Programming Surveys for the MIT Information Acceleration Project

by

Jimmy M. Hsu

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Bachelor of Science in Electrical Engineering and Computer Science

and

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1996

Author .....
Department of E Science
May 22, 1996

Certified by.....
William J. Qualls
Associate Professor
is Supervisor

Accepted by .....
Morgenthaler
Chairman, Department Committee on Graduate Theses

Programming Surveys for the MIT Information Acceleration Project

by

Jimmy M. Hsu

Submitted to the Department of Electrical Engineering and Computer Science
on May 22, 1996, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science in Electrical Engineering and Computer Science
and
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The goal of this thesis project is to describe and document the recent programming that was done for the MIT Information Acceleration Project. Our objective was to create extremely user-friendly programs to collect marketing data for research studies. In order to meet the desired program specifications, new features such as video-conferencing and data storage were added to an existing multimedia programming framework. Meta-software was also created to rapidly generate marketing surveys without requiring any programming knowledge. This meta-software will be used as a homework exercise for the MIT course "Advanced Marketing Management" to allow students to quickly create their own studies.

Thesis Supervisor: William J. Qualls
Title: Associate Professor

Acknowledgements

This thesis is dedicated to my family whose generous emotional and financial support has made my dreams come true. No parents could have set a better example as role models for me than Dennis and Anne Hsu. It's also nice to have wonderful grandparents who are so loving and proud. I would especially like to thank my brother, Peter Hsu, for his unwavering support through times thick and thin.

I would also like to acknowledge the support of the other members of the Information Acceleration team at MIT: Prof. William Qualls, Jon Bohlmann, Conan Dailey, Miran Rechter, and Ahmed Benabadji. They were a wonderful source of inspiration, and many of the innovations in this thesis grew out of their suggestions and ideas. Without them, this thesis would not have been possible.

Heartfelt thanks go out to Prof. Qualls for his trust and support. Thanks also go to Florian Zettlemeyer, who is a helpful and handy officemate. Thank you, Miran, for helping me with my flowchart, among many other things. Thanks go out to Jon and Ahmed for their patience and understanding. It was a real pleasure to work on IA designs with them. I also want to thank Mila Getmansky, Dina Mayzlin, Peter Madden, Sean Chiu, Pushpinder Singh, and Yuan Lu for their friendship and moral support.

Contents

1. Introduction

Interactive multimedia marketing surveys are a key element of the Information Acceleration (IA) paradigm described by Urban et al. [1,2] IA is a new market research and modeling methodology that is used to provide customers now with information that will characterize the future product adoption decision. Survey participants then give responses through an electronic questionnaire.

The implementation of IA surveys is discussed in the first section. An outline of the IA design process is given. The choice of hardware and software used in IA studies is discussed. Guidelines for user-friendliness in IA design are also given.

Section two discusses many recent improvements to the IA multimedia framework. Techniques for improving code efficiency were discovered. In particular, the size of IA programs could be reduced by careful re-use of code. Also, a group of objects can be replaced by a single well-written object.

Also in section two are descriptions of many recent additions to the software. New movie objects were written to improve movie behavior. The ability to record data into a text file was added. Finally, a special video-conferencing feature was added.

Finally, meta-programming techniques were used to create software tools used in a classroom exercise for marketing students. This technology was used by students to rapidly generate customized marketing surveys without doing any programming. This classroom exercise will be described in the third section.

2. Section I: IA Technology

The technology that is currently used for interactive multimedia marketing research at the MIT Information Acceleration Project shall now be discussed. There are many issues concerning the choice of hardware and software used in IA studies.

Additionally, guidelines for user-friendliness that are relevant to IA design shall be given. Although easy to overlook, user-friendliness should be at the forefront of the concerns of a software designer. It is certainly essential in ensuring the success of an interactive multimedia research study.

The IA design process begins with a clear design specification, which is a blueprint of the marketing survey. This design specification consists of a sequence of screens to appear in the survey. Also, text to appear on each screen is given. The design specification also explains the functionality of each screen.

Implementing the IA is a three-part, iterative process. (A flowchart of this process is given in Figure 1 below) First, objects such as buttons, text, or graphics for each screen is arranged into a layout following the guidlines for user-friendliness. A prototyper may be used at this point to check the layout for problems. These problems are corrected early.

Next, media for the program is created. The media is designed to fit into the space allotted by the layout. Graphics, movies, text, and sound are all used as media in an IA program, hence the description "multimedia marketing survey".

The program is then written, again following the guidelines for user-friendliness. The size and position of objects are dictated by the layout. The functionality of the objects is dictated by the design specification. The program is then compiled and run with the finished media.

Errors in the program code are then iteratively corrected in successive versions of the IA program. Once the program is error-free, it is ready for a pre-test on volunteers. Testing the IA often reveals changes that must be made. This can lead to changes in the design specification, which propagates down the flowchart to lead to an updated program. The IA program is finished only after it is thoroughly tested.

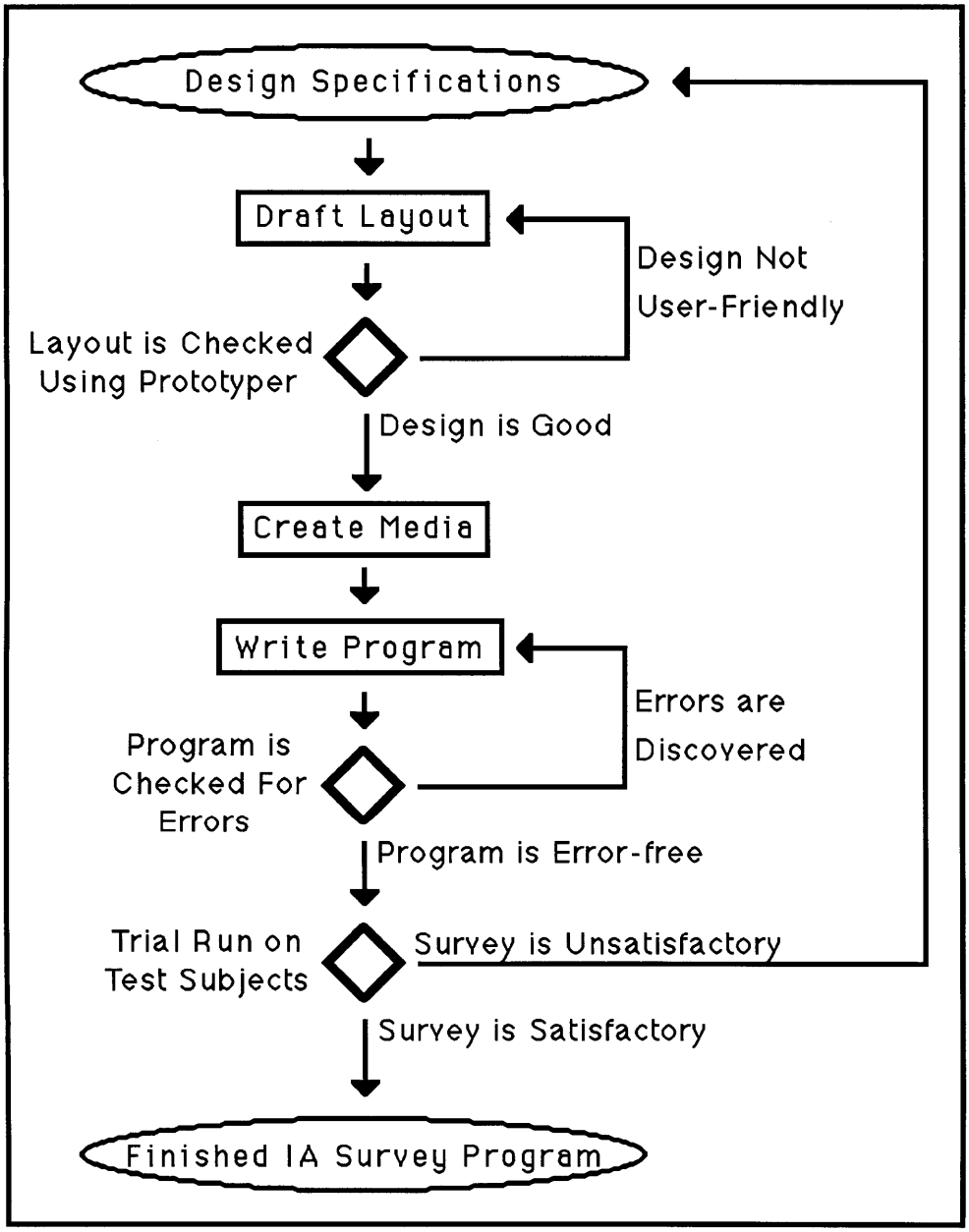


Figure 1: Flowchart of IA Design Process

3. Computer Hardware

The Apple Macintosh[1] was selected as our hardware survey platform because of its user-friendliness. Participants in IA studies used a simple one-button mouse to interact with the IA program. Because there were no prior computer experience requirements, a survey participant might be using a computer for the very first time. However, even first-time computer users were able to quickly pick up the procedure and were entering responses within minutes.

