# Engineering Aspects of Web Hypermedia: Examples and Lessons from the GR*i*NS Editor

Dick C.A. Bulterman
CWI: Centrum voor Wiskunde en Informatica
Kruislaan 413, 1098 SJ Amsterdam, The Netherlands
Dick.Bulterman@cwi.nl

**Abstract.** This paper discusses the work-flow control features provided by the GR*i*NS editor for creating SMIL presentations. We start with an overview of the generic presentation creation process workflow and then introduce the general features supported by GR*i*NS. We then follow with a detailed set of examples of how these features can be used to create a simple slideshow of the type that can be played in mobile devices. We then provide an analysis of the use of GR*i*NS's feature set and contrast these with features found in other SMIL editors. We close with a set of directions for future work in supporting a presentation workflow.

## 1 Introduction

This paper looks at engineering-related aspects of creating and managing hypermedia content. Such an overview of the presentation modelling process is useful in understanding why creating media presentations — even for lightweight mobile devices — can be a complex endeavour.

In the sections below, we consider the requirements for creating a simple slideshow presentation. Such presentations represent a class of application that are likely to be supported on a wide variety of mobile devices that will be in the hands (literally!) of many millions of users world-wide. On the surface, such presentations should be easy to construct using a text editor and a simple presentation template. In reality, the complex interactions of presentation features, bandwidth limitations and device characteristics provide a demanding collection of authoring requirements.

This paper provides a use-case survey of the GR*i*NS editor [1]. The latest version of this editor provides a compelling collection of tools, but the high cost of the software has left most of the facilities to be unknown or under utilized. In anticipation of a more accessible version of this software as part of the Ambulant player project [4], this paper provides a general overview of GR*i*NS's facilities.

We start with a short description of the presentation creation workflow process. We then provide a general introduction to the facilities of the GR*i*NS editor. We follow with an overview of several aspects of creating a presentation from a standard template, including aspects related to presentation, environment and device modelling. We conclude with a brief analysis of other SMIL creation tools and provide several directions for future work.

## 2 Presentation Creation Workflow

Creating a slideshow presentation usually consists of supporting the following steps:

- *Define a presentation structure, either by specifying a custom design or applying a standard template*. For many presentations, a standard template (either wizard-based or via a text file) will be the initial choice for users creating slideshows.
- *Specify a target bitrate for the presentation*. Most users do not have a first-hand knowledge of performance issues associated with a presentation, but these aspects will contribute more directly to how the content is received and processed than any other aspects of the workflow. For desktop systems, this aspect is often overlooked, but for mobile presentations, performance aspects are of critical importance.
- *Select media objects and add them to the presentation*.
- *Preview the presentation*. Note that object selection and previewing occur as a feedback loop during design; for hand-edited presentations, this is not possible.
- *Fine-tune the presentation by adding transitions/animations*.
- *Publish to the desired media player* (such as the Ambulant open player [2]) or to a proprietary platform.
- *Upload to a media server.* This can be as simple as a local website, or as complex as an MMS delivery structure for mobile use [3].

A presentation can range from a single media object (such as a video) to a complex presentation that integrates hundreds of objects [4] Although SMIL provides a flexible multimedia container format, most of the presentations using SMIL have been of only limited scope. The presentation format described in this paper — that of a slideshow with images, text and background audio which adapts to the available facilities of the environment — is near the upper end of mass-production presentations.

## 3 Overview of the GR*i*NS Editor

This section considers basic features of the GR*i*NS/Pro editor for SMIL 2.0 and it discusses the special features of SMIL that GR*i*NS supports.

### 3.1 Overview of the GRiNS Editor

GR*i*NS uses a structured editing paradigm to describe streaming media documents. GR*i*NS exposes SMIL's <*par*> / <*seq*> / <*excl*> / <*switch*> structure to create maintainable, reusable and adaptive documents.

While the structured paradigm provides a powerful mechanism to free users from the constraints of timeline-based editing (which has fundamental limitations when working with switches, interaction and database-feed dynamic content), it is clear that many users are comfortable with a timeline based authoring model. For that reason, the GR*i*NS environment provides users with a unified *structured timeline* interface. The activation of the timeline is optional in GR*i*NS, where it can be activated selectively on individual structure containers (or *groups*). In general, GR*i*NS templates start by exposing a single presentation-wide timeline.

The GR*i*NS editor supports the following views of a SMIL 2.0 document:

- *Embedded Previewer*: the internal GR*i*NS player allows a user to preview the entire document, a sub-part, or a single node. Nearly all SMIL 2.0 functionality may be previewed directly in the editor before publishing to a SMIL player.
- *Structured Timeline view*: this view allows the user to create the logical structure of the document and provides support for all aspects of the SMIL 2.0 group hierarchy.
- *Layout view:* this view allows full creation/editing/maintenance control over presentation layout. The layout view also provides an integrated animation control and previewer. GR*i*NS/Pro provides extra support for multi-window layout.
- *Source view*: this view provides a complete source viewer and editor. The source editor has integrated syntax checking and error identification/resolution support. The source editor is context-coupled: objects selected in other GR*i*NS views are auto-highlighted in the source view.
- *Assets view*: the assets view allows a library of media objects and structure templates to be associated with a presentation template.
- *Transitions editor*: this view allows new transitions to be defined and configured.
- *Hyperlink view*: this view allows full control of internal and external anchors. It is also integrated as part of the linking creation functionality of the editor.

