# Using Multimedia in Design Instruction

by

Sepehr Kiani

B.S. California Polytechnic State University
1991

Submitted to the Department of
Mechanical Engineering in Partial Fulfillment of
the Requirements for the Degree of

Master of Science in Mechanical Engineering

at the Massachusetts Institute of Technology
June 1994



Signature of Author _____
Department of Mechanical Engineering
June 3, 1994

Certified by _____
David Gordon Wilson
Professor, Mechanical Engineering
Thesis Supervisor

Accepted by _____
Ain A. Sonin
Chairman, Graduate Committee

# ABSTRACT

The design and development of a multimedia education program called EDICS (Engineering Design Instructional Computer System) is discussed in this thesis. The system is an upgrade from earlier work intended to teach undergraduates the basics of mechanical design elements. A survey of multimedia technology is first given, including digital video, digital audio, authoring environments, networking, and storage issues. The software design is presented, including a discussion of the environment chosen. The Microsoft Visual Basic program uses an overall database-centered approach. EDICS is used to create a tutorial chapter on the process of design by presenting a case study on the design of a human-powered hydrofoil. By conducting a study on this chapter of EDICS, using MIT undergraduate mechanical engineering students as subjects, both overall effectiveness and specific design are evaluated. The results of the study indicate that the students enjoyed the multimedia approach to learning, but that the tutorial still needs work in specific areas and that not all the intend subject matter was successfully conveyed. Recommendations as to the future of EDICS are made. The overall conclusion of this work is that multimedia has great potential in conveying large amounts of diverse information, but that both the technology and approach are still in their infancy.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ACKNOWLEDGMENTS

A project of this magnitude could never be done without a large team of skilled people, as did this one. They provided the diversity of experience required to make this project a success.

First I would like to thank professor David Gordon Wilson for his guidance and encouragement as principal investigator for EDICS and my faculty advisor. He gave me the freedom to try new and different approaches to problems; giving just enough rope to hang myself, but never letting me hang from it. Co-principal investigator Ernesto Blanco brought heart and character to the project, and always many words of wisdom.

The large number of undergraduate students made this project a learning process from all sides. Their hard work and enthusiasm carried me through the low points. Each of them brought their own abilities and experiences too many for me to mention here: Acee Agoyo, Fred Ackerman, Dalia Ali, Sumit Basu, Chris Berg, Bryson Gardner, Dennis Burianek, Joe DeMare, Richard Domonkos, Ryan Ehlert, LaTaunynia Flemings, Sherondalyn Johnson, Rob Kim, Jason (Scrapy) Mueller, Bruce Padmore, Chang Suh, and Yannick Trottier.

Building the Skeeter was probably the largest part of this project and never would have been possible without professor Mark Drela and Ph.D. student Matthew Wall volunteering enormous amounts of time on the design and construction of it. They enjoy a large portion of the credit for its success. I also want to give special thanks to Norm Berube for all his help with last minute machining and the guidance he gave to Fred Ackerman (an aspiring undergraduate machinist and designer). There were many other people who volunteered on the hydrofoil project in large and small ways: Julie Yang, Li Shu, Ben Lindes, Ian Kaye, Tom Washington, Harold (Guppy) Youngren, professor Woodie Flowers, Marc Schafer, and Shin Choi. I would also like to thank Ted VanDusen and Composite Engineering for letting us use his facilities and kayak molds.

The Center for Advanced Engineering Studies (CAES) needs mention here for its initial sponsorship and later allowing us to finish work in their facilities. I would like to specifically thank Cecil Feaver for his good humor and support. Without him we never could have kept all the computers running.

Finally, my friends and family who kept me going with their constant emotional and moral support. Special thanks to Bill Baker and Sami Farhad for their help with editing this thesis, to Hood Gaemmaghmin for his crazy food runs to China Town at three in the morning, and finally to Sasha Bashirelahi for her love and cheer throughout my busiest times.

