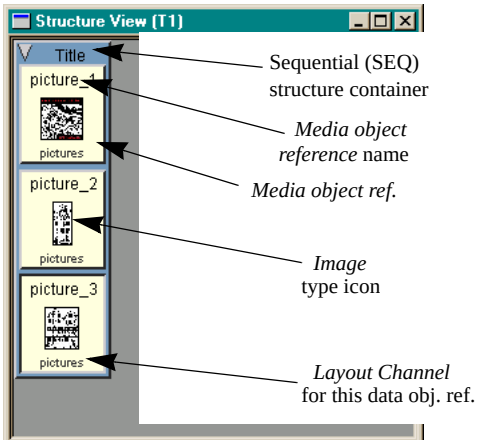
The ability to selectively activate and preview nodes or sub-parts of a presentation is a powerful tool that can help you fine-tune you presentation with a minimum of authoring effort.

NOTES:
1. In our examples, we encourage you to play the entire presentation. The GR*i*NS Editor also allows selective playing: you can either select one node (the Play node entry in the Play menu) or identify a starting point other than the beginning by selecting Play from here in the Play menu.

Additional GR*i*NS Features

Understanding the Structure View

The Structure View gives you an overview of the structure of any SMIL presentation. If we look at the contents of the presentation just created, the following detailed information is shown in the Structure view:



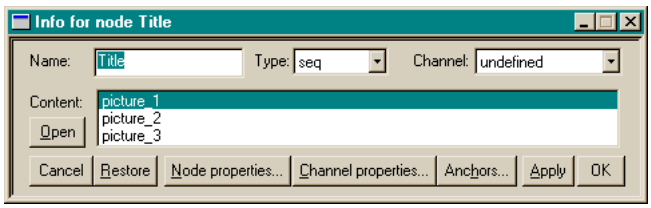
The presentation consists of three media object references (the light-colored boxes) and one structure node (the blue box labelled *Title*). Each of the media object references, labelled *picture_1*, *picture_2* and *picture_3*, refers to a place where you can insert an image in to the presentation. An object reference describes all of the uses of a particular object that is stored either locally or at a remote site. If one image is used multiple times, each use will be described by one image object reference. Each reference will describe the duration of the image, its location on the screen and other properties associated with the use (or instance) of this image. All objects have defaults associated with the type of media being rendered.

It is important to understand that media object references (the light colored boxes) have attributes that control how the media objects themselves are played during the presentation. Each of the structure elements also have their own attributes that can be used to tailor a presentation.

Changing attributes of GRiNS *Objects*

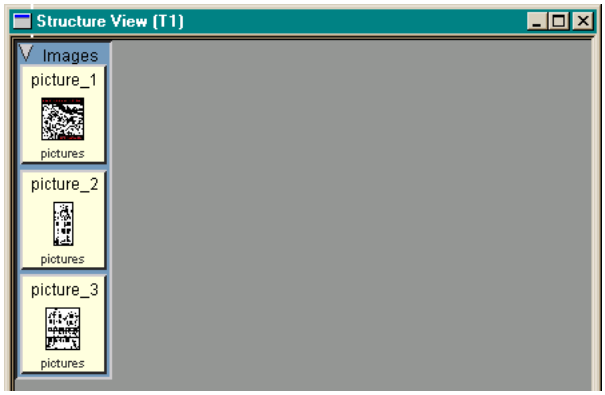
In order to get a feel for working with the attributes of objects, let's change the names of each of the objects shown in the presentation. This isn't strictly required, but can help make you SMIL code more readable by others — and it becomes important when you want to work with hyperlinks or if you want to do complex timing operations on your file.

Start by bringing forward the Structure view; if this view is obscured, you can always bring it forward via the Windows menu. Now, select the node named *Title* and bring up its Info property sheet using either the right mouse button or via the Edit menu. The Info sheet contains high-level properties of the object:

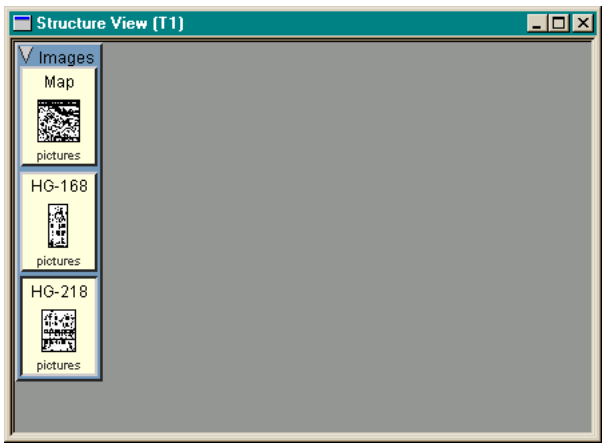


The Info sheet says: *Title* is a sequential node containing three children.

Change the name to *Images*, then click on OK. The resulting Structure view is shown, with the *Images* as the title of the structure node.



You can also rename the individual media objects to: *Map*, *HG-168* and *HG-218* by opening their respective Info sheets. The yields the following Structure view:



Closing Comments on Tutorial 1

The purpose of this tutorial was to acquaint you with the basic features of the GR*i*NS Editor. In this tutorial we worked primarily with the Structure view, which is the main editing view in GR*i*NS. We also used the Player view, which contained the actual presentation contents. GR*i*NS also supports other views of the presentation, such as the Timeline View and the Layout and Hyperlink views. You can always select the views you want open from the View menu; you can have as many open as you need, and you can move each view within the GR*i*NS overall canvas to help organize your editing session.

In the next tutorial, we'll revisit these features and show how you can also create more complex SMIL presentations easily by manipulating the structure of an application.

For now, take a break and give *Tutorial 2: Extending an Existing Presentation* a try when you are ready.

Tutorial 2: Extending an Existing Presentation

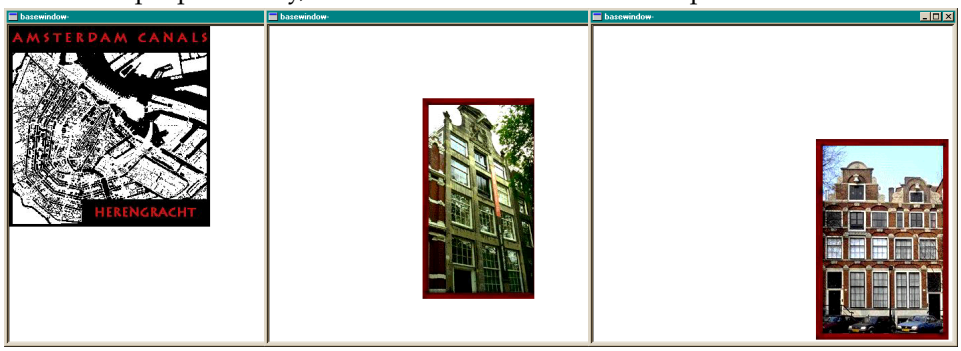
Overview and Goals

In this tutorial you will extend a SMIL presentation that displays a sequence of images in three separate regions on the screen by adding background music.

In order to edit this presentation, you:
1. start the GR*i*NS Editor and open the *Walk.smil* presentation;
2. preview the presentation;
3. learn to navigate through the structure of the presentation;
4. add some new nodes containing background music;
5. fine-tune the timing relationships between objects using property sheets;
6. further manipulate timing relationships using the GR*i*NS Timeline View; and
7. get a taste of manipulating the layout of the presentation via Layout Channels.

This tutorial assumes that you have the GR*i*NS Editor available, and that you have developed the basic skills illustrated in *Tutorial 1*. Some of that material will be reviewed here, but your time can be most productively spent if you look at Tutorial 1 before you complete this one.

The following sequence of images shows what the first part of the tutorial will yield. (Note: the audio is not shown.) Each of the images is bordered by a box showing the boundaries of the respective regions used. These are shown for illustrative purposes only, and are not shown in the actual presentation.



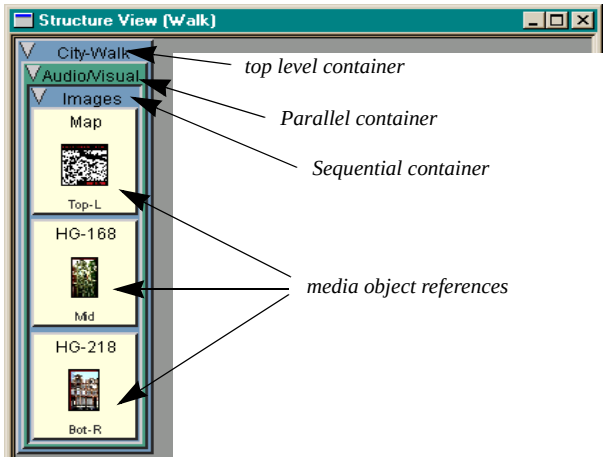
(a) use of Top-L Region *(b) use of Mid Region* *(c) use of Bot-R Region*

Creating the Presentation

In the first tutorial you opened a standard GR*i*NS template. In this tutorial, you open an existing presentation and expand it with new content. You can do this with any legal SMIL presentation — whether created by hand with a text editor or with someone else's editing product.

Open the Walk.smil presentation

Start-up the GR*i*NS Editor and open the *Walk.smil* presentation by selecting Open from the File menu, and browsing the *Tutorial* folder in the GR*i*NS system folder. When the document opens, the Structure View is activated. This will give you the following Editor view:



The structure in this view is the same as that in the first tutorial, except that:

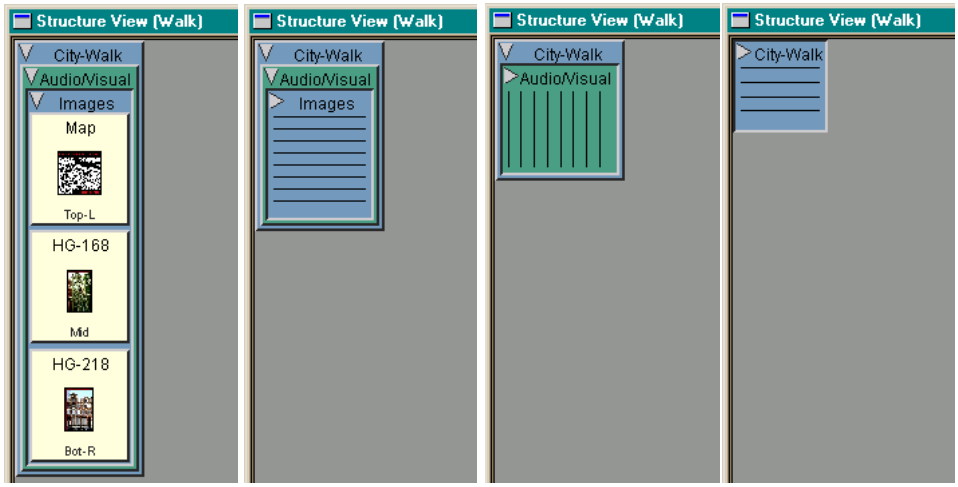
- the media objects are slightly different (they are color images)
- there are additional structure containers (*City Walk* and *Audio/Visual Show*.)

Once you open this presentation, you can edit and save it under its existing name or you can save it under another name (such as *Walk2.smil*). If you save it in another directory, make sure you also copy the *Assets* folder, since the presentation uses relative path names for media objects.

Revisiting the GR*i*NS *Structure View*

In the first tutorial, you saw that a blue structure box with a stack of three items was the GR*i*NS method of representing a SMIL sequential structure container (SEQ). In this document, you see the same structure in the container named Images. There is also an additional green structure container outside: a parallel (PAR) container labeled *Audio/Visual Show*. There is also an outer blue sequential container named *City Walk*.

In order to see how GR*i*NS can help you manage the complexity of the Structure view, click on the small triangle at the upper left corner of the *Images* node. This has the effect of collapsing the details of the node, leaving you with only a "closed" structure container. The results of closing the containers available are:



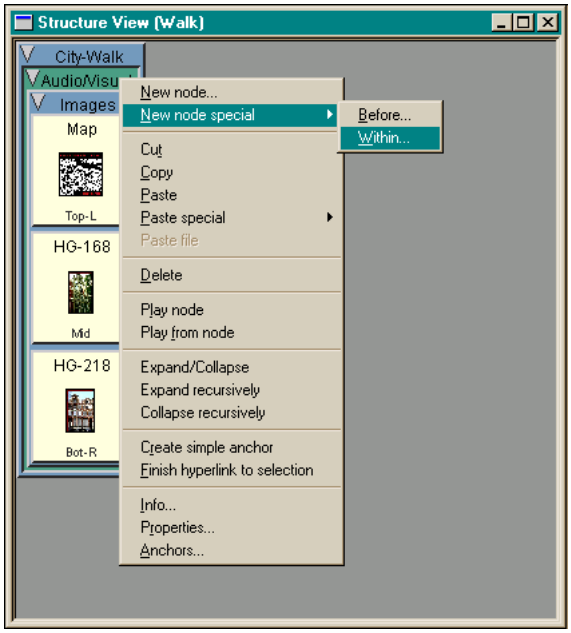
In this simple presentation, opening and closing boxes is not necessary, but this facility is really handy once you start to edit more complex SMIL presentations. But: enough about the future — let's get some work done!

The first thing you should do is to play the existing presentation. You can do this by selecting the VCR-like Play button on the tools bar or by selecting Play from the Play menu. You can also play individual nodes with Play Node or Play From Node.

Adding a new media object reference

A presentation like this short walking tour of a city is nice, but you can liven things up a bit by adding some music. In order to do this, you first need to tell the Editor that you want to define a new node in the presentation, and then fill the node with a particular media reference.

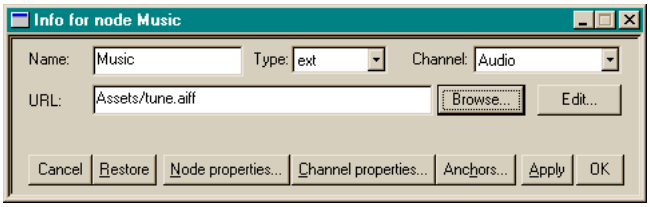
To create a new node, make sure the target structure container is open (by selecting the appropriate triangle), which in our case is the green node labelled *Audio-Visual-Presentation*. Now create a new node by going to the Edit menu and selecting New Node Special -> Within or by using the New Node Special -> Within shortcut under the right mouse button, as shown below:



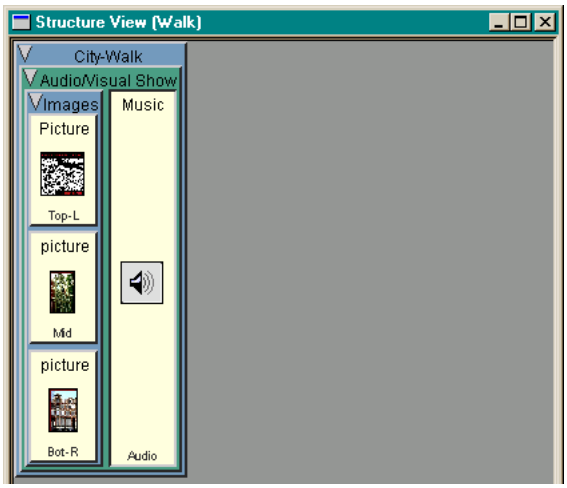
This operation says: in the currently selected structure node (in this case, a parallel node), add a new element within the node. We could have also placed the new node before or after the structure node by using the appropriate commands.

Whenever you create a new node, a *node creation dialog* pops up, in which the basic properties of this node can be defined in the node's Info sheet. Fill in the following properties, as shown below:

- *Name*: Music;
- *Type*: use the selector switch to choose ext (for external object reference);
- *Channel*: use the selector tab and choose *Audio* as the name in the pop-up box;
- *URL*: you can either use Browse to select the name of the media object, or you can use the drag-and-drop facility to put a data node on the media object reference.



If you've already used GR*i*NS drag-and-drop, try Browse and locate the file *tune.aiff* in the *Assets* folder. (You can, of course, simply continue to use Drag and Drop.) The result is:



The structure view shown above says: the presentation consists of one parallel node that contains two elements: a piece of background music and a sequential structure element that itself contains four images.

Once you get used to this representation, you can immediately see what happens in parallel and what happens sequentially in a presentation. The Structure View given in the presentation corresponds to the following SMIL structure:

```
<seq id="City Walk" >
   <par id=”Audio-Visual-Presentation” >
      <seq id=”Images” >
         <img id=”map” ... />
         <img id=”HG_168” ... />
         <img id=”HG_218” ... />
      </seq>
      <audio id=”music” ... />
   </par>
</seq>
```

Spend a few moments to understand how the SMIL code and the Structure View are related. Once you understand this, using GR*i*NS to make complex SMIL presentations is a snap!

Manipulating Structure and Data Objects

GR*i*NS allows you to manipulate both structure elements in a presentation and individual data objects. Why? GR*i*NS does this because being able to manipulate structure makes presentations easy to expand, edit and maintain. By separating structure from content, you can reuse the same structure dozens of times. We can also substitute and update the individual data objects without changing the base presentation. We can also copy and paste the structure from one presentation to another, something you’ll see in a later tutorial.

Previewing the Results

The presentation will display a sequence of three images (for five seconds each) in parallel with a short background audio track. You can see and hear the result by selecting Play presentation from the Play menu or by using the VCR-like icon on the shortcut bar. Try this now.

