"iTheater" Interface Design:
Integrating an educational user interface with a non-linear story engine

by

Alon Mozes

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 5, 2000

Author [illegible]
Department of Electrical Engineering and Computer Science
May 5, 2000

Certified by [illegible]
Glorianna Davenport
Thesis Supervisor

Accepted by
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

"iTheater" Interface Design:
Integrating an educational user interface with a non-linear story engine

by

Alon Mozes

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 5, 2000

ABSTRACT

User interface design is an important part of any software application, especially educational programs. iTheater is a powerful application development tool created by Echo Bridge Productions. It is used to implement non-linear interactive stories. This thesis involves enhancing iTheater to allow for video components for the stories, which should greatly improve the user interface for the resulting projects. Taking advantage of the new features, iTheater is then used to create an educational program for health care professionals. The result is effective, although the program does not take full advantage of iTheater's non-linearity.

Thesis Supervisor: Glorianna Davenport
Title: Principal Research Associate, MIT Media Laboratory

1. INTRODUCTION

From the early days of interactive video, an effective interface for such videos has been critically important.. Recent developments in compression algorithms for sound and video have broadened the range of applications and audiences for interactive video and allowed for more immersive interfaces. Clinical education, in particular, will be increasingly facilitated by powerful multimedia processors with broadband networks that can deliver applications to the home or office. However, many of these applications have not been fully explored, and the state of computer-based professional education is one of unrealized potential (Henderson 1997). New educational applications that seek to take advantage of this technology should follow some primary development principles, and pay attention to user interface lessons learned in the past.

User interfaces can enable users to quickly grasp the purpose and power of various applications, or, if poorly designed, can prohibit the use of even the simplest programs. VisiCalc, a spreadsheet program developed in 1978, is one of the first applications with an interface design transparent enough for an ordinary user to easily understand its functionality. Despite the complexity inherent in spreadsheet programs, VisiCalc provides an intuitive interface for users to perform calculations that had never been previously computerized.

Since this milestone in interface design, other developments have utilized advances in processing power. Apple developed its QuickTime movie player, which displays digital movies with a controller. This controller provides an intuitive interface for playing the movie. It takes advantage of elements from older interfaces, like the symbols on a VCR, while applying it to an entirely new domain. These precedents will likely affect the future of interface design, and have played an important role in this thesis.

2. DESIGN PRINCIPLES

This thesis involved the design and development of enhancements to iTheater, which is a non-linear software development tool, and the creation of a clinical training application, "Understanding Partner Abuse", using iTheater with its new enhancements. User interface design was a critical part of developing the enhancements and application. Principles extracted from concepts in scientific visualization, user-centered design, and graphic design allow one to create and evaluate user interfaces. Some basic principles include:

- Develop a metaphor
- Keep it simple
- Make changes clear and consistent
- Design within constraints
- Design an interface that teaches

2.1 Develop a Metaphor

An excellent way to begin to design a user interface is to identify a metaphor, something in the real world that the application resembles. A user interface that acts as a metaphor can provide instant information about the functionality of the back end software. A successful metaphor requires two elements: the metaphor must be identifiable and well understood by the user; the metaphor must be appropriate for the software it represents. The first is achieved by selecting a metaphor that is prevalent in everyday life and that one could reasonably expect a user to understand. The second principle is more difficult to realize. For example, TV broadcasting has been used as a metaphor for a newspaper publishing business. While there may be some similarities, the user could be "led astray" by the implication that newspaper businesses involve instantaneous transmission of data like television. In reality, newspapers have delivery delays (Erickson 1990).

iTheater uses a metaphor of a stage play to enable the construction of non-linear stories. Characters are placed on a stage and perform scenes in which they may say a line or take action. They are puppets that can be controlled by the computer or the user. Each story is represented as a separate theatrical production played out in the theater that iTheater creates.

2.2 Simplicity

One of the most basic principles is to "keep it simple" (Kobb 1994). As software becomes increasingly complex, user interfaces also tend to be more complex in order to give users access to the full functionality of the software. Unfortunately, this can easily result in a totally incomprehensible interface. Achieving an effective, simple interface means that a designer must strip away all superfluous features and graphics, leaving only the most basic functions displayed in a clear way.

Interactive educational applications are especially tricky because users may not be technically savvy. There may be many options and settings that are non-intuitive. Tutorials, help files, and instruction pages can all aid in explaining how an application works, but in and of themselves these methods are employed to clarify an obscure user interface design. A good user interface, in contrast, allows users to dive into the application and learn *from* the software, rather than having to take the time to learn *about* the software.

The user interface for an iTheater application is simple so unnecessary features do not distract the user. The user answers a few simple questions to set up the game, like choosing a character, and then begins the game. This allows the user to be immersed in the story rather than worry about setting software parameters.