# CHAPTER 1. INTRODUCTION

## 1.1. HISTORY OF EDICS

The Engineering Design Instructional Computer System (EDICS) was initially funded in 1984 by MIT's Project Athena with Professor David Gordon Wilson as principal investigator and Seichi Tsutsumi as co-principal investigator. Later Professor Ernesto Blanco as co-principal investigator and Professor Woodie Flowers as a consultant joined the project. Initial work was done on a DEC IVIS (Interactive Video Instructional System). This system was not very practical and the project eventually fell into limbo.

In 1988 EDICS was reinvigorated with funding from the National Science Foundation, with Wilson, Blanco, Tsutsumi and Flowers continuing their roles. Graduate student Douglas Marsden was hired to develop three chapters on bearings, attaching rotors to shafts, and connecting and capping cylinders [Marsden 90]. Each chapter represents a complete tutorial section of EDICS. The platform was changed to an Apple Macintosh running HyperCard and controlling a separate video disk and monitor.

In 1990 David Crismond replaced Marsden as graduate research assistant concentrating his efforts on converting the software from version 1 to version 2 of HyperCard (EDICS v2) and conducting an extensive evaluation of the program's effectiveness [Crismond 92].

In 1992 Sepehr Kiani replaced Crismond, funding having shifted to the ECSEL (Engineering Coalition of Schools for Excellence in Education and Leadership) program, also under the National Science Foundation. Work went forward on the Macintosh adding a drafting and sketching section, and debugging the code so that it could be sold under an agreement with Intellimation. However, in the Fall of 1992 funding of the project was taken over internally by MIT's Center for Advanced Engineering Studies (CAES), which, hoping to include EDICS with their video sales, required that the agreement with

Intellimation be terminated. Market forces pushed for the program to run on the IBM PC platform also, and for delivery to be on CD-ROM using digital video instead of a separate video disk player. Work began in summer of 1993 developing EDICS v3 using Visual Basic running under Microsoft Windows. The bearings chapter was converted, and a new chapter on the design process was added, incorporating a case study of the design of a human-powered hydrofoil.

## 1.2. OLD EDICS REVIEWED

EDICS was designed to provide users with some practical experience in mechanical design, using video of real devices. Aimed at students who may not have mechanical design experience it covered topics, such as bearing selection and shaft/rotor connections, that students tend to have difficulty with.

The version of EDICS prior to this thesis was running on an Apple Macintosh computer requiring a minimum of 5MB of RAM and 40MB hard disk space, monochrome or color monitor, and a laser-disk player with a separate monitor. The authoring environment was HyperCard using third-party software to control the laser-disk player and to play animations.

Each subject, except drafting, has the following sub-sections: introduction, physics, specification, types, good practices, examples, selection system, and self-test. The treatment relies heavily on real hardware examples, often asking the user to guess quantities like the speed of a dentist drill. The user navigates through the system using a menu structure and a sequenced page metaphor. A map can be accessed to help guide the user though the program.

EDICS has been put in the workshop (right next to the cut-off saw) for a short course offered at MIT called "Taking Things Apart." In this course, students disassemble machine components, and describe them using EDICS as a resource. The students are also required to go linearly through separate subjects in EDICS. Copies of EDICS have been given to

twelve other universities and a couple of high schools for evaluation. While feedback has been minimal, it has been positive.

A study of EDICS was done by Crismond to compare its effectiveness to written text. He found that "[s]tudents who used the computer version of EDICS had significantly higher scores than the text versions in the following areas: analogical reasoning; estimation; and certain vocabulary measures, location-task skills, and a timed-list-generation or brainstorming task." [Crismond 92].

## 1.3. RELATED WORK

There are many other efforts around the world to implement computer-aided instructional systems (CAI). Most are more modest efforts specifically designed for a particular class or lab. Many of them concentrate more on the simulation side than on informational media.

Rochester Institute of Technology department of electrical engineering has been using a HyperCard tutorial in their third-level-design laboratory. The laboratory course requires students to design and build electronics circuits. The RIT faculty found that the current student pool had widely varied experience and that they needed to provide less-experienced students with a head start. To do this they developed a tutorial using the computer to teach the design process in conjunction with separate video tapes [Spina & Mukund 93].