If you used our data objects, you will see three images in sequence (each displayed for 5 seconds, for a total of 15 seconds) displayed at top-left, middle and bottom-right on your screen, and you hear a bit of music in parallel that lasts for about 4 seconds. Having the music be this much shorter than the full presentation is not so nice, but it gives us something to fix later in this tutorial!

While you have just previewed the entire presentation, the GR*i*NS Editor also allows you to preview only parts of the presentation. For example, select the audio node, and then select Play node from the Play menu or with the right-mouse button. You now only hear the audio. You can do this for any simple data node, but you can also do it for any structure node. For example, select the structure node labeled Image Sequence once, and then select Play node from the Play menu. All of the images in the sequence (that is, the contents of the selected node) will be previewed, but the accompanying audio — which was not selected — will not.

Note

1. Be aware that the player window will be pushed to the background if the Structure View is accessed; you should get into the habit of moving the Structure View a bit to the right to see if the Player is being blocked.

Congratulations: you've made your second GR*i*NS-based SMIL presentation.

Fine-Tuning the Timing of a Presentation

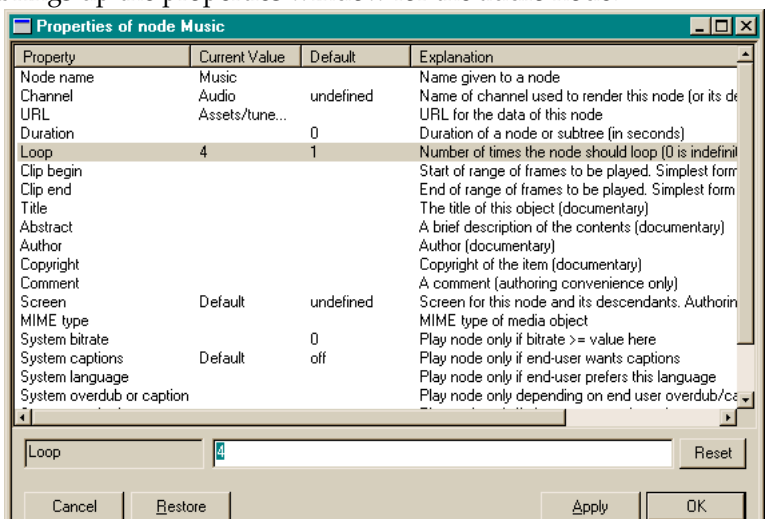
In this section, we show four simple editing operations that will start to develop your SMIL skills for extending the control among objects in a presentation:

1. you can tell GR*i*NS to loop your data object for a certain number of iterations;
2. for objects with a behavior like background music, you can set the loop to be determined by its context in the presentation;
3. in a parallel group, you can explicitly say which object controls the end of all of the members of the group;
4. you can use synchronization arcs in the Timeline view to control relative start times on objects in a structure container, and
5. you can change the assignment of objects to Layout Channels, and resize or move the Layout Channels to another area of the screen.

Each of these are discussed in the following sub-sections.

Setting an Object Reference to "Loop"

Open the Structure View and select the *Music* data reference by clicking on it once. Using either the right mouse button, or the Edit menu, select Properties, which brings up the properties window for the audio node:



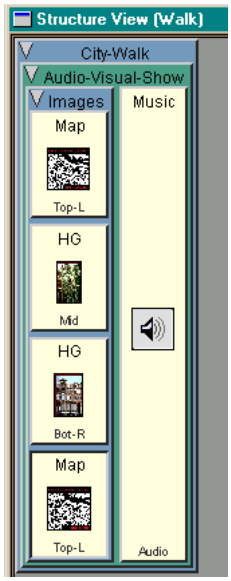
Using either the arrow keys or the mouse, select the line labeled Loop. In the attribute editing area at the bottom of the box, set the value to *4*. This tell GR*i*NS to loop the audio four time during this part of the presentation.

Now, preview the new presentation by selecting Play from the shortcut bar, the right mouse or the CTL-P shortcut. The audio will keep repeat until all of the images have been shown -- a total of 16 seconds.

> ***GRiNS Tip:***
> *Not all data types can be set to loop. For images or text (or any object that is discrete — that is, that has no 'natural' duration), looping does not make sense. For all continuous, time-based objects, looping makes sense, but not all players support looping functionality. For streaming media (such as RealAudio), looping is not supported, since it requires an entire stream to be buffered in the player.*

Setting an Indefinite looping period

SMIL give you lots of flexibility in controlling the duration of nodes, but using the kind of explicit control shown above can be dangerous if you want to reuse the structure of a presentation. For example, assume that you hav.e defined a Loop counter of *4* for an effective object length of 16 seconds.



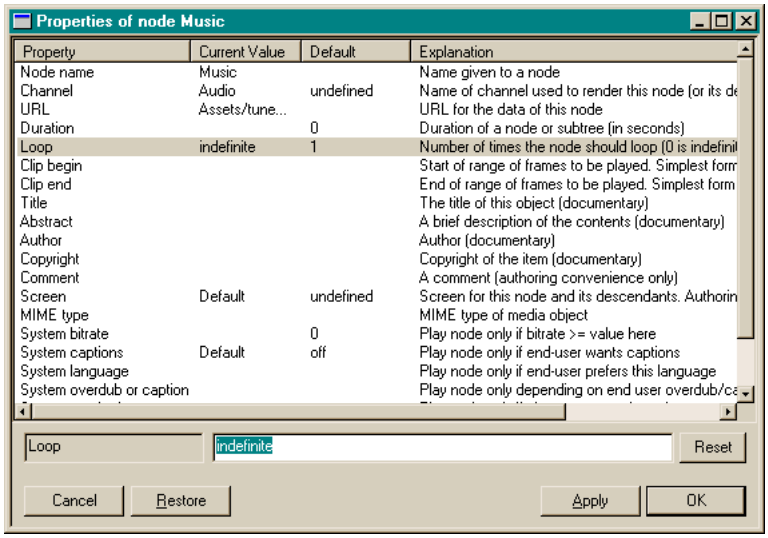
Now, edit the presentation as follows:

1. select the node labelled *Map* in the *Images* container
2. using either Edit -> Copy, the Copy function under the right mouse or CTL-C, copy the *Map* element.
3. select the last image in the three-image sequence (labelled HG-218), and do a Paste from the Edit menu, via the right mouse or using CTL-V.

This makes a copy of the node Map, so that the presentation now looks like the image at right.

If you now preview the presentation, you get a 20 second image show, but only 16 seconds of audio — not exactly what you would like!

You can solve this type of problem easily with GR*i*NS. First, open the property sheet for the *Music* object. This is shown below.



Now, replace the loop count of 4 with the word *indefinite*, as shown. What this says is: keep looping the object until the end of the object's parent container. The duration of the parent is determined by the length of all of the images, so the music loops until all the peer-level objects have been completed.

It is important to realize that *indefinite* is not the same as *forever*: the duration of an object with indefinite looping is determined by the object's context. When the context ends, the object ends.

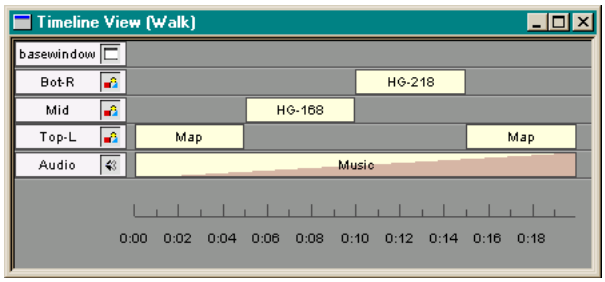
There are several other ways of influencing the timing of a node or of a structure container. We consider these later in this tutorial.

Advanced GR*i*NS Features

This section presents some of the more advanced features of the GR*i*NS editing system.

Introduction to the Timeline View

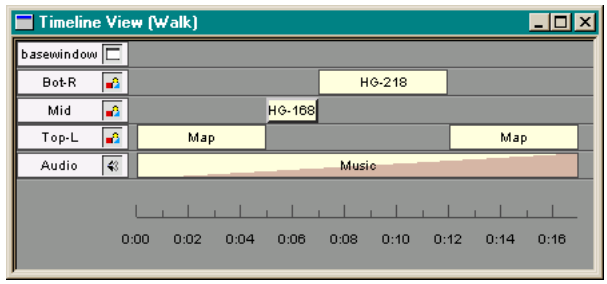
The Structure view shows the logical grouping of elements in a presentation. At runtime, the structural view is mapped to a presentation timeline. An indication of this timeline can be seen by looking at the Timeline View of a presentation. You can bring up this view by selecting it from the Windows menu. A picture of the timeline for this presentation is:

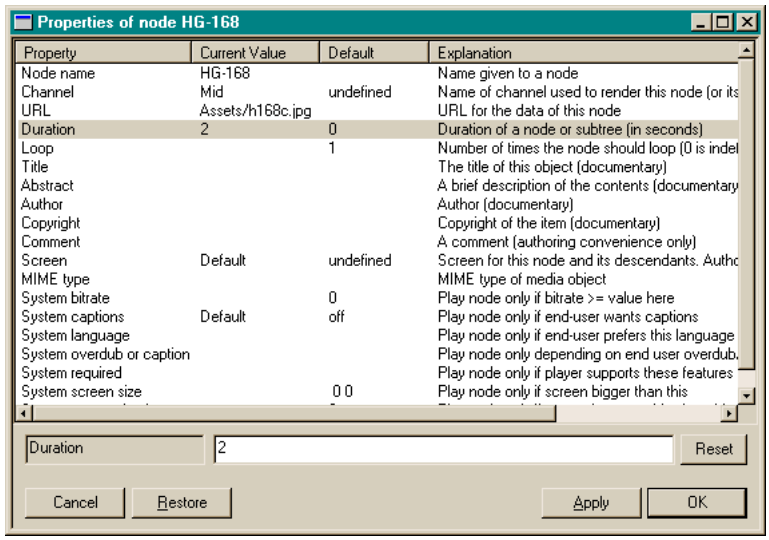


This view shows the *audio* object and *Map*, *HG-168*, *HG-218* and *Map*. Now, select *HG-168* and bring up its properties via the right mouse button. Find the *Duration* property and change it from 5 to 2 seconds.

The property sheet for this node (shown on the next page) contains all of the SMIL and GR*i*NS-related properties associated with this node. The use of most of these properties will be covered in other tutorials.

The result of this is that the length of the *HG-168* and *Music* objects are scaled automatically. The runtime behavior can be confirmed by previewing the node or the entire presentation using the Play options.





Notes:

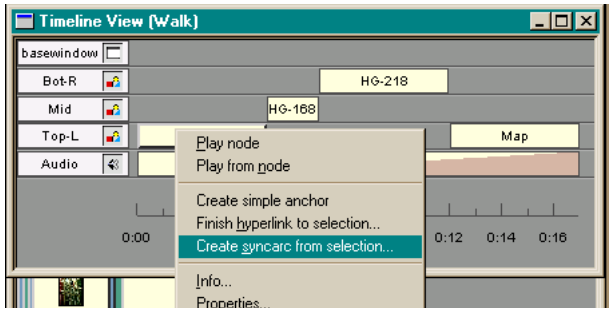
1. The Timeline displays the either the natural or effective duration of objects. In the example above, the effective length is shown, which is the length provided by the runtime environment. For some datatypes (where information on duration cannot be drawn from the definition), the natural length or a best-estimate is shown.

Fine-Tuning via Synchronization Arcs (Sync Arcs)

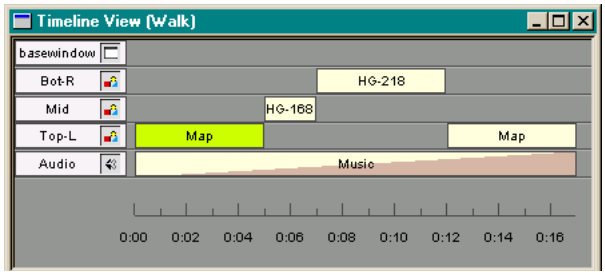
Let's suppose that you wanted to take our presentation-under-construction and make some fine-grained adjustments to timing. For example, let us assume that you want the audio fragment to begin slightly later than the first image. Perhaps you are doing this to balance the delivery load of the presentation, or maybe you just want to create a particular effect.

GR*i*NS allows for this fine-tuning via *synchronization arcs*, or *sync arcs*. Sync arcs provide a visual representation of special relationships between objects in the timeline. They are only available in the timeline, since they have to do with the execution state of a presentation rather than its logical state.

To create a sync arc, select a node in the timeline view. For example, select the picture of the map (named Map) and then, using either the right mouse menu or the Linking menu, select the create sync arc from location command:

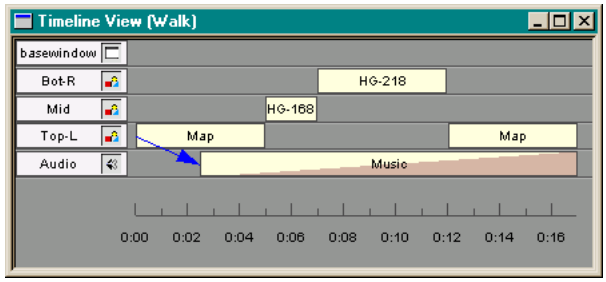


This will define the source part of a sync arc. The source is colored amber to highlight it in the Timeline View.



Once the amber node is shown, select the node named Audio. This will bring up the Sync arc properties box, as shown top right. The GR*i*NS default is to define a relationship between the end of the source to the beginning of the target. In our example, change *From* to *Begin* and give a delay of 2.5 seconds, as shown bottom right.

When the sync arc is accepted, the arc is drawn as a blue arrow between the start of *MAP* and the start of *Audio*. It says that the audio node will start 2.5 seconds after the start of Map:



Once a sync arc is created, you can edit its properties by selecting the arrow head associated with the arc. If you make changes, the timeline will be redrawn automatically for you.

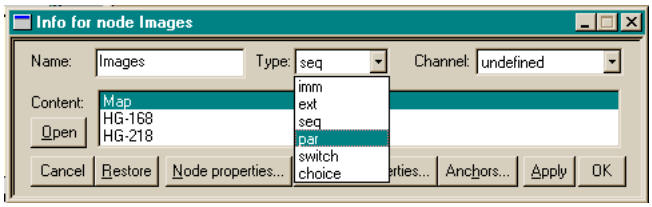
Notes:

1. SMIL places structural restrictions on the scope of sync arcs. In SMIL V1.0, both the source and target must be have a sibling relationship. This is expected to be relaxed in version 2.0.

More on Timing in Parallel and Sequential Structure Nodes

The examples in this tutorial have shown you how to add nodes and adjust timing in a presentation. We conclude with looking at manipulations on SMIL structure nodes.

Go to the Structure View, select the Images structure node and remove the second copy of *Map* from the presentation. Now, bring up the Info box. In the middle part of the top of this box is the Type selection box. If you highlight the alternatives, you get:



If you select *par* (as shown above) instead of the original *seq*, GR*i*NS will take the image sequence and turn it into a parallel structure group in which all of the images are displayed simultaneously.

If you confirm the selection of par and then preview the presentation, you will see that the images are played together. The images *Map* and *HG-218* are displayed for five seconds each, and *HG-168* for two seconds. The audio lasts as long as the longest image (instead of the sum of the image times).

A copy of the screen at the start of the presentation is shown on the next page.

.

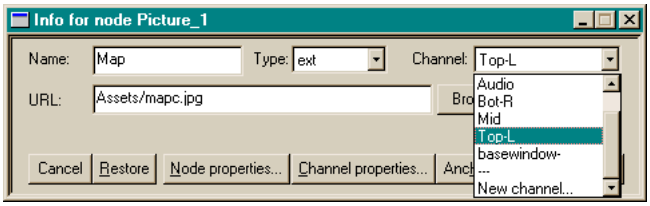
The presentation as it now stands has a delay defined for the start of the audio. This means that the *Map* image is shown for 5 seconds, but *Music* starts 2.5 seconds after the *Map* has been shown.

Selecting Layout Regions

All of the examples you have used in this tutorial play their output on to a Resource Channel, which is the GR*i*NS way of identifying which screen and audio resources are used to render the presentation. In the first tutorial, you used the default resources available, as defined in the SMIL standard. In the second tutorial, you used four Resource Channels that were defined as part of the *Walk.smil* presentation.

In the next tutorial, you take a deeper look at Resource Channels. To get a taste of this kind of presentation control, we end this tutorial with a simple introduction on how to associate any data object with its presentation space.

Open the Structure View and select the Audio-Visual-Show object. Go to its Info sheet and make sure that the object type is PAR (for parallel node). Confirm this choice with OK and then select first element in the *Images* — this is the *MAP* object. Next, bring up the Info box for this image by selecting Info from the Edit menu (or from under the right mouse button). In the upper right hand corner of the Info box is the Channel selector. When the list of available channels is exposed, you will see the following options:



If you select Bot-R, the *MAP* image will be displayed at the bottom-right part of the screen. Select Bot-R, confirm the selection and then preview the presentation using PLAY. If you also change the second image (*HG-168*) to display its output in Bot-R, all three images will be displayed in the bottom right part of the display.

Closing Comments on Tutorial 2

The purpose of this tutorial was to acquaint you with the notion of parallelism in a SMIL presentation and to let you get an idea of SMIL abilities to manage the flavor of a presentation by using looping and various forms of activation and duration control.

Unlike the first tutorial, this one also made use of the ability to explicitly place objects using GR*i*NS Resource Channels. The next tutorial looks more closely at the types of control you have via GR*i*NS layout.