IA survey programs were compiled to run on Macintosh computers with 8MB of available RAM with a monitor that can display 640x480 resolution at thousands of colors. The use of a large monitor is recommended to ensure visual clarity.

The Power Macintosh 8100/80 AV is used as a development platform because of the built-in video capture capabilities. The AV board is used to create standard QuickTime[2] movies using video footage from VHS videocassettes.

In the past, IA surveys have used laserdisk technology to display high-quality video at realtime frame rates. Although the quality of the laserdisk video was superior, pressing custom laserdisks present several drawbacks. Chief among these is the delay from the time video is sent to be pressed onto disk and to receipt of a finished disk. As a result, disks may need to be pressed more than once, leading to waste. Laserdisk equipment, and laserdisk production costs were expensive, and was not on a trend to become cheaper. This was due to insufficient popularity of laserdisks in the US. Finally, controlling

---

[1] Macintosh is a trademark of Apple Computers

[2] QuickTime is a trademark of Apple Computers

laserdisk equipment within the IA software became problematic as laserdisk support diminished and software firms stopped producing laserdisk software drivers.

On the other hand, QuickTime technology was steadily improving, and hard drive costs were decreasing. By switching to standard QuickTime movies, the production cycle was reduced significantly because development did not have to wait for the turnover delay of pressing laserdisks. With less equipment to acquire, set up, and maintain, conducting a survey becomes easier. Another benefit of QuickTime is the ability to perform on-line video editing. However, the use of QuickTime movies does require that there is sufficient space on the hard drive of the survey platform to store the movies used in the study. The price of larger hard drives is now cheaper than the cost of purchasing multiple laserdisk players and producing laserdisks. Thus, QuickTime is more economical than laserdisks.

For the video-conferencing add-on that is described later, Videolabs FlexCam[3] color digital video cameras were used. The Macintosh computers were connected to a 10Base-T local Ethernet hub, which provided high-speed networking capability with sufficient bandwidth to support good quality real-time video-conferencing.

---

[3] FlexCam is a trademark of VideoLabs

4. Software Development Platform

The following is a list of software packages that was used to create surveys at the MIT Information Acceleration Project.

- Adobe Photoshop 3.0
- Adobe Premiere 4.0
- AppleScript 1.1
- Apple Media Tool and Programming Environment 1.2
- Metrowerks CodeWarrior C/C++ CW5

4.1 Creating Media

Adobe software was used for creating media for IA programs. Adobe Photoshop 3.0 was used for creating graphics for buttons and screens. Photoshop was also used to scan in graphics from magazines and brochures. Adobe Premiere 4.0 was used to create videos and animations.[4] Premiere has excellent movie editing capabilities.

4.2 Rapid Prototyping

Rapid prototyping is used to quickly plan out the areas of a survey screen. In this way, screen layouts can be tested out at an early stage in the development cycle, when

---

[4] Photoshop and Premiere are registered trademarks of Adobe

changes are easier to make. Changes are more difficult later on when a lot of programming has been done. Several tools exist for rapidly prototyping an IA design.

FaceSpan[5], which is included with AppleScript, is a fairly powerful interface builder for creating "front-ends" to applications. Objects, such as buttons, text, or graphics, can be moved around the screen by simply dragging them around with the mouse. FaceSpan programming is done in an AppleScript/HyperCard[6] hybrid language. This language is used give prototype objects limited functionality. Buttons can be programmed to start a movie, for example. Text or graphics can be updated after the user performs an action, such as clicking on a survey response. Unfortunately, it is difficult to extend FaceSpan beyond its the built-in features. Another drawback of using FaceSpan is that screens with a large number of objects exhibited slow-down and poor response times.

Another prototyping tool is the AMT meta-program. This meta-program is similar to FaceSpan in that it can generate rudimentary IA programs with limited functionality. Screen objects can be moved around or copied back and forth among screens. Screen flow can be specified by programming continue buttons to transition between screens. The AMT meta-program outputs program code in the AMT programming language, which can be compiled into an efficient application that is both quick and responsive.

The output source code can be modified to add extra functionality to allow the IA program to go beyond the restrictions of the built-in features of the meta-program. To make the source code easier to comprehend and debug, comments should be added to AMT's generic code output. Also, objects should be given more descriptive names.

---

[5] FaceSpan is a registered trademark of Software Designs Unlimited

[6] AppleScript and Hypercard are registered trademarks of Apple Computers

4.3 Programming Tools

CodeWarrior[7] C++ is used to create and debug C programs for the Macintosh. However, because of the excessively long development cycle for writing C programs, it is not practical for programming an entire IA survey. Instead, C is used for writing specialized functions when there is a need to access the underlying Macintosh operating system.

Apple Media Tool[8], or AMT, is a object-oriented software suite that is used to write entire IA surveys. The AMT programming environment has many multimedia constructs already built in, which simplifies the creation of information sources used in our studies.

4.4 Macintosh System Extensions

Several important Macintosh system extensions are needed for proper IA performance. The most recent extensions for QuickTime and SoundManager[9] provide needed multimedia capabilities used in the IA software. RamDoubler improves performance on PowerMacs through better memory management and SpeedDoubler[10] provides a more effective cache for processor emulation. Finally, MacTCP[11] provides network capabilities used in the video-conferencing add-on described later.

---

[7] CodeWarrior is a registered trademark of Metrowerks

[8] Apple Media Tool is a registered trademark of Apple Computers

[9] QuickTime and SoundManager are registered trademarks of Apple Computers

[10] RamDoubler and SpeedDoubler are registered trademarks of Apple Computers

[11] MacTCP is a registered trademark of Apple Computers

5. Guidelines for User-Friendliness

The factors in Human-Computer Interaction (HCI) which govern user-friendliness were of key concern as no prior computer background was assumed from subjects who participated in our IA marketing studies. This section gives four guidelines to follow when designing an IA. The guidelines are adapted from Hicks and Essinger [2] for use in IA studies.

- 5.1 Visual Appearance
- 5.2 Visual Clarity
- 5.3 Consistency
- 5.4 User Guidance and Feedback

IA screens are divided into two distinct functional types: survey screens and information search. It is important to make this distinction because the two functional types have different objectives and should be treated differently. Survey screens are used to gather data about the user's response to survey questions. Information search, on the other hand, provide information to the user.

5.1 Visual Appearance

Visual appearance is paramount in the design of a graphical user interface. A poorly designed interface can seriously impair the collection of survey data. In the survey section, the goal is to make the interface as unobtrusive as possible so the user can glide through a lot of screens quickly and effortlessly.

Survey screens should be kept as simple as possible, without any distracting graphics or logos or background wallpaper. This allows the user to focus on the survey without distraction. A minimalist approach in designing a simple screen that consists of a question and a row of buttons is most effective in eliciting an accurate response. (See Figure 2 below)

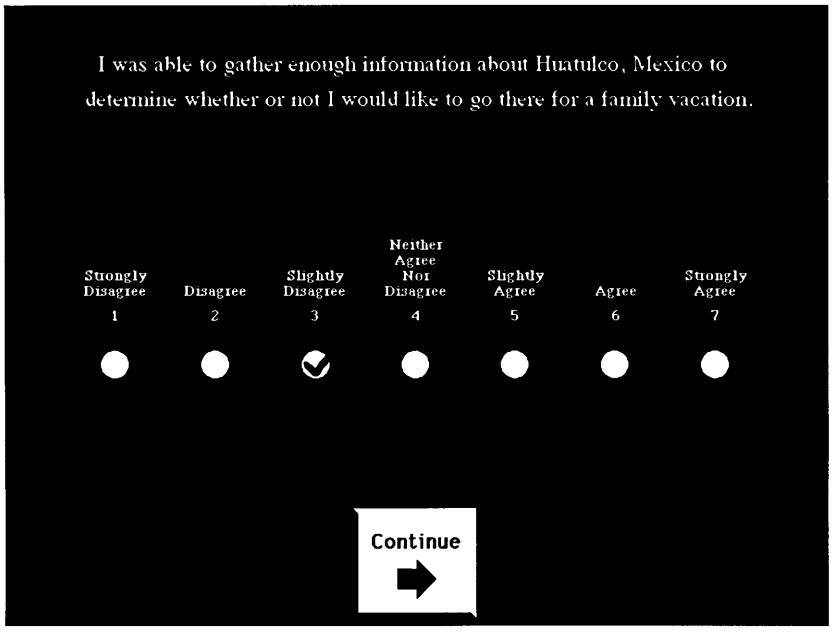
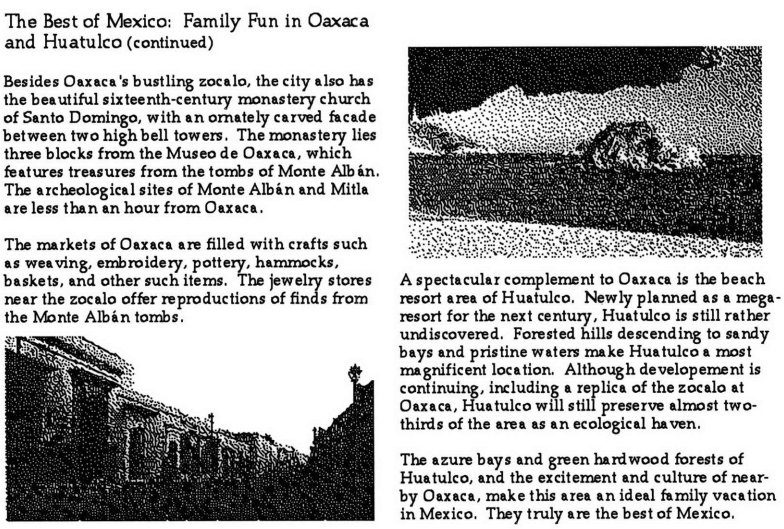