A tutorial on the First Law of Thermodynamics for closed systems has been developed at the University of Louisville department of mechanical engineering. It is the first part of an eight-module series designed to reduce the high attrition rate for the thermodynamics course. Using Asymetrix's Toolbook authoring system on a Windows-based system, the tutorial is both projected in class using an LCD projection panel and out of class by individual students. The program uses hypertext and animation [Coburn 92].

At Rensselaer Polytechnic Institute a system called the Flexible Education Module (FEM) has been developed for a lab course on embedding microprocessor controllers. It is a HyperCard program running on Macintosh Quadras using EyeQ DVI (Digital Video Interleave) cards set up at each laboratory workstation. A local hard disk stores frequently used video files while a large server stores less-used video files. The information, organized in a tree structure, can be accessed through a glossary/index or by topic using a menu structure. FEM also provides links to commercial software on the system, allowing students to launch data-analysis software, word processors, etc. A controller card in the Macintosh is used in parts of the experiments. Initial tests of the program showed that students wouldn't use it until it was well integrated with the laboratory experience, i.e., until a computer is sitting on the lab bench running analysis software. Student response to FEM has been positive for the most part. Overall use is slightly lower than was hoped; however, when the students used the program they benefited from the experience [Borkowski 92].

Iowa State University Engineering College has been developing a system called the Studio for Exploiting Technology to Teach Technology (Setforth). The program uses a Macintosh-based system and Authorware Professional as an authoring environment. They conducted experiments exploiting the cross-platform capabilities of Authorware to port course material to IBM PCs. Courseware for two computer engineering courses were developed for both projection in class and student tutorial use. Duplicates of lecture screens are available to students in hard-copy and on-line. Old exams and problem-set solutions are on the system for student access. Student rating of the system suggests that the most important aspect of the program was having old exams and problem-set solutions on the computer [Smay & Genalo 92].

In the Laboratoire EPC in France a five-module tutorial on power electronics has recently been developed. It is a Windows-based system using Icon Author as an

environment. The tutorial uses digital video and color graphics to present the topic of fly-back converters [Jaafari, Picard & Bessège 93].

## 1.4. THESIS SUMMARY

The eight chapters of this thesis follow the basic chronology of the work. Chapter two gives an overview of the current state of multimedia looking at all the elements and where they may go in the near future. The third chapter begins the discussion of the design of the new EDICS. The entire database structure and approaches to data entry are given here. The fourth chapter is an overview of the design of the EDICS program. The EDICS-chapter on the presentation of the design process is discussed in chapter five. The case study for this EDICS-chapter is the design and construction of a human-powered hydrofoil. The next chapter discusses the findings of an evaluative study on the effectiveness of the EDICS design-process chapter. Ten MIT undergraduate mechanical-engineering students give their feedback. Recommendations are made in chapter eight as to the future direction of EDICS. The program design outline is giving in appendix A and the actual program code in appendix B. The database structures are given in appendix C. Appendix D has the complete outline of the EDICS design-process chapter.

# CHAPTER 2. MULTIMEDIA HARDWARE AND SOFTWARE

## 2.1. THE CURRENT (1994) STATE OF MULTIMEDIA

For more than a decade various forms of multimedia have been available on computer systems, in the form of graphics and text controlled by the computer. Currently available are digital audio, digital video, and full-color graphics, all running on powerful, inexpensive personal computers that have large memory capacity.

### 2.1.1. Digital Audio

In the last ten years, with the introduction of the compact disk, digital audio has almost completely taken over the recording industry. Digital audio is now available on all major computer systems either on the motherboard or as a low-cost addition. Compression and decompression, which significantly reduces storage requirements, are easily within the capability of modern computers.

The future will bring hardware prices down further and standardize computer motherboards to include sound chips. Digital signal processors (DSPs) are becoming widespread. They can do general digital-to-analog and analog-to-digital processing of audio, modem, analog video capture and fax-type functions on one chip. Combined audio and video file structure and compression standards are currently evolving (as discussed below).