For now, take a break and give Tutorial 3: *Working with Templates, SMIL Regions and GRiNS Resource Channels* a try when you are ready.

Tutorial 3: Creating Templates, Working With Regions and Using GR*i*NS Resource Channels

Overview and Goals

The purpose of this tutorial is to create the *Walk.smil* SMIL presentation that was used in Tutorial 2. The *Walk.smil* presentation contains references to four media objects and it makes use of three visual rendering areas on the screen. It also makes use of one audio rendering area. Since tutorials 1 and 2 show how to define and integrate media objects into a presentation, this tutorial will concentrate on defining SMIL Regions by using GR*i*NS Resource Channels.

You will perform the following steps in this tutorial:
1. Open the editor and select the *Empty.smil* template;
2. Define three Resource Channels, one for each of the visual regions;
3. Define the structure and media object referenced to be used in the template and define a Resource Channel for the audio object;
4. Assign default timing to nodes where appropriate;
5. Save the presentation for later use.

An Introduction to Layout Manipulation in GR*i*NS and SMIL

GR*i*NS helps you allocate and organize presentation resources by using a tool called a Resource Channel. A Resource Channel is a place where you can put information that is all managed using a common policy. Often, the Resource Channels will help you manage space on the display screen. In more complex presentations, Resource Channels may help you control an audio device (or one channel on a stereo audio device), or a printer, or a control/communication program or a special-purpose device.

In most initial examples of GR*i*NS/SMIL, it is convenient to think of a Resource Channel being the same as a SMIL *Region*. A SMIL *Region* is a rectangular area of the screen. It is defined by an X, Y base coordinate, a height and width (in pixels or as percentages of the available space) and a 'depth' coordinate (called Z) which tells you how individual objects are ordered on top of one another.

Where the SMIL *Region* provides a means to partition the presentation canvas, the Resource Channel in GR*i*NS provides a broader perspective on managing presentation resources. Using GR*i*NS Resource Channels, you can give individual collections of data objects a common resource control policy, which can help the Player decide how best to process your documents.

A general discussion of Resource Channels is beyond the scope of these tutorials. In this tutorial, we concentrate on using Resource Channels to manage the SMIL-defined properties associated with *Regions*. To simplify the text, we will use the name Channel to denote a GR*i*NS Resource Channel and a Region to denote the properties specific to a SMIL *Region*.

Layout and the Walk.smil presentation

In the first version of the Slideshow in Tutorial 1, we made use of the default rendering regions available with any SMIL presentation. In Tutorial 2, we made use of three SMIL regions, one of each picture. In this tutorial, we show you how to define and manipulate these regions, and how to customize them for special needs.

When you previewed the *Walk.smil* presentation, you saw that each of the three images were placed in different parts of the screen: the *Map* was placed in the upper left part of the display, the first house near the middle, and the second house in the bottom right. Each of these areas corresponds to one Channel. There was also a fourth Channel in the presentation: the background area that defines the size of the player window. The fifth resource used in the presentation was the audio device — this is a device that is not specifically controlled in SMIL presentations, but one you can influence when you use GR*i*NS.

In *Walk.smil*, each media object is placed on a separate channel. This is acceptable for small presentations, but it is typically not a good idea for larger presentations. GR*i*NS provides facilities for helping manage all the resources in the presentation, but you can help GR*i*NS by reusing resources when appropriate. You will see how to do this in the examples below.

In conventional HTML presentations, the user is not given very much control over the visual area in the presentation. This makes a lot of sense for the text-flow

documents that make up the majority of HTML presentations, but it is very frustrating to graphic designers and media presentation creators, who often want much greater control over the relative placement of objects on the display. Although several mechanisms exist in HTML to gain more control over the display space, using these often require a full implementation of CSS-2 (the optional Cascading Style Sheet extensions to HTML browsers). In order to provide a basic degree of control over SMIL presentations, the creators of SMIL decided to provide a simple, light-weight layout mechanism that any SMIL player could easily implement. This mechanism is called *SMIL Basic Layout*.

SMIL Basic Layout allows you to define a base window in which your presentation is played. The size of the base window can be defined in terms of pixels or in terms of external unit measure (typically millimeters or inches). Each of the SMIL Regions (if they are used) are placed inside the presentation's base window. Each Region is defined by an X, Y coordinate within the base window, followed by either an absolute size (in pixels, mm or inches) or a relative size (as a percentage of the base window).

When you define a presentation (or a presentation template), you specify how large a base window you want, and then how many independent regions you want to define to segment the space inside the base window. Of course, you don't need to specify any base windows or regions at all: in this case, you make use of the default layout properties of the Player you are using.

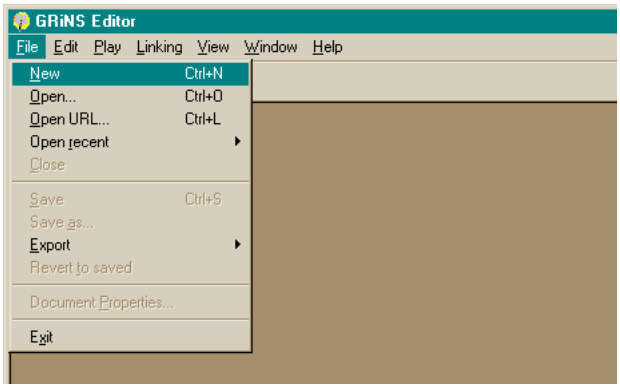
You specify a SMIL Base Window and the individual SMIL Regions via (Resource) Channels in GR*i*NS. The Channels provide a container for basic layout attributes and they provide an extra measure of control by allowing you to assign media types to individual SMIL Regions.

The GR*i*NS Editor also allows you to define *Layout Screens* (or simply Screens); Layouts let you group a number of Channels that you would like to have available during a particular part of a presentation. This allows you to change the look of your presentation over time -- something that most viewers will appreciate.

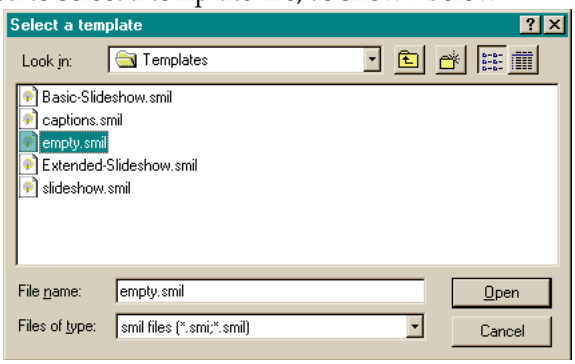
In this simple presentation, we will ignore the concept of a Screen. Instead, we make the decision to define three regions on the screen for displaying three images in sequence in different places.

Creating the Presentation

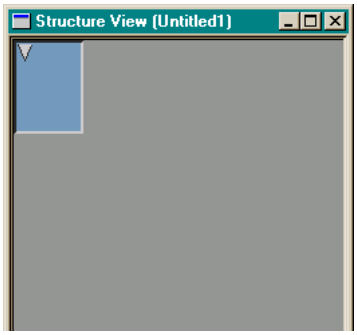
If you have not already done so, start-up the GR*i*NS Editor; go to the File Menu and select New.



You will be asked to select a template file, as shown below:



Select the template named *Empty.smil*. The Empty.smil template contains the SMIL boiler-plate information required to build a SMIL-based presentation.



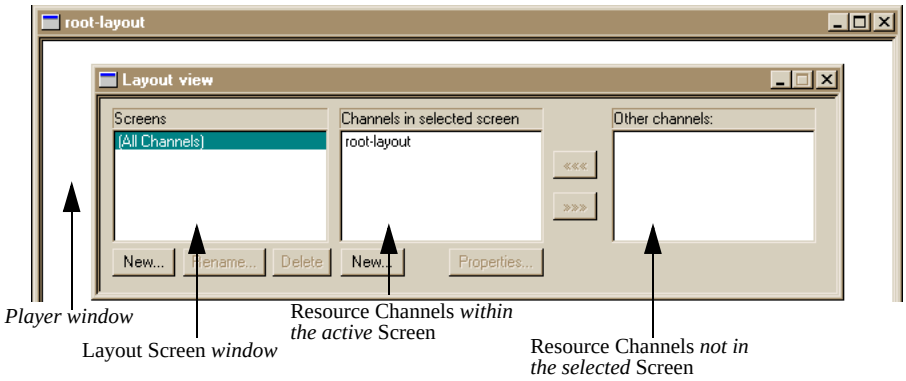
When you open the *Empty.smil* document, you get the default GR*i*NS Structure View, which in the empty document contains a single empty, unnamed sequential structure container.

Once the Empty.smil template is opened, you can start with the process of designing your application (or application template). There are many ways of building a SMIL presentation. You can take a *data-centric* view by first defining all of the objects in the presentation and their relative order or activation, or you can take a *presentation-centric* view by first designing your layout region(s), and then later assigning object to each of these regions. In most design cases, you may switch between the two approaches — first defining a few regions, then defining media objects, and then adding some more regions.

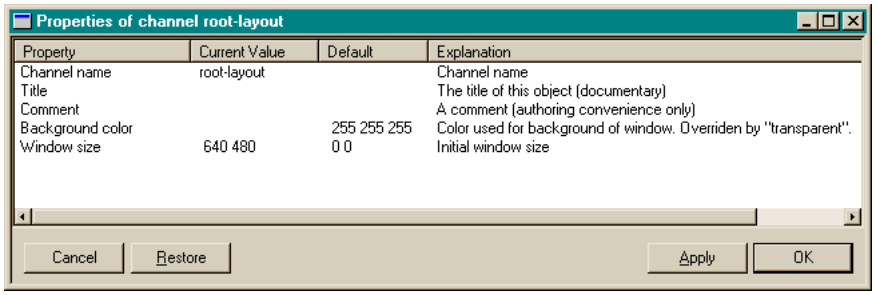
For the sake of simplicity, this tutorial takes the latter view: we start with designing a simple layout space and then identify the objects to be presented.

Define Regions for Displaying Presentation Components

In order to define and manage your presentation resources, you should open the Layout View from the Windows menu. The Layout view looks like this:



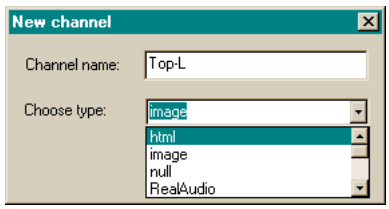
GR*i*NS provides you with a *root-layout*, which is the rendering canvas for the presentation. The default size of this canvas is 640 x 480 pixels. You can resize the window using the standard window resizing techniques for your computer (such as dragging the lower-right corner up or down, or left or right), or you can edit the various properties associated with the *root-layout* by selecting the name *root-layout* in the middle (Channel) window of the Layout View, and then selecting the Properties button. This brings up the following Property Sheet for the *root-layout*:



The main elements to alter are the name, the background color of the window and the window sizes. We don't need to alter these defaults now, so close the window to expose the Layout view window.

Define SMIL Regions via GR*i*NS Resource Channels

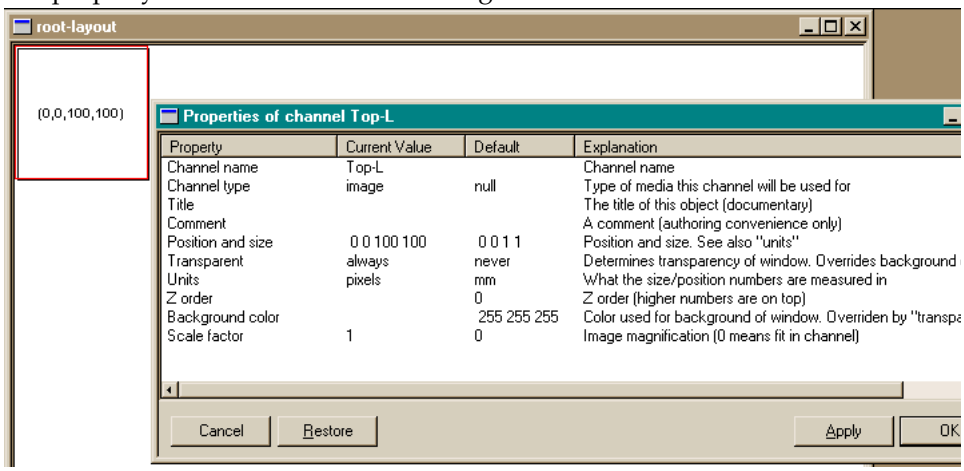
A SMIL presentation allows you to put any number of regions inside a presentation root layout. Find the section labelled Channels in the middle of the Layout view, and then select the New... button. This will bring up a dialog box that allows you to define the Channel.



A Channel consists of a name and a type. In GR*i*NS/SMIL, the following channel types are supported:

- HTML: a channel in which HTML-formatted documents are rendered;
- Image: a channel in which various types of image formats are supported;
- Layout: the GR*i*NS name of the SMIL Base Window;
- Null: a channel without any particular resource policy;
- RealAudio, RealPix, RealText and RealVideo: channels that allow you to integrate RealNetworks' RealMedia objects in a presentation;
- Sound: an audio channel.

You should name this Channel *Top-L* (for Top-Left) and give it a type of *Image*. Once you confirm the selection, GR*i*NS draws a red box containing the new Channel in the default SMIL position: at the top left of the canvas. You can change its size and location in several ways. The first way is to open the property sheet associated with *Top-L* selecting it in the Channels list and then selecting Properties. The property sheet contains the following elements:



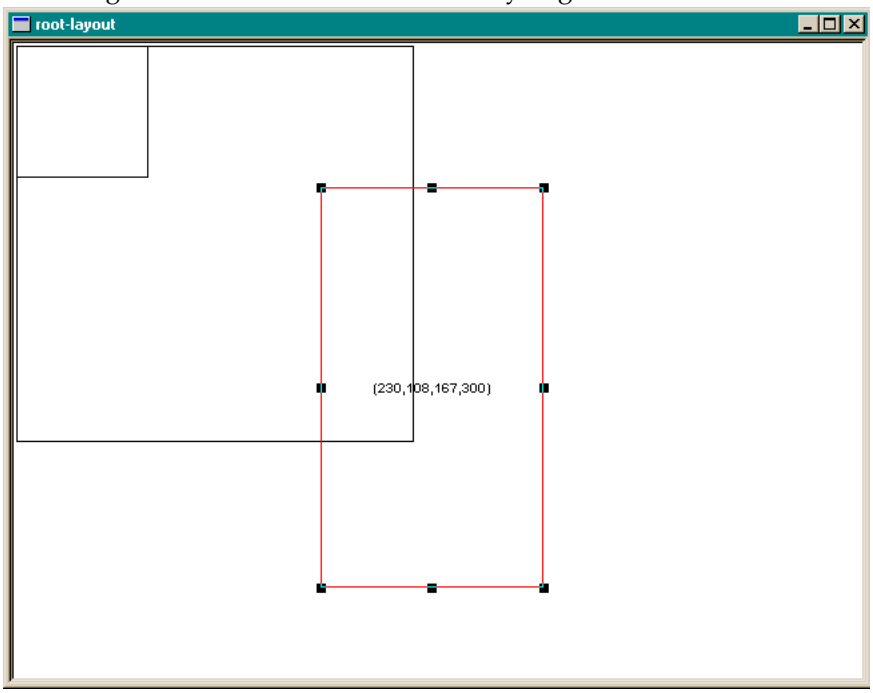
(The default window is shown at top left, and the new window parameters are shown in the property sheet.) You can change the location of the window and its size by entering the parameters directly into the property sheet. This is useful if you want pixel-level control over your application.

You should keep *Top*-L positioned at 0,0 pixels (the upper-left part of the canvas), and increase its size to 300x300 pixels. Once you accept the changes, the Top-L Region will be defined and drawn at the appropriate part of the screen.

You should now create a second Channel, this time named *Mid* (for Middle). It should also have a type of *Image*. Once the Channel is created, it is drawn with resizing tabs and it is placed in the default position at the top-left of the root window. You can place the mouse anywhere inside the box, and reposition it by keeping the left-mouse key depressed. You can reshape the channel by selecting the resize tabs.

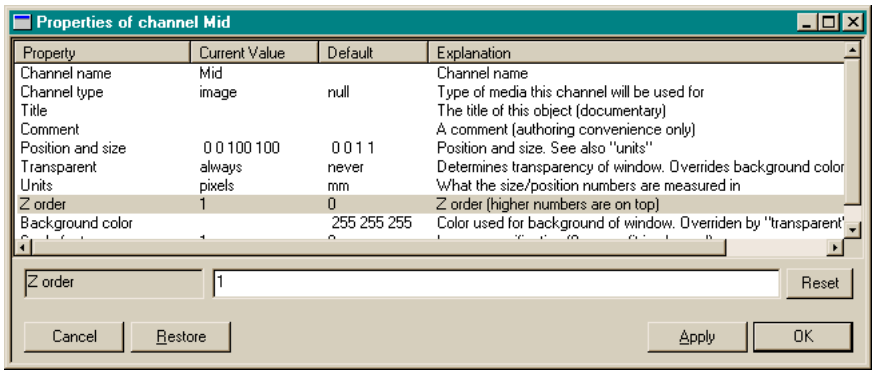
You should reposition the channel to (230, 108) and give it dimensions of (167,300).

The following illustration shows the view that you get:



Once you are done with the positioning, you can accept the changes by clicking anywhere outside the channel.

