Experiences with multi-level scripting for responsive multimedia

Stefan Agamanolis and V. Michael Bove, Jr.
{stefan, vmb}@media.mit.edu
MIT Media Laboratory
20 Ames Street, Cambridge, Massachusetts 02139 USA

ABSTRACT

Responsive multimedia refers to a melding of media forms that has the ability to react in some way to audience identity, circumstances, equipment, and actions. Cutting-edge responsive multimedia works often involve a collaboration among many people of varying expertise and require innovation at many levels, from the incorporation of a new sensor or display technology to the structuring of media objects and their behaviors. Tools should therefore be designed to support the multi-level nature of such projects, in terms of both authoring and user-base. In this article we present a scripting environment designed with this goal in mind and our experiences using it to develop several major multimedia prototypes ranging from interactive stories to live communication systems.

Keywords: scripting languages, responsive multimedia, multi-level authoring

1 Introduction

In a classic medium, a message is conveyed from a transmitter to a receiver over a one-way channel. We define a responsive medium as one in which the channel is effectively two-way—where information from the receiver may be sent back to the transmitter and utilized to modify the message in some useful way. Responsive multimedia can then be thought of as a melding of several media forms that has the ability to react to information gathered from the receiver, such as viewing equipment or conditions, audience identity or profile, or direct interaction. In the latter case, the more common term interactive applies.

When responsiveness is mediated by a computational device, many interesting opportunities to enhance the effectiveness of a message for particular circumstances arise, allowing for a much finer granularity of control and increasing human leverage throughout the process. Key elements of artistic expression and storytelling, such as mystery and identification, can be heightened and even directly driven by the audience of an experience. Device-mediated responsiveness can also serve to enhance interpersonal communication and collaboration—making communication spaces smarter and filling in where natural human capabilities fall short.

Creators of new responsive multimedia experiences have many kinds of authoring environments from which to choose, each of which approaches the practical problems of media design in a different way. Each of these approaches, in turn, has its advantages and disadvantages.

Regarding basic system functionality, many tools, especially those developed commercially, are intended for creating content for mass consumption and must therefore cope with limitations in what interface and delivery options are available. Mainstream input devices, like keyboards and mice, are often the only options because presentations must be able to operate on any of the current popular platforms. Output is often limited to a single computer monitor and one pair of speakers. Functionality may be restricted to simple operations on image and sound data and limited by processing capability. The requirement of platform independence can introduce extra complexity, both internally and externally. If mass consumption is not a concern, then it is important for the tool to extend and adapt easily and efficiently to special circumstances.
Tools also differ greatly in the kind of interface they provide to an author for harnessing a system's power. Non-scripted visual platforms have the distinct advantage that there are no written programs and all work is done graphically. Not surprisingly however, because they often operate at a high level of abstraction, they can also overly constrain the creativity of the author by forcing the use of a particular set of mainstream transformations or certain models of media presentation such as card-based, frame-based, or hyperlink-based metaphors. On the other hand, completely text-based scripting systems may carry a higher level of expressivity, but they sacrifice some of the intuitiveness that often comes with visual representations of the authoring process. Visual or not, it is hard to imagine adding support for a new sensor or display technology without writing some specific low-level code to drive the device.

2 Multi-level authoring

Given these simple observations, it is no surprise that the most popular tools allow the creator to work at different levels within the same system. The visual interface of Macromedia Director and its companion scripting language Lingo is one of the best current examples of such a multi-level environment. The visual part of Director allows the creator to invent simple object motions and transformations using a frame-based timeline metaphor. When the visual layer does not provide a needed operation, Lingo may be used to express more complex processes and relationships, including user interaction. Various so-called "Xtras" or externally written libraries serve to extend the functionality of Lingo in various ways to handle nonstandard devices, access databases, and so on.

The Apple Media Tool, Supercard, Toolbook, mTropalis, Everest, and Gain Momentum are some additional examples of commercially produced tools that feature both a visual interface and a scripting language. In some cases they also allow for extensibility by means of traditional computer programming languages. Digital Chisel and the Summit Authoring System have changeable "user expertise" settings, from novice to advanced, that modify the authoring interface in various ways.