2.3 Clear and Consistent Changes

One of the essential characteristics of a good user interface is consistency. This principle asserts that "mechanisms should be used in the same way wherever and whenever they occur" (Tognazzini 1990). Inconsistency can be detrimental to a user's understanding of an application. For example, an infamous inconsistency exists in the Macintosh interface. A user deletes files by dragging them to the trash can, an understandable metaphor. Ejecting a disk from the floppy drive, a very different operation, requires a similar method of dragging an icon into the trash can. This inconsistency can be confusing to a novice user who may worry that they are erasing files on a CD-ROM when dragging it to the trash.

In new interactive educational programs, users are often unfamiliar with the mechanics of the application. At the onset, many applications require the user to make a decision, like choosing a direction to navigate or choosing an object to pick up, and then communicate that decision to the machine. The method for decision making should be obvious and consistent throughout the experience. Changes following each decision should be very clear. After a navigational decision, for example, there should be a change of view on the screen. The latency of the change should be kept to a minimum, so the user can make a clear connection between the action and the consequence. The user should be able to understand immediately what effect the decision has had and understand how a similar decision will affect the program in the future.

2.4 Design Within Constraints

Paying attention to software constraints can simplify the developer's task of taking an idea from theory to practice. Common constraints in a distributed application are found in the operating systems of the client and host computers and the brand of browser used to display the application. Different operating systems and browsers often have different performance results as well as graphical displays. Some applications are better suited for particular platforms.

For example, a QuickTime application could be significantly faster on a Macintosh machine since that is its native platform. Also, a Java panel in an applet has inconsistent results when displayed in Microsoft Explorer and Netscape Communicator.

These differences can be discovered during testing, but it may be too late at that point to debug unwanted side effects. Testing can also present a difficult design choice. Ideally, application developers conduct user testing by getting people to use the final product before it is released. However, this sort of testing can take an extensive amount of time as revisions and retesting is conducted. Therefore, deadlines should also be taken into account when designing an application so that enough time remains to fully test the program.

Distributed programs may face problems with constraints of accessibility. The ideal situation would be to allow users to run an application from any computer. Two problems interfere with achieving this. First, not all computers have Internet access, which is necessary to access remotely located files. Second, even those computers that do have Internet access do not necessarily have a connection that matches the performance required by the application. A CD-ROM application could avoid these problems, but using a CD-ROM involves distributing physical hardware rather than just sending bits over a network. Such tradeoffs are best determined in advance (Mountford 1990). Designing with these choices in mind can prevent having to deal with surprises caused by different operating systems, deadlines, and accessibility issues later in the development process.

2.5 Designing an Interface for Learning

The user interface can be the key to clarifying the functionality of the program. Menus, for example, can quickly convey many of the features of an application without overloading the interface. An interface can also act as a coach or guide for a user. Prompting the user with questions and providing clear error messages are important parts of using the interface as a coach. Instead of inundating the user with a screen of various parameters, the application can ask the user a series of questions, one by one, to make sure that the user is aware of all the options. Also, error messages that explain the error and how to avoid it may teach the user about the program. The http protocol demonstrates a poor message with its response for a failed page-load, "404 ERROR: FILE NOT FOUND", which could imply a missing file, a typo in the URL, or a downed server. More detail in the message could help the user discover the source of the problem. An example of a coach is the popular paper clip guide in Microsoft applications. It is always available to teach the user about various features and also prompts the user with questions in order to verify the user's understanding of their actions.

3. BACKGROUND

Although many applications have taken the leap from text to video, it is useful to analyze this new interface for iTheater applications within the context of clinical training. Clinical training videos provide a forum for teaching healthcare professionals, thus augmenting the clinical one-on-one interactions with patients. . Videos can offer realistic scenarios and can be produced to simulate one-on-one interaction or third person viewing. Learners can use the video repeatedly to practice interviewing techniques and diagnostic skills and to ensure knowledge retention. Various designs for these training videos have been used, ranging from a simplistic multiple-choice video game to a more complex simulation of patient interactions.

3.1 "Condom Conundrum": An Interactive Sexual Education Program

In the spring of 1999, I worked with Gillian Steel, a student at the Harvard School of Public Health, to develop an interactive game to teach teenagers about public health issues, focusing on the realm of sexual education. Gillian developed a script with a Mission Impossible plot and with the "in-your-face" style of You Don't Know Jack by Bezerk Software. We, along with the help of the Interactive Cinema Group of the MIT Media Lab, filmed the live action scenes and combined them with graphics and animations in Macromedia's Director 7.0. The result was an entertaining, educational game on CD-ROM.

There were a few limitations to "Condom Conundrum". The story was totally linear so that the user only learned about how well they performed by reading a score, rather than experiencing varying outcomes. This simplicity was designed to appeal to teens of all educational levels. A non-linear story, while more complex, would seem to be a more immersive approach, especially with the inclusion of some simulated interactions with STD patients.

Also, the application was developed on CD-ROM, which meant that any user must have a copy of the disc and install it onto their computer. An Internet application that is freely available on a browser would be an easier form of distribution.

3.2 "Virtual HIV Clinic": An Interactive Health Training Video

"Understanding Partner Abuse", the application developed with iTheater, is an interactive application that will be used to teach professional health care workers about domestic abuse. Joseph Henderson, a doctor at the Dartmouth Medical School Interactive Media Lab, has conducted a significant amount of research at the lab on the subject of using technology to train health care professionals. One of the major projects, the "Virtual HIV Clinic", explores the training of health care professionals to interact with HIV patients in various stages of the disease.