Now, pull up the property sheet for *Mid*, as shown below. (Note: you may need to

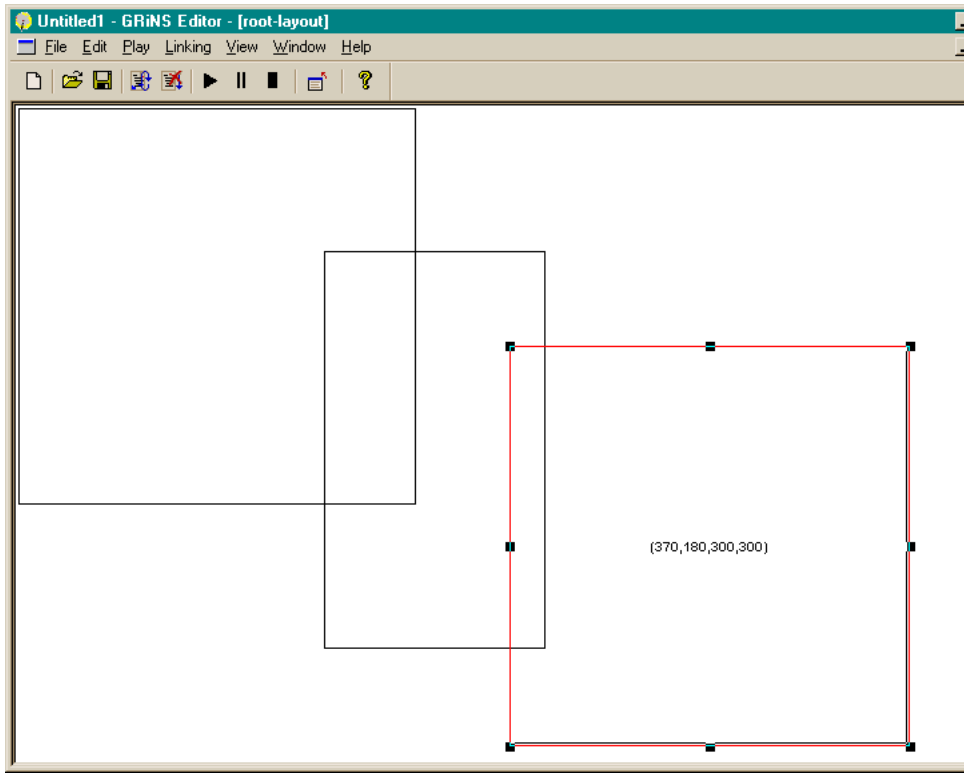


reselect the Layout view from the Windows menu to re-expose the layout view.)

Go to the *Z order* selection, and change the Z order from 0 to 1. This means that, in case of an overlap of images, Channels with a higher Z order will be (relatively) on top of the display. You also have the opportunity to change the *scale factor*. There are two choices that we now consider: if you set the scale to '0', the image will fill the entire window. If you leave it at '1', the image retains its natural size.

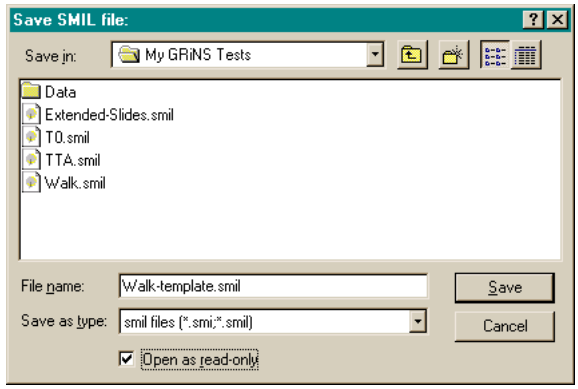
You can now go on to create the third channel by selecting New from under the Channels list. Give the Channel the name *Bot-R* (bottom right) and give it a type of *Image*. Using either the property sheet approach or the resizing via the mouse, place the Bot-R channel in the bottom right part of the canvas and give it a size of approximately 300x300 pixels. (Note that you may need to expand the GR*i*NS Editor view to 100% of the screen to see the entire player window.)

If we used the mouse-based resizing, we would get the following view:



At this point the presentation contains one base window and three rendering Channels, named *Top-L*, *Mid*, and *Bot-R*. Each of the Channels is set to render images in the format defined by the author. Note that we don't know if the images will appear in parallel or sequentially, or if each Channel will be used for one or more images. This depends on the specification of the presentation in the Structure and Timeline views.

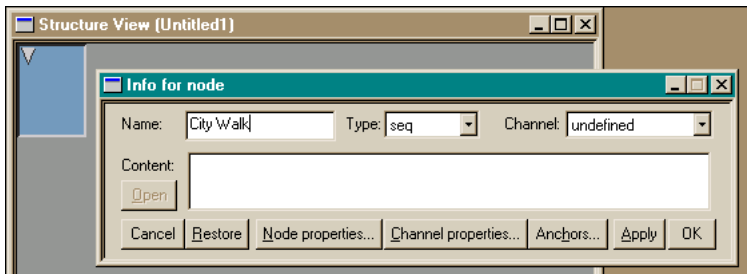
If you haven't done so already, this is a good time to save your work. Use the name *Walk-template.smil*, and select the 'open as read-only' check box to create a template. The save box looks as follows:



Defining the Structure and Contents of the Presentation

The *Walk.smil* presentation used in Tutorial 2 contained a single piece of audio that accompanied a sequence of three images. In SMIL-ese, we need a parallel node containing the audio and a sequential node, with the sequential node providing the images.

We can construct this presentation using the facilities available in the Structure View. Close the Layout View and expose the Structure View. Inside this view, we start by giving the top-level sequential structure a name. We do this in the same way in did in tutorials 1 and 2: by bringing up the structure's Info box.

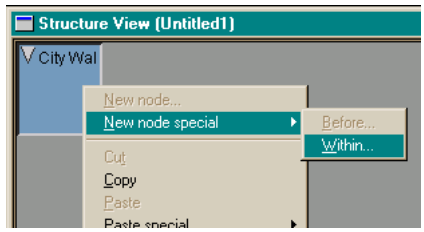


Inside, we label the structure node *City Walk*. The other properties are already filled in by the Editor.

Note:

1. The use of a top-level sequential node is optional in the sense that this node could also be transformed directly into a parallel node containing the audio node and the sequential node containing the images. The reason for using the top-level node will become clear in Tutorial 4.

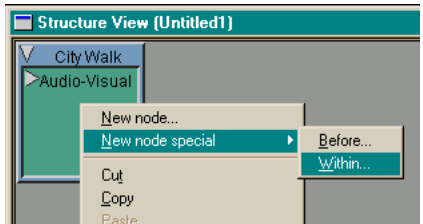
We start the development of the presentation by creating a new node within the *City Walk* container by selecting New node special -> Within.



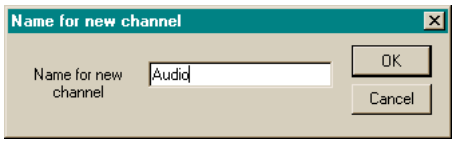
The GR*i*NS Editor is aware that we only have one option here: you can't put anything before or after the top-level container, so the only choice is to put a new node inside (or within) that container.

The type of node we are about to create is a parallel structure container, which will hold the audio and images in the presentation. Once the node has been created, select its Info box and name the node Audio Visual Show. In the Info box, you should also set the node type to be 'PAR', for a parallel container.

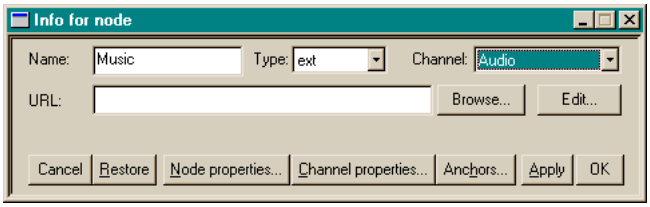
We now continue building the presentation by placing a new node inside the Audio Visual show. We again select New Node Special > Within, but this time we create a media object reference instead of a structure node. Name the node *Music*, sets its type to *ext* (for external) and select *New Channel* under the channel window.
The filled-in dialog box is:



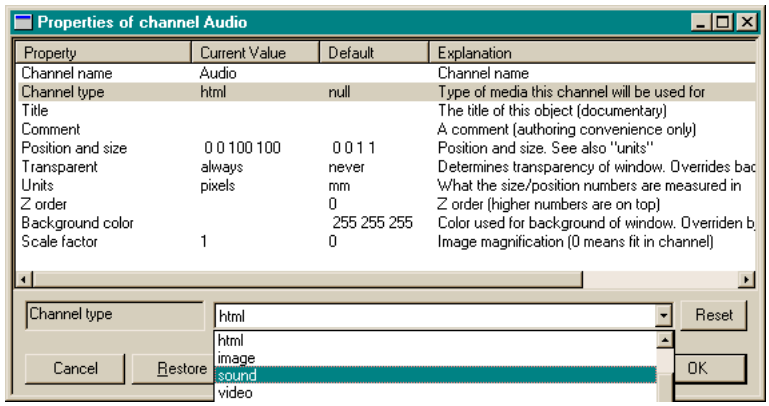
When you select New Channel, you will be given a *Channel name* pop-up box. You should name the new channel *Audio*:



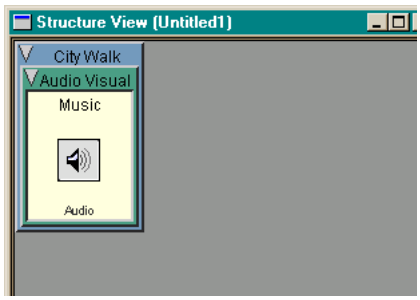
The resulting Info box is:



Confirm these selections with Apply. Recall that we haven't defined a layout region for *Audio*, but by creating a new channel named *Audio*, GR*i*NS has given this channel some default properties. The default media type for this new channel is HTML. This is probably not what we want for an audio channel, so it is probably a good idea to define the properties that we actually want for the Audio channel. We do this by opening the channel's properties, using the Channel properties button at the bottom of the Info box. Select *sound* as a type, as shown:



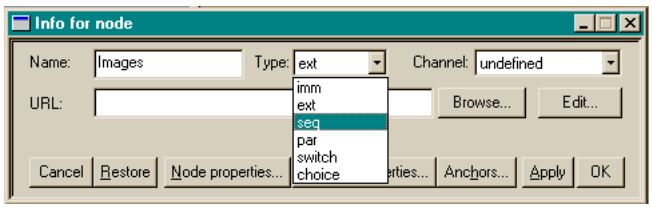
After creating the audio node, the Structure view will look as shown at left. What we have defined so far is a presentation structure with a single audio media reference.



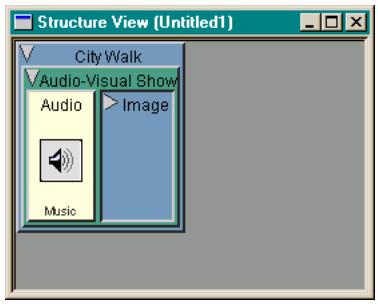
We now want to define the sequence of images in parallel with this object.

First, we need to create a sequential structure container. There are several ways of doing this, the most obvious of which is to first select the newly-created Music node and then select New Node Special > After from the right mouse button. (Since it is not possible to define a node within a media object, this option is not shown.) We will create a sequential structure node after the Music object.

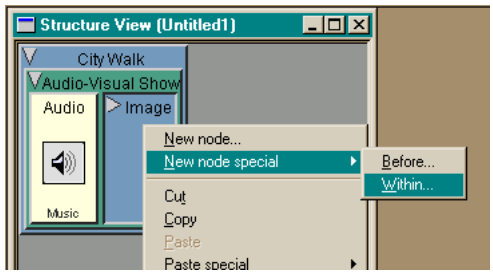
Once the new node is created, open that node's Info box and define the high-level characteristics of the node. Name it *Images*, give it a type of *seq* (for a sequential node) and leave the channel type undefined. The resulting Info box is:

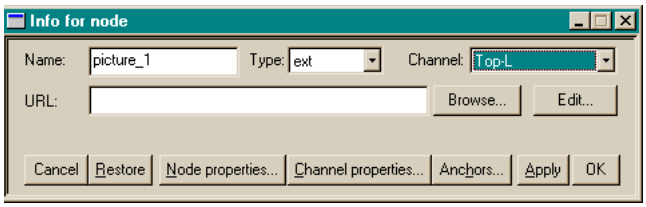


When you accept the changes, the Structure View is:



All that is left to do is to fill the sequential node with the image references. We start by defining a new node within the Image Sequence container:

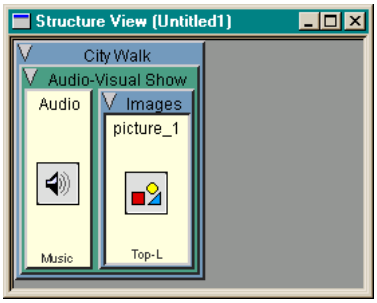




We open the Info box associated with this node, and define an image reference. The name is *picture 1*, the type is *ext* (for external) and the channel that we will use to display this image is *Top-L*, which is one of the channels we defined earlier in this tutorial.

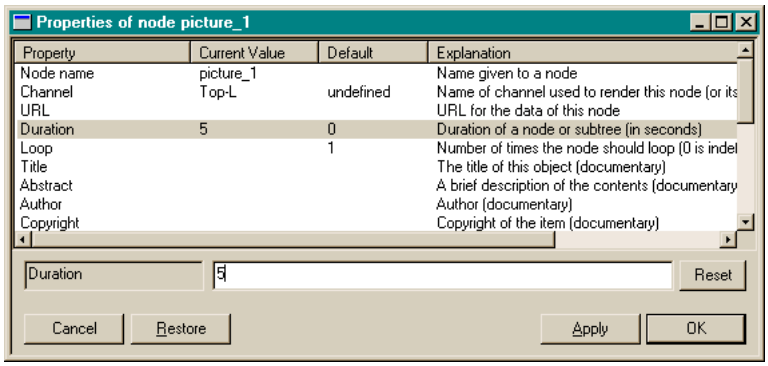
The node is defined as *picture 1*. This is not the name of an actual file (or network URL), but simply a text line that will tell the user of the template what kind of

reference should be put in the ultimate presentation. You can also put in the name of a real file (or use a full network URL), but this may create problems for the user during previewing of the application. If you do wish to include a file name or an explicit URL, you can type it in directly or you can browse the local file system for the desired asset.



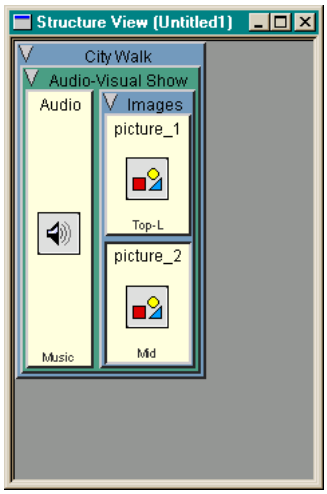
Once you accept the information in the Info box, a media object reference will be created and the Structure view will be updated with a tan block named *picture 1*, and containing an image icon. (If an actual file name was used, the object reference box would contain a thumbnail image of the referenced file.) At the bottom of the object reference is the name of the channel on which this image will be rendered (in this case, Top-L).

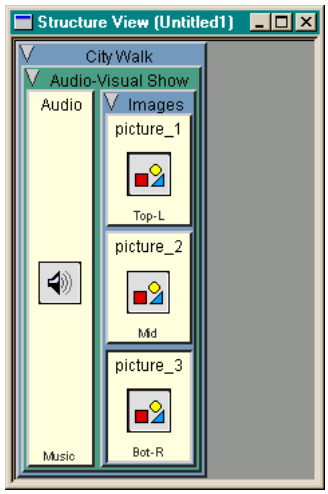
IMPORTANT: what you have created is a place-holder object reference for an image. One of the essential features of an image is that it is timeless — that is, it has no inherent duration. In order to give your template some predictable performance if a user drops an image onto the icon, it is **strongly recommended** that you define a default duration for this place-holder. To do this, open the property sheet for the object (via the Edit menu or via the right mouse button) and set a default duration of, say, five (5) seconds:



Once the *picture 1* image reference is created, we can use the same procedures to create image references for *picture 2* and *picture 3*. You can also use the GR*i*NS copy and paste facility by selecting copy from under the right mouse (or from the Edit menu) and then Paste After. If you use copy and paste, be sure to open the Info boxes for each node and change the node name, the channel assignment (Mid for *Picture 2* and Bot-R for *Picture 3*), and the name of the file/URL to be referenced.

The resulting Structure Views after each step are:





Make sure that all the information in your template matches that in the examples above. Note that if you used copy/paste to create picture_2 and picture_3, the duration of five seconds is inherited by the new objects.

The template is now complete. You should save it for later use.

