Generating Multimedia Presentations Automatically using TYRO, the Constraint, Case-Based Designer's Apprentice

Ron MacNeil
Visible Language Workshop
Massachusetts Institute of Technology
Cambridge, MA. 02139

ABSTRACT
TYRO [MacNeil 89,90] is a visual programming environment that uses a case-based reasoning approach to capturing and reusing knowledge about the design of multimedia presentations. In TYRO, the designer constructs a case library by demonstrating solutions to prototypical multimedia problems and defining constraining relations between object sequences. Adaptation and augmentation of the case library takes place as trial presentations reveal failure conditions. The designer constructs rule objects which are combinations of condition objects and action objects. Condition objects trigger when the failure condition is detected and action objects fire and revise the constraint network, or revise the sequence of objects. The resulting cases generated are stored and indexed for future use by the system.

INTRODUCTION
As multimedia communication replaces desktop publishing as the norm, the number of multimedia authors will increase, as will the need for multimedia design expertise in the workstation. As remote transactions over broadband networks increases so too will the need for complex data bases that can show you what they are all about using short, personalized, automatically generated multimedia presentations.

Conventional tools for building multimedia presentations like MacroMind Director and Videoworks provide only a cut and paste approach to reusing design elements.

TYRO is an environment for capturing multimedia design knowledge and reusing it to make automatically generated presentations. It provides a visual programming front-end to a powerful object-oriented inferencing environment interfaced to a multimedia graphics environment. With it the designer builds a case library by demonstrating solutions to typical design situations and helping the system generalize them to generic templates where possible. New presentations are generated when the system's planner takes a goal statement as input and builds output scripts by adapting and concatenating applicable cases.

The underlying approach taken in TYRO is a combination of constraints [Leler 88] and case-based reasoning [Riesbeck, Schank 90]. We like this approach because it seems to be the way people and especially designers think and work. Young designers learn their craft by studying the work of the old masters and then trying to solve new problems with those as context. They develop over time their own canonical approaches, or "style".

Design Example
The example we will work with was developed by a young multimedia designer, Steve Librande, at a small design firm which specialized in low cost promotional slide tapes. The goal of the piece was to give viewers facts about the company, a sense of the employees and the kinds of jobs they do. The form of the piece is simple and elegant. The broad outline of the piece follows the classic sequence: introduction, body of communication, recapitulation. There is essentially one module which gets repeated 8 times with variations. The module begins with an approximately 10 second lead-in comprised of a company logo as background with engaging music or sounds. Near the end of the lead-in the logo is overlayed with an image that summarizes the section to follow, then dissolves into the main body of the section, usually a short biography or job description with accompanying engaging music or sounds.
TYRO Environment
TYRO is built in Apple’s Macintosh Common Lisp [Steele90], Common Lisp Object System (CLOS) [Keene89], ARLOtje/IDX [Haase91] and runs on a MacSE/30 with 8MB of RAM and a SuperMac high-resolution color card.

ARLOtje is an experimental representation language intended for building representation languages and provides basic inferencing mechanisms, IDX is a recursive and persistent framework for media description and representation. Together they support symbolic constraints, case-based and analogical reasoning.

Logging the material with ARLOtje/IDX
Logging the material is one of the first steps in any multimedia project. In order for the reasoner to operate we must represent both the media elements to be used in the presentation as well as the elements which are the subject of the presentation; the characters, objects and actions of the story line. And we need to represent the story as well.

Media elements are organized into browsers either by media type or subject matter. Fig. 1 shows several browsers, which provide a way to organize and access the various element types. The ARLOtje/IDX Browser shows the representation we have built for the 'accident' script. On the bottom of the browser is a representational hierarchy which we have pruned to a few elements of interest. On the top we are adding an annotation to the image of the dispatcher on the phone, to the effect that in this particular image she is talking to a driver.

ARLOtje has a parser for stories that builds a representation of the story structure and content from the objects it has on hand. Fig. 1 also shows the story editor with the simple story outline that describes the central organizing drama of the presentation. The Planner uses the story representation as a guide to choosing elements when generating new presentations.
TYRO's Multiple Views
The cases and elements that make up each case can be viewed from three different perspectives: logical, spatial and temporal.

In the Logical View, Fig. 2, the element types are organized according to the constraining relations between them. This view serves as a debugging aid, helping the designer make sure that the correct constraints have been invoked, and makes retracting constraints, cutting, and pasting easier.