The application's layout is broken down into several rooms in a clinic, each containing the patient at different stages of the disease and one room for learning resources. Navigation is achieved by either clicking on the master navigational map (Figure 3.1) or by travelling "inside" the clinic (Figure 3.2).

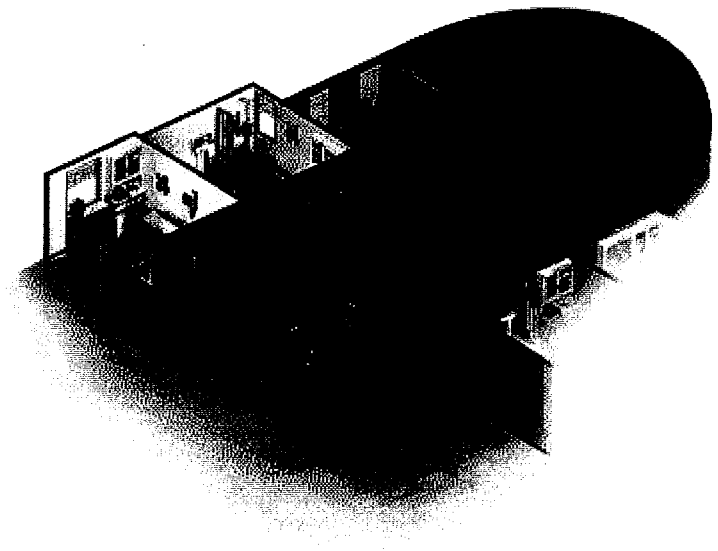


Figure 3.1: Overview of the clinic

Figure 3.2: User's perspective for navigating the clinic

Once inside one of the rooms, the application uses a range of interactive methods to train the user. One method includes displaying recorded video for "mini-lectures" and case presentations. Some of the more interactive "computer-based" methods include unscrambling jumbled lists, labeling diagrams, categorizing challenging statements, and other similar puzzles. Through these modes of interactivity, the application simulates "learning through experience" (Henderson 1997).

4. iTHEATER

4.1 iTheater Introduction

iTheater is a software tool developed by Echo Bridge Productions, which can be used to easily develop non-linear interactive stories and games. The metaphor of a theater allows developers to quickly understand and use it. A development team (including an author, producer, programmer, film crew, and any others involved in design and development) design the story by creating characters and scenes. A character in iTheater is treated just like a character in a theatrical play, and has a name, description, and optional "states of being". For example, the developer can use these states of being to attribute characteristics such as skill or trust to a particular character. A scene contains a character performing an action, saying a line, or entering a room. The interface for entering the data into iTheater is a text file, which contains all the characters (see Figure 4.1) and scenes (see Figure 4.2).

iTheater is designed to allow developers to create scenarios which play out differently depending on specific actions of an audience participant. iTheater enables the developer to provide criteria that determine which scene should become active. The story engine uses these criteria, often based on character states or scene order, to determine which scenes should be selected next in the story. For example, a level of trust, which is a possible character state, can be a criterion for determining whether a secret should be revealed. If a computer-controlled Actor A does not trust a human-controlled Actor B, then Actor A will never reveal a secret to Actor B. These criteria are input by the developer when creating the text file.

```
<character>
  ID = 6
  stage = 1
  title = Dr.
  firstname = Reginald
  lastname = Doctor
  referent = the doctor
  description = the doctor in charge of this case
```

Figure 4.1: Text to create a character in iTheater

```
<scene>
  ID = 14
  stage = 2
  name = doctor chooses
  character = 6
  lastScene = 13
  criteria = *
  factor = 4,trust,>=,B
  movieline = 4, http://localhost/upa/sport.mov , 0 , 6
  movieline = 6, http://localhost/upa/crossfad.gif , 0 , 0
  description = You say, "Tell me what happened"
  nextCharacter = 3
```

Figure 4.2: Text to create a scene in iTheater

iTheater also allows for user interaction in the story. The user can choose to play any of the characters and alter the story by making decisions. For example, by controlling Actor A, the user may be able to persuade Actor B to trust him and reveal the secret.