Figure 2: Survey Screen Example -- An Agree/Disagree Question

In an information search screen, just the opposite is true. We want to visually convey a sense of a realistic situation in order to transport the participant into a simulated future decision environment. For example, we may want to expose the user to a television commercial of a really new product. The commercial would not be seen for a few years. We use a realistic living room background to help the participant imagine such a situation.

Brochures used in the information search feature bright, colorful graphics on a white background, similar to a typical promotional brochure. (See Figure 3 below) Again, this is intended to make the users feel as if they are really browsing through an actual brochure.



The Best of Mexico: Family Fun in Oaxaca and Huatulco (continued)

Besides Oaxaca's bustling zocalo, the city also has the beautiful sixteenth-century monastery church of Santo Domingo, with an ornately carved facade between two high bell towers. The monastery lies three blocks from the Museo de Oaxaca, which features treasures from the tombs of Monte Albán. The archeological sites of Monte Albán and Mitla are less than an hour from Oaxaca.

The markets of Oaxaca are filled with crafts such as weaving, embroidery, pottery, hammocks, baskets, and other such items. The jewelry stores near the zocalo offer reproductions of finds from the Monte Albán tombs.

A spectacular complement to Oaxaca is the beach resort area of Huatulco. Newly planned as a mega-resort for the next century, Huatulco is still rather undiscovered. Forested hills descending to sandy bays and pristine waters make Huatulco a most magnificent location. Although developement is continuing, including a replica of the zocalo at Oaxaca, Huatulco will still preserve almost two-thirds of the area as an ecological haven.

The azure bays and green hardwood forests of Huatulco, and the excitement and culture of near-by Oaxaca, make this area an ideal family vacation in Mexico. They truly are the best of Mexico.

Figure 3: Information Search Screen Example -- A Promotional Brochure of Mexico

Careful use of color can be an extremely powerful tool in interface design. The color scheme used in the survey screens is white text on blue background. This maintains a high level of contrast, which improves readability. Also, white and blue are both neutral colors, with fewer cultural connotations, making them good choices for a research study. [4] Pure blue is a good background color because blue appears more distant to the human eye. Blue light has a shorter wavelength, which causes it to always focus in front of the retina. This makes a blue field appear defocused, as if it is receding. [5] The eye has a lower sensitivity to shorter wavelengths, such as blues, which reduces the fatiguing effects of staring at a monitor. [6]

Black on white is also a good color scheme, and is used in the information search for such screens as brochure pages or magazine articles. Black on white produces excellent contrast, which improves the readability of the text that appears on these screens. It is also the same color scheme as the documents, which reinforces the sense of realism.

5.2 Visual Clarity

We use a monitor setting of 640x480 pixels on all of our screens. This limits the amount of readable text that can fit onto a given screen. We try to keep the font as large as possible for most of the text that appears in the survey. For instructions to the user and for survey questions, we use either 18 point or 24 point Times. In the advertisements and brochures, font should not be made smaller than 12 point , otherwise the text becomes unreadable. When a brochure is scanned in, all text is re-typed to maximize visual clarity.

For labeling the scales, the font should not be made smaller than 9 point. The restriction on scale labels is less strict because the labels are read less frequently. In some fonts, 9 point is more than adequate, but on other fonts, 9 point is completely illegible. Ultimately, the designer must make a subjective decision on the readability of small text. When the IA program is field tested, pay particular attention to whether the user is having trouble reading the text.

Larger (and therefore wider) fonts tends to crowd together when placed as a label over a row of buttons. A helpful trick for this situation is to manually adjust the spacing between individual letters in the text. This can make a slight difference in text width, which may be enough to make the text just fit. Furthermore, it improves the visual clarity.

5.3 Consistency

Consistency is important because it helps maintain a mental model of the underlying marketing survey. [7] One way to maintain this consistency is to use a set of conventions that group together objects with similar functions. By giving all logically similar objects a similar look, users can instantly recognize a type of object and quickly know its intended use.

For example, many of the buttons for screen navigation are identical in appearance. They follow similar color schemes, have similar shapes and use similar graphical arrow icons to give the user additional clues about the functionality of the button.

The same idea can be applied in reverse. By intentionally giving objects with different functionalities a different appearance, it helps set the objects apart. For example, survey screens were set apart from information search by the use of different color schemes, which further reinforced the difference in the purposes of these screens. To set different scales apart, distinct fonts are used in the scale labels. To set certainty ratings apart from regular ratings, we use five boxes close together instead of seven circles spaced apart. The "X" mark is different from regular checkmarks. (See Figure 4 below)

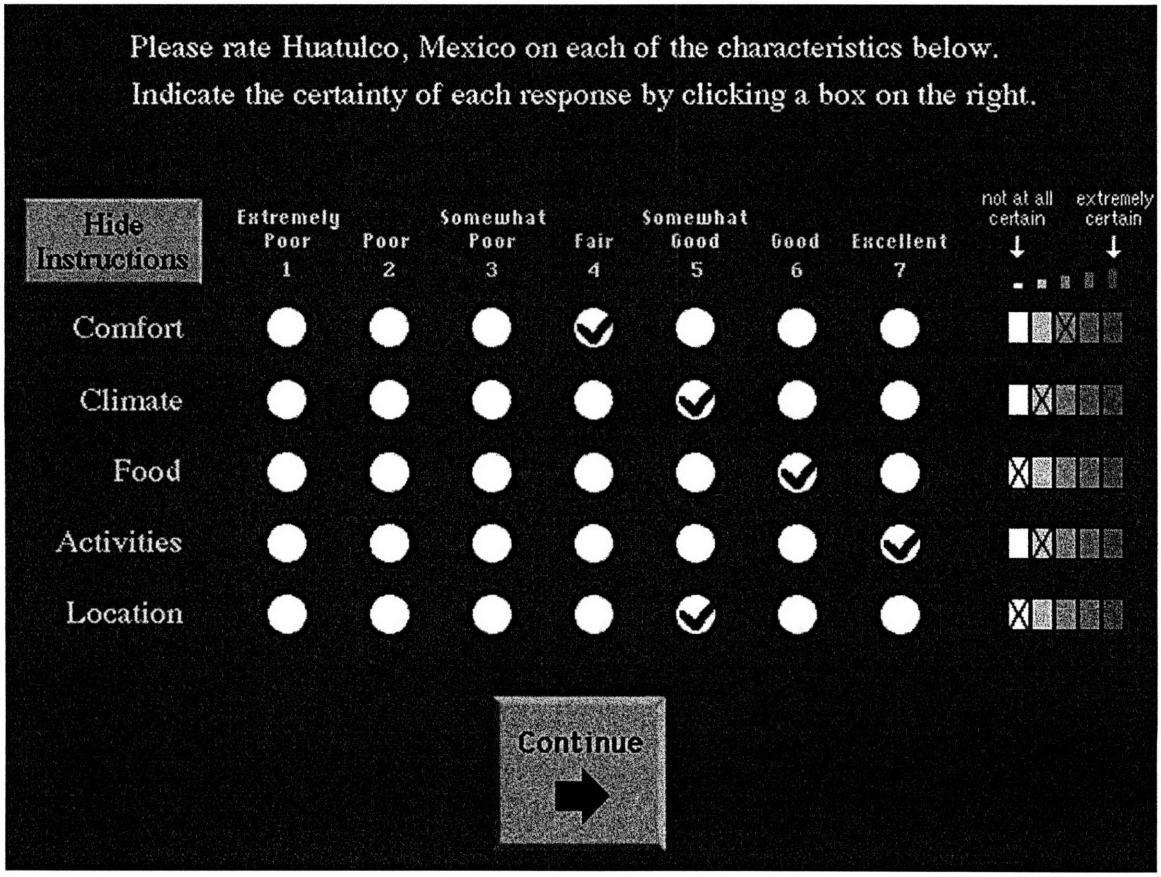


Figure 4: A Characteristics Rating Screen

5.4 User Guidance and Feedback

It is important to guide users through more complicated screens, where the potential for confusion is greater. In the screen on Figure 4, a highlight bar is used to guide the user through a sequence through each question. Although the user is free to go back and change a previously entered response, the program does not allow the user to skip ahead. This ensures that all the questions are answered before the user can move on.