In addition to the functionality of the views described above, the Editor provides the following editing workflow support:

- publishing support for RealONE, including data object conversion.
- import conversion to native SMIL 2.0 for legacy RealPix content.
- integrated bandwidth modelling and pre-roll/stall prediction.
- support for multi-configuration previewing and modelling (including previewing for particular *bandwidth*/*language*/*systemTest* configurations).
- link and event creation and viewing using both accelerated (semi-automatic) link creation and full manual link control.
- integrated external media editor access directly from the editor.

GR*i*NS/Pro also includes initial support for export to the XHTML+SMIL profile (i.e., InternetExplorer 6.0 and later), to SMIL 1.0 players (such as Real's RealPlayer 8) and support for the SMIL Basic/3GPP profile for mobile applications.

## 4 The Creation Workflow

In this section, we consider how GR*i*NS can be used to create a simple media presentation. The presentation model is that of a slideshow, containing a background audio track and one or more foreground slides — each containing an image and an (optional) text caption. This type of presentation is typical of the expected use of MMS/PSS SMIL [3] within the mobile telephone market place.

GR*i*NS allows these kinds of presentations to be built using either an empty document or a document template. We will a GR*i*NS *Slideshow* template, since it focuses the discussion on user-level functionality.

```
[1]<smil  xmlns="http://www.w3.org/2001/SMIL20/Language"
          xmlns:GRiNS="http://www.oratrix.com/">
[2]  <head>
[3]   <meta name="title" content="GRiNS Mobile Slideshow"/>
[4]   <meta name="template_name" content="Basic Mobile Slideshow"/>
[5]   <meta name="template_description"
           content="A slideshow with background music and image."/>
[6]   <meta name="template_snapshot" content="T-icons/slideshow_P1A.bmp"
[7]   <meta name="project_html_page" content="external_player.html"/>
[8]   <layout>
[9]    <root-layout id="Basic_PPC_Slideshow" width="240" height="270"
        GRiNS:editBackground="192 192 0" GRiNS:showEditBackground="1">
[10]   <region id="audio" GRiNS:type="sound"/>
[11]   <region id="bk_image" width="240" height="270"
        GRiNS:type="image" z-index="0" GRiNS:editBackground="192 0 192"
        GRiNS:showEditBackground="1"/>
[12]   <region id="Images" fit="meet" top="7" left="7" width="233"
        height="240" z-index="1"  GRiNS:showEditBackground="1"
        GRiNS:type="image" GRiNS:editBackground="0 192 192"/>
[13]  </layout>
[14]  <transition id="fade" type="fade"/>
[15]  <transition id="slideover" type="slideWipe"/>
[16]  <transition id="push" type="pushWipe"/>
[17] </head>
[18] <body GRiNS:hidden="true">
[19]  <par id="BasicPPCSlideshow" GRiNS:thumbnailScale="false"
        GRiNS:thumbnailIcon="T-icons/slideshow_P1A.bmp"
        GRiNS:project_default_region_image="Images"
        GRiNS:project_default_region_sound="audio"
        GRiNS:project_default_region_text="Images"
        GRiNS:showtime="bwstrip" GRiNS:timezoom="14">
[20]   <seq id="BkgdClip" region="audio" GRiNS:emptyColor="#f7f388"
        GRiNS:emptyText="First Drop Background Music on Icon"
        GRiNS:emptyIcon="T-icons/dropSound.tif" />
[21]   <seq id="ImageSet" fillDefault="freeze"
        GRiNS:emptyIcon="T-icons/dropImage.tif"
        GRiNS:emptyDur="8s" GRiNS:emptyColor="#f7f388"
        GRiNS:emptyText="Drop Images on Icon"
        GRiNS:nonEmptyIcon="T-icons/dropImage.tif"/>
[22]   <img id="BkImg" fill="freeze" region="bk_image" src=Back3P.gif"/>
[23]  </par>
[24] </body>
[25]</smil>
```

Figure 1. A GR*i*NS Slideshow Template.

### 4.1 Select a Slideshow Template

A typical MMS-style presentation can be created by selecting the Basic Slideshow template, as shown in Figure 1. This template contains basic SMIL 2.0 structure definitions. The preview will allow content that is added to be analyzed based on the selected bitrate settings.