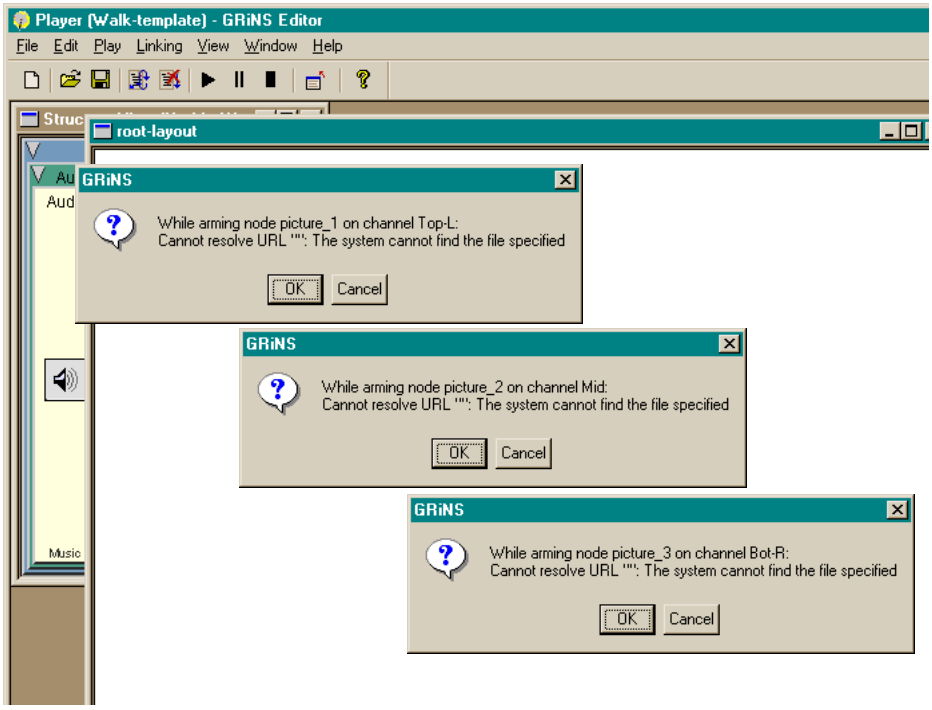
Previewing the Presentation

The Walk-template.smil presentation contains all of the structure and object information required to construct the presentation. As it now stands, however, each of the objects only contains a place-holder name. As a result, there isn't much to see in this presentation.

Even though the template is empty, you can still preview the presentation. This may be useful if only a part of the object references have been resolved, or if you are defining a complex presentation where you are not sure what has been defined and what not.

The embedded GR*i*NS previewer will walk through the presentation-in-progress and inform you of any missing references. For each unresolved reference, a GR*i*NS pop-up box is displayed with the name of the object that cannot be resolved. If you provided timing information in terms of the duration of objects, then this timing will still be enforced in the presentation.

The image on the following page show a time-lapsed rendering of what would happen if we previewed our presentation. Note that in an actual preview, you would get the notification boxes one at a time rather than all together, as shown.



Closing Comments for Tutorial 3

The steps taken to build a simple presentation have been defined in some detail, so that you can get a feeling for the GR*i*NS general structure. Naturally, we could have made things easier by simply putting all three images on one channel -- or not defining channels at all and having GR*i*NS define them for us. Hopefully, by taking the somewhat longer road, you have gotten a bit of a better idea of how GR*i*NS works. While this may seem a bit counter-intuitive if you are used to time-line editors, the pay-back comes when you start to edit larger presentations -- or when you need to define alternate content for adaptive presentations.

Take a break and then learn about creating adaptive presentations in Tutorial 4.

Tutorial 4: Working With Transitions and RealMedia

Overview and Goals

The facilities available in SMIL-1.0 for placing and activating images provide the basic support needed for many multimedia applications. Some individual data formats extend the facilities of SMIL by providing support for standard image operations such as *fade-in*, *fade-out*, *wipe* and *crossfade*. GR*i*NS works together with the RealSystem G2 player and Real's RealPix format to support this functionality. RealPix objects can be created using GR*i*NS and existing RealPix files can be extended or modified within the GR*i*NS environment.

In order to use RealPix effectively, several aspects of the format must be understood. One of the most important things to know about RealPix is that it makes use of a timing and layout model that are different from that in SMIL. As a result, care needs to be taken to make sure that the content of a RealPix object is defined in such as way as to allow smooth integration into a SMIL presentation.

This tutorial constructs a RealPix slideshow that is not unlike that which was created in tutorials 1, 2, and 3. The main differences in the presentation created here are:

- the slideshow starts with a fade-in of an image, followed by a fade-out;
- the GR*i*NS Layout Channel containing the images get sub-divided into a number of sub-regions, each containing a picture; and
- image movement is introduced into the presentation.

This tutorial starts with a quick introduction to the conventions used in creating a RealPix object and then goes through a step-by-step summary of how you can make effective use of RealPix via GR*i*NS. This tutorial let you build a RealPix object from scratch — things are even easier if you use a GR*i*NS template file.

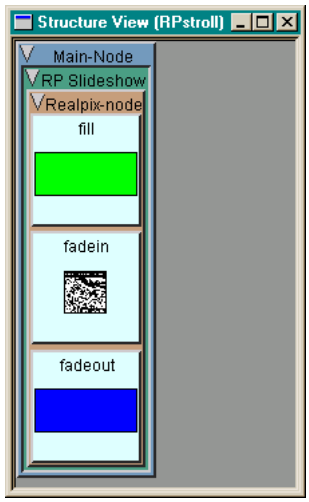
> **Important note:** a RealSystem G2 player must be installed on your system before you can preview RealPix images in the GR*i*NS Editor. If you are working on a UNIX system or any system that does not have a G2 player installed, you can still create RealPix presentations, but you can't preview them without G2.

Modeling RealPix-Based Objects

RealPix is an object format that is based on changes to the bitmap inside of a SMIL region. All manipulations with RealPix specify a sequence of changes to a RealPix window that is placed inside a SMIL region.

GR*i*NS models a RealPix object as a single, complex datatype. You start by creating a RealPix container object. You then specify a sequence of changes (transitions) to the contents of that object over time. Each transition consists of a number of attributes:

- *Type*: the kind of transition being specified, such as *fade-in*, *fade-out* or *wipe*;
- *Image*: the data being added to the existing bitmap;
- *Start time*: the moment that the transition is to start to take effect;
- *Duration*: the length of time given to allow the transition to complete;
- *Location*: the part of the window that is to be changed with the transition.



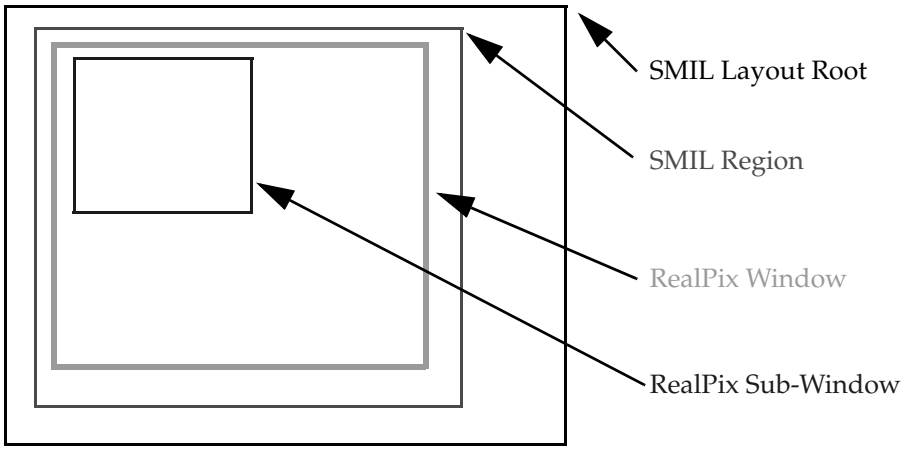
An example of a RealPix object is shown at right. The entire object has a special light red color to identify it as a RealPix object. Inside the object are collections of images that are mixed into the background bitmap. The attributes of any transition (type, image, start time, duration and location) are defined inside each of the items in the RealPix object.

We see a parallel node named Slideshow that contains a single RealPix object (called *RealPix-node*). The *RealPix node* contains three components: an initial fill of the region with a green background, followed by a fade-in of a *Map* image, followed by a fade-out to a blue background. This view tells us what the components of the object are and the general ordering of the transitions, but not their detailed timing. This is in each components's property sheet.

You can fill individual objects using GR*i*NS drag-and-drop, and you can use standard copy/paste, but the specific nature of RealPix brings with it some restrictions that are discussed in this tutorial.

SMIL Layout and RealPix Windows

When creating or editing a presentation containing a RealPix node, you should take care to manage screen resources correctly. The following illustration shows the relationship of both SMIL's and RealPix's use of screen resources:



Any SMIL presentation is defined in terms of a *root layout* window. All of the visually rendered objects will be placed inside this window.

One or more *SMIL Regions* can be placed inside the root layout window. In GRiNS, a Region is modeled as a *Logical Channel*. The behavior of overlapping regions is determined by each region's Z value.

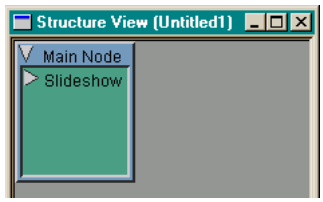
A RealPix object defines a *RealPix window.* The is the visual space that the object uses during rendering. In general, the *RealPix window* should be the same size as the *SMIL Region* that contains it. If the *RealPix window* is smaller, then there will be used screen space in the region; if the *RealPix window* is larger than the region, then information will be lost.

As we will see, a *RealPix window* can also be divided into *sub-windows*. Unlike SMIL's regions, *RealPix sub-windows* don't have a Z value. Instead, the most recent contents of a *sub-window* over-writes the old contents.

Creating a New RealPix Object

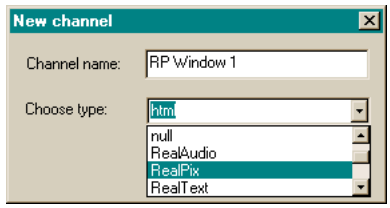
Open the GR*i*NS Editor and create an empty presentation using New in the Edit menu and then selecting the *empty.smil* template. Using the techniques learned in Tutorial 3, perform the following steps:

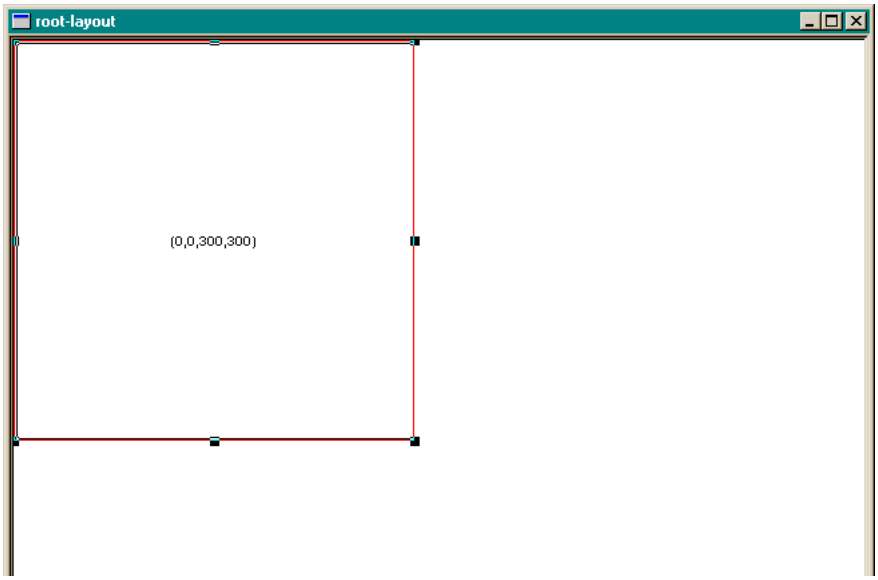
1. create an outer sequential structure container named *Main Node*;
2. place a parallel structure within *Main Node*, and name it *Slideshow*;



This yields the Structure view at right.

3. go to the Layout view and create a GR*i*NS channel named *RP Window 1* and give it a type of RealPix:
4. when the channel is created, resize it in the Layout view — using direct manipulation or the Channel's property sheet — to a size of 300 x 300 pixels, as shown below:





At this point, you have a *root layout* sized 640x480 pixels and a single *RP Window 1* region sized 300x300 pixels.

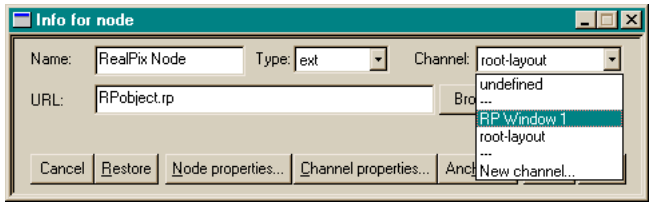
> **Important**: save your presentation now in a directory that will serve as the root directory for your RealPix object—that is, in the directory in which all your images are located, or in which a directory is located that contains your images. This is a RealPix requirement.

You should save this presentation is a file called *RPshow.smil* in the *Tutorials* directory. (Put it here so that you can include objects from the *Assets* directory that is a sub-directory of *Tutorials*.)

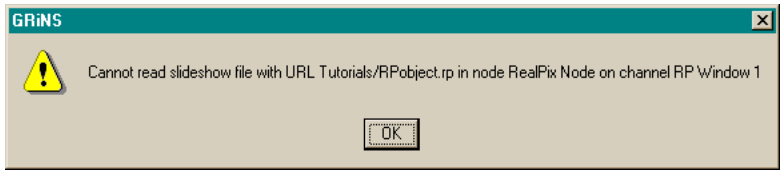
Defining the Basic Properties and Attributes

Once you have created a GR*i*NS Layout Channel to reserve screen space for your RealPix object and after you have saved the presentation as *RPshow.smil* in the *Tutorials* directory, you are ready to create the set of transitions and images that will make you your object.

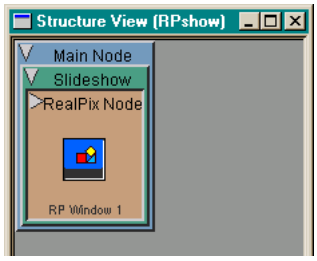
Go back to the Structure view and select the *Slideshow* parallel node. Now, add a new node within the parallel node by selection New Node Special -> within from under the right mouse or via the Edit menu. This brings up the Info node for the new GR*i*NS object. You can either select an existing RealPix object by entering the URL of that object, or you can create a new object by specifying its name in the URL field. Enter the name *RPobject.rp*; GR*i*NS will place it in the same directory as your *RPshow.smil* presentation. (The *.rp* extension identifies this as a RealPix file.) Fill-in the rest of Info box as follows: the *name* is RealPix Node, the *type* is ext and the Channel is the *RP Window 1* that we just created.



Once you accept the information by selecting Apply, you see the following:



This is an informational warning that you are creating a new RealPix object, rather than opening an existing one. Select OK. You get the following Structure view:



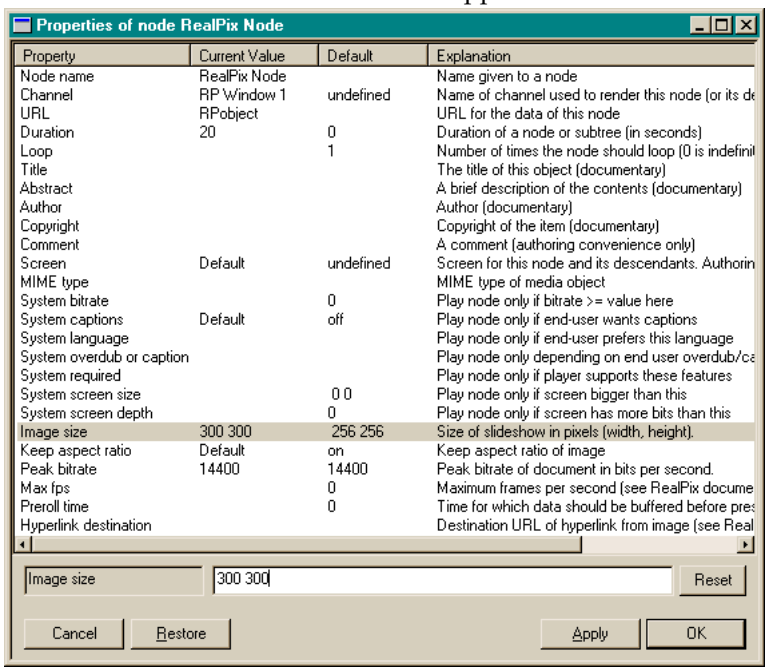
Normally, GR*i*NS works with existing data objects. As a result, GR*i*NS attributes are tailored to helping you control when and how an object gets activated. In the case of RealPix, you use GR*i*NS to edit the content of the object itself. This means that new attributes are available that control when and how individual images and the various transitions within a RealPix file get built.

The RealPix object contains all of the normal GR*i*NS attributes, plus some special purpose attributes that tell the G2 system how to process the file. These include:

- *URL*: the location containing the (root of the) images for this object;
- *Duration*: the SMIL duration of the entire object. You should start with an estimate of the total length — for our example, fill in *20*.
- *Image Size*: the space within *RP Window 1* that will be used for images — this should be equal to or smaller than the dimensions of the RealPix Channel;
- *Keep aspect ratio*: enlarge objects to fill the space, but keep aspect ration same;
- *Peak Bitrate, Max FPS, Preroll time*: RealPix attributes that we will visit later.

If you had opened an existing object, these would have been filled in for you based on that object's definitions.

Since this is a new object, fill in the property sheet with the following values. You can either study Real's documentation for RealPix or you can take these values as magic numbers that will do the trick for most applications.