It would also not be unreasonable to say that HTML, JavaScript, and Java form a multi-level suite of tools for authoring content for the world wide web. At the highest level, HTML is used for raw text and formatting information. JavaScript provides simple scripting functions from within an HTML document but not available directly in HTML. Java then provides full low-level programming capabilities and the ability to access external platform-specific libraries.

The problem with some multi-level tools such as those described above is that the individual levels often do not complement each other, perhaps because they were not designed to be multi-level from the start but rather emerged that way over time, either intentionally or accidentally, because of the needs and demands of their users and their projects.

Lingo was not part of the original design of Director—it was added later, as a response to the need for greater extensibility. In many cases, Lingo overrides many of the operating assumptions made in the visual part of Director, creating a situation where novice users have to relearn some of their expertise when moving to Lingo. For example, use of "puppet" sprites in Lingo is a major departure from the way sprites are controlled in the strictly frame-based visual score interface. As more and more operations are "puppeted", or moved to Lingo control, the score no longer expresses what is happening in a production and becomes more of an encumbrance rather than an aid to the authoring process.

JavaScript, originally known as LiveScript, was developed to look like Java so that experienced Java users would have less trouble using it. But beyond language syntax, there is not much similarity between the two. Their interpretation is handled by completely different engines. It is difficult to manipulate Java objects from JavaScript and vice versa, and HTML is effectively isolated from both, mainly because a collective design strategy underlying all of these layers is lacking. Such issues stand as barriers to those developing prototypes that push the limits of the medium.
3 Projects

In order to illustrate some of the issues faced in developing responsive multimedia applications, below we describe four major projects and demonstrations that we have built at the MIT Media Laboratory. What all of these projects have in common is that they were created within the same scripting environment, Isis. Following these examples we present details regarding the multi-level design of Isis and how it was harnessed in each case.

3.1 The Museum

An early experiment in our group was intended to explore the possibilities for capturing the director’s intent when video is to be displayed across a broad range of display sizes and aspect ratios. Beyond merely rescaling the video or cropping it as the screen or window changes, we wanted to specify more complex behaviors. The Interactive Cinema and Television of Tomorrow groups at the MIT Media Laboratory in late 1994 produced a short Isis video titled The Museum in which the set was represented as a three-dimensional model produced from multiple uncalibrated two-dimensional camera views.[1] and the actors and other foreground features were specified as separate two-dimensional objects which could be composited back into the rendered background.

The problem faced here was how to maintain the actors’ expressions at a recognizable size across a broad range of image sizes, while simultaneously giving a sense of the setting. Image scaling would make the actors too small on small windows, while cropping would remove too much of the background. The solution (Figure 1) was to change both the focal length and distance of the camera smoothly as a function of display size—a difficult move with real cameras but relatively simple with a virtual one. A more subtle effect slightly shifts the camera side-to-side with size to minimize the intrusiveness of the hanging empty picture frame in the scene.

3.2 Wallpaper

In 1995, we set out to break from traditional forms of storytelling and create a new kind of responsive narrative experience. Instead of allowing the viewer directly to control elements of the intrinsic narrative (like plot and characters), we decided to experiment with a possibly more fruitful model of interaction in which the experience controls both the spatial and subjective points of view from which a story is told. Changes in the point of view
Figure 2: Wallpaper allows viewer control of a virtual camera along a constrained path so that the action is always visible.
are rendered by altering aspects of only the *narration* (such as editing, shot composition, music, sound effects) of an underlying story that remains constant. Our purpose was not to take control away from the director, but rather to give the director more freedom to control how a production would change when told under different circumstances.

The video is based on a film adaptation of the short story *The Yellow Wallpaper*, by Charlotte Perkins Gilman, and content was captured while the film was in production. Scenes of a present-day woman (Kathy) writing this story are interwoven with the world of the character she is creating in the past. The film revolves around her struggle to contact and understand the parallel world in the past while at the same time contending with an abusive husband (John) in the present who is completely oblivious to her inner turmoil. This particular adaptation was perfect for our purposes because of its simple yet powerful conflict between the woman and the man, offering two extremely opposing points of view of the same story world and presenting many opportunities to explore the nature of reality and perception.