Programs produced in AMT have a user-friendly feature of dynamic cursor appearance. Depending on whether or not a region of the screen is active, the cursor changes from an arrow (active) to a "no" slash (inactive). This provides a subtle but useful clue to the user about which areas of the screen can be clicked. (See Figure 5 below)

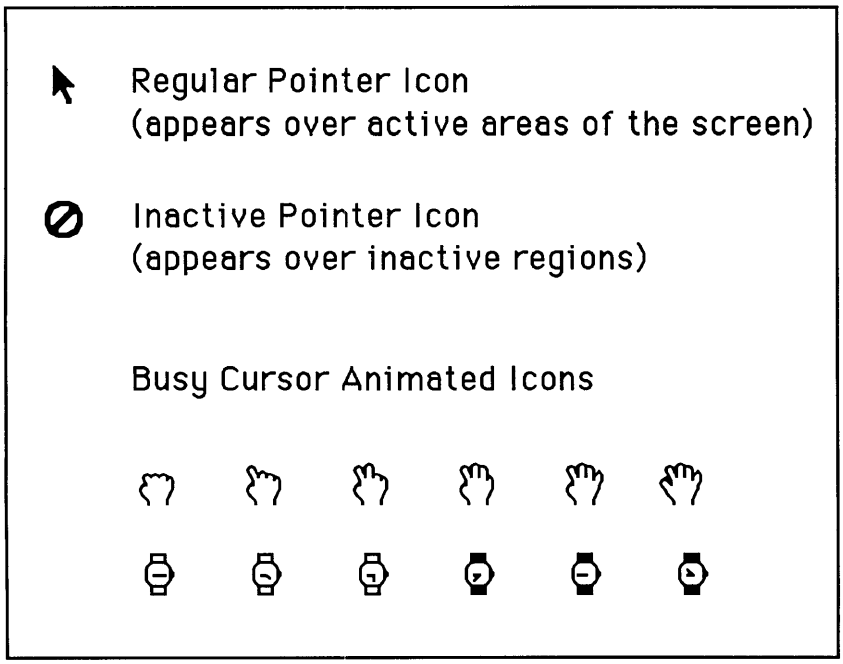


Figure 5: Cursor Icons from Apple Media Tool

The IA programs are written to be as responsive to the user as possible. Users must be given immediate feedback when they affect a change. Responsive feedback is the key to dynamic checkmarks that can easily be "erased" and changed to a different answer.

This corresponds to the user's expectations of the behavior of the checkmark based on the user's mental model. For lengthy operations, such as screen transitions, the cursor changes to a "busy animation", which indicates that the machine is too preoccupied with a calculation to respond. This gives the user a polite signal that they should wait before trying to give new inputs to the computer. This is much more than just a courtesy because without busy animations, the user might be frustrated and confused with clicks that are not immediately acknowledged.

6. Section II: Recent Improvements

This section discusses recent improvements to the Apple Media Tool (AMT) framework used to produce IA programs. Three main methods were utilized when improving code within the AMT programming language.

First, code was re-written to be made more efficient. Basically, this is achieved by doing more with less. Objects are re-used when needed for the same tasks. This reduces the size of the program and the amount of source code that is needed. Less source code is generally easier to debug. Also, by efficient usage of hotspot regions, a single well-written object can replace several screen objects. This speeds up the program significantly, eliminating otherwise noticeable delays.

Second, new features were added to the framework by writing new object classes. The existing movie objects were upgraded. New object classes for data storage were created.

Finally, for a sophisticated video-conference feature, custom C code was written and added to AMT through the external code interface. This video-conferencing feature is described in detail.

7. Improving Code Efficiency

7.1 Re-using Code

The object-oriented AMT design environment allows much of the code written for IA programs to be re-used. In an IA marketing study, questions are often repeated and the responses compared. The effects of intermediate screens (for example, an information search) can thereby be evaluated. A limited set of screen types are used to collect information on a wide range of questions. For example, a seven-point agreement screen might be used for a whole series of agreement-type survey questions. The objects used for many of these screens can simply be recycled each time they are needed. This makes all the screens of a certain type completely identical, which is important in maintaining consistency throughout the IA.

In many cases, the same object is re-used many times over the course of the survey. One example of this is the ClockText object that is used to time the information search. [See Appendix A for source code] The clock appears in every information search screen, and counts down the remaining time left for the user to gather information.

ClockText is a text object that displays the time remaining in the upper right corner of the screen. The **Idle**() procedure is called by the application every second or so, which updates the ClockText graphic. ClockText only counts when its internal variable, **IsCounting**, is true. This is because the clock is supposed to only count down when the user is actually searching an information source, not when the user is deciding which source to view next. Since ClockText is a single object, its internal local variables can keep track of the remaining time regardless of which screen the user is currently exploring.

7.2 Effective use of HotSpots

AMT's excellent handling of hotspots allow us to define specialized regions for the user to click. By clever use of hotspots, multiple buttons can be consolidated into a single object. This is very useful in our application domain, because survey questions are often answered by clicking on a row or series of buttons.

An example of this technique can be found in Appendix A. The source code is taken from an object that displays a row of circles on the screen and accepts mouse clicks to any of the circles in the row. Exactly which circle to mark is deduced by examining the x-coordinate of the mouse when it was clicked. A checkmark graphic is then moved over the correct circle.

In this row object, a hot region of seven disjoint circular regions is used. The hot region consists of a variable-length hexidecimal string that uniquely specifies a mask region on the screen. This string is produced using the AMT meta-program application.

This single object takes the place of seven regular objects. By using these row objects instead of individual circles, a screen with many buttons such as the one depicted in Figure 4 (reproduced below) may have quite noticeable improvements in response time.

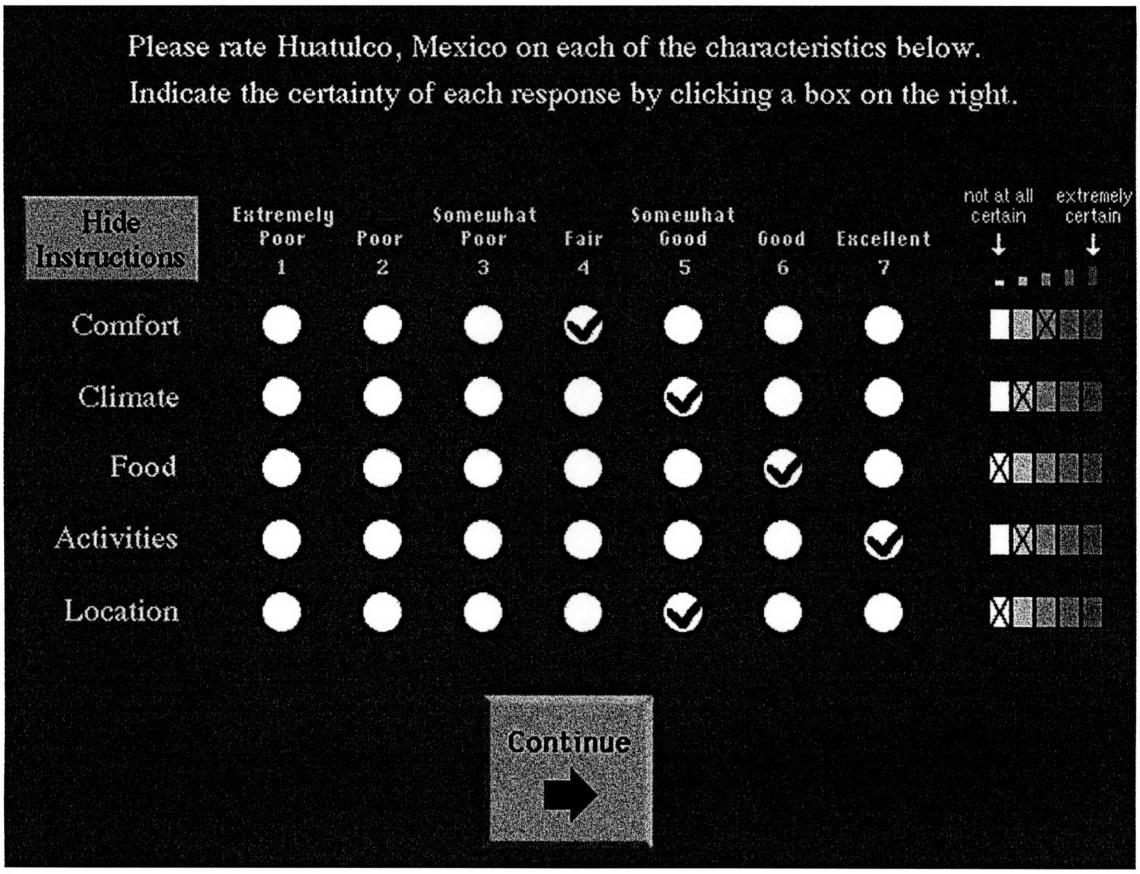


Figure 4: A Characteristics Rating Screen (repeated)

By way of comparison, if above screen was implemented in FaceSpan, dramatic delays with durations over five seconds might be observed. Using AMT, delays with durations of approximate a second might be observed on the same machine. With hotspots, this delay is made unnoticeable. This demonstrates the power and effectiveness of this particular code optimization.