The Temporal View shows the co-linear temporal relationships between the media elements themselves as well as graphic controls for building fade and loudness constraints on images and sound, and position and size constraints on images.

The Spatial View is where 2-D composition of graphic elements is done, shown in Fig. 3.

Building a Generic Template
Remember that case-based reasoners solve new problems by adapting solutions that were used to solve old problems. If the old problem is different enough from the current problem the reasoner will need some abstraction of the old problem. Conventional, non-visual, case-based reasoning systems have built-in methods for developing those abstractions. In TYRO they take the form of Templates, and it is the job of the user to provide them.

Templates are the generic "recipes" or "scripts", the raw outline of the multimedia items and their constraining relationships, temporal and otherwise.

The Planner uses these much the way the user does; they get chosen according to their applicability to the task at hand, and then the details get filled in.

Fig. 2 shows a generic template built to support a recurring specific kind of event, the visual synchronization to the rhythmic unit 2,2,4. It also illustrates use of multimedia constraints.

Multimedia Constraints
Multimedia systems must have a facility for guaranteeing that multiple events happen in parallel at the prescribed time. TYRO makes use of a beta release of Apple's QuickTime facility which can not only play digitized video, sound and other event files at any rate up to the limits of the hardware, but can coerce one rate to another, which makes possible both unidirectional and bidirectional constraints.

TYRO affords graphical interfaces to symbolic bidirectional constraints on top of QuickTime. Fig. 2 illustrates the constraint "fit-visual-to-audio", one side of the bidirectional constraint. We have selected the four notes we wish the last visual element to coincide with and have then invoked the constraint, chosen from the menu of constraints applicable to these element combinations. The constraint manager calculates the new value for coercing the visual element, and Quicktime does the coercion at run time.

In the Temporal View one builds unidirectional constraints directly on loudness and fade by using controls on the audio and visual channels, see Fig. 3. Clicking on a new place in the graph adds a new line segment in the graph representing the attenuation values.

Building the Case Library
Cases are the telling examples, the chunks of design experience the designer feels are worth saving. In our example presentation there is a module or "phrase" which recurs with variations many times. One version of this module is about what an ambulance dispatcher's job is like, and what kind of a person makes a good dispatcher. It begins with the company logo fading in from black, accompanied by a bright bass guitar rhythm, melody for about 4 seconds. Then the dispatcher begins talking and an inset portrait of him fades up against the the logo background. Then the logo and portrait cross-fade to a sequence of other cross-fade images that relate to the monologue and to the overall script as well.

This assembly process starts by finding the elements in the various browsers that might be applicable. In Fig. 3, we show an image browser. A track is cho-
sen and the selected element is sent to that track, where it can be dragged to any time position. The window can be scrolled and zoomed out to a factor of 8, to get a wider view. If the duration of an item needs to be changed from the default of two seconds, each element has a handle on its right side which can be dragged, and in the case of image elements causes the right hand edge of the image to be repeated to show the position of the new duration.

The Spatial Editor can be used to do a trial playback of a sequence to check timing. It can also be used to constrain the position and size of graphic elements by moving and resizing the element directly at any given Temporal Editor time setting. If one wants to create a movement or change a parameter over time one can either demonstrate the change by moving the element in the Spatial Editor under "demonstration mode" or can modify a graph in the Temporal Editor associated with the parameter. A size control graph shows that the proportional size remains constant over the portrait's duration, although somewhat smaller than full frame. If the user wishes to constrain parameters of an element in relation to some parameters on other elements, the propagation constraint system described in earlier papers [MacNeil 89], [MacNeil 90] is available.

Fig. 3. also shows how a new annotation can be added to the prototype of an object found in an image or other element. We want the close-up image of the dispatcher's mouth and headset microphone and to appear just after the word "communicate" is spoken. After placing the image and selecting the word in the Transcription track, then double clicking on the close-up image, the Image Element Browser discloses the image representation. By further clicking we can edit the prototype so that every subsequent image of a mouth will know that it can be used for communicating, a fact that is obvious to us, but was a piece of missing knowledge until added.

Building an Interview Template
After constructing several of these interview cases, one from the point of view of the president, an EMT, an ambulance driver, we realize that we can save ourselves some time if we make a generic template for this type of interview case. The system has a rather unsophisticated tool to generate such a template automatically by instantiating the generic element types and setting parameters at a value which is an average of the several source cases, Fig. 4.
Adaptation

We still don't have a case that will work well as an introduction to the presentation. The introductory case is simpler than the dispatcher case in that it isn't showcasing an individual but trying to set the stage and focus on the company as a whole. The solution is to linger on the company logo longer and then go directly into the story, without superimposing a portrait of an individual.