To facilitate spatial movement, using a process developed at the Media Laboratory, a three-dimensional model of the scene was extracted from uncalibrated two-dimensional photographs of the room where the original film was shot. This model was used to synthesize camera views of the scene that were not originally captured. Actors were shot separately, from several calibrated angles simultaneously, on a blue-screen stage.

Three master “presentation variables” control different aspects of the playout:

- *Camera pose*—controls the position of the virtual camera in the room.
- *Closeup intensity*—controls the general number of close-ups that are shown during playout.
- *Story perspective*—expresses what subjective point of view of the story should be presented.

Instead of giving the viewer a full six degrees of freedom in movement, we restricted camera motion to a specific path that passes near the angles from which the actors were originally shot. This was done to prevent the viewer from missing action and to ensure that synthetic angles would look as convincing as possible. The appropriate actor views, of the five different views captured, appear in the correct position and scale in the virtual space according to the current camera pose. Remarkably, this produces convincing results, with much less of a “ratcheting” effect than one might expect (Figure 2). Spatialized audio is rendered for the corresponding viewpoint.

The story perspective variable ranges between the two extremes of John’s and Kathy’s points of view, and the system is able to generate a blend of their perspectives by interpolating the playout parameters associated with these two extremes.

The actual presentation is responsive to both the display conditions and direct user interaction. Shot composition is altered based on the display size. For example, at one particular point near the beginning of the scene, the viewer may see the master shot or a close-up of Kathy typing, depending on the size of the output window (Figure 3). More
Figure 4: Shot composition varies in Wallpaper based on the current story perspective setting (which is controlled by the viewer). These three images represent a John-biased, neutral, and Kathy-biased composition at one moment during the story.

Interestingly, near the middle of the scene, the viewer might see John grabbing a rag from the shoulder of a dress form in a dark gray corner of the room, or the same action superimposed over a bright cloud background, or John grabbing a handkerchief from the hand of a third mysterious character, all depending on the current setting of the story perspective variable which is controlled interactively by the experimenter (Figure 4). The virtual acoustics of the room are smoothly interpolated between two extremes, as are the volumes of the ambient sounds. A small room with almost no reverberation is at one extreme (John’s perspective), while a much larger and quieter room with a high constant of reverberation is at the other extreme (Kathy’s perspective).

The narration space created in the implementation gives a very cold, abusive, claustrophobic tone to the scene as perceived by John. Backgrounds are full of solid greys. The typewriter is loud and piercing, and a harsh wind sound is heard in the background. The acoustics are tuned so that any noise seems very close by. Kathy’s point of view is completely the opposite. Her inner world is spacious and liberated, almost as if she is not in the room at all. Colorful cloud images are used as backgrounds in several shots. The sound of the typewriter is muted, and the wind is still present, but it is more distant and is heard accompanied by chimes. The acoustics are modified so that the room sounds extremely large and more reflective. Kathy’s world is also inhabited by the character she is writing about, and consequently fleeting images of this third entity appear at various points.

3.3 Sashay

One of the foci of research in our laboratory’s Interactive Cinema group is the harnessing of alternate modes of interaction to enhance dynamic story experiences. Sashay, an installation conceived of by Freedom Baird, allows a participant to interact with a virtual video character; the Sleeper, by using gesture.[3] As each gesture is interpreted by the system, a corresponding animated event takes place on a wall-sized screen facing the viewer (Figure 5), such as a spiraling image of a ghost or a shower of images of cigarettes. As more gestures are performed, these animations accrue on the screen and play out simultaneously. Electric field sensing technology, designed by Josh Smith and Joe Paradiso from our Physics and Media group, is employed to capture hand gestures.

The multi-layered animated “dream” constructed by the viewer affects the state of the Sleeper. Internally, two variables, one representing wakefulness and the other agitation, are modified based on each new animated layer’s emotive quality. The participant can build a peaceful animation that soothes the character to sleep, one that produces nightmares, or perhaps one that calms the Sleeper but keeps her awake.