### 2.1.2. Digital Video

The video industry is following the footsteps of the audio world. The video laser-disk player uses the same storage technology as the CD but stores both digital video and audio. These systems convert the digital information on the video disk to analog output that is then played on television monitors. This is still one step away from having video on the

computer screen. The video production industry has been flirting with the prospect of full digital recording for a few years now. The industry foresees video being transformed, in the same way that desktop publishing changed the printing industry, to "desktop video" [Cornell 92]. The main limitation is the large storage capacity required for this application; fortunately data-compression techniques are under continual development and higher-density storage is on the horizon.

Video and audio compression methods are similar, in that they condense information. Video compression is accomplished by storing only changes between video frames (temporal compression) or by compressing the data in each frame separately. The compression ratio can be increased by sacrificing image quality and making moving images "blocky" along their edges. The video industry has set its first standard for digital video codec (compressor/decompressor) called MPEG (Motion Picture Experts Group) and is working on a new scaleable international standard called MPEG II slated for 1995. Meanwhile the computer industry, impatient for standards, is waging a battle for the multimedia standard, with chief players being Apple Video, Microsoft Video, SuperMac's Cinepak, and Intel's Indeo.

There are two approaches to digital video codec. Software-based codec simply uses the main processor to run compression and decompression algorithms. Hardware-based compression uses specially designed processors to compress and decompress video images. This frees the main processor for other tasks.

Hardware-based codec performance is very good, allowing video resolution of 640x480 pixels playing at frame rates up to 30 frames per second. These specialized processors can be programmed to run any current or future codec algorithms. Most compression chip-sets are manufactured by C-Cube and Intel. Cost is the big drawback of dedicated video compression. Low-cost playback (decompression only) cards, however, have recently been coming into the market.

Because general computer processors are beginning to have the speed to perform video compression, software developers are starting to produce software-based video codecs. The compression algorithms are the same as those used in the hardware-based systems: the only difference is that the main CPU handles these algorithms. These software-based systems that run on PCs currently include Apple QuickTime on Macintosh and Windows, and Microsoft's Video for Windows. Also MPEG players for workstations and PCs are now available. MPEG is much more computationally intensive and requires high-performance systems to playback at a reasonable rate. Playback that is approximately equivalent to television is 640x480 pixels by 30 frames per second. The current crop of high-end PCs can decompress at about half that rate. Most developers are currently using the Cinepak codec to put 320x240 pixel at 15 frames per second on CD-ROM, a reasonable compromise.

As a general rule non-dedicated hardware will win out in the long run, and this will be the case with digital video. Full-rate video playback on standard PCs will be software-based within the next few years, gradually reducing the video-playback hardware market. The other main trend is with video compression, which is 50-to-100-times more computationally intensive than decompression. This means that a 50 to 100 times increase in CPU speed is required for a standard PC to process video in real time. This sort of performance increase is five to seven years away. Therefore production-level digital video recorders will depend on hardware-based systems in the years to come. The prices of these systems, however, will drop dramatically.

### 2.1.3. Storage

Optical data storage has come into the low-price arena due to the explosion of CD sales in the music industry. The 650MB compact disk (CD-ROM) at first seemed to be an enormous amount of storage. One of the first available encyclopedias claimed to take only 20% of one disk; but, it did so mostly because it contained only very compact text, with

little or no multimedia components. With the addition of 8-bit color (i.e., 256 colors), and later 24-bit color (i.e., 16.4 million colors) the storage situation becomes more difficult. The addition of digital video requires even greater storage capacity.

Using storage media that were designed for audio as computer-based storage, however, creates a whole series of problems, the largest one being speed. Slow-spinning CDs have data rates of 150 kB/s and access times of 1000ms. Newer CD-ROM drives have doubled and even quadrupled that performance, boosting data rates to a level high enough for full video playback. Access times still remain very slow (100 to 300ms) when compared to hard-disk storage (10 to 20ms).

A recently available technology that solves the speed problem and allows for the writing of data to the disk is the magneto-optical disk. With storage densities similar to those of the CD-ROM, these systems use magnetic fields to distort locally the optical surface of the disk. The speeds of these systems are much closer to those of hard-disk storage. Magneto-optical drives are, for now, too expensive to have the mass appeal of the CD-ROM.