The most important changes for our tutorial are:

- the *Duration* is initially set to 20 seconds
- the *Image Size* is set to 300x300 (as determined by our data requirements)

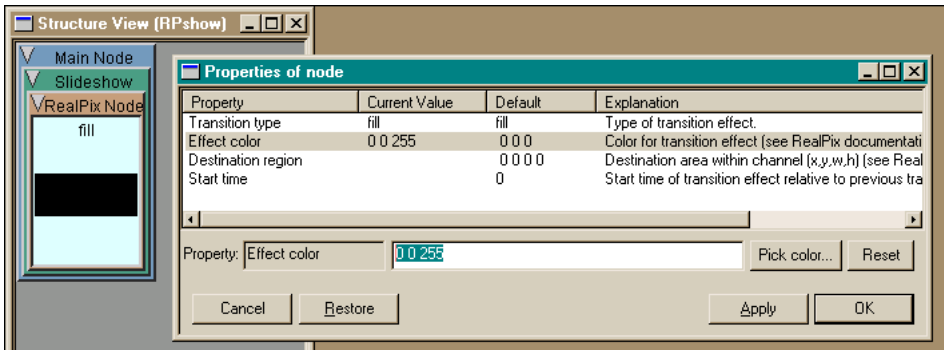
Other key attributes are set by GR*i*NS automatically during editing.

The duration of the actual presentation we will put together in this tutorial is 35 seconds, but we start with a duration of 20 seconds to show how you can manipulate and debug presentations as they develop.

Adding Images and Transitions to the RealPix Object

We now have done all the set-up that we need. To add a component transition to a RealPix node, simply select the node and use the standard GR*i*NS technique for adding a new components: select New Node Special -> within from the Edit menu (or under the right mouse).

Where this usually brings up an Info box, for RealPix nodes you get a property set up for the default transition: a *Fill*.



The Fill allows you to paint a particular color over all or part of the drawing space for a RealPix object. This "drawing space" is defined as either the Image Size set in the RealPix object's property sheet (for us, a 300x300 area) or some subset of this area, as defined by the *Destination region* attribute in the property sheet above. Leave this blank for now — we'll play with it in a later component.

You should select a color by typing in the RGB code or using the Pick Color option on the data entry line. In this example, choose Blue (which is 0 0 255).

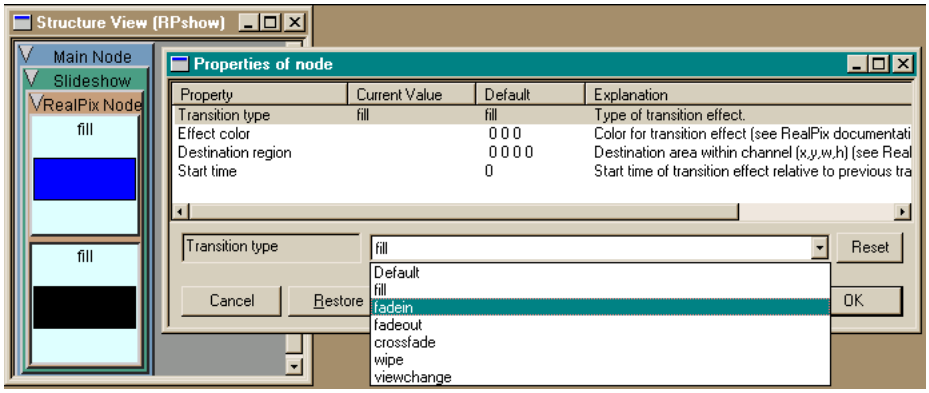
The last important attribute is the Start time. This is set to "0" to say that we want the fill to take place right away.

You should preview the presentation at this point. You will see a blue field that takes up a 300x300 square in the top left part of the player window, which is what you'd expect given the attributes for *Fill*. Note that the Structure view shows a Fill transition icon, with a sample of the fill color used.

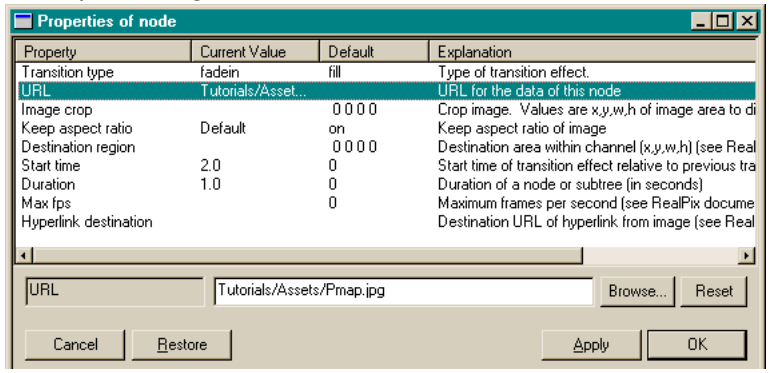
Fading In an Image

The *Fill* transition sets the background to a particular color (in our case, blue). You can now specify that you want to fade-in an image over this background by saying New Node..., which put a node after the one you last created.

When you specify New Node..., a new transition property is created. The default type is *Fill*, which you should change to *Fadein*:



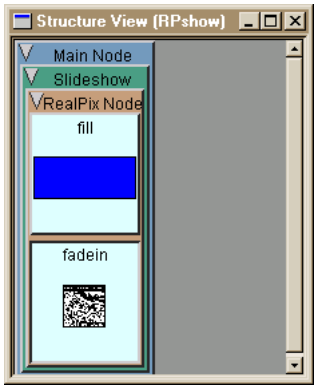
Select Apply and you now get the *Fadein* transition sheet:



In this sheet, we have selected the *Pmap.jpg* image in the Assets sub-directory, we've set the Start time to 2.0 seconds, and the Duration to 1.0 seconds.

What this says is: take the specified image (*Pmap.jpg*) and start to fade it into the background starting at 2.0 seconds after the most recent transition specified, with a duration for the fadein of 1.0 seconds.

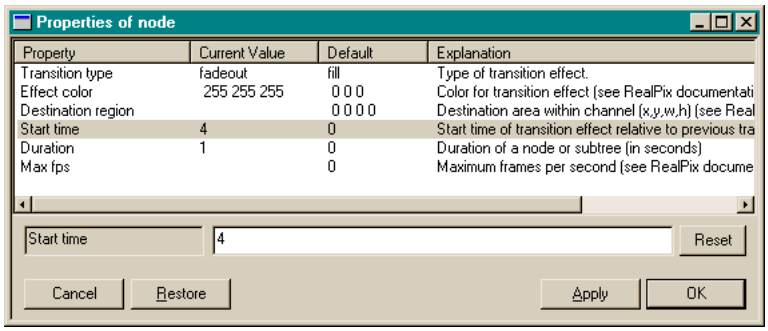
> **IMPORTANT**: the start time used by GR*i*NS is the *time since the specification of the last transition*. This is not the same as in a RealPix file, which uses the time since the start of the object. By using the GR*i*NS method of specifying a time relative to the start of the last transition, you can cut and paste within a RealPix object without have to update all of the other image times.



This Structure view after specifying the transition is shown at left. If you now select Play, you will see a blue square, followed by a fade-in of the Pmap image after two seconds. Since we set a duration of 20 seconds in the RealPix Node, the image stays up for about 17 seconds.

You can now add a few more transitions to experiment with RealPix functionality. Start by doing a Copy on the component labelled *fadein*, and then select Paste from the Edit menu. This makes a copy of the component, and places it after the map image.

Open the property sheet of this node and change the transition from *fadein* to *fadeout*. Hit Apply, which changes the property sheet to one for *fadeout*:



Set the effect color to white (255 255 255) by typing in the RGB code or using the Pick Color option, and set the start time at 4 and the duration at 1. Accept the changes with OK.

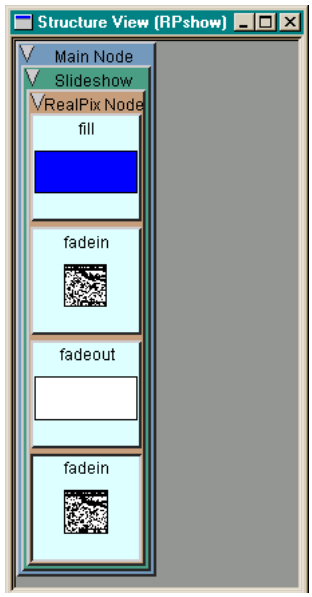
If you now select Play, you see a blue field, followed by a fade-in of the *Map* image, followed by a fade-out to white four seconds after the map first started to appear.

NOTES:

1. All of the images used in a RealPix presentation **must** be JPEG images in a streaming-ready format. This is most easily created though a utility called Jpegtran, which translates the images to a format compatible for use with streaming presentations. The utility is available from RealNetworks at: *http://www.real.com/*.

Making Use of Sub-Regions

The presentation built so far always places an image into the full area defined in the RealPix node. You can also tell RealPix to place in image in a smaller part of the object's window by specifying *destination region* within any transition's property sheet.



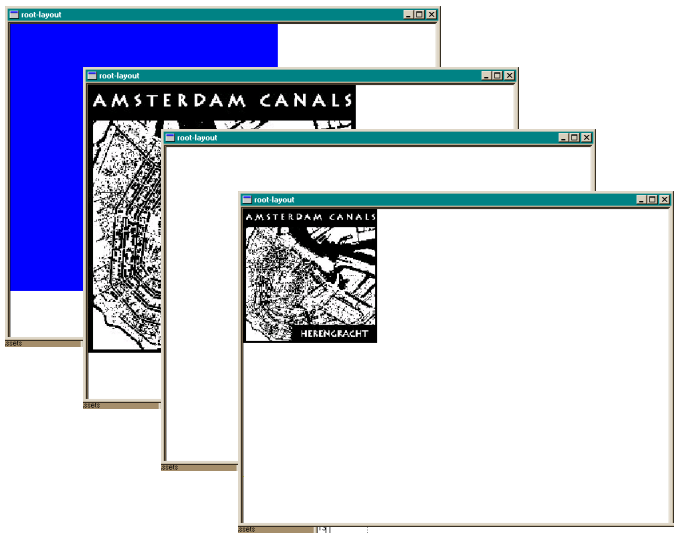
To see how this works, go to the Structure view and make a copy of the component labelled *fadein* by using the Copy/Paste functionality. This creates the Structure view shown at right.

Open the property sheet for the second *fadein* component and make the following changes:

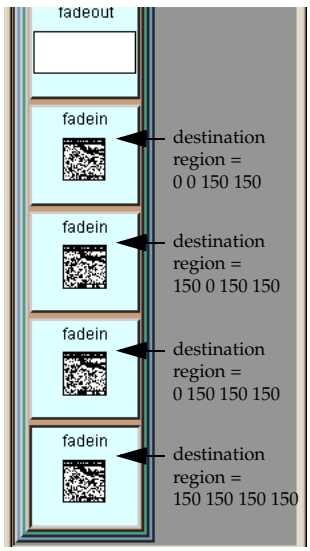
- *Start time*: 4 seconds
- *Destination region*: 0 0 150 150

This says: four seconds after the last transition started (in this case, the fade-out), start a fade-in of the image in the component lasting one second and place the object in a sub-region of 150x150 pixels, anchored at the 0,0 position of the object's window.

A time-lapse view of the resulting presentation is:



You should now create three more copies of the last fade-in component and paste them on to the end of the *RealPix Node*. The Structure view of the lower part of the object is shown at right. You should now edit the property sheets of each of these objects so that the *Destination regions* are set as shown in the illustration.



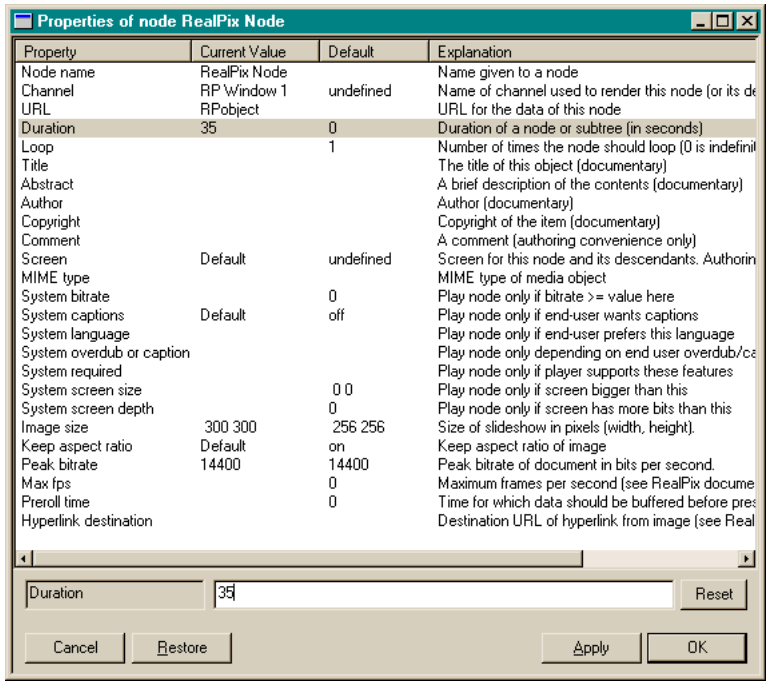
This will produce a checker-board pattern of images, each displayed for several seconds before the follow-up images are brought in.

Now, select Play to preview the presentation. Everything works fine until the next-to-last image is entered into the presentation. After it is displayed for about 2 seconds, things stop and the final image is never shown!

What's going on? Before you send mail to the GR*i*NS help-desk, recall that when we created the RealPix Node, we set a Duration of 20 seconds. This is a SMIL-level directive that tells the SMIL player to stop rendering the node 20 seconds into the object — regardless of the object's internal length.

The player has done just that: two seconds into the third checkerboard square, the 20 second time limit has expired, and the (SMIL) presentation ends.

Go to the enclosing object (labelled RealPix Node) and bring up its property sheet. Change the Duration to 35 seconds:

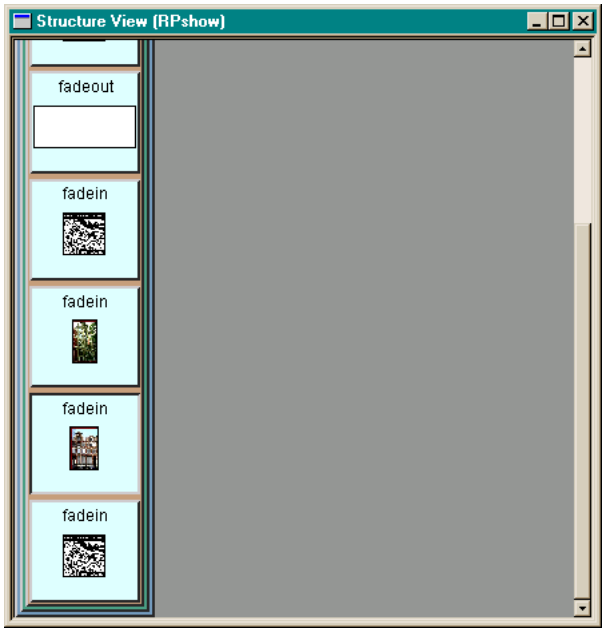


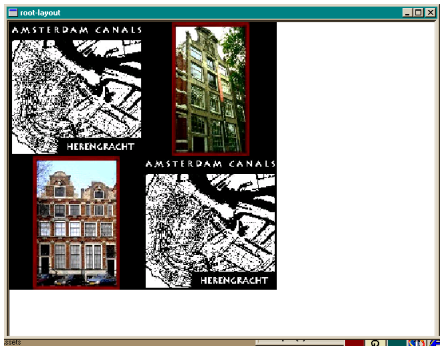
Now, replay the presentation. You will see the entire checkerboard displayed.The checkerboard remains until the full 35 seconds of the presentation have elapsed.

The existing presentation ends with the checkerboard pattern at left. If you wanted to liven this up a bit, you could put in two different images in the lower-left and upper-right squares.

To do this quickly, use the GR*i*NS drag-and-drop funtionality by dragging the image *Ph168c.jpg* on to the second component in the four-object sequence, and *Ph218c.jpg* onto the third component. (Both images are in the *Assets* directory.)

This gives the following Structure view fragment:





Now, when you play the presentation, the final state of the checkerboard is as shown. (This illustrates one of the artifacts of RealPix: if an image does not completely fill a given (sub-) region, the background is made black.)

If your presentation uses another background color, you can either edit the source images or you can make use of image crops, as discussed next.

Using Wipes and Viewchanges

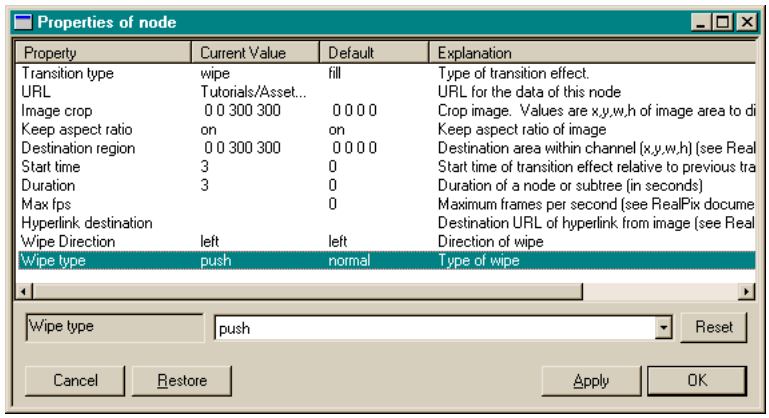
The images used in the presentation so far have been static: the are faded in and out, but they don't move. We can add basic image movement using *wipes* and *viewchanges*.