4.2 iTheater Advantages

Other software tools exist that allow developers to easily script interactive stories. Many of these products have inherent disadvantages that iTheater improves upon. Macromedia Director is a popular example of such an application. Director 7.0 requires a movie score (though the latest versions just released may not), which adds a built-in timeline to every project. Lingo is Director's scripting language and can be used to control elements within the score as well as the flow of the entire application. Using Lingo, developers can use certain keywords to integrate loops or sidestep the timeline, but such non-linearity can quickly get convoluted and difficult to program. Doepel, principal investigator at Echo Bridge Productions for "Understanding Partner Abuse", explains the disadvantage inherent in Director's linearity:

> "[Director] requires programmers to create all possible decision branches in the 'score' (i.e. timeline) of the program and to use 'go to' commands to direct the program to the appropriate next step. Any additions, deletions, or changes made to the logic tree would then have to be tracked and re-programmed individually throughout the entire program resulting in a time delay and potential anomalies in the program (requiring additional debugging time). (Doepel 1997)"

iTheater has no requirements about where a character or scene can be entered. This development preference allows for a nearly unlimited cast of characters and events that can be entered into the text file in an arbitrary order. This lack of structure enables the creation of the most random stories along with the creation of strictly linear stories.

iTheater's versatility has a number of benefits. Janet Murray, former Director of the Program in Advanced Interactive Narrative Technology (PAINT) at MIT, describes the "multiform story" and provides some insight into the possibilities of nonlinear stories. These stories include a "single plotline in multiple versions". As implemented by iTheater, users can investigate these "alternate realities" through interactive decision making. iTheater can trigger different scenes depending on what decisions the user makes, bringing the user through a different path to different conclusions each time a story unfolds (Murray 1997).

The user also has the choice of which character to play. In a game like Zork, developed at MIT's Lab for Computer Science, a user generally plays the role of a single main character and goes through the interactive adventure, with one of the ultimate goals to slay a dragon. Using iTheater, it is easier to build a story in which a user can choose to play one of several characters; the goals of the game change according to the character's perspective. This would be the equivalent in Zork of playing the dragon and assuming the new goal to avoid being slain by the main character.

Applications are also easily extensible with a tool like iTheater, unlike with some Director productions. If a developer wishes to add characters or scenes, these new elements are simply appended to the end of the text file. In Director, by contrast, a developer must keep track of where other elements are in the score in order to know where to place new pieces. This is a direct result of the built-in linearity of the score metaphor, an unwanted feature that iTheater avoids.

4.3 User Interface for iTheater Applications

iTheater applications begin with users choosing the character they wish to control from a list of controllable characters in the story (see Figure 4.3). Users can choose to control any of the main characters. While in the current version, this choice is made by selecting from a list-box in which the user sees all possible characters, one could, if appropriate, use icons or other selection methods.

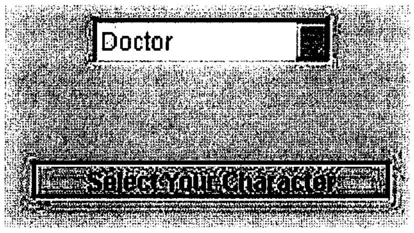


Figure 4.3: iTheater applications allow users to select their characters

For a health training application, selecting a point-of-view is an especially useful feature. One can choose to be the doctor and try to earn the patient's trust, or one could choose to be the patient and reenact a scenario they may have experienced in real life. A user can also choose to play an "outsider" and just watch the story unfold. This feature allows the user to set the appropriate skill level, in the form of a standard letter grade (see Figure 4.4), of other characters in the story so that one can learn from mistakes in addition to learning from a well-handled situation.

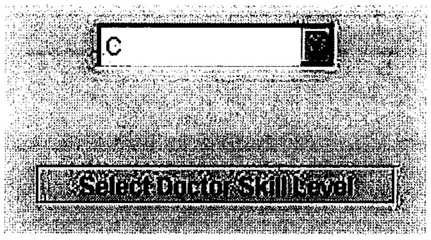
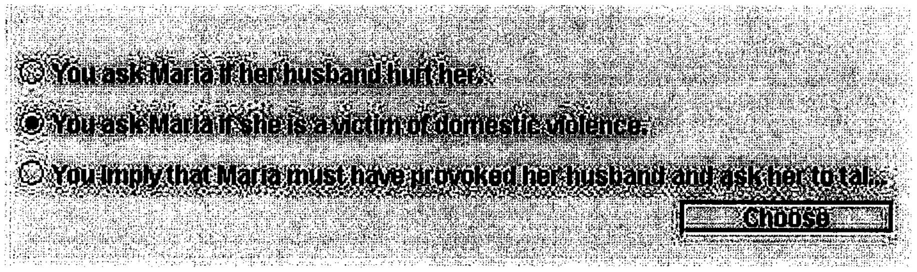


Figure 4.4: Users can select the skill level of computer controlled characters

Once characters are chosen and skill levels are set, the story begins. The story continues until it is time for the user to make a decision. A user makes decisions in iTheater applications through an interactive multiple choice panel (see Figure 4.5). The choices are displayed when the presentation reaches a point at which multiple scenes can be triggered. For example, if the user is controlling the character of a doctor, and it is the doctor's turn to say something to the patient, there may be a few different lines from which the doctor can choose. If the computer were controlling the character of the doctor, then it would select a line automatically, based on various criteria, such as a character's skill level.



- You ask Maria if her husband hurt her.
- [✓ You ask Maria if she is a victim of domestic violence.]
- You imply that Maria must have provoked her husband and ask her to tal...