A large database of video clips and musical score elements are edited together on the fly to convey dynamically the current state of the Sleeper. To determine which clip and sound file will play next, a concise set of rules is followed that were designed to maintain cinematic continuity. For example, two close-ups of the Sleeper may not be shown in a row, and a shot of the Sleeper lying down may not be followed by a shot of her standing up without first inserting a clip of her rising from her bed.
3.4 Reflection of Presence

In contrast to the preceding examples which use pre-recorded audio and video objects in their output, Reflection of Presence is an example of an intelligently responsive system for the composition of live objects to reflect presence and center of attention. The ultimate aim of this ongoing project is to develop more natural and intelligent environments for interpersonal communications and telecollaboration. While other researchers have presented videoconferences in which all participants appear in the same window [4][5], and the idea of automatically identifying the “center of attention” in a multipoint videoconference can be traced back at least 25 years [6], this system is novel in that it is implemented totally as a set of cooperating scripts, enabling rapid experimental changes in the behavior of the prototype, and allowing video and audio each to affect the presentation of the other in useful ways.

The software layout of the system consists of one server and several clients all running on different machines and connected via TCP. Audio and video are captured on each client workstation and an image of each participant is segmented from his or her surroundings using a background-adaptive luminance/chrominance difference algorithm. These data are sent to the server whose job is to compose the images of all the participants into a single frame and output it for distribution back to the clients. This basic cycle needs to occur in real-time, many times per second, with relatively low lag, and needs to be synchronized with the distributed audio as well.

One of the most intriguing aspects of the system is its “reflection” metaphor. Each participant’s image is reversed left-to-right, such that it looks like looking into a mirror, but a mirror also occupied by other participants, creating a shared space with unique social characteristics, the psychological impact of which will be a subject of further study as its development progresses.

Beyond performing the compositing task, the system listens to how loudly each participant is talking and watches how vigorously each is moving. This information is used to make judgments about who is the center of attention, which is then used to drive decisions about how to render all of the participants in the output frame (Figure 6). For example, if a participant speaks above a certain loudness or raises a hand, she is brought to the front layer and becomes opaque. If the same participant stops talking and remains still for a few moments, she relinquishes the front layer to the new center of attention and slowly fades into the back layers, becoming partly transparent. If more than one participant wants to be the center of attention, a “first-come-first-served” strategy is employed in which they all appear opaque and are composited in the front layers of the frame, but the first participant who seized the center of attention will retain the very front layer until relinquishing it as described before. In addition to these visual effects, the audio is spatialized to reflect the position of each participant’s image in the output frame, thereby enhancing a
listener’s ability to distinguish who is speaking.

A specially-colored object, usually a ball, provides a wireless and tangible means of interacting with the system in various ways (Figure 7). Clients watch for the presence of this object and transmit information about its position and proximity to the server.

The server provides several different “modes” of communication. The most common allows each participant to place images or documents in the background and navigate through them using the specially-colored objects. In this case, the positions of the objects act as vectors controlling the movement of the background. If several participants’ objects are visible simultaneously, the motion vectors obtained from each are added together to determine the actual motion of the background.

In another mode, a digitized video clip may be played with synchronized audio. The colored objects serve as a “shuttle” control. If no object is visible, the video plays out normally, but when one is brought into frame, the video pauses and the object may be moved left or right to rewind or fast-forward the video at different rates.

Another mode allows participants to collaborate on a painting that appears in the background. In this mode, the colored objects serve as paint brushes, and the proximity of the object controls the opacity of the paint. Different colors are selected by “dipping” the object into a palette displayed at the bottom of the screen. In a variation of this mode, participants may simply use their bodies to paint images into the background.

The performance of the system is highly dependent on the speed of the processing hardware and network involved and whether or not image compression is performed. Our prototype at the Media Laboratory (ATM network, 320
Figure 7: Participants in Reflection of Presence may use a specially colored object to interact with the space in various ways. In one mode, the object controls the movement of the documents and images displayed in the background (top images). In a different mode, the object serves as a paint brush, allowing participants to annotate documents or collaborate on a drawing (bottom images).

by 240 resolution, no data compression) runs reliably between 10 and 15 frames per second with 3 participants, with a latency that hovers at approximately 200 milliseconds.

We commonly demonstrate Reflection of Presence using full-motion video for all participants, but this is not a necessity. If, for instance, one client lacks video capability due to hardware or bandwidth limitations, a still image “avatar” of that participant could be shown to video-equipped participants. If the client also lacks audio capability, then text or synthesized speech could be rendered. In addition, in the current version, all participants view the same video stream—there is no personalization of the video output for each client. However, experiments are underway in which each viewer’s video and audio output are tailored to correspond to local processing and display capabilities and user preferences.