The work being done in laboratories is encouraging. Super-high-density magnetic-storage and optical-storage systems are being developed [Bell 93]. The density of data on a magnetic medium is dependent on how close the read/write head is to the disk. The performance of these systems is being improved by using disk media such as glass and liquid films instead of air to reduce the read/write gap—thereby boosting an already mature technology. Work also continues in solid-state storage systems; industry coalitions have been formed to develop flash memory that stores information in chips without needing continuous-connection to electrical power.

For now (approximately the next 5 years) the CD-ROM will be the main delivery medium for multimedia. Within ten years typical storage capacities will increase a hundred to a thousand fold.

### 2.1.4. Networking

The technology for high-capacity optical networks currently exists and is being used in some limited ways. The bandwidths of optical networks are such that multiple digital video and audio signals can be sent on one fiber-optic line. There are major forces creating large-scale, high-capacity networks internationally and in the US. These proponents include phone companies, the television industry, the federal government, and the computer industry. Development on this scale will be slow; large-scale networking will likely take five to seven years to fully implement. High-speed local networks, however, capable of transferring the quantity of information required for multimedia, are currently in use. Loeb warns, however, that even in the long term these networks may not allow for arbitrarily complex multimedia applications [Loeb 92]. Lower capacity networks, i.e., most of the current Internet system, is connecting academia and many corporations.

### 2.1.5. Personal Computers

Demand for the personal computer is increasing, pushing the prices of high-performance computers down. The current crop of low-cost desktop computers have the power that workstations had only a few years ago (see Figure 2.1). Manufacturers, now poised to attack the consumer audio/video market in the next few years, envision consumers buying computers that will also serve as home entertainment and education systems. Computer processors will need to make a significant increase in power per dollar for this to be possible.

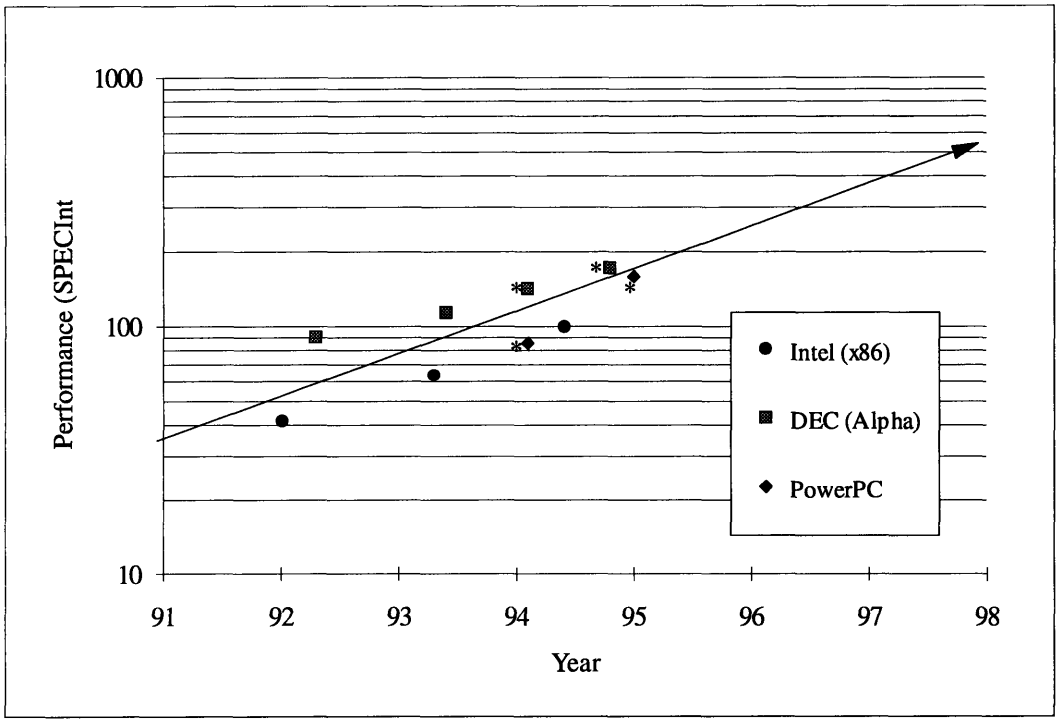



FIGURE 2.1: POWER VERSE TIME FOR MICROPROCESSORS.
(Data from manufactures)
* estimated