We could build a special case for this but instead we will illustrate an adaptation technique.

In case-based reasoning the adaptation process looks for salient differences between the retrieved case and the input and then applies rules that take those differences into account. In TYRO those differences are detected by a predicate held in a Condition object while rules are held in a Rule object.

Building the Condition Object.
The process starts in the Condition Editor, Fig. 4. The goal of the Condition Editor is to help the user define a predicate which captures the event or confluence of events, and save that inside an instance of a Condition object. When the 'detect' button in the Condition Editor is pushed, the system looks at the state of the selected objects, or the objects which are enabled at the current time point in the temporal editor, and tries to build a predicate. If the request is ambiguous a dialog box pops up with the alternative objects or variables from which to choose.

In this scenario the condition we are trying to detect is that the phrase being constructed is the first phrase. In the Planner's terms this translates into a Lisp predicate of the form '(equal *phrase-number* 0) which the user can simply type into the pattern field in the Condition Editor.

Building the Rule Object
We want the planner to change the 'most-applicable-case' that it retrieves to better fit the criteria for an introductory phrase. In this simple example all it needs to do is remove the portrait that is overlaid onto the logo background at the beginning of the case. But how do we tell it to do that?

In the Rule Editor, the user chooses remove-element from the menu of commands, and next selects the element to be removed from the retrieved case's parent template, the Interview Template. The system makes an action object and puts in the THEN part of the Rule Editor. The bottom Temporal Editor in Fig. 4 shows the phrase which would be constructed when that rule fires.

Related Work
HyperAdaptive, [Fineblum 91] automatically generates a multimedia presentation by using knowledge-based representations of different presentation styles as guides in its search through a database of richly described multimedia data.
MUSE [Hodges-al89] is a toolkit for building multimedia learning environments developed at MIT's Project Athena. It integrates full motion video with graphics and sound, and has the notion of constraints between multimedia objects.

COMET [Feiner, McKeown90] is an environment for generating multi-media explanations developed at Columbia Univ. Its architecture includes a single content planner, media coordinator, bidirectional links between text and graphics generators and a media layout component. Each of these components has at least one knowledge base.

The Electronic Scrapbook [Bruckman91] is a system to help users edit home video. Semantic knowledge representation is used to log the video and then simple knowledge-based templates help the user to construct stories.

The Director's Workshop [Davis91] is an iconic interface for the logging of multimedia information in domain-independent, multimedia archives. The interface makes use of a temporally indexed, semantically structured representation of multimedia content for logging, searching, composing, and repurposing temporal multimedia.

Conclusions and Future Work
TYRO has been extended with the addition of multimedia element Browsers, Temporal Editor, ARLOtje/IDX Browser to allow multimedia designers to build case libraries which represent high level chunks of design knowledge. Work needs to be done in several areas. Earlier versions of TYRO allowed the user to build most of the primitive constraints and extend object types in Editors using visual programming techniques, and this capability needs to be restored. The logging process and the interface to ARLOtje/IDX needs to be more intuitive. Finally we need to find ways of visualizing the reasoning process so users can extend the reasoner's capabilities.

Acknowledgements
Major support for the work described in this paper was provided by NYNEX and DARPA. I would like to thank Nathan Felde and Sergio Canetti at NYNEX's Media Lab as well as Prof. Muriel Cooper, Prof. Nicholas Negroponte, Dr. Henry Lieberman of MIT's Media Lab for their continued support of this project.

The Visible Language Workshop is also supported by grants and equipment from Paws Inc., Apple Computer, Digital Equipment Corp. and Hewlett-Packard.

This effort was sponsored in part by Rome Labs, by the Air Force System Command and the Defense Advanced Research Projects Agency (DARPA) under contract # F30602-89-C-0022. The views and conclusions of this document are those of the author and should not be interpreted as necessarily representing the official policies, expressed or implied, of the RADC, of the AFSC, DARPA or the United States government.

References
[MACNEIL 90] Ron MacNeil, ADAPTIVE PERSPECTIVES: Case-Based Reasoning with TYRO, the Graphic Designer's Apprentice, Proceedings of the IEEE 1990 Workshop on Visual Languages.