4 Isis Overview

All of the projects described above were built in Isis, a scripting environment, named after the ancient Egyptian goddess of fertility, specially tailored from its inception to address the multi-level nature of multimedia projects, both in terms of functionality and user-base. Its design draws from the basic realization that cutting-edge multimedia usually involves innovation at many levels, from low-level concerns of incorporating new sensing and display technologies to higher-level media structuring issues, and almost always involves a collaboration of creators of different levels of technological expertise.
(set factorial
  (proc (x)
    (if (= x 0)
        1
        (* x (factorial (- x 1))))))

Figure 8: For experienced computer programmers: This is a recursive factorial function written in Isis.

To date, we have ported Isis to a variety of UNIX systems, as well as to one non-standard platform (the Magic7 operating system on the Media Laboratory’s Cheops hybrid data-flow processor [7]). In this section, we outline the concerns taken into account in the design of Isis core scripting language and highlight some of the multi-level tools we have built inside of it. Later, we revisit the projects described above for details on how Isis was utilized in each case.

4.1 Scripting language

At the core of Isis is a scripting language that was specially designed to support the needs dynamic multimedia projects. It serves as a common foundation for tools of varying levels of abstraction that are built inside of it so that basic language operation is the same no matter what level of interface is used.

Isis is a interpreted functional language that on the surface resembles the Scheme programming language, but otherwise is completely dissimilar from it, especially in its internal operation. Isis is a much smaller language in comparison. With fewer and simpler concepts to learn and a more English-like syntax, it is hoped that Isis can be mastered by those not experienced with programming more readily than other more complex languages. At the same time, Isis must also support those who are seasoned hackers and wish to work at a lower-level or in concert with others at a higher level. Isis is a complete language for this reason, capable of all the things expected from a good programming language, such as nested environments and recursion. Additionally, although the language is inherently functional, object-oriented styles of scripting can be simulated by creating procedures with private local environments. Most of the higher-level tools described later are built and accessed in an object-oriented fashion.

The core language provides six basic types (integer, real number, boolean, character, procedure, and address). Higher-order lists of any kind of value may also be formed. All types are first-class, meaning they can be passed to, created within, and returned from procedures. A full complement of mathematical, list manipulation, file input and output, and debugging primitives is provided. Strings are modeled as lists of characters and are manipulated using the list primitives. Conditional evaluation is handled using if-then-else, cond, while, and switch constructs. Statically-scoped local environments are created using a local statement. Procedures are defined using a proc construct and may accept a variable number of arguments. When a procedure is defined, it includes a reference to the environment within which it was defined (a static store accessible only to that procedure), thereby providing support for object-oriented scripting.

The internal design of Isis is optimized in a number of ways to foster real-time performance critical to responsive media applications. Our design goals included making the core language small enough to support execution by a relatively small interpreter with no precompilation stage. An Isis core interpreter executable for a RISC processor is typically around 128 kilobytes in size. Lists and many other internal structures are based on arrays, not linked-lists, for fast retrieval of data. Isis employs its own memory management system, thereby avoiding inefficiencies in dynamic memory allocation that may exist on many platforms and in systems that use the standard C or C++ mechanisms. In addition, unlike Scheme, there are no “mutators” for standard data types in the syntax of the language, thereby simplifying the problem of internal garbage collection.

In the realm of multimedia in particular, a long list of system features is relatively meaningless without a way of extending and adapting the system for special situations, and so an integral part of Isis is its mechanism for interfacing with externally libraries. Functions written in C can be used from within Isis. From the interpreter’s point of view, C functions look like and behave in exactly the same manner as scripted procedures. In languages with this feature, efficiency can be a concern since internal representations of data types must be converted (possibly in very costly
(set framesize [320 240])

(dig-init-read-frame dig-rgb framesize)
(set rgb-buffer (new-image 3 framesize)
(set alpha-buffer (new-image 1 framesize))
(set port (tcp-open "gibsons.media.mit.edu" 1234))

(dig-read-frame rgb-buffer)
(image-chroma-key rgb-buffer alpha-buffer)
(write-image port rgb-buffer)
(write-image port alpha-buffer)

Figure 9: This short low-level script grabs a frame from a video digitizer, chromakeys it, and sends the result to another machine on a TCP link.

routines) to and from normal C types in order to link with external functions. In Isis, all internal script data are stored using standard C types, thereby minimizing this conversion phase.