The template contains basic SMIL directives and a number of semi-empty structure containers. (For players that do not support the GR*i*NS namespace, these will appear fully empty, meaning that the template will always represent a valid document.) The meta information is used by the editor to dynamically support new templates. There are a series of editor-specific directive that are used to manage and customize the editor UI. In the body section, a <*par*> is defined that contains a background image and audio file and a set of images. In order to keep our examples simple, we have removed the text caption associated with each image.

When the template is opened, several document views are available. The default view is the *Structured Timeline* view. The Structured Timeline shows the SMIL 2.0 hierarchy of the presentation, mapped to a timeline. This timeline shows a green <*par*> (parallel) group and two blue <*seq*> (sequential) groups. Media objects are placed into these containers. The view in Figure 2 shows how the presentation above is represented to the author.

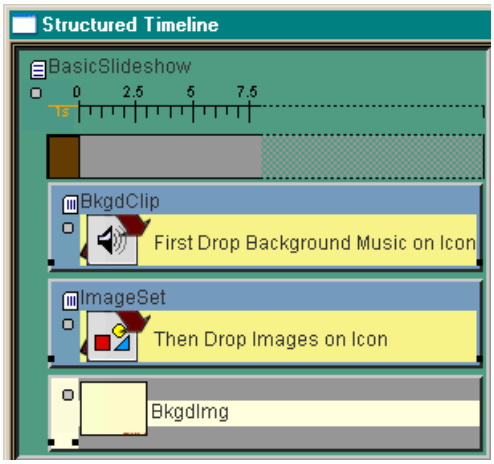

Figure 2. A GR*i*NS Slideshow Template.

At the top of the structured timeline is the *timescale* bar. This timescale is not fixed: it changes to highlight the timing of the presentation and to identify performance problems. Yellow or red gaps in the timeline indicate performance related issues that affect timing. A dashed timeline indicates that GR*i*NS needs extra space to draw structure.

### 4.2 Setting a Presentation Bitrate

Before the template is populated with media content, it is usually wise to define a target bitrate for the presentation. This bitrate is used by the GR*i*NS performance modelling system to help build a presentation that 'fits thru the pipe'. GR*i*NS provides substantial control over performance-related aspects of the presentation (such as defining alternative content that can adapt to the available client bandwidth), but everything starts with selection the baseline presentation bitrate.

The presentation bitrate is set via the previewer control panel shown in Figure 3. GR*i*NS uses this in setting values in the Bandwidth Usage strip and in selecting which objects get previewed in an adaptive presentation. (In the examples below, we'll initially assume a 56K Modem bitrate.) Note that for presentations with a wide range of test settings (language or device capabilities), additional settings fields may appear in the control panel.

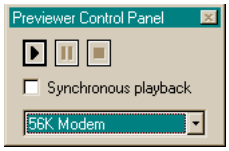

Figure 3. The GR*i*NS Control Panel.

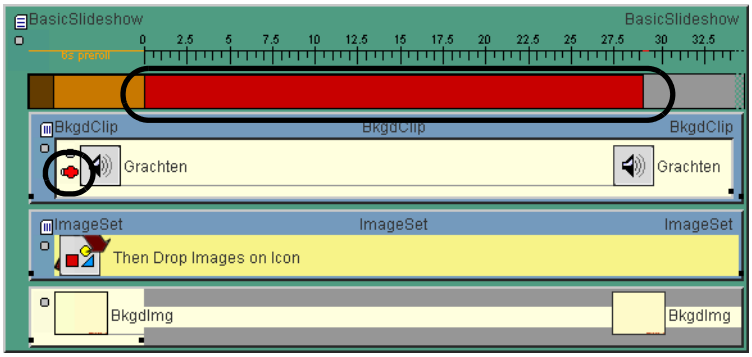


Figure 4. Adding an audio object to the presentation.

### 4.3 Adding Media Assets to the Presentation

The *Slideshow* template contains all of the information required to create a presentation *except* the actual images used and the background audio. The media objects can be added by selecting and dragging the object from the operating system file navigator and placing them on top of *Place Background Audio Here* for audio and *Place Images here* for the pictures.

Existing content can be replaced by dropping new object on the old ones. In general, timed media objects should always be added first (such as audio or video). Since these objects have the most critical streaming requirements, adding them first sets a baseline for performance evaluation of the presentation.

**Adding Audio Objects.** Figure 4 shows the effect of adding an audio object to the Structured Timeline. Note that the Bandwidth Usage strip is red and that the new audio node (*Grachten*) has a red blocked-pipe icon.(Both red areas are circled.) This is the GR*i*NS indicator that an object needs more bandwidth than is currently available.

The editor will provide an estimate of how much bandwidth is required to display the object by clicking on the blocked pipe. This gives a conservative indication of the bitrate (for continuous media) or time (for static media) deficiency of each item. The estimated over-subscription of resources for the audio node is shown in Figure 5.


Figure 5. Quantifying the bandwidth deficiency.