Using either New Node Special -> within or Copy/Paste, add three nodes to the end of the *RealPix Node* object. Edit the properties of the first node as follows:

- Transition type: set to *wipe*
- URL: set to *Phouses.jpg* (in the *Assets* directory
- Image crop: this is the part of the full image that we want to use; set it to the values: 0 0 300 300 (the entire image size is 520x300)
- Destination region: this is the place we want to put the image in the presentation: set it to 0 0 300 300; since this is the entire region, you can also leave it empty
- Start time: set to 3 seconds
- Duration: set to 3 seconds
- Wipe Direction: set to *Left* (this is the default)
- Wipe type: set to *push*.)

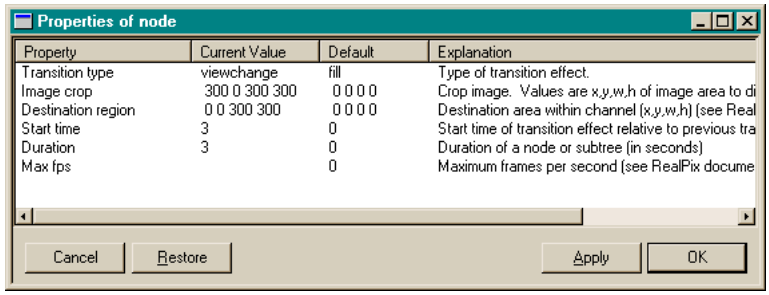
These attributes say to take the image *Phouses.jpg* and to move it into the presentation from right to left, pushing out the old bitmap along the way. (The default behavior is not to push out the old bit map, but to replace it incrementally.

The completed property sheet is:



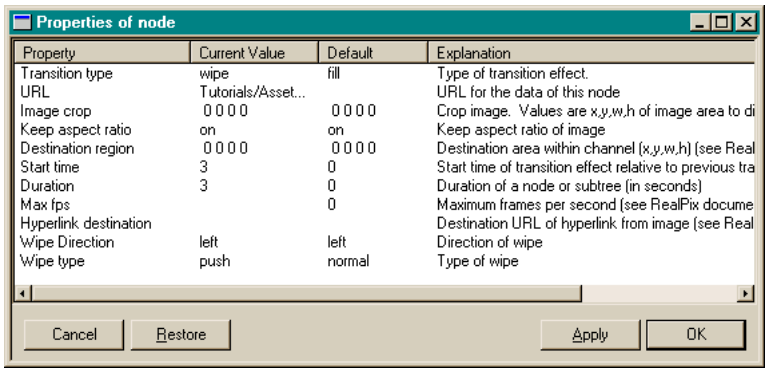
Another type of movement is *pan and zoom*. In the second of the three new components, set the following attributes:

- Transition type: set to *viewchange*
- Image crop: this is the part of the full image that we want to use; set it to the values: 300 0 300 300)
- Start time: set to 3 seconds
- Duration: set to 3 seconds



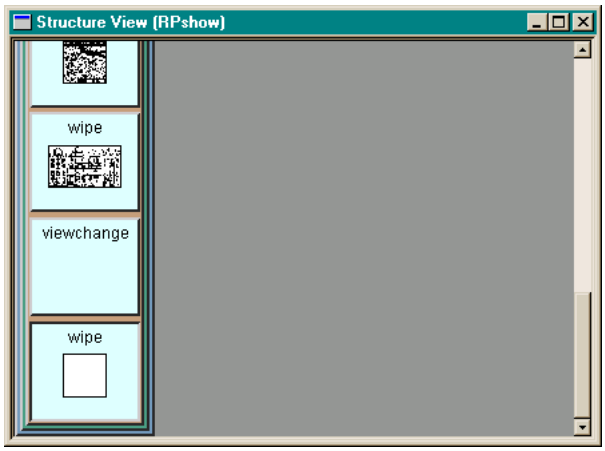
This allows you to pan across an image that has been placed in the bitmap. The pan takes place from left to right.

In the last box, we do one more *wipe* to remove all of the images we've seen so far. We do this by wiping in a white field, using the image *Pwhite.jpg*. The filled in property sheet is:



Here we use the default image sizes for source and destination and once again use a three second transition time. Note that we also once again use a wipe type of *push*.

The tail-end of the Structure view after adding these three operations is:

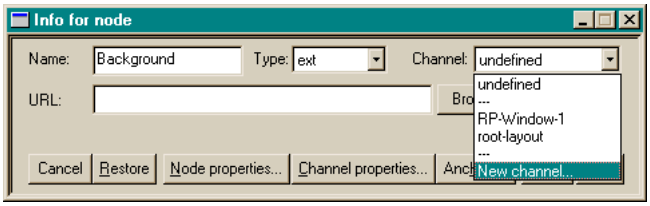


If you haven't done so already, preview the presentation.

Finishing the Presentation

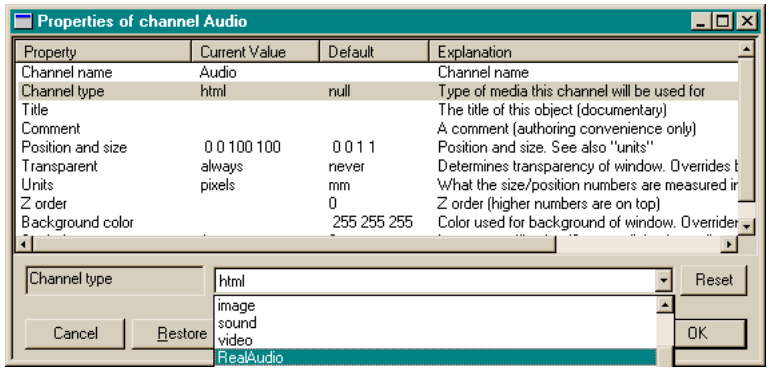
As a last touch, we add some background music to the presentation.

Go to the Structure view and select the *Slideshow* parallel node. Create a new node by selecting New node special -> within, and call the node *Background*. It will default to a type of ext. In the Channel area, request the creation of a new rendering resource:



Name the new channel *Audio*, confirm with OK and then select Apply in the Info node.

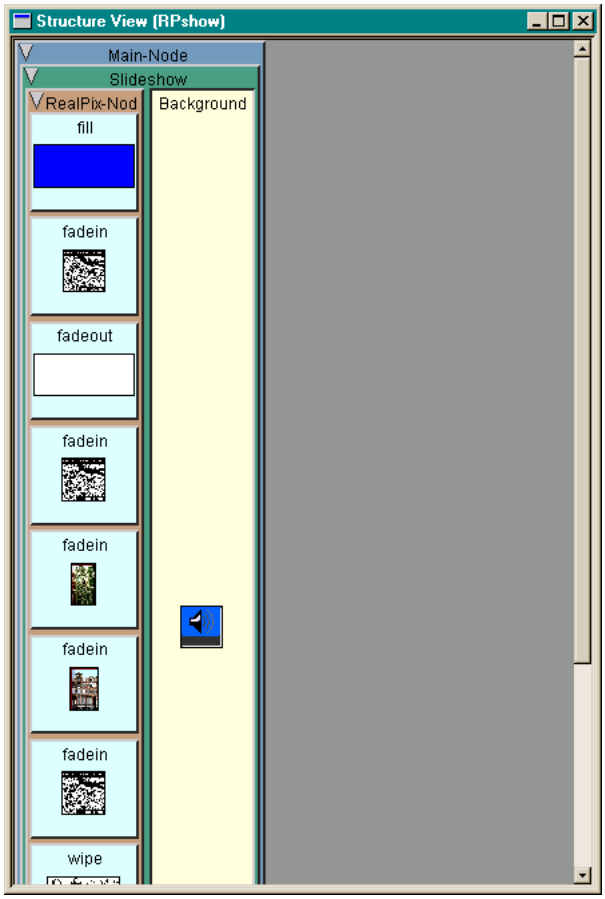
Next, select Channel Properties to define the type of data that *Audio* will accept:



Set this to RealAudio and confirm with OK.

We now have a RealAudio channel on which we can play an audio object.

Using either GR*i*NS drag-and-drop or the Browse button in the property sheet, select the object *Pmusic.ra* in the Assets directory. A portion of the resulting structure view is shown below.



Preview the presentation. (If you done the other tutorials, you'll recognize part of this tune!)

Opening and Editing an Existing RealPix Object

This tutorial has focused on creating a RealPix object from scratch. If you work with an existing SMIL file that contains a RealPix object, you can edit and extend the object as if you had created a new object directly.

When editing an existing object using GR*i*NS, you should keep the following in mind:

1. When GR*i*NS reads in the existing RealPix object, it will either use the region associated with the RealPix file or (if no region was specified) it will automatically create a GR*i*NS channel in which the object get rendered. The channel's dimensions are taken from the RealPix file.
2. If you don't want to alter the original file when saving your presentation, give the object a new URL in its property sheet. If you give a new URL, make sure that it is either in the same folder as the original object, or that it is in a folder than contains all of the referenced images. (This is a RealPix requirement.)

GRiNS will assume reasonable defaults for most values associated with a RealPix file, but you should monitor the values of things like object duration, channel size and channel Z-ordering to make sure they meet the new object's needs.

Closing Comments

The purpose of this tutorial has been to show how you can create sideshows using RealPix and RealAudio functionality. Although RealPix is a proprietary data type requiring the use of the G2 player, it does provide functionality that makes many SMIL presentations more attractive. Similar functionality is likely to be added directly to future versions of SMIL.

In the next tutorial, we return to manipulating and customizing generic SMIL presentations.

Tutorial 5: Adapting Contents to Meet the Needs of the User or System Environment

Overview and Goals

The purpose of this tutorial is to show how GR*i*NS can be used to build *adaptive presentations*. An adaptive presentation is one that can adapt to the needs of the environment when it is being rendered. Adaptation can be based on:

- the speed of the network connection
- the (natural) language preference of the user
- the type of display available to the user
- whether captioning or dubbing of audio material is required

The basic adaptive presentation support available with GR*i*NS matches the facilities available in SMIL. SMIL has a construct known as the *switch* statement, which lets authors define what the alternatives are, and the order in which these alternatives should be evaluated. GR*i*NS gives the author full control over the SMIL switch.

This tutorial illustrates several types of adaptive control. We start with the control over a single media object and then move to controlling alternative groups of media objects.

You will perform the following steps in this tutorial:

1. Open the editor and create a document based on the *Extended Slideshow* template;
2. Fill the document with media objects;
3. Edit a system test attribute on one of the media objects to control the object's rendering;
4. Define a Switch container and create a new media reference for an alternative audio object;
5. Define a Switch container to hold alternative GR*i*NS structure containers.

This tutorial assumes skills developed in tutorials 1, 2 and 3. Please complete those tutorials before starting this one.

An Introduction to Adaptive Presentations in GR*i*NS and SMIL

The Web is a heterogeneous place. This heterogeneity is reflected in (at least!) the following aspects:

- *Bandwidth*: the bandwidth available to view a presentation depends on a host of factors that are unknown at the time you create your SMIL document. The user many be connected via a relatively slow modem (14.4Kbps or 28.8Kbps) or via a multi-megabit LAN connection. The server where you place your presentation may be dedicated to your application, or it may be shared with lots of other users. The section of the "information superhighway" between you and your customers may be wide (that is, it can accommodate lots of high-speed connections) or you could live along an Internet side-road. Whatever the potential speeds, your part of the road can be heavily travelled or relatively lightly accessed, all of which impacts the options you have available.
- *Equipment*: there are may different computers and computer configurations available on across the Web. These computers run different operating systems—or many version of the same operating system—and they have different processing resources available. Some machines will have 800x600 displays, others 1600x1240. Some will have multimedia acceleration hardware, others not. Some machines will have multi-channel audio, others will have one (or none).
- *Users*: It is generally assumed (especially by English speakers) that English is the *lingua franca* of the Internet. For the first generation of use, this was true, but as the Internet becomes more an a world-wide tool, the use of a single language within a presentation can form a serious barrier to broad attention of your message. Even within a single language, your personation may be experienced by a wide range of users, including the sight- and hearing-impaired. These last two groups are much wider than you might expect: all of us are sight- and hearing-impaired at one time or another during Web use. If you want to 'listen' to the news during a meeting, you are hearing impaired (and would welcome a written summary instead). If you want to 'watch' a presentation on your hand-held computer, you may find it more useful to get an audio summary of the show instead.

In each of these cases, a *one size fits all* approach to document design has its limitations. You can significantly enhance the audience for your presentation by providing alternatives to the information content of your work. For example, if you want to show a video to a wide audience, you may want to make both high-resolution and low resolution version available so that users with high-end systems will get a better experience (without losing customers at the low end). Doing so makes your work more attractive to a wider audience, which will increase your visibility.

The current way of supporting multiple representations of a presentation is to make multiple presentations. This is costly in terms of initial effort, and makes a general presentation very difficult to maintain and enhance.

GR*i*NS provides an integrated approach to creating these adaptive presentations. You can start with the baseline presentation, and incrementally add enhancements as you go. In so doing, you can make full use of the SMIL *switch* statement.

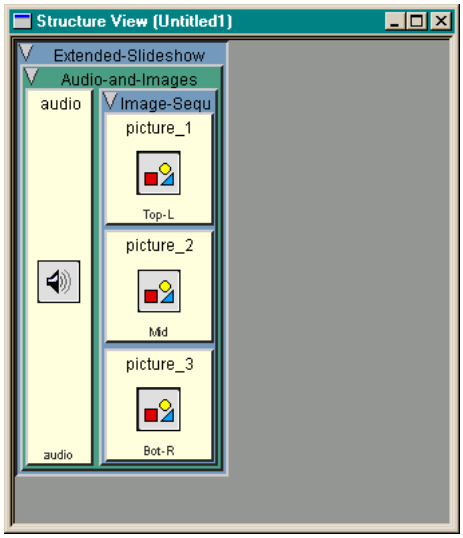
Two important notes:
1. GR*i*NS manages the process of making a presentation with lots of alternative content, but you have to provide the content itself.
2. Not all SMIL players provide full support of the SMIL *switch*. If you want to provide adaptive presentations, use a player such as the RealSystem G2 or the GR*i*NS Player.

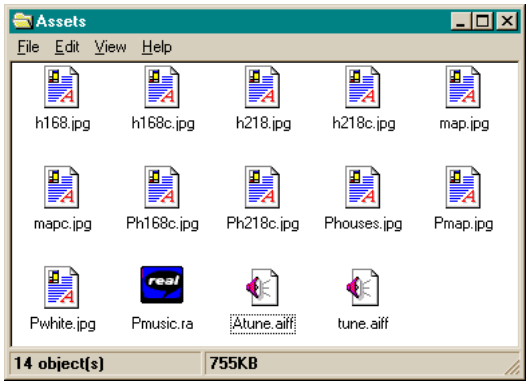
The following sections will show you how to use the GR*i*NS facilities for creating and maintaining adaptive presentations. This tutorial is set up to teach you a few new GR*i*NS skills, as well. These include:
- advanced editing techniques
- monitoring bandwidth use during a presentation
- using GR*i*NS preferences to set presentation environment settings

All of the examples in this presentation build on our running theme of a city walk. In this tutorial, we look at making the walk a more meaningful experience for a wide range of user.

Creating the Presentation

If you have not already done so, start-up the GR*i*NS Editor; go to the File menu and select New. Select the *Extended-Slideshow.smil* template. The Structure view will look something like:





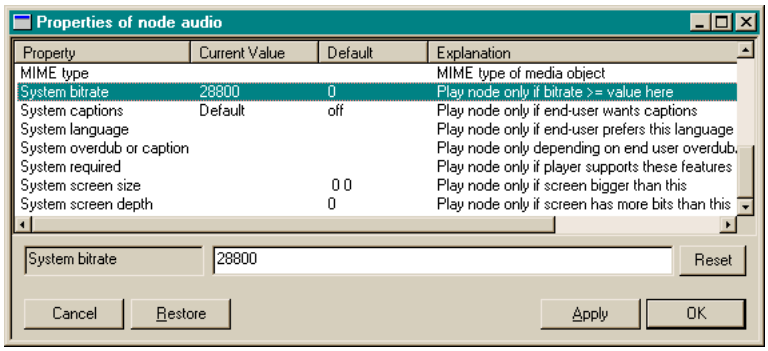
Now, use your computer's window system to open the directory *Assets*, usually located in the *Tutorials* directory of the GR*i*NS root folder. The *Assets* directory is shown at left.

Drag the images named *map.jpg*, *h168.jpg* and *h218.jpg* onto *picture_1*, *picture_2* and *picture_3*, and drag the audio object *Atune.aiff* onto the object named *Audio*. Preview the presentation.

Selectively Enabling a Media Object

Suppose you wanted to tailor the presentation so that the audio object was rendered only if your user had an network connection rated at 28800 (or greater) speeds. GR*i*NS provide a simple facility to do this.

Open the property sheet associated with the *Audio* object. Locate the item that is labelled System bitrate, and set the value to 28800:

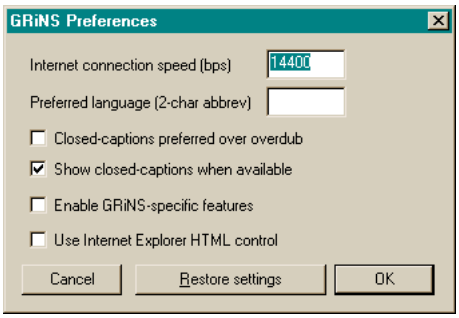


This tells the SMIL player to only render this object if the bitrate set at the client's player is 28800 or higher. (Note that the actual bitrate will depend on network conditions; this variable refers to a testable setting on the SMIL player.)