This effect is more pronounced on screens with an abundance of objects. It is less noticeable on screens with few objects.

8. New Object Classes

Because AMT is object-oriented, we can define a new class of objects to override an existing class specification. This method was used to improve the MovieClient objects, to store data, and to add video-conferencing to AMT.

8.1 Improving Movie Behavior

The original movie objects in AMT were limited to the standard movie functions, such as playing and stopping. A method was needed to force everything else to wait while one movie was running. This was accomplished through the **WAIT_MOVIECLIENT** class. Any movie instantiated with **WAIT_MOVIECLIENT** inactivates other objects when it is running. It does this by intentionally using up extra clock cycles, preventing the user from clicking anywhere else while the movie is running.

Many of the screens required actions to be triggered upon the completion of a movie. This is accomplished by adding a **Finished()** procedure to the **MOVIECLIENT** class to produce a new class, **FINISH_MOVIECLIENT**. [See Appendix B] For example, if a user clicked on an information source to display a TV commercial, the IA would transition to a screen and run a QuickTime movie of the commercial. At the end of the movie, the **Finished()** procedure would be triggered and the IA would return to the original screen.

8.2 Recording Data

Another important customization that was done was adding the ability to store data collected in the survey. This was done through the new **STORE_STRING** class.

Because saving data involves a call to the Macintosh file system, it had to be done through the external C interface.

The data had to be stored to unique files based on numbers entered at the start of the IA to identify the participant. The data is stored as ASCII strings in a standard text file, which can be readily imported into a spreadsheet package for numerical or statistical analysis.

To utilize the new feature, objects such as continue buttons are instantiated with the **STORE_STRING** class instead of the typical **MEDIADELEGATOR** class. The new **STORE_STRING** object inherits the procedures of the **MEDIADELEGATOR** class. Thus, the new data storage feature is added on top of the existing **MEDIADELEGATOR** class.

When this continue button is clicked, it records any responses that the survey participant has given for the current screen. It does this by calling its **StoreText()** procedure with a string to be stored as its argument. For example, **StoreText(** "Hello, World!" , "myfile") would append the string "Hello, World!" at a new line at the end of the file named "myfile". If the text contains an integer value, this integer must first be type-cast (i.e. changed) into a string.

9. Video Conferencing

9.1 Why Do We Need Video Conferencing?

One area of active research by the IA project team is the study of multi-person decision behavior. [8] For many purchase decisions, multiple decision participants are involved, which makes face-to-face interaction desirable. This may be accomplished through a video-conference session.

Ellis et al. [9] studied group interaction through shared computing environments, by comparing face-to-face and distributed sessions among groups of three. Face-to-face sessions were conducted in the same room and distributed sessions were done on computers running electronic whiteboard software and using a conference call on speaker phones for voice communication.

They found that without face-to-face communication, discussion was more difficult and distributed sessions "seem to require more concentration and are more tiring." Distributed sessions encouraged groups to split up work on different parts of a task, whereas in "face-to-face" sessions, people tended to work together.

Video-conferencing therefore adds an important face-to-face element to a distributed session. Gallagher and Krant [10] found that audio and video conferencing can help users coordinate their activities by simulating aspects of face-to-face interaction. Furthermore, video-conferencing provides "assurances" [11] to the users about the activity and participation of the other party, enhancing the sense of collaboration.

9.2 A Video-Conference Program

A commercial video-conferencing package is not ideal for an IA study because of the complex nature of configuring a session. Each time a session is started, someone would have to interact with a complicated interface. What is desired is a way to launch a video-conferencing session with a single mouse-click. The user should also be able to quickly terminate a video-conferencing session and resume the IA survey.

Conan Dailey's research into a Macintosh multimedia server using QuickTime [12] resulted in a video-conferencing program written in C. Improvements to his code was suggested in his thesis paper, and these modifications had to be made in order to allow his program to properly interface with AMT. In particular, a routine for gracefully exiting a video-conferencing session was written. (Dailey's original test programs could not be stopped.) All of Dailey's background processes and code threads are terminated upon exit so that memory can be deallocated and the computer restored to a state from which the IA to continue.

9.3 Launching a Video-Conference Session

In order to add video-conferencing capability to AMT programs, a provision was needed that would allow an external program to be launched from within a program compiled with AMT. Because this is essentially a call to the Macintosh file system, it is best implemented using the AMT external code interface. A new object class, **LAUNCH** was created which has the procedure **DoLaunchConference()**. [See Appendix B] This procedure calls a C function through the external code interface. The C function performs a standard operating system call which launches the video-conferencing program.

Typically, this would be implemented as a button object that appears on an IA screen. When the user clicks on the button, it calls its **DoLaunchConference()**

procedure, which launches the session. The user sees two windows. The first window displays a view from the user's own video-conference camera. Essentially, this acts as a mirror, letting the user know what is being sent to the other user. The second window is a view coming from the other user's camera. This window is blank until the second IA launches its video-conference session. As soon as both computers are running the video-conference program, the survey participants appear on the screen and can see and hear one another.

They can have an online discussion of information they were recently exposed to. Once they are finished with their discussion, respondants simply click their mouse, which calls the exit routine and terminates the video-conference session. The IA then continues from that point on.

10. Section III: The 15.820 Software Tools

The issues of IA design shall now be examined in the context of a set of software tools developed for a homework exercise for the MIT course "15.820: Advanced Marketing Management." The software tools consists of three functional blocks:

- Builder meta-program compiler
- IA survey interpreter
- Top-level control scripts

10.1 The 15.820 Development Path

The goal of the 15.820 Software Tools is to take the place of a programmer team in the IA development path. Figure 6 (below) is a diagram of this development path. The survey designers are at the top of the diagram. A survey designer produces a design specification of a desired marketing survey. On the left path, this design specification is passed along to a team of programmers who implement the design and produce an IA survey program.

There is an alternate path on the right which depicts the use of the 15.820 Software Tools. In this path, the design specifications go directly into the 15.820 builder which produces an IA survey program. Note that the 15.820 path, as drawn, is shorter than the typical path. This was done intentionally to emphasize the reduced development time using the 15.820 Software Tools.

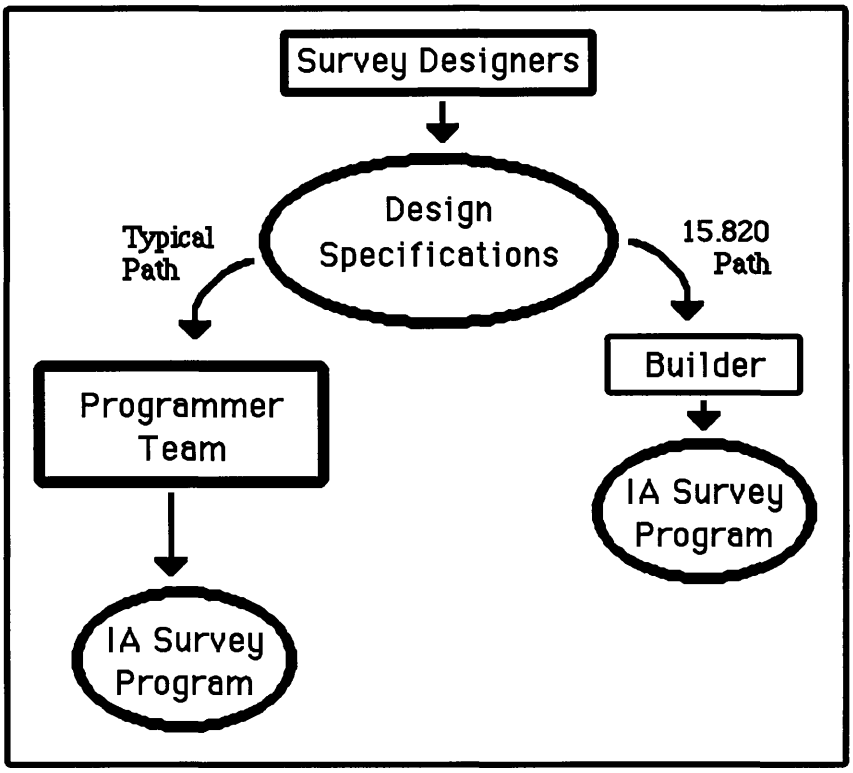


Figure 6: Diagram of IA Development Path

10.2 What is a Meta-Program?

Abramson and Rogers define the term "meta-program" as any program that treats another program as data. [13] Meta-programs are a class of programs that operate under a framework of formalized grammar rules that are used to describe another program. The 15.820 Software Tools operate under a defined syntactic framework that is used to uniquely describe a class of marketing survey applications. Thus, because the 15.820 Software Tools is a program that treats another program as data (in this case, a survey program), it is a meta-program by definition.