While a deficiency of 400 bps is not extreme, it does mean that all of the network's resources will be used to deliver the audio, meaning no time for images! Unless you are running a radio station, this is probably not what you want.

Often, resource over-subscription can be solved by down-sampling the media object. (This is especially true for audio.) In Figure 6, the music node is scheduled to be trans-coded from being a stereo, high fidelity audio clip to a mono, lower fidelity object. GR*i*NS has a built-in media converter that resamples files on export. To have the Bandwidth Strip use the post-conversion bitrate,

a post-conversion resource use rate can be specified in that object's property box. In our example, we turn on the *Convert Data* checkbox, select *28.8K* and set the audio type to *Music (Mono)*.

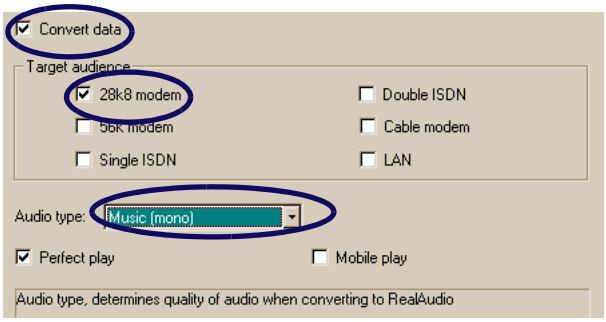


Figure 6. Specifying publish-time resampling.

Setting the audio conversion parameters results in a major reduction in bandwidth use, as shown in the Bandwidth Usage strip. Note that the pre-roll time is now 4s.

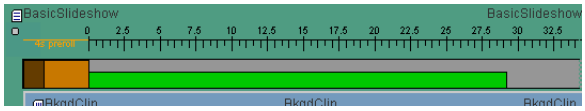


Figure 7. The effect of resampling on bandwidth use.

**Adding Images to the Presentation.**

After specifying the baseline temporal objects, images can be added to the presentation. The object can be added by dragging an image from the operating system file interface onto the container with the *Then drop images on icon* text. This results in the illustration in Figure 8.

This view contains very useful new information. First, the pre-roll time has increased to 9 seconds (the image is defined to appear at the start of the presentation). Second, the drop text has been replaced with an image icon. Third, a Drop Icon has been added to the end of the container. Note that we've also zoomed out the timeline to compress the view.

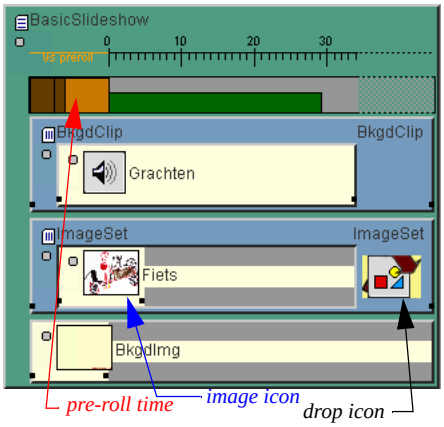


Figure 8. Adding the first image to the presentation.

Additional images can be added to the presentation by dropping them on the template.

The illustration in Figure 9 shows the effect of dropping five images into the presentation. The illustration also shows a number of performance issues (indicated by the blocked pipes and the tall blocks in the bandwidth markers). Often, they are caused by adding high-resolution images to a presentation (which the presentation device then needs to sub-sample). Like audio, these can be automatically down-sized during publishing but, unlike audio, it is often better to crop out unwanted content first.

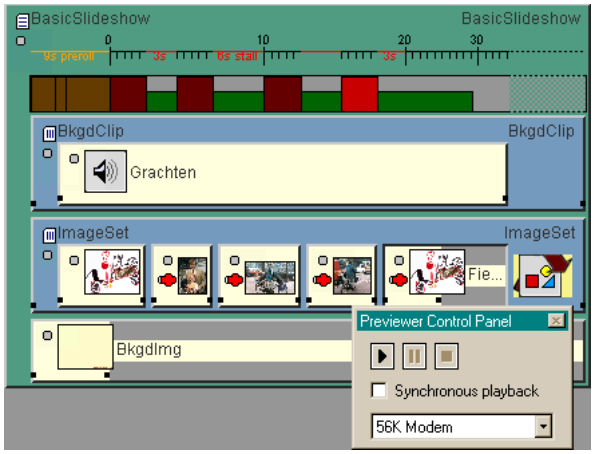


Figure 9. The raw presentation, oversubscribed.

### 4.4 Previewing the Presentation

One of the most important facilities than any editing system can provide is the ability to preview a work-in-progress presentation. GR*i*NS supports two choices: an entire presentation can be previewed from start to end, or a subset of the presentation can be selected for preview. The presentation subset can be as small as a single object or scale up to the entire presentation.

The fact that GR*i*NS has a built-in SMIL player — it is not simply a wizard interface — means that it can play selective parts of the presentation. This feature is supported internally by transforming the presentation rather than simply adjusting a timeline.