There are several *System Test Attributes* available to control the SMIL presentation:

- Bitrate: the declared Interconnection bitrate, as defined on the player;
- Captions: only play the object if the user has captions turned on;
- Language: only play the object if the user has specified this as the preferred language;
- Overdub/Captions: display captions instead of dubbed audio, if available;
- Required: only play the object if the named feature is present;
- Screen size: only play the object if the client's screen size is equal to (or larger) than this;
- Screen depth: only play the object if the client's graphics adapter has more bits than this.

The legal values for these attributes are given in the SMIL specification.



Now, preview the presentation. If you are running GR*i*NS as configured at installation, you will see the slideshow but won't hear the audio. Why? Open the GR*i*NS *Preferences* dialog by selecting Preferences in the Edit menu. GR*i*NS will not play the audio file unless your GR*i*NS preferences have been set to reflect the fact that your connection speed is at least 28k8. (Note: if the GR*i*NS Preferences had been set to a value greater or equal to 28800, the playback result would be different than that described above.)

Change the *Internet connection speed* to 28800 and play the presentation again. Now you should see and hear all objects.

The GR*i*NS Player allows you to set the connection speed and the following system characteristics via the Preferences dialog:

- *preferred language* (using ISO 2-letter codes);
- *Closed-Captions preferred over overdub*;
- *Show closed captions*.

GR*i*NS also provides two GR*i*NS-specific preferences:

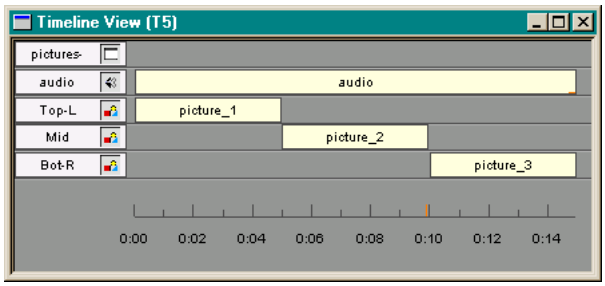
- *Enable* GR*i*NS-*specific features*: these are SMIL extensions that are only supported by the GR*i*NS Player; turning these extensions on will enable new menu options and new document processing capabilities (which are not described in this tutorial).
- *User Internet Explorer HTML control*: GR*i*NS uses the Webster HTML-3 control by default. On Windows-95/98/NT systems, you can choose to use the Microsoft IE control instead when you render HTML in a GR*i*NS document.

The use of these controls is beyond the scope of this tutorial, but you can try to set the various options on and off and then experiment with GR*i*NS to see how you can customize document behavior.

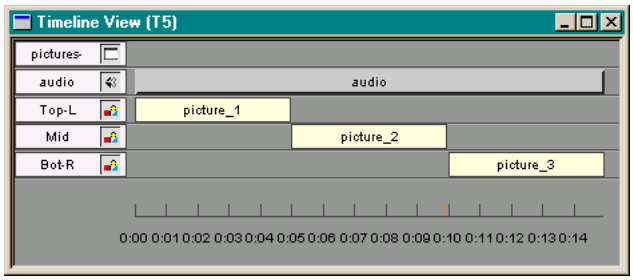
Viewing Selective Activation in the Timeline View

Using Edit -> Preferences, set the *Internet connection speed* to 28800 (or greater).

Now, open the Timeline view. As we saw in Tutorial 3, the Timeline view shows a temporal projection of the presentation and is generated automatically by the GR*i*NS Editor:



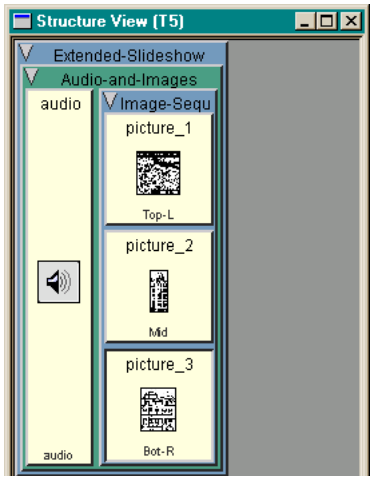
Next, go back to the Preferences dialog and set the *Internet connection speed* to 14400 (or any speed less than 28800). The Timeline view now changes to:



The illustration reflects the fact that the Audio object will not be available at the currently-specified bandwidth setting (it is drawn in grey).

Specifying Alternatives for a Group of Media Objects via the Switch

While SMIL's *test attributes* provide a useful way of controlling object activation, they become more powerful when combined with the SMIL *switch* statement. The Switch allows you to specify a group of alternatives, one of which will be chosen by the SMIL player at runtime.



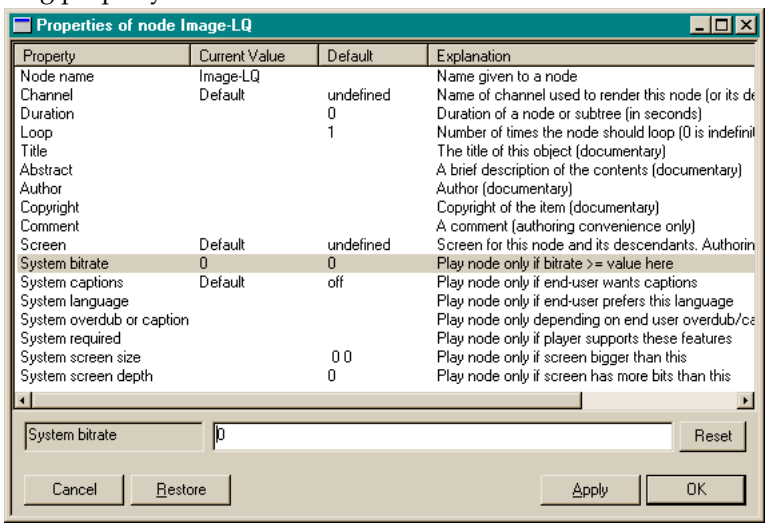
The SMIL Switch is a structure element, much like a sequential or a parallel node. The Switch does not have any rendering behavior in and of itself: it simply tells the Player that a number of candidates must be evaluated at runtime, and to choose the first item which meets all of the selection criteria attached to the objects in the Switch.

To see how the Switch can work within an application, open the Structure view and select the object labelled *Image sequence*, as shown at left.

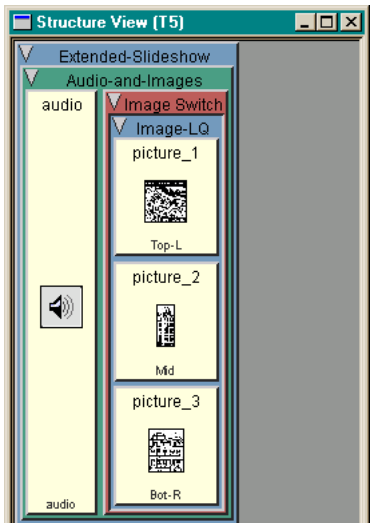
Open the property sheet for the node Image sequence, and make the following changes:

- change the Name from *Images sequences* to *Images-LQ* (for Low Quality);
- set System Bitrate to 0 (this is the default and allows playing at any bitrate.

The resulting property sheet is:



After you accept these changes, go to the Edit menu, and select:
New node special -> Switch parent.
This creates a new parent node to Image LQ of type Switch. Name the node *Image Switch*, click OK.

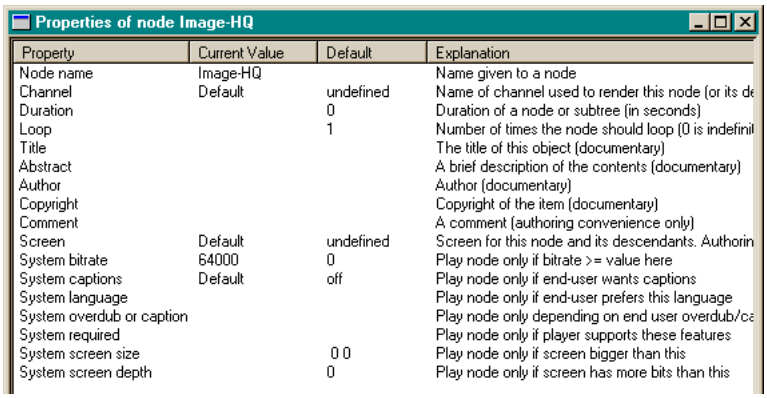


The new Structure view will contain a red Switch node containing the contents of the *Image LQ*. You will see the View at left.

Now, copy the entire object named *Image-LQ*, and paste it before the *Image-LQ* object using the Paste special -> before directive.

This makes a copy of the entire *Image-LQ* node and places it in before the old copy. Once this is done, go into the property sheet for the new version of *Image-LQ* and:
- change the name to *Image-HQ* (for High Quality);
- set the System bitrate to 64000.

The property sheet for the new object is:



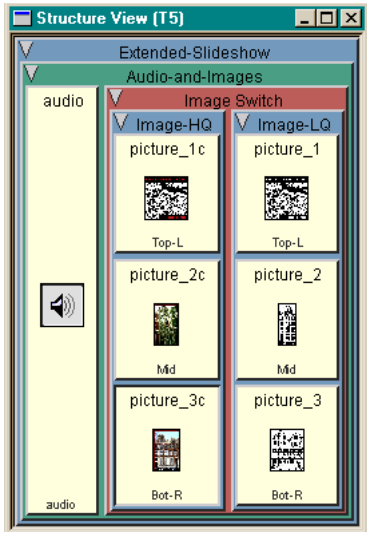
Property	Current Value	Default	Explanation
Node name	Image-HQ		Name given to a node
Channel	Default	undefined	Name of channel used to render this node (or its de
Duration		0	Duration of a node or subtree (in seconds)
Loop		1	Number of times the node should loop (0 is indefinit
Title			The title of this object (documentary)
Abstract			A brief description of the contents (documentary)
Author			Author (documentary)
Copyright			Copyright of the item (documentary)
Comment			A comment (authoring convenience only)
Screen	Default	undefined	Screen for this node and its descendants. Authorin
System bitrate	64000	0	Play node only if bitrate >= value here
System captions	Default	off	Play node only if end-user wants captions
System language			Play node only if end-user prefers this language
System overdub or caption			Play node only depending on end user overdub/ca
System required			Play node only if player supports these features
System screen size		0 0	Play node only if screen bigger than this
System screen depth		0	Play node only if screen has more bits than this

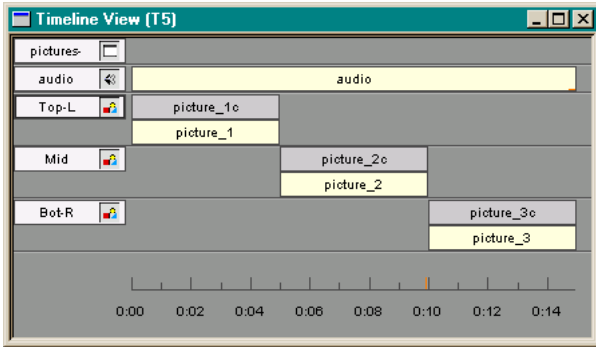
After creating the new structure, you should fill the media object references with new contents. Using GR*i*NS drag-and-drop, fill the contents of *Images-HQ* with the following objects:

- Picture_1: replace *map.jpg* with *mapc.jpg* (this is a color version)
- Picture_2: replace *h168.jpg* with *h168c.jpg*;
- Picture_3: replace *h218.jpg* with *h218c.jpg*.

You should also give each of the structure container a new name; this gives each object a unique ID, which will become useful when you make detailed timing references or hyperlinks within or across documents. Add a 'c' after each name to create the unique IDs.

The resulting Structure view is shown at right.





To get an idea of what will be happening over time, open the Timeline view. This shows the objects defined per Switch construct. (The bitrate was set to 28800 when this screen shot was made.)

You can now preview the presentation using the various Play options with GR*i*NS. Depending on the Preferences settings, you will get three different presentations:

1. If the *Internet connection speed* is set to less than 28800, you will see a sequence of black and white images, but you won't hear anything;
2. If the *speed* is set to between 28800 and 63999, you will see a sequence of black and white images, and hear the audio;
3. If the *speed* is set to 64000 or higher, you will hear the audio and see a set of color images.

NOTES:

1. GR*i*NS 1.0Beta has a redraw error that sometimes generates an incorrect Timeline view, or that refuses to show the Preferences dialog. If this (intermittent) problem occurs, please save your file and then restart the Editor and reopen your file. This will solve the problem in all known cases.
2. Not renaming media containers may cause your presentation to be rejected by some XML parsers. When copying and pasting collections of node, make sure you give newly-created nodes unique IDs.

Implications for Timing, Synchronization and Resource Use

You can make the contents of a Switch arbitrarily complex. The Switch can also contain nested Switches. When using the Switch, keep the following in mind:

- the Switch will play the first peer-level object that meets all of the enabling conditions set for that node.
- always order the contents of a Switch in a most-resource-intensive to least-resource-intensive order. This will allow your presentation to render correctly.

If you make use of complex levels of Switching, you may need to carefully design you other media objects to make sure that overall timing of your presentation is maintained. In general, avoid the use of explicit durations in non-continuous objects that are played in parallel with Switched elements — GR*i*NS will adjust timing automatically for you in these cases.

Closing Comments for Tutorial 5

This tutorial was meant to help you understand the basics of creating adaptive presentations. We used controlling bandwidth via connection speed as a convenient example, but there are many other ways to generate adaptive presentations. You should experiment with Language, Captions and alternative media encodings -- such as building a Switch that allows you to substitute a piece of text for a video under low-bandwidth conditions.

To get a better understanding of Switch construction and use, look at the presentations in the *Examples* directory.

Tutorial 6: Working with Hyperlinks

This tutorial is being revised for V1.0beta.

GR*i*NS Quick Reference Information

SMIL Compliance Information

The GR*i*NS Editor for SMIL and the GR*i*NS Player support the entire SMIL v1.0 specification, with the exception of the constructs listed below. Documents that make use of these constructs are parsed correctly, but the features are ignored during rendering.

- begin and end attributes in the anchor element.
- fit="fill" and fit="scroll" attribute values in region element.
- name="pics-label" attribute value in meta element.
- alt attribute in media object elements.
- fill attribute.

These features are expected to supported in a future release.

Supported Media Table

The following chart gives a listing of the media types supported by various versions of the GR*i*NS Editor and Player for SMIL.

MIME type	Extensions	Windows 95/98/NT	Mac	SGI Irix-6.3	Sun Solaris-2.5
audio/32kadpcm		no	no	no	no
audio/basic	au	yes	yes	yes	yes
audio/x-aiff	aiff, aifc, aif	yes	yes	yes	yes
audio/x-wav	wav	yes	yes (1)	yes (1)	yes (1)
image/cgm		no	no	no	no
image/g3fax		no	no	no	no
image/ief		no	no	no	no
image/jpeg	jpeg, jpg	yes	yes	yes	yes

MIME type	Extensions	Windows 95/98/NT	Mac	SGI Irix-6.3	Sun Solaris-2.5
image/naplps		no	no	no	no
image/png	png	yes (2)	yes	yes	yes
image/tiff	tiff, tif	yes	yes	yes	yes
image/x-portable-anymap	pnm	no	yes	yes	yes
image/x-portable-bitmap	pbm	no	yes	yes	yes
image/x-portable-graymap	pgm	no	yes	yes	yes
image/x-portable-pixmap	ppm	no	yes	yes	yes
image/x-rgb	rgb	yes	yes	yes	yes
image/x-xbitmap	xbm	no	yes	yes	yes
image/x-xpixmap	xpm	no	no	no	no
image/x-xwindow-dump	xwd	yes	no	no	no
	bmp	yes	yes	yes	yes
	fts	no	no	no	no
	pm	no	no	no	no
	ras	yes	no	no	no
	tga	yes	no	no	no
video/mpeg	mpeg, mpg	yes	yes	yes	yes(6)
video/quicktime	qt	yes	yes	yes	no
video/x-msvideo	avi	yes	yes(5)	yes (5)	no

MIME type	Extensions	Windows 95/98/NT	Mac	SGI Irix-6.3	Sun Solaris-2.5
video/x-sgi-movie	mov	no	no	yes	no
text/html	html, htm	yes (7)	yes (3)	yes (4)	yes (4)
text/plain	txt	yes	yes	yes	yes

Notes
1. Uncompressed WAV only.
2. Support seems to be somewhat buggy.
3. Very limited.
4. HTML 2.0 only.
5. Not all encodings supported.
6. With Quicktime 3 or QT MPEG extension installed
7. Both IE4/5 and WebsterPro controls supported

Support for RealMedia (G2) Data Types

The following chart describes the levels of support provided for the listed RealMedia data types used in the RealNetworks G2 player:

RealMedia type	Extension	GR*i*NS Editor		GR*i*NS Player	
		Recognized	Rendered	Recognized	Rendered
RealAudio	ra, rm	yes	yes	yes	yes
RealVideo	rv, rm	yes	yes	yes	yes
RealText	rt	yes	yes	yes	yes
RealPix	rp	yes	yes	yes	yes
RealFlash	rf	yes	yes	yes	yes

Notes

1. *Recognized* in the Editor means that RealMedia regions can be specified as part of a document. A G2-compatible presentation can be authored. Timing of RealMedia objects will not be displayed correctly.
2. *Recognized* in the Player means that data objects containing RealMedia items will be parsed correctly and will not cause the Player to crash.

Access methods

The GR*i*NS player supports the following access methods (URL schemes).

- file
- http
- ftp
- data
- gopher

References and Links

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