10.3 The 15.820 Meta-Program Syntax

Builder is a program that is capable of compiling an IA design into an IA survey program. (See Figure 6) After someone inputs a full survey description into Builder, it compiles the design and outputs a set of specification files that follows the 15.820 meta-program syntax. This compiled survey design is then loaded by the IA survey interpreter, which emulates the online interactive multimedia survey that was created by the student.

Twenty-two specification files are needed to uniquely define a survey. Together, these twenty-two files make up a compiled survey design. These files consist of:

- "Sequence Data": specifies the sequence of screens encountered by the user.
- "Info Search": specifies which buttons are to be enabled during information search.
- "Text1" through "Text20": specifies header text to appear in each of twenty available survey screens.

A key constraint of the homework exercise is that surveys always consist of a linear sequence of screens selected from a set of available screens. These screens are always encountered by the survey participant in the order of this sequence.

Each available screen is assigned a numerical screen type value. Thus, to uniquely characterize the student's sequence of screens, a list of screen type values is stored. For example, "2 6 4 1" would describe a survey that begins with a Type 2 screen, and is followed by a Type 6 screen, a Type 4 screen, and finally a Type 1 screen. This list is stored in the file "Sequence Data".

Another constraint in this homework exercise is the choice of information sources available to the student. The student selects from a set of five available information sources. To store which sources the student has selected, a list of five boolean 0 or 1 values are saved. Each of these values represent the availability of a different information source in the IA.

These values operate independently, so it is entirely possible for the student to create a survey with all five information sources disabled. (In this case, Builder would store the string "0 0 0 0 0" into the "Info Search" file.) However, because the continue button on the information search can only appear after at least one source is explored by the user, the student would quickly discover an error when a trial run of the IA is performed. All screens after the information source would never be seen because the IA would always stop at the information search screen in this example.

The final specifications that must be stored by the Builder are the header text files to appear at the top of each screen. The text can be any string typed into the Builder by the student. Although there is no formal size restriction on the length of the header text, there is a practical limit of about 5 lines of text. Any text after the fifth line of the text field is scrolled beyond the display area and will not appear to the survey participant. The student knows this because Builder displays the header text of each screen exactly as it appears to the user. ("What You See Is What You Get") It is sized, formatted, and centered the same way as would appear in the IA survey.

11. Builder

The main challenge of the 15.820 homework exercise was to make it possible for students to create a customized marketing survey without doing any actual programming. A facilitator program, called Builder, was designed towards that end. When Builder is run, the user sees the Builder workscreen (see Figure 7). The Builder workscreen is divided into four major areas:

- Header Text: Any text to appear at the top of the screen is typed into here.
- Current Screen: The lower left region contains a preview of the screen that is currently being constructing. The arrows select the type of screen that will appear in the IA simulation.
- Screen Sequencer: The lower right region allows graphically depicts the working sequence of screens as a series of miniaturized thumbnail graphics.
- Save To Disk/Exit Program: The "Save To Disk" button allows students to save their work midway. Clicking on "Exit Program" automatically compiles and saves the survey onto the hard drive and then quits Builder.

The student implements a survey with the Builder program by first arranging a sequence of screens in an order that the user is to encounter. This is done by selecting a screen from the Screen Sequencer. The student then uses the arrows to select the type of the screen that is to appear for each position in the sequence. Any text to appear at the top of each screen must be added. Finally, the student clicks "Exit Program", which saves this survey and quits out of Builder.

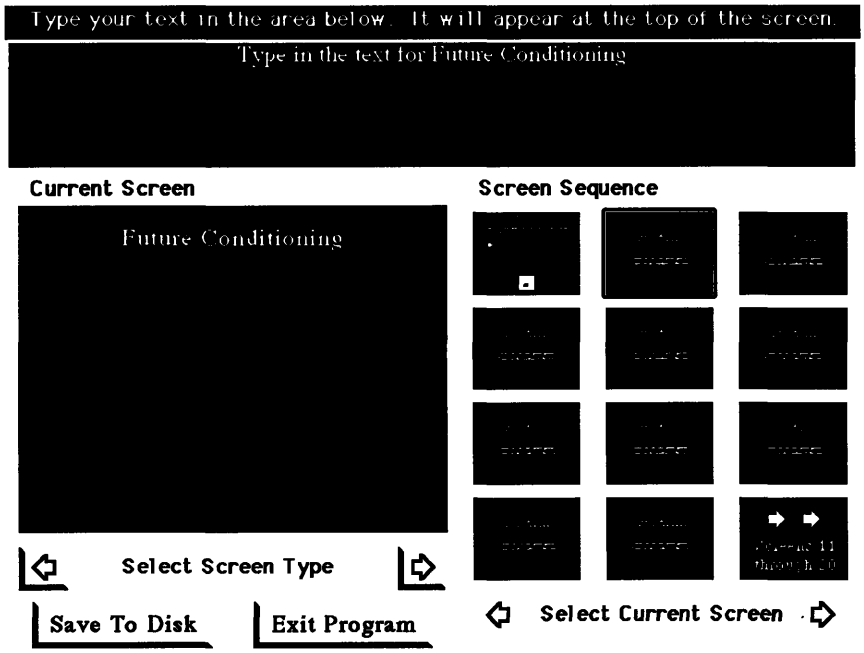


Figure 7: Builder Workscreen

11.1 HCI Issues in design of Builder

When designing a custom survey, the student must keep track of the screen sequence that has already been built up. The Builder assists the student in this regard by maintaining a updated graphical representation of the sequence as a string of thumbnail graphics. By looking at the thumbnails, the student can visualize the shape of the survey.

A bright red highlight is used to keep track of which screen the student is currently working on. The attention-drawing red makes this highlight easy to see and makes it an effective marker. It is updated immediately if the user moves to a different screen, giving immediate feedback to the user.

The workscreen is divided up into four separate areas. Subtle color hints reinforce this screen structure.

Finally, a voice-over reminds the student that the first screen (used for Future Conditioning) is never changed. This is necessary, because there is no other signal to the user indicating a special case for the first screen.

12. The 15.820 Survey Interpreter

The IA survey interpreter is an application that emulates a compiled survey design. The IA program reads in all twenty-two specification files and displays the Welcome screen. At this point, the computer is ready for the survey participant. Whenever the user clicks on the "Continue" button, the program determines the type of the next screen and transitions to the appropriate screen. It also puts the appropriate header text to appear at the top of the screen.

The default screen type is the Final screen. Thus, the survey always ends on the Final screen, which instructs the participant to see the assistant. Because there is no "Continue" button on the Final screen, the survey participant cannot exit the IA with the mouse. This prevents the survey participant from accessing the computer. The survey assistant can exit the IA program by typing "Command-Q" on the keyboard.

During each transition, any results that was entered on the preceding screen is saved to the data file using the **StoreText()** procedure described in section 8.2. This ensures the survey participant will be able to continue midway through a survey if there is a computer malfunction. Although a system crash is rare, it is important to be able to quickly reboot the machine and reset the survey to the last screen if a crash does happen. This lets the user continue on with only a minor delay. The second time the survey is run, it appends its output data to the previously created data file. The old results are not lost. Afterwards, the data file must be manually corrected, to delete any intermediate garbage values stored by the IA program.

13. Control Scripts

The control scripts are implemented in AppleScript, which allows complex operating system tasks to be automated and hidden underneath a regular double-clickable application. We used the following four scripts:

- Run Builder
- Copy To Floppy
- Run IA
- Print Survey Results

The control scripts were motivated by a need to simplify routine tasks for students who may not be familiar with the Macintosh operating system. A complete survey design consists of a set of twenty-two ASCII specification files. These specification files can be found in any of four separate locations on the local hard drive, depending on the student's stage in development. (See Figure 8 below) The specification files are also saved to and restored from a floppy disk provided to the student.

In the beginning, fresh template spec files can be found in the NEW_SPECS folder. These files must be copied into the INPUT_SPECS folder in order for Builder to access them. Builder outputs the student's work into the same folder as the Builder application. So newly-built files can be found in BUILDER FOLDER. The student is asked to copy these files onto a floppy disk to be turned in at the completion of their exercise. Finally, the IA survey program loads spec files from the IA_SPECS folder.