### 4.5 Making Adaptive Presentations

The presentation modelled in Figure 9 contains a number of problems that need to be resolved. The red pipes on media objects and the red fields in the bandwidth strip identify potential performance problems. To remove the red pipes (and thus get better performance), we can take one of two approaches:

- *design for the low-end*: take the minimum acceptable bitrate and design the presentation based on this constraint; or
- *design an adaptive presentation*: make a version that does something sane on a low speed connection, but also has some added features for users at the higher end.

**Designing for the Low-End.** To make a minimalist presentation, this bitrate should be defined in the player control panel. (In our example, we only consider bitrate, but there could also be restrictions on the memory in the device or other parameters.) Once the resource model is defined, GR*i*NS has a feature that allows scheduling adjustments of be made based on the available network bandwidth.

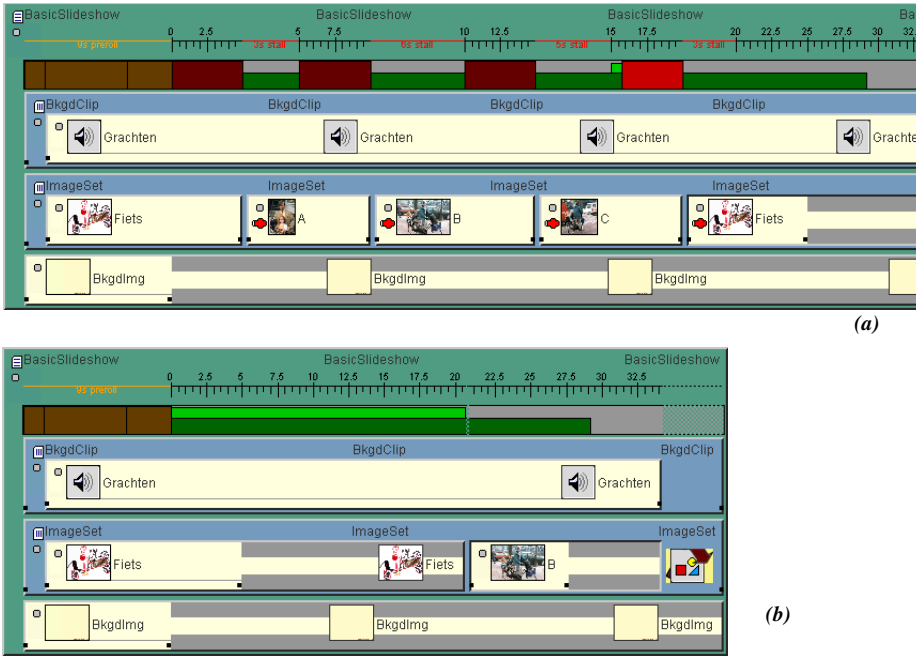


Figure 10. Automatically transforming a resource-constrained presentation (a) into a bandwidth conformant version (b).

By clicking on any of the pipe icons a pop-up box is brought up that quantifies the resource limitation in terms of a scheduling adjustment. Figure 10 shows the result of applying automatic timing corrections. Note that since the audio of is of fixed length, most of the images have been removed from the presentation to meet the temporal constraints.

**Designing an Adaptive Presentation.** So far, we've shown how the presentation would perform on a 56K Modem connection. If the bitrate in the Preview Control is set to 112K, Figure 11 shows that the original presentation had plenty of headroom for the audio and the images at the higher bitrate.

Since we have some extra resources available at the high end, it makes sense to construct an *adaptive* presentation. The strategy we will use is:

- the high-end gets the full presentation, and
- the low-end gets the same slides, but with no audio.

This is done in GR*i*NS by opening the audio object's *System properties* tab and setting a Bitrate constraint value to 112K Dual ISDN. This says: only play this object if the player has determined the bitrate to be 112K or above. (The 112K applies to the entire presentation, *not* the converted bitrate used by this single object.)

If the presentation is previewed at 112K or above, the slides and audio would be rendered. If the previewer was now set back to 56K modem, the illustration in Figure

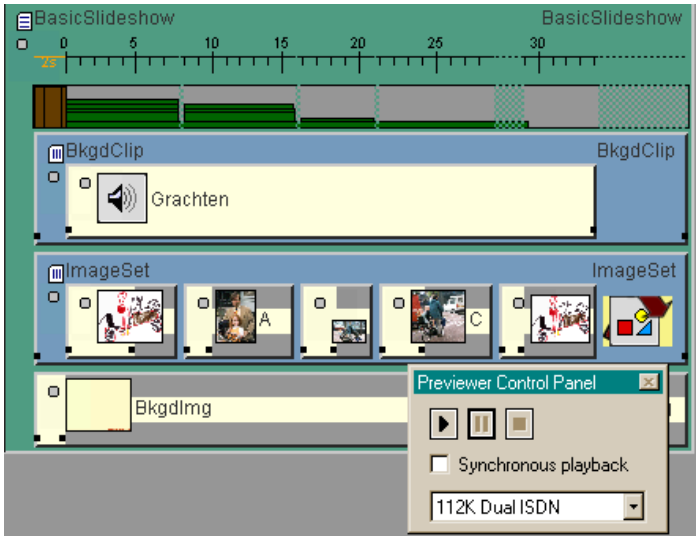


Figure 11. Evaluating the base presentation when extra resources are available.