Figure 4.5: Users make decisions through a multiple choice panel

These decisions are limited to the few selections that the story engine presents to the user. The application is not open-ended, but the application frames the lesson with limited responses in a productive way to teach the user the lesson. The developer essentially preprograms these options when the text file is initially created. This format limits both the number of branches in the story and the amount of input that the user has. The advantages of such a format are that the designers can control what lessons are presented to anyone who uses the application.

iTheater is intended to create interactive stories for an educational goal. The pedagogy of current applications in iTheater provides the user with surrogate experience. More open-ended interactivity has been used with some success in earlier developments like Eliza (Murray 1997). These applications use rule-generated behavior in order to respond to users' open-ended textual responses. As an entertainment tool, this open-ended decision format may have been a better design choice. However, as an educational medium, a more limited format seems to be more effective. Doepel describes the advantages of limited branching over open-ended role-playing:

> "Programmed branching in the 'virtual visit' scenario anticipates potential correct and incorrect choices the learner can make and provides the opportunity to explore these pathways with appropriate 'teaching' feedback along the way. Furthermore, using interactive educational media, educators can avoid the limitations of traditional role-play teaching. Research has shown that role plays are typically not standardized, making the learning experience of different learners inconsistent. Verbalizations can change in nuance and content, thereby changing the potential outcome of the interaction. And often role plays focus on generic skills rather than on specific knowledge or skill (Neimayer and Pfeiffer, 1994)."

5. ENHANCING THE UI FOR iTHEATER AND iTHEATER APPLICATIONS

The crux of this thesis involved improving iTheater and using the improvements to iTheater to develop an interactive healthcare training video. The improvements to iTheater included the addition of a movie component to the user interface, whereas the only previous output format was text-based. The healthcare training video was a group effort as medical professionals, writers, editors, and programmers worked together to create the project. The script and all film work were completed by Echo Bridge Productions; my role was to use iTheater to construct the final product from those pieces.

5.1 iMovieLine and MoviePanel: the interface in iTheater for displaying movies

iMovieLine is a Java class that adds the capability to store information that can later be used for display in MoviePanel, a class that allows for the display of QuickTime movies. On the first design pass, this information took the form of a local file. The application would reference the file and create an instance of a Movie (found in QuickTime for Java). The problem with this design was that an applet would generate a security exception if it tried to reference a local file on a client machine. Applets are inherently untrusted by client machines and therefore any attempt to reference a local file will cause a show-stopping error. One way to avoid the problem was to design the program as a local application, which could be run from a CD-ROM, rather than an applet. However, in order to be displayed from any browser, the application needed to take the form of an applet.

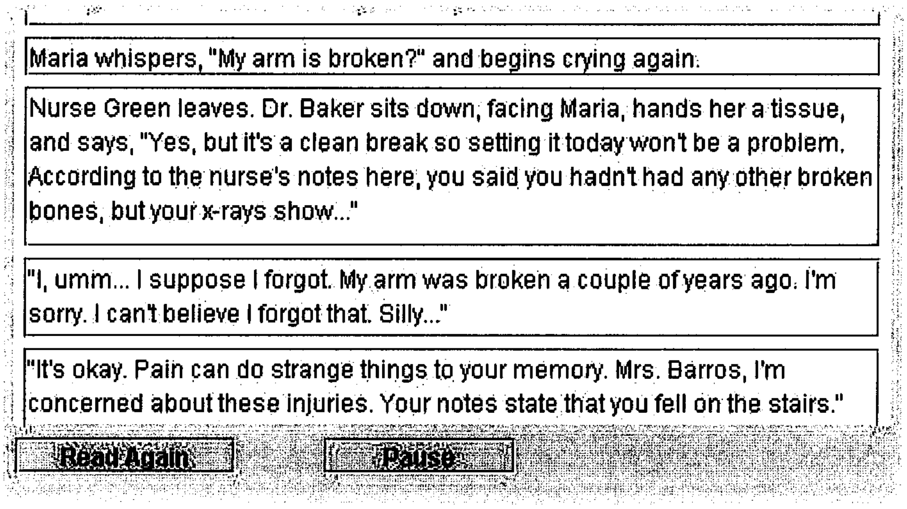
This led to the design of an applet that uses streaming movies, or movies that are located on remote hosts and are "streamed" into the display environment. iMovieLine was revised to accommodate the use of URL's instead of local file references. The MoviePanel class would then create a DataRef object (see Figure 5.1), so that the URL will ultimately be streamed into a displayable Canvas object (Maremaa 1999).

```
//The following code creates a data reference object from a text
//object that represents a URL. A movie object is then created
//from the data reference object, and then a movie player is
//created for the movie. The movie player is then added to the
//canvas
DataRef dr = new DataRef(firstSegment.getMovieURL());
currentSegment = currentSegment.fromDataRef(dr,1);
movPlayer = new MoviePlayer(currentSegment);
myQTCanvas.setClient(movPlayer,true);
```

Figure 5.1: Code to display movie streamed from a URL

Using streaming movies avoids the security problem of the applet and adds the ability to store the QuickTime movies on any host reachable by the Internet. This implementation adds a delay in the playing of the movie. The delay is caused by the attempt to stream the movie, which potentially references another machine to retrieve the movie. This method is generally slower than referencing a local file. The delay depends on the size of the movie being streamed, but it averages a few seconds. While the user waits the few seconds for the movie to load, the last frame of the previous movie is displayed in the viewing area, which is preferable to a blank screen (This is primarily a function of "QuickTime for Java" and not a design choice). Following the delay, it successfully displays the movie in the movie panel.