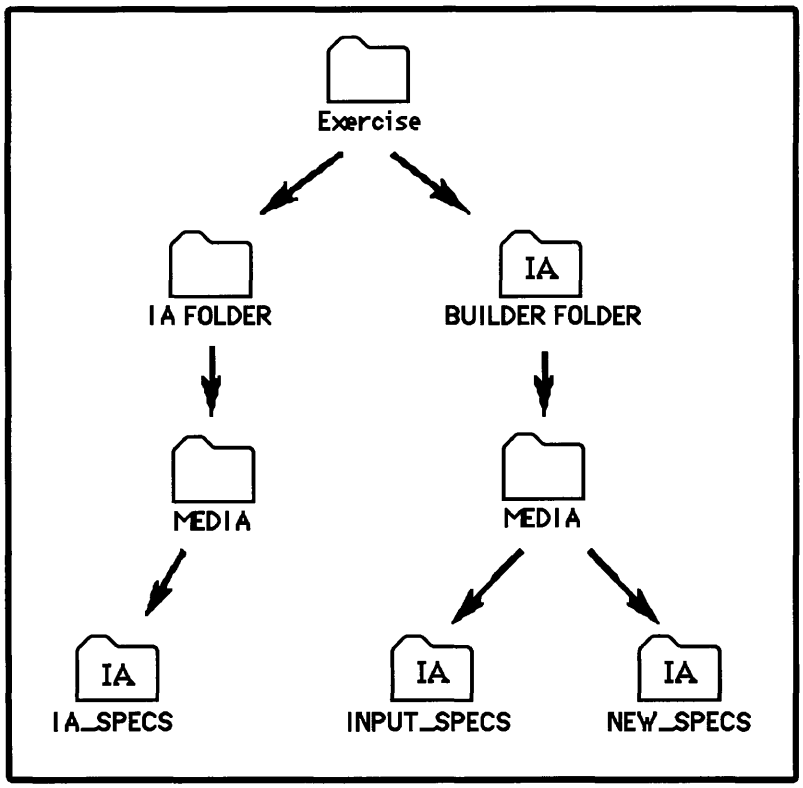


Figure 8: Possible Locations of Specification Files (marked with "IA")

Instead of asking the student to perform the multiple steps needed to copy files to the correct location, the complex task of file management is simplified into a series of dialog box choices. (Such as the one shown in Figure 9 below)

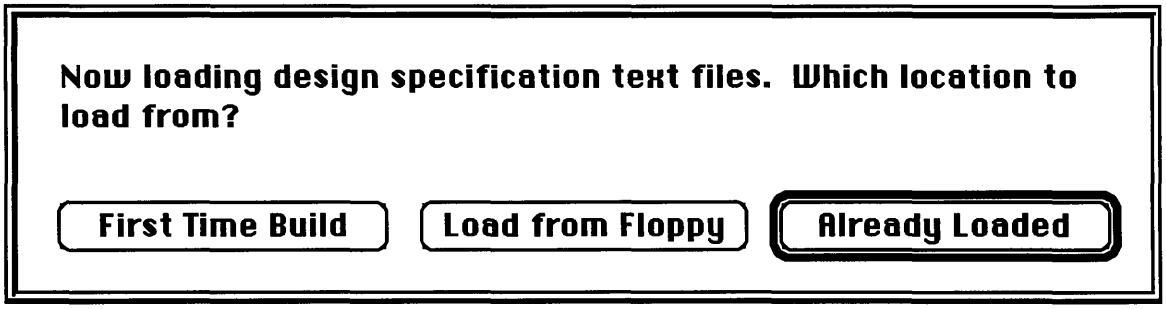


Figure 9: An Example of A Control Script Dialog Box

14. Comparison of Application Creation Software

It is useful to compare the 15.820 Software Tools to other application creation software to form a context by which to understand the strengths and weaknesses of the software. We now compare the 15.820 Software Tools to five other software packages which have been used in the past to create IA surveys. The software will be compared along the following four characteristics:

- Ease of Learning: How quickly can new users begin using the software effectively?
- Ease of Use: How easy is it to create the IA? How easy is it to debug?
- Development Time: How quickly can an IA design be implemented?
- Power: How easy is it to implement new design ideas?

Metrowerks Codewarrior represents one extreme end of the spectrum. It is extremely difficult to write and debug a large program in C. Furthermore, writing large programs in C takes an extremely long time. However, the primary benefit of C is its almost unlimited power. C will implement almost any functionality that can be reasonably from a computer. It is clear that C is adequate for short pieces of code. At certain times, it is needed for its power, such as when writing a video-conferencing program.

The Apple Media Tool Programming Environment is similar to C, but not as far to the extreme. It is somewhat easier to debug AMT code, and is somewhat easier to use to write large survey programs. This translates into a shorter development than C, however still lengthy compared to the other software packages. The redeeming benefit of AMT is its

extensibility, which gives AMT the unbridled power of C through the external code interface. We saw many examples of this in Section II.

In the past, MacroMedia Director 4.0 was used to create IA programs. Code drivers were available for Director to drive the laserdisk players. We moved away from Director because of the difficult of implementing special customizations. We were unable to customize the laserdisk drivers, or add new functionality, such as video-conferencing.

FaceSpan and the AMT meta-program boast short development times and are fairly easy to learn and use. However, the major drawback of FaceSpan is the limitations that one encounters when one attempts to implement new design ideas. AMT can output source-code for the AMT programming language, which makes it more attractive than FaceSpan. However, the value added by using AMT is minimal, because it is relatively quick and simple to write such source-code from scratch. Because of their short development cycles, both of these tools can be useful for prototyping, since a prototype does not have to be fully functional in order to get a feel for how a screen will work.

The 15.820 Software Tools fall at the other end of the spectrum. Extremely easy to learn and use, a custom IA can be created almost overnight. However, the drawback is that not every IA can be done. The 15.820 software places heavy restrictions on what can and cannot be implemented. Table 1 summarizes the above comparisons.

Table 1: Comparison of Application Creation Software Titles (Summary)

•Software Title	Ease of Learning	Ease of Use	Development Time	Flexibility and Power
•Metrowerks Codewarrior CW5 C/C++	Difficult	Extremely Difficult	Extremely Lengthy	Excellent
•Apple Media Tool 1.2 Programming Language	Difficult	Somewhat Difficult	Lengthy	Excellent
•MacroMedia Director	Average	Average	Lengthy	Somewhat Limited
•FaceSpan	Average	Easier	Medium	Somewhat Limited
•Apple Media Tool 1,2 Meta-Program	Easier	Easier	Short	Extremely Limited
•15.820 Software Tools	Extremely Easy	Extremely Easy	Extremely Short	Extremely Limited

15. Future Directions for Research

15.1 Video Conferencing

Currently, the video-conferencing software is built around a two-client arrangement. The performance is quite good. As documented by Dailey [12], moving to three-client video-conferencing might be problematic. The performance of the existing code degrades noticeably when an additional client is added. The quality of the audio and video drops significantly. This might be remedied either through improvements in software or hardware. Installing newer releases of system software has improved performance in the past. Also, upgrading to faster computers may also yield the desired boost in performance.

It is highly desireable to pass values across the clients. Thus, another improvement to the video-conferencing would be to allow IA clients to communicate data and share responses that were previously given. This can be useful for conducting research of group decision behavior. (See Bohlmann [11] as an example of this type of research)

15.2 The 15.820 Software Tools

Although the current meta-program framework fully implements the needed functionality for the 15.820 homework exercise, future improvements might allow users to create more generalized surveys.

It is certainly possible to make more screen types available to the student. With the video-conferencing add-on, a new screen for online discussion can be added to the roster

of screen types. However, the more screen types there are, the longer it takes to scroll through them using the Builder.

The ideal goal of the meta-software would be to allow the student to create a highly-customized survey without the restriction of choosing from a suite of available screens. This level of sophistication can be worked into the meta-software framework by adding an object-level architecture as described by Abramson. [13]

It is possible to implement a toolbox of standard survey objects without crossing the line of requiring the student to write programs. By using a "drag and drop" interface, the student might be able to create more customized screen layouts. The positions of each object and the context of its use would be stored in the survey specification following the meta-software convention.

Also, a more expanded set of available information sources might be useful. It might even be possible to generalize the information that appears to the survey participant. Currently, the 15.820 design tools limit the students to five pre-designed informations sources about a vacation package to Mexico. By careful manipulation of media files, the IA survey could play any movie or display any graphic in lieu of the standard default Mexico information.

Perhaps the student would select from a library of available information sources. Or, if the student can somehow create new graphics tailored to a particular study, they would no longer be restricted by what is available. Unfortunately, the process of creating custom information search files may simply be too difficult and time-consuming for a homework exercise.