11 would be seen in the Structured Timeline shown in Figure 12. The audio object is drawn with a dark background (indicating that it is inactive for this bitrate). If the presentation is previewed, all of the images — which consume all of the bandwidth — but the audio would not be rendered. (The nice thing about this approach is that low-end users don't know they are low-end: they simply don't hear the audio.)

There are many ways to manage complexity in a presentation. We could have also made separate slide sets (one with two images for low-bandwidth sites and one with five for high-bandwidth sites), or we could substitute video for images.

The ability to build adaptive presentations within GR*i*NS gives a powerful tool to address a broad audience without having to make several independent presentations.

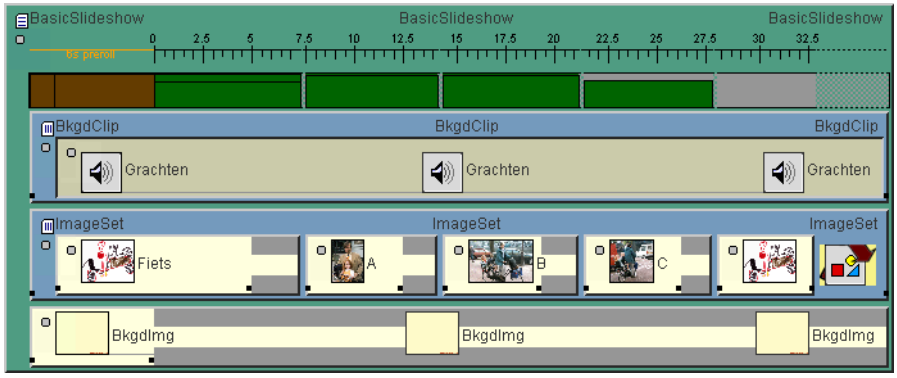


Figure 12. Visual feedback about the presence of an excluded object.

### 4.6 Special SMIL Feature Support

Thus far, we have discussed fairly standard aspects of defining an controlling a media presentation. The GR*i*NS interface also allows more exotic parts of SMIL to be supported. Several of these are discussed.

**Adjusting Presentation Timing.** GR*i*NS provides a uniform interface to specifying SMIL's *begin* time and *duration* behavior for all objects. Every object in a presentation — even <*par*> and <*seq*> and <*excl*> structure containers — can have implicit or explicit begin times. For <*par*> and <*excl*> containers, the implicit begin time is the scheduled start of the <*par*>/<*excl*>. For <*seq*>, the implicit begin time is the end of the preceding object

Every object also has a duration in the presentation. The implicit duration of an object depends on the object type:

- the duration of a piece of continuous media (audio/video) is the length of that clip;
- the duration of a discrete media item (an image or text) is set to the default in its template (in this case, 5 seconds) or it assumes the language default of 0 seconds;
- the duration of a <*par*> or <*seq*> is the result of computing a timeline for the contents of the container.

Every object can also have an explicit duration set that over-rides the implicit duration.

**Specifying the Fill Behavior.** An object's fill behavior determines what happens to the object when its duration ends. Does it disappear or do the bits stay on the screen until they get covered by something else? In the Slideshow template, all objects have a fill behavior of freeze. This means that if no new object is scheduled after the end of a current object, its rendered duration (but not scheduled duration!) is extended to the end of its time container.

The representation of *fill="freeze"* in GR*i*NS is shown in Figure 13; it is given by the colored bar that extends to the beginning of the next object or, if there is no successor object, to end of the time container. The illustration shows two sequences, each with identical objects — that is, they have the same begin times and durations — but with different fill behaviors.

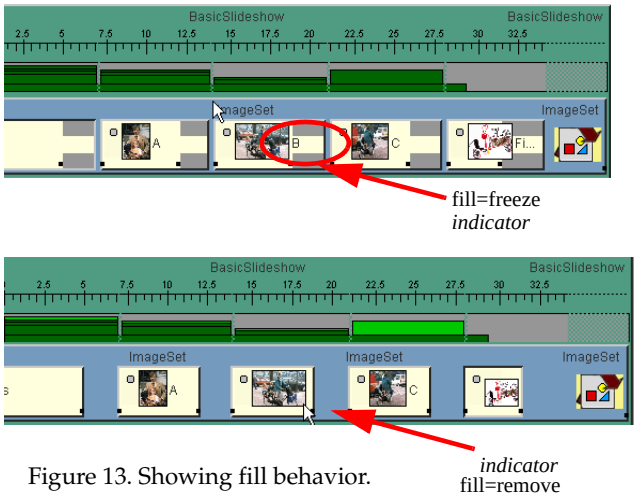


Figure 13. Showing fill behavior.