Once the basic functionality for displaying a movie was created, the next step was enabling the user to choose from multiple points-of-view. This is a simple task if the display is entirely text-based. As far as the interface is involved, a user that plays the role of the doctor has the same perspective as any other character (see Figure 5.2).



Maria whispers, "My arm is broken?" and begins crying again.

Nurse Green leaves. Dr. Baker sits down, facing Maria, hands her a tissue, and says, "Yes, but it's a clean break so setting it today won't be a problem. According to the nurse's notes here, you said you hadn't had any other broken bones, but your x-rays show..."

"I, umm... I suppose I forgot. My arm was broken a couple of years ago. I'm sorry. I can't believe I forgot that. Silly..."

"It's okay. Pain can do strange things to your memory. Mrs. Barros, I'm concerned about these injuries. Your notes state that you fell on the stairs."

Figure 5.2:Text panels can only offer a third person point-of-view

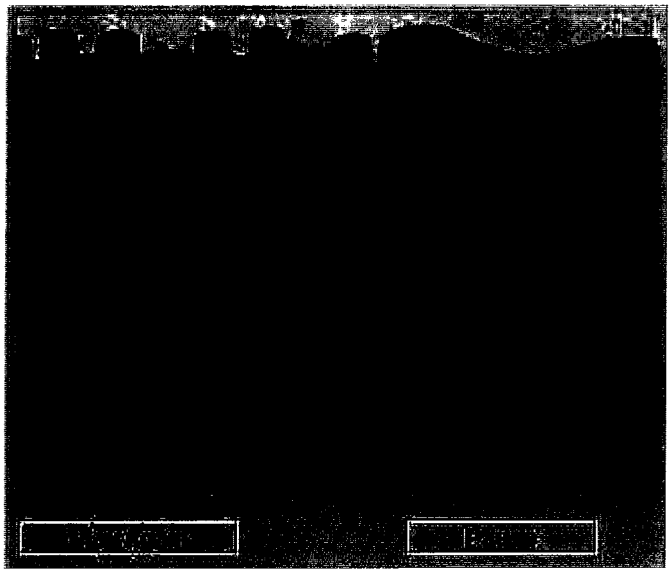


Figure 5.3: The patient explains her injury to the user, who is acting as the doctor

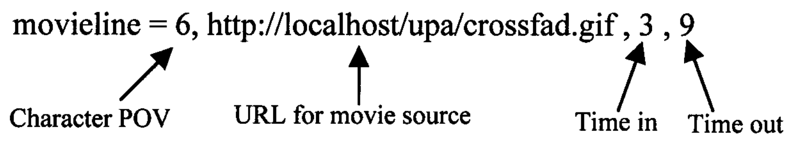
However, the introduction of movies to the display adds a new dimension to the interface. Ideally, if a user plays the role of the doctor, the movie should portray all the scenes from that viewpoint, meaning that the doctor should never be visible in the movies (see Figure 5.3). This implies that separate movies must be created for each point-of-view and that the software should be equipped to choose from multiple movies for the same scene. This code (see Figure 5.4) is added to the applet rather than the MoviePanel in case multiple users open the application.

```
//checks if the character controlled by the user is the same as the
//POV of the character in the current movie about to be displayed
if (this.humanControlled == movieline.getPOV()) {
        //display the movie since it is from the user's POV
        .
        .
        .
}
```

Figure 5.4: Code to select movies with the appropriate POV

Users will have their own applet open, which allows control over any character they choose. Each applet will determine which movie has the appropriate point-of-view for that particular user.

The next development regarding the addition of movies to the interface benefits the developer. Since so many different QuickTime movies would be necessary to achieve any point-of-view for any version of the story, a simpler design would allow a developer to create one master QuickTime movie and input start and stop times to indicate which segment is appropriate for the scene. These times can be added along with the URL reference in the text file (see Figure 5.5).



```
movieline = 6, http://localhost/upa/crossfad.gif , 3 , 9
```

Figure 5.5:Adding a movieline to a scene

The "time in" and "time out" indicate the start and stop times for the movie referenced. In figure 5.5, for example, the movie will play from the third second in the clip until the ninth second. The only reference that would be necessary would be one master movie that includes a compilation of all the scenes. This makes film editing much easier since the programmer can essentially do much of the film editing in the text file. The downside is that QuickTime for Java is slow to pre-roll movies to a particular point. This significant trade-off implies that the clips should be separate movies for the sake of the user's experience.

5.2 Applying iTheater: "Understanding Partner Abuse"

iTheater was used to implement an interactive healthcare simulation entitled "Understanding Partner Abuse" (UPA). This simulation is designed to teach emergency room professionals to recognize signs of domestic violence. The story was developed by Echo Bridge Productions and involves a doctor, a nurse, a female patient, and her husband. The user can control the doctor or patient, or the user can take an outsider's perspective and not play any of the characters.