The trade-off in adding power and extra flexibility to the software is the increased complexity of the tools, and the corresponding increase in learning time needed to become proficient in using the software.

16. Conclusion

In the course of the above research, many exciting new directions were explored at the MIT Information Acceleration Project.

First, a stable IA development platform was realized and described. Several principles in the field human-computer interaction were given to guarantee the user-friendliness of IA software.

Secondly, many new features were added to the Apple Media Tool framework, allowing new possibilities for marketing research studies. One such feature is the exciting video-conferencing technology. This demonstrates that IA technology can be expanded in new directions to accomodate growing marketing research needs.

Finally, a meta-program was written to enable marketing students to rapidly develop customized surveys. The software achieves the goals of being user-friendly, easy to learn, and avoids the need for the students to program. This technology could be used to potentially speed up the IA development process.

17. Appendix A: AMT Source Code Examples

17.1 ClockText Example

```
object IN_10_clocktext is MEDIADELEGATOR
has
      searches;
      is_counting;
      theCount;
      minutes;
      seconds;
      timer;
      time_is_up;

      OffScreen()
            do
                  self.Enable(false);
                  self.Show(true);
                  self.timer := ( Clock.GetTime() +
self.theCount.ToInteger() );
            end;

      SetTime()
            do
                  self.seconds := self.theCount / 1000;
                  self.minutes := self.seconds / 60;
                  self.seconds := self.seconds % 60;
                  if (self.theCount < 0) then
                        self.time_is_up := true;
                        self.Target.SetString("0:00");
                  else
                        if (self.seconds < 10) then

      self.Target.SetString(self.minutes.ToString() + ":0" +
self.seconds.ToString());
                        else

      self.Target.SetString(self.minutes.ToString() + ":" +
self.seconds.ToString());
                        end;
                  end;
            end;
```

```
      Idle()
            do
                  if (self.is_counting = true) then
                        self.theCount := self.timer.ToInteger() -
Clock.GetTime();
                        self.SetTime();
                  else if (IN_10_main.searched_once = false) then
                        self.SetTime();
                  end;
            end;

with
      searches is 0;
      is_counting is false;
      theCount is 1200000;  -- 1200000 milliseconds = 20 minutes
      timer is 0;
      time_is_up is false;
      Target is FIELDCLIENT with Supplier is Mclock; Duration is 0; end;
      Duration is -1;
      X is 572; Y is 13; Width is 56; Height is 22;
end;
```

17.2 HotSpots Example

```
object T3_row1 is MEDIADELEGATOR
has
      selection;

      OffScreen()
            do
                  self.Enable(true);
                  self.Show(true);
            end;

      MouseDown(theX)
            do
                  if (T3_main.currenthilite = 1) then
                        T3_boxes1.Enable(true);
                        T3_hilite1.Show(false);
                        T3_hilite2.Show(true);
                        T3_main.currenthilite := 2;
                  end;

                  AClickSound.Run(true);
                  T3_main.RestoreDescription();
```

```
                    if ( theX < ( self.X + 57 ) ) then
                          self.selection := 1;
                    else if ( theX < ( self.X + (2 * 57) ) ) then
                          self.selection := 2;
                    else if ( theX < ( self.X + (3 * 57) ) ) then
                          self.selection := 3;
                    else if ( theX < ( self.X + (4 * 57) ) ) then
                          self.selection := 4;
                    else if ( theX < ( self.X + (5 * 57) ) ) then
                          self.selection := 5;
                    else if ( theX < ( self.X + (6 * 57) ) ) then
                          self.selection := 6;
                    else
                          self.selection := 7;
                    end;

                    ACheck1.MoveTo( ( self.X + ( self.selection * 57 ) -
42 ) , self.Y + 5 );
                    ACheck1.Show(true);
              end;

with
      selection is 0;
      Target is PICTURECLIENT with Supplier is Mrow; Duration is 0; end;
      Duration is -1;
      X is 110; Y is 161; Width is 395; Height is 40;
      HotRegion is {
            $00020000, $0025000E, $0007002D, $00400066, $0079009F,
$00B100D8, $00EA0111, $0123014A,$015C0184, $00280000, $FFFFFFFF
      };
end;
```

18. Appendix B: Additions to AMT

18.1 New MovieClient Classes

```
class FINISH_MOVIECLIENT
is
      MOVIECLIENT;
has
      Finished()
            do
                  self.Container.Finished();
            end;
end;

class WAIT_MOVIECLIENT
is
      MOVIECLIENT;
has
      Run(runIt)
            do
                  self.MOVIECLIENT:Run(runIt);
                  from loop         -- Wait until finished running...
                        self.Idle(1);
                  while self.IsRunning();
            end;
end;
```

18.2 StoreString

```
-- Class specification for saving data

class STORE_STRING is MEDIADELEGATOR;
has

      StoreText(theString,theFile)  -- Store a string

            theString is? STRING;
            theFile is? STRING;

            use
                  error;
            do
                  -- Save it into the given filename
```

```
                error := self.wrapper_do_store(theString,theFile);

                if error = -1 then
                      -- file I/O screwed up
                      Application.Beep();
                end;
          end;

      wrapper_do_store(aString,bString) -- wrapper method
            external "do_store";
end;

#include "key.h"
#include <stdio.h>

/* this function stores a string into a designated data file in the */
/* current directory                                                */

void do_store(key *the)
{
      FILE *myfile;

      /* open for append or create a new file */
      myfile = fopen(keyToString(the[ARGUMENT(2)]), "a");

      /* if NULL, then I/O error */
      if (myfile == NULL)
            the[RESULT] = keyFromInteger(-1);
      else
            /* we opened the file, now write to it */
            /* this overwrites the existing file */
            fprintf(myfile, "%s", keyToString(the[ARGUMENT(1)]));

      /* close the file */
      fclose(myfile);
}
```

18.3 Launch

```
-- Class specification to enable conference launch

class LAUNCH is MEDIADELEGATOR;
has
      DoLaunchConference(aVal)
      use
            error;
      do
            error := self.wrapper_do_launch(aVal);
            if error = -1 then
                  -- file I/O screwed up
                  Application.Beep();
            end;
      end;

      wrapper_do_launch(aVal) -- the wrapper method for "do_startvid"
            external "do_launch";
end;

#include "key.h"
#include <stdio.h>
#include <Files.h>
#include <Processes.h>

/* this function launches application "launch"        */

void do_launch(key *the)
{
      LaunchParamBlockRec     LaunchParams;
      FSSpec                  mySpec;

      keyIfError(FSMakeFSSpec(0, 0, "\plaunch", &mySpec));
      LaunchParams.launchBlockID = extendedBlock;
      LaunchParams.launchEPBLength = extendedBlockLen;
      LaunchParams.launchFileFlags = 0;
      LaunchParams.launchControlFlags = launchContinue +
launchInhibitDaemon;
      LaunchParams.launchAppSpec = &mySpec;
      LaunchParams.launchAppParameters = nil;
      keyIfError(LaunchApplication(&LaunchParams));
}
```

19. Bibliography

[1] Urban G. et al. "Validation and Lessons from the Field -- Applications of Information Acceleration." (August 1995)

[2] Urban G., Weinberg B. and Hauser J. "Premarket Forecasting of Really-New Products." *Journal of Marketing*. 60. pp. 47-60. (January 1996)

[3] Hicks R. and Essinger J. Making computers more human: Designing for human-computer interaction. Elsevier Science Publishers. (1991)

[4] Durrett H.J. and Trezona J. "How to use color displays effectively." Byte 7(4). pp. 50-53. (April 1982)

[5] Murch G.M. "Physiological Principles for the Effective Use of Color." *IEEE Computer Graphics and Applications* 4(11). pp. 49-55. (1984)

[6] Marcus A. "Color: A Tool for Computer Graphics Communication." in *The Computer Image*, Greenburg D. et al. Addison-Wesley. (1982)

[7] Shorrock B. System Design and Human-Computer Interaction -- A Practical Handbook. Sigma Press (1988)

[8] Bohlmann J. Disconfirmed Expectations and Group Decision Behavior. MIT Ph.D. Thesis (1996)

[9] Ellis C. et al, "Design and Use of A Group Editor" in *Engineering for Human-Computer Interaction*, Cockton G. Elsevier Science Publishers. (1990)

[10] Gallagher J. and Krant R.E. "Computer-Mediated communication for Intellectual Teamwork: A Field Experiment in Group Writing." in *Proceedings of CSCW '90.* pp. 65-78. (October 1990)

[11] Seligmann D, Mercuri R., and Edmark J. "Providing Assurances in a Multimedia Interactive Environment." in *Human Factors in Computing Systems CHI '95 Conference Proceedings*. (May 1995)

[12] Dailey C. A Design for a Multimedia Server using QuickTime. MIT EECS M.Eng Thesis (1995)

[13] Abramson H. and Rogers M.H. Meta-Programming in Logic Programming. The MIT Press (1989)