2.1.6. Authoring Systems

Authoring systems are the word processors of multimedia, allowing multimedia and hypertext "documents" to be written and edited. They are essentially high-level programming languages; Figure 2.2 gives a general comparison. The purpose of multimedia is much less defined than desktop publishing was when it got started. HyperCard, a program developed by Apple, was the first to lay down tracks in this area. It allowed for interactive text and graphics as well as powerful programming features. The current crop of development tools resembles anarchic mayhem, with dozens of players fighting for supremacy; but no one knows the exact needs of the users. Key terms advertised include "cross-platform compatibility," "hypertext," and "QuickTime or digital video support." When the dust settles there will likely be a few big players left standing.

Much of the battle will ensue when Microsoft, the Apple and IBM alliance, and Macro Media release rumored products.

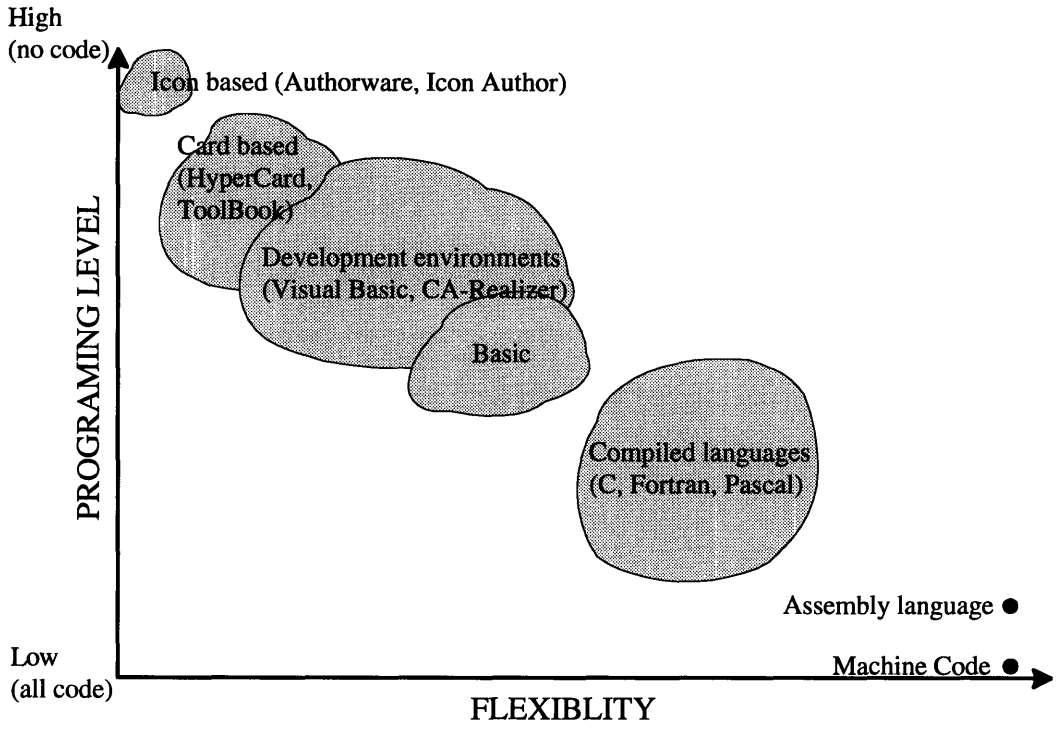



FIGURE 2.2: COMPARISON OF PROGRAMMING LANGUAGES AND AUTHORING ENVIRONMENTS

No one program or system will win out; there will be numerous applications running on different types of computers all trying to share information. One effort on the Internet is Mosaic, developed by the University of Illinois, which is multi-platform front-end to Internet information. Another group working to set a standard for multimedia objects is The Multimedia and Hypermedia information coding Expert Group (MHEG) [Kretz & Colaïtis 92].

### 2.1.7. Modeling and Animation

Graphics tools that