**Adding Visual Effects to Media Objects.** A streaming media player works hard to deliver streaming media on schedule. Once that media has arrived, however, the player can also help make things look more fluid and interesting by adding transitions, animations and layout positioning. The interesting thing about all of these features is that they consume zero bandwidth. They do, of course, make some additional demands on the client's platform.

GR*i*NS can be used to add transitions to a media object by opening the *Transitions* tab of the property sheet for the object and selecting an input or output transition. (New transitions can also be defined by this tab.)

The *rendering position* and *size* of an object can be animated in GR*i*NS, not the content itself. (For animated content, SVG objects should be placed in the presentation.) On the left side of the Layout window shown in Figure 14 is the region/ media selection tab. At top right is the placement window, and at bottom right is the Animate control. Animation is activated by enabling the Animate check box. The scale on the Animate bar defines the duration of the animation in terms of the active duration of the media object. (That is, excluding freeze time.) If the duration of the object is later made longer in the Structured Timeline, the animation will occur more slowly.

The slider can be placed to a particular offset to control the key time at which an animation is to take place relative to the local timeline of the object. The required size and position can then be specified for that key time. The player will compute the interpolation steps to control animation. SMIL uses a 'return to rest' animation model. That is, each key time defines a non-standard position, and the animation will return the object to its original setting.

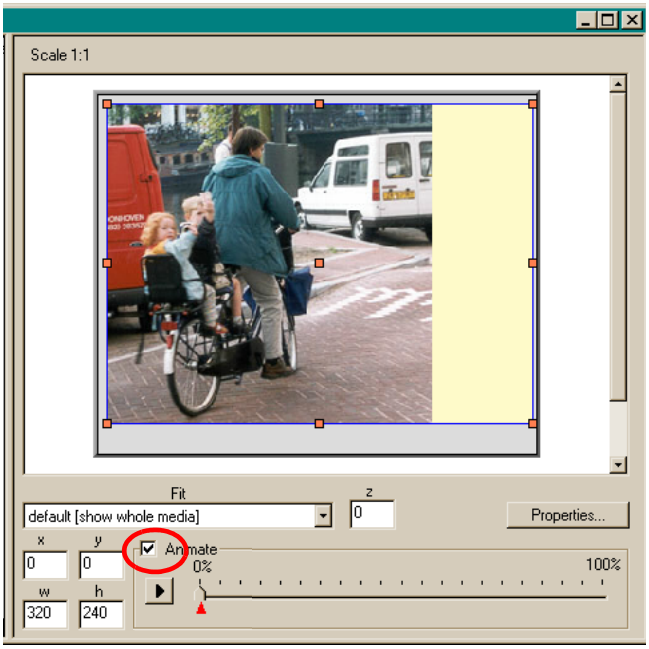


Figure 14. Combining layout and animation control.

**Adding Presentation Links.** GR*i*NS supports the construction of self-firing and external links. The following discussion shows a three step process for creating links to the external browser pane in the RealPlayer[5]:

1. The node in the presentation that will serve as the source anchor for the link is selected in the structured timeline view. (This is the node that, when played, causes the linked content to appear.)
2. Using the linking toolbar in the GR*i*NS/Pro Editor (shown in Figure 15), select the icon with an arrow pointing to the *browser* pane.


Figure 15. The *linking control* toolbar.

3. Enter the URL (either as a full Web path or as a local name) into the property dialog box for the newly created anchor. The property box and an example URL are shown in Figure 16.

Now, when you play the presentation, the Web page appears in the RealOne *media browser* pane. Timing offsets and other special attributes can be added via other GR*i*NS views.

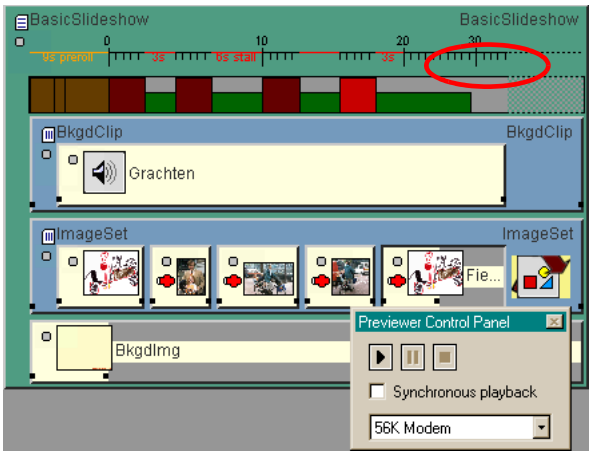


Figure 16. The *linking* property sheet.