4.2 Multi-level tools and libraries

Below we survey some of the more important or innovative libraries and objects we have written for Isis to support our projects at the Media Laboratory. In our work, we tend to add features as we need them and then include them into the database of tools we have at our disposal for the next project. We consider the "standard" Isis package of features to be those that can be considered necessities or that have evolved to a point of stability. However, every library is open to changes and improvement in our laboratory. If a needed piece of functionality is not present for a project, the usual response is to implement it, either by modifying or creating a new Isis script or C function, and to incorporate it, including documentation, into the system for future use.

4.2.1 Media manipulation and composition

The most important area of functionality in any multimedia tool is undoubtedly the suite of operations that can be applied to visual and aural media. At the lowest level, we have written a library of optimized image processing primitives, such as composition, scaling, gain/bias, arithmetic, statistics, and so on. There are also some more complex operations available such as chroma-key, noise removal, edge and feature detection, color balancing, and background segmentation. All of these are written in C and bound into Isis so that they may be accessed directly if desired. When a new piece of functionality is needed, often it can be tested and tweaked more easily using a combination of previously installed operations before it is written in C.

At the mid-level, we have developed a image buffer protocol so that all of the lower-level functionality described above may interface more cleanly and elegantly. This protocol, written completely in Isis, handles the allocation and deallocation of memory space for images and simplifies isolating specific parts or channels of images for an operation. Images are treated like objects instead of lists of memory addresses and offsets.

At the highest level of image manipulation is Amazon, a package of scripted objects for handling complex compositions of images and video. This system provides satisfactory support for experiments in the use of model-based representations as the basis for responsive media [8], which was one of our main reasons for developing Isis in the first place. In Amazon, the author may place any number of visual media "actors" in a three-dimensional space along with a "camera" that represents the viewer. The view attained by this virtual camera is rendered on an output device.

Seven kinds of Amazon objects can be instantiated and manipulated by the author: engines, windows, cameras, environments, actors, media sources, and effects. Each object contains parameters which control its state. Some of these parameters will contain references to other objects in the system, allowing a complex web to be constructed.
(set tl (new-timeline))
(tl 0 [50 50])
(tl 10 [150 -100] interp-linear)
(tl 20 [-150 400] interp-cubic)
(tl 30 [0 42] interp-linear)

-> (tl 5)
[ 100 -25 ]
-> (tl 7.4)
[ 124 -61 ]
-> (tl 15)
[ -3 163 ]
-> (tl 17.94837)
[ -110 342 ]
-> (tl 33)
[ 0 42 ]

Figure 10: The top part of this example shows a short Isis script in which a timeline object is created consisting of four key points. Each point is placed by specifying a time, a value at that time, and an optional interpolation type. In this case, the points are two-dimensional positions. Cubic interpolation is selected between the middle two points, and linear interpolation between the others. The points do not have to be specified in order, and other kinds of interpolation may be added if desired. Points may also be changed or removed at a later time. The bottom part of this example shows a transcript of an Isis session which retrieves the value of this timeline at various times. The arrow is Isis’ prompt at which the user types an expression to be evaluated, the result of which appears on the following line.

For example, an engine has a reference to a list of actors that will be rendered, and these actors, in turn, contain data about their position and orientation in the virtual space and references to media sources and effects that will be used to render them. All media and device interaction is controlled from within the objects—all the author has to do is create the web of connections between the appropriate objects and write the script such that their parameters are updated to appropriate settings depending on the values of timers, sensors, user identity, and other factors. The current configuration may be “realized” to the output devices at any point. Several webs, each containing its own engine, window, etc., may be created and used at the same time, and single objects may be referenced multiple times if desired. For example, a single media source may referenced by several different actors, or a single window may serve as the output destination for more than one engine.