The plot in this simulation concerns the patient, Maria, who comes to the hospital to have the doctor look at her arm. As the doctor, the user can influence Maria and how much information she divulges. If played well, the user learns of past fractures that Maria has suffered, gets Maria to admit her husband abuses her, and convinces Maria to consider using a hotline or marriage counseling. If played poorly, the user simply makes Maria eager to have the examination completed so she may leave the hospital.

The user also has the opportunity to play the part of Maria. Maria has no choices to make in the story, as there is only one response for any action that the doctor takes. The importance of choosing Maria is that the user has Maria's "on-stage" point-of-view in the story (see Figure 5.6), meaning that the doctor appears to be speaking directly to the user. This interpersonal point-of-view is important for the inclusion of the user in the role-playing format.

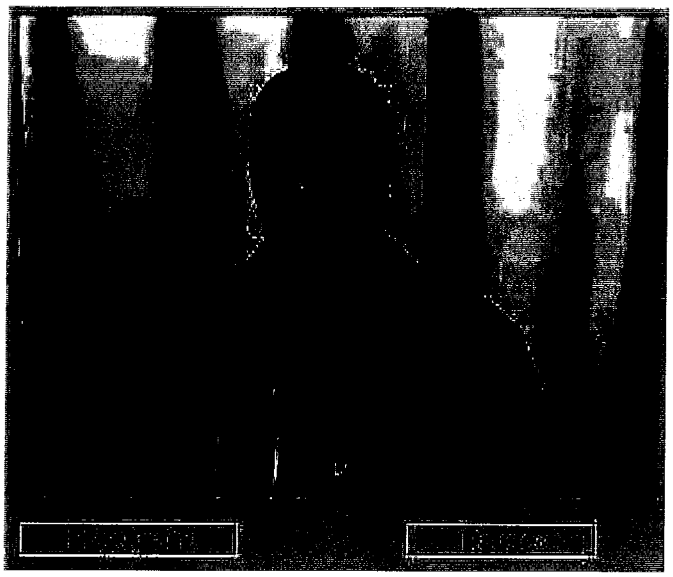


Figure 5.6: The patient's POV in "Understanding Partner Abuse"

The third option is to play the role of the outsider. Again, the outsider has no decisions to make so the entire story unfolds. The difference from this point-of-view and Maria's point-of-view is that in this case the user sees the master scene, in which the user views all characters involved. The user, as a passive observant, can watch the doctor speak to Maria.

If the user chooses either the patient or the outsider, then the user is prompted with a choice of skill level for the doctor. This determines how "well" the computer-controlled doctor interacts with Maria. This is an effective option since the user can choose to watch the doctor handle the situation well and convince Maria to get counseling or choose to watch the doctor demonstrate what not to do.

Even though "Understanding Partner Abuse" allows for some user interaction, it is not as powerful as it could be. It does not fully take advantage of iTheater's potential to create non-linear stories. It contains few factors upon which to base computer generated decisions. A greater variety of variables in the story could make for a more interesting application when played over and over. The story is fairly linear and branching is minimal (see Figure 5.7).

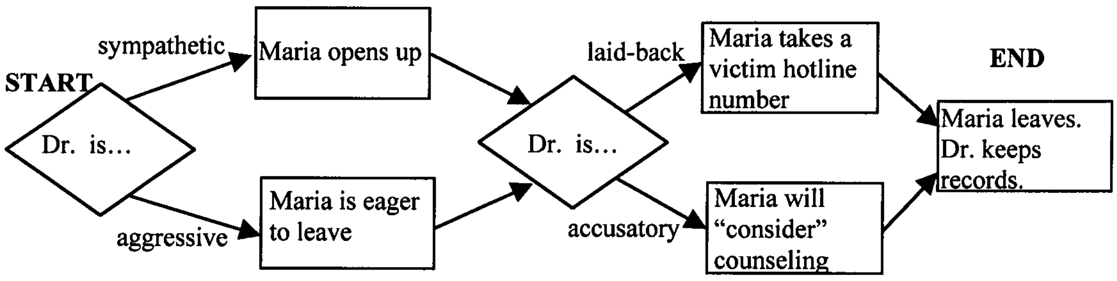


Figure 5.7: A simplified flow chart of UPA. The same end point is reached through minimal branching, though different paths imply different results.

However, "Understanding Partner Abuse" does successfully demonstrate iTheater's capability to easily construct a non-linear story, and it displays iTheater's user interface and use of decision making.

6. EVALUATION

Many features and improvements were not included in this project for various reasons, including time constraints and limited resources. Alternatives to some of the design choices could be used for future developments in order to make the application more immersive and user-friendly. In particular, some ideas that occurred to me during the development of this thesis include real time movement and navigation, continuous decisions, and multi-player forums.