## 5 Analysis of GR*i*NS Features

The GR*i*NS editor has been defined over a series of generations to provide sophisticated support for a wide variety of SMIL 2.0 applications. It is the most comprehensive SMIL editor available and has been applied to a range of scientific and professional applications.

During the development of the editor, a number of elements have become clear that will impact the design and implementation of any workflow support software. We summarize these points in the following list:

- In order for a multimedia presentation creation tool to be successful, it must be geared at presentation designers, not hand-coders of SMIL (or other) documents. Most hand coders prefer a text based interface and the flexibility of cutting and pasting code, while most graphic/ presentation designers appreciate the important side effects of being able to manage assets in a visual manner.
- An editing tool must be geared to presentation maintenance as much as initial presentation creation. As with software, the dominant long-term cost of multimedia design is the ability to reuse, update and incrementally expand presentation structure. Unless the editor is able to assist in this task, it will not be used.
- Incremental previewing of complex presentations is a fundamental feature that all editors must support to be accepted.
- Fine grain control over individual properties must be available for expert use, but these properties must be hidden for novices. Initial users of complex editing systems want to focus on solutions, not the highlighting of problems.
- Automated solutions to resource constraint management is a key area for additional work.

One of the frustrations of supporting complex visual editing of SMIL presentations is that most SMIL-based multimedia rarely has exceeded the complexity of a single slideshow. While examples exist of interactive news and other demonstrations, these are not yet being created by content authors. It is unclear if the dominant problem is related to the underlying technology or the education of the user community.

## 6 Comparing GR*i*NS With Other Editors

GRiNS provides a comprehensive visual interface to controlling the presentation workflow creation process and exposing all of SMIL's functionality. Several text editors exist that ease the process of source-level typing, but they provide no runtime feedback or bandwidth analysis [6], [7].

The LimSee2 editor [8] provides a standard timeline interface for creating slideshows that couples a storyboard model for placing individual pieces of media content on a canvas. The timeline allows constraints to be defined that govern application behavior, but there is no support for event-based interaction or for a-temporal scheduling (such as with the SMIL <*excl*> container. Users are constrained to only a basic subset of SMIL features.

The WGBH/NCAM MagPie editor [9] provides an interface for creating special-purpose captions in either RealText or Quicktime format as an adjunct to a single (typically video) media object in a SMIL presentation. It does not allow the SMIL presentation itself to be edited.

Adobe's GoLive HTML editor [10] contains a simple plug-in for creating basic SMIL slideshows, based on an adaptation of a Quicktime editor. It does not allow the definition of event-based, interactive, animated or adaptive presentations.

## 7 Conclusions and Future Work

The primary goal of this paper has been to survey a range of multimedia presentation facilities. Our expectation is that other designers can benefit from our experiences when designing new tools.

The design of the GR*i*NS editor has been an interesting exercise in the development of declarative authoring support for interactive presentations. The goals of the GR*i*NS environment were to be able to manage multi-dimensioned presentations that could cleanly adapt to various environment.

During the initial development of GR*i*NS, the resource limitations of the presentation workstations and the Internet have tended to relax based on new technology rather than because of user adaptation of SMIL code. It is clear that, for a new mobile generation of SMIL presentations, the technological progress will lag behind that of the public Internet. In this sense, we expect that there will be increased interest in supporting the multiplexing of presentation targets.

We expect that the following areas will be of key interest in the development of future versions of our environment:

- The development of more comprehensive active device models, in which technological and user constraints can be codified and processed by the editing engine.
- The development of multi-target publishing support so that a single presentation can be distributed on a wide range of target players.
- The integration of more extensive end-to-end presentation environment models.
- The integration of better support for automatic presentation adjustment so that resources can be managed in a more efficient manner.

It is also clear that continued attention needs to be paid to providing demonstrations systems for more comprehensive use of SMIL facilities. We are actively engaged in finding content partners in this area.

## 8 References


[1] Oratrix, The GRiNS/SMIL 2.0 editor. See: www.oratrix.com/GRiNS/ .

[2] Bulterman, D.C.A. et al, The Ambulant Open Source SMIL Player, Proc. ACM Multimedia 2004, New York.

[3] 3GPP, Multimedia Messaging Service (MMS): Functional Description, Stage 2 (Release 5), 3GPP TS 23.140.

[4] The Ambulant Project, The Ambulant News Demo. www.ambulantPlayer.org/ .

[5] RealNetworks, The Real Player. See: www.real.com/ .

[6] Allaire, The Homesite SMIL editor. See: www.allaire.com/products/homesite/ .

[7] HotSausage, Inc., The SMIL Composer SuperToolz. See: autodownload.sausage.com/ .

[8] INRIA, The LimSee2 SMIL editor. See: wam.inrialpes.fr/software/limsee2/ .

[9] WGBH/NCAM, The MagPie Captioning System. See:ncam.wgbh.org/webaccess/magpie .

[10] Adobe, The GoLine-6 Editor. See: www.adobe.com/products/golive/overview .