A similar multi-level set of operations is provided for handling audio. Basic audio transformations are written directly in C. At a higher level are “audio player” objects which run as separate process threads and provide exact audio synchronization to other system events. At the highest level is a three-dimensional environment for the playback of sound, designed in a manner similar to Amazon, in which several “voices” and a “listener” may be placed anywhere in a virtual space which has certain acoustical characteristics. The composition of audio that the virtual listener hears is rendered on speakers surrounding the real listener.[9]

4.2.2 Data handling

The simplest way databases may be handled is by using Isis lists, which internally are based on arrays rather than linked-lists for speed in access. We have also implemented a number of higher-level storage constructs that use lists as their foundation. For example, the structure construct may be used for arbitrary key-value pairs.

We have designed a timeline data storage construct which might be considered as an array with real-number indices. Values of any type are placed at real-numbered points, or “times,” along a one-dimensional “line.” An interpolation method to apply between each successive pair of points is also specified. Therefore, a timeline is evalutatable at
(load "amazon.isis")

(set eng (new-engine))
(set win (new-window))
(set act (new-actor))
(set squirrel (new-source-url "file:squirrel.jpg"))

(act [ac-source squirrel]
    [ac-position [100 100]]
    [ac-scale [2.0 1.0]])

(win [win-title "A very fat squirrel"]
    [win-size [200 200]])

(eng [eng-actors act]
    [eng-window win]
    [eng-render])    ### renders and displays

Figure 11: In this high-level script, the Amazon library is used to display a single scaled image at a particular location in a window. If the media source was an image sequence, a frame number could be selected. 3D positioning and various built-in or user-defined special effects could have also been specified.

arbitrary real-numbered points. Linear and cubic interpolators are built in, but other user-defined routines may be used as well. Timelines are handy for multimedia purposes since they are helpful for coordinating time-dependent activities and expressing multi-dimensional spatial variations (or variations in any other analogous domain). In many of our scripts, timelines control the parameters of Amazon objects. Dynamic behaviors are made possible by the fact that points on a timeline may be added, changed, or removed at will.

4.2.3 Networks and distributed processing

Some of the most ambitious and breakthrough responsive multimedia projects require the power of several processes running on several machines simultaneously, all synchronized with each other and communicating in the most seamless manner possible. Isis has TCP/IP networking capabilities to support distributed processing. To ensure a high degree of interoperability, the same file input and output primitives may be used to read and write coded Isis values or raw data to disk files, TCP connections, serial ports, etc. Most importantly, timers in separate Isis processes or on multiple machines may be synchronized.

The low level I/O primitives allow Isis scripts to be written that communicate with foreign network protocols. High-level procedures that send email and retrieve documents and images directly from the world wide web are among the most commonly used in our projects. Amazon “media sources” allow the specification of URLs and use these procedures to access their documents. External processes may be created and communicated with in a similar manner. For example, we launch a public-domain postscript interpreter in a separate process to render postscript text and documents from within Isis.

4.2.4 Sensing mechanisms and output devices

In our laboratory, we use Isis mostly on Unix-based platforms with X window displays, and we have therefore developed a simplified environment for accessing the X window system in Isis that we use for many standard window-based applications. Images may be displayed to windows and simple events (such as mouse motion and key clicks) may be received. Low-level details, like color palette and visual selection, are handled automatically. The goal was to provide a smaller and more streamlined interface to a windowing system. As Isis is ported to other platforms, support for other windowing systems will have to be added. Many objects have been written on top of this interface,
such as an image file viewer/selector, an image "clipboard," and a more sophisticated "smart-window," an object that remembers what areas of a display buffer have been updated so that only modified areas need to be refreshed to a window.

Hardware is attached to many of the workstations in our laboratory that supports asynchronous digitization and display of video and audio. We have written drivers for this hardware in C that are bound into Isis. One may use the low-level access functions or higher level routines that return data in the image buffer protocol format described above. "Windows" in Amazon may render their material to this hardware. This hardware supports compression and decompression of video and audio as well, which is used by mid-level movie objects and movie "media sources" in Amazon. Moving to different digitizing hardware would mean adding new low-level drivers, but the high-level objects could remain the same.

We have also developed objects for controlling some unusual input devices, including several that were developed at our laboratory which include sonar, radar, and electric field sensors [10]. Midi translators and other devices using serial port communication may be accessed using the built-in file I/O primitives.