6.1 Real Time Movement and Navigation

In Understanding Partner Abuse and other iTheater applications, there is no place for any spatial navigation. In Henderson's Virtual HIV Clinic, there exists some spatial geometry but movement is in discrete mouse clicks as a user can move from one room to the next. An improvement on these methods would be to create a Virtual Reality environment through which a user can move in real-time. This continuous time navigation would better simulate real life situations and enhance the user's experience. A story could include multiple patients waiting in different rooms, giving the user different cases to handle. It could provide a more realistic space, enabling the user to truly feel as if the setting was a hospital or doctor's office rather than a computer screen.

Some prohibitive limitations may interfere with such a development, namely computer resources and performance. A virtual reality environment is significantly more taxing on an individual computer's resources, let alone in a distributed environment where the simulation must be sent over a network. In addition, handling continuous movement throughout the story is a significantly more complex task for the underlying story engine. The potential for improvement, however, would make this a worthwhile endeavor in the future.

6.2 Continuous Decisions

Currently, in iTheater applications, decisions are made in a multiple-choice panel. These decisions cause a distinct break in the telling of the story. The text will stop scrolling and the movie will stop playing so that the user has time to read and think about the options. While this may provide the user with some benefits, it detracts from the reality of the experience.

In order to avoid this pause in the action, decisions could be made such that a delay in decision making has an effect on the story. This could keep the user aware of how time can affect decisions made in real life.

6.3 Multi-Player Forum

One of the benefits of having an application written in Java and available on the web is that it is easily extensible into a multi-player forum. When iTheater was created by Echo Bridge Productions, it was intended to ultimately be a tool to create applications that could include many players simultaneously. Due to the complexity involved in multi-threaded applications, a multi-user forum was not included as part of this thesis. Future developments, however, should allow for multiple players.

The benefits of a multi-player system to the realm of clinical training would be tremendous. Healthcare professionals could work together in counseling a patient or take adversarial roles as one professional plays the part of the patient while the other plays the doctor. As a web-based application, it would allow learners to work together across any distance. It could potentially team up skilled doctors with eager students that would otherwise be unable to work together. Multiple opinions could provide a better learning experience for users.

Using movies and Internet distribution to enhance user interaction is an important next step in educational simulations. With rapidly progressing technology, the possibilities for real-time decisions, virtual spatial navigation, and multi-player forums become more likely. The potential for immersive technology grows with computer and network performance and soon will only be limited by reality itself.

7. BIBLIOGRAPHY

Ambron, S. (1990). Learning with Interactive Multimedia. Redmond, WA, Microsoft Press.

Ambron, S. (1988) Interactive Multimedia. Redmond, WA, Microsoft Press.

Bolt, R. (1984). The Human Interface. Belmont, CA, Wadsworth Inc.

Chan, P. Lee, R. (1998). The Java Class Libraries Second Edition, Volumes 1 and 2. Reading, MA, Addison-Wesley.

Doepel, D. (1997) "Understanding Partner Abuse." Proposal for government grant to National Institute for Mental Health. Miami, FL.

Eckstein, R. (1998). Java Swing. Sebastopol, CA, O'Reilly & Associates, Inc.

Erickson, T. (1990). "Working with Interface Metaphors." The Art of Human Computer Interface Design. B. Laural. New York, NY, Addison-Wesley.

Halliday, M. (1993). Digital Cinema – An Environment for Multi-Threaded Stories. Cambridge, MA, MIT.

Henderson, J. (1997) "Comprehensive, Technology-Based Clinical Education: The Virtual Practicum." Interactive Media Lab, Dartmouth Medical School, NH.

Kobb, M. (1994). Simplifying the Graphical User Interface. Cambridge, MA, MIT.

Kim, U. (1990). A Graphical User Interface for Lucy. Cambridge, MA, MIT.

Maremaa, T. Stewart, S. (1999) QuickTime for Java: A Developer's Reference. San Diego, CA, Academic Press.

Mountford, J. (1990). "Tools and Techniques for Creative Design." The Art of Human Computer Interface Design. B. Laurel. New York, NY, Addison-Wesley.

Murray, J. (1997). Hamlet on the Holodeck. Cambridge, MA, MIT Press.

Nicol, A. (1990). "Interfaces for Learning: What Do Good Teachers Know That We Don't?" The Art of Human Computer Interface Design. B. Laurel. New York, NY, Addison-Wesley.

Nelson, T. (1990). "The Right Way to Think About Software Design." The Art of Human Computer Interface Design. B. Laural. New York, NY, Addison-Wesley.

Norman, D. (1982). Learning and Memory. San Francisco, CA, W. H. Freeman and Company.

Norman, D. (1986). User Centered System Design. Hillsdale, New Jersey, Lawrence Erlbaum Associates.

Ravden, S. Graham, J. (1989). Evaluating Usability of Human-Computer Interfaces: A Practical Method. New York, NY, Halsted Press.

Tognazzini, B. (1990). "Consistency". The Art of Human Computer Interface Design. B. Laurel. New York, NY, Addison-Wesley.

Tufte, E. (1983). The Visual Display of Quantitative Information. Cheshire, CT, Graphics Press.

Tufte, E. (1990). Envisioning Information. Cheshire, CT, Graphics Press.