5 The projects revisited

In both The Museum and Wallpaper, high-level Amazon scripts were used to establish the framework of visual and auditory media actors of which these productions are composed. Timeline interpolation enabled the director of The Museum to describe the variations in the camera and scene elements smoothly over a range of window sizes rather than in discrete steps. The same could have conceivably been done for a range of aspect ratios.

In Wallpaper, each of the high-level presentation variables is represented in the script as a real number within a certain range. These variables index into nested timelines to determine the composition of each frame over time, different story perspectives, and different dexterity settings. Timeline interpolation is applied to generate blends of the two extreme story perspectives while playout is in progress. Based on the current camera pose, the correct actor views are selected by changing the Amazon media sources associated with each of the actors.

Since Wallpaper was a collaboration among several researchers, the multi-level nature of Isis was put to the test. Some worked on low-level functionality such as the spatialized audio and Cheops image processing libraries, and others who had less expertise in programming were focused on high-level Amazon structure and interaction. Isis provided a common design environment for everyone involved and allowed the latest research from several groups to be harnessed for a single piece.

Seshay was also a collaboration among a small but diverse group of people. Baird developed the design, content and physical structure of the piece while directing two assistants who implemented the software for the system based on her storyboards. One worked mainly on an Isis script to recognize gestures using low-level sensor interface routines; and another utilized Amazon elements to describe the video presentation. Both Baird and her assistants explored methods of constructing complex animation templates and databases of annotated audio and video clips in Isis. In this case, Isis was an instrumental factor in fostering the collaboration. The system was powerful enough to support a completely scripted gesture recognizer utilizing in-house sensing technology, yet it also allowed Baird successfully to remain in a higher-level directorial role for the duration of the project, experimenting with the characteristics of the animations without undue concern about the underlying way in which the features would be implemented.

In Reflection of Presence, the server script utilizes Amazon to structure the composition of the video output, while the client script harnesses lower-level image manipulation procedures to capture the segmented image of each participant. Audio is recorded and processed in separate scripts from the video. All of these scripts run as separate processes on several different machines, interconnected via TCP. Profiling indicates that, in this particular case, the majority of time is spent in image processing operations whose efficiency undoubtedly could be improved.

Reflection of Presence is the most complex application built in Isis to date, and the several people involved on the project all agree that using Isis has been advantageous, from providing a common environment for the individual pieces of the puzzle to come together, to creating a complex and dynamic network of cooperating processes, to allowing changes and additions to be made and tried out with fairly little effort.
6 Conclusion

As a model for scripting environments, what makes Isis interesting is the fact that the entire system has been specially designed from the ground up to foster the multi-level character of responsive multimedia projects. It enables applications to mix high-level constructs with low-level in-house specializations within the same language foundation, and it enables collaborators of various levels of expertise to find a common base of operations. As a continuation of our bottom-up approach, we are currently developing visual interfaces in Isis that augment several of its features, including tools to aid in defining timeline databases and creating webs of Amazon objects.

Complete documentation on every aspect of Isis is currently available on the World Wide Web at [11].

7 Acknowledgments

The authors wish to thank the many individuals who have contributed to this project, including Alex Westner, John Watlington, and in particular Glorianna Davenport. This research has been supported by the Digital Life Consortium at the MIT Media Laboratory.

8 Biographies

Stefan Agamanolis is a doctoral candidate at the MIT Media Laboratory where he is creating new tools and languages for utilizing computational media for collaboration and artistic expression. His research led him to design the Isis media scripting environment. He received a Masters degree from the Media Lab in 1996, and before that he studied computer science, philosophy, and film at Oberlin College.

V. Michael Bove, Jr. holds an S.B.E.E., an S.M. in Visual Studies, and a Ph.D. in Media Technology, all from the Massachusetts Institute of Technology, where he is currently Principal Research Scientist at the Media Laboratory. He is the author or co-author of over 40 journal or conference papers on digital television systems, video processing hardware/software design, multimedia, scene modeling, and optics. He holds patents on inventions relating to video recording and hardcopy, and has been a member of several professional and government committees. In December 1995, Boston Magazine named him one of the “People Shaping Boston’s High-Tech Future.” He is on the Board of Editors of the Journal of the Society of Motion Picture and Television Engineers, and served as general chair of the 1996 ACM multimedia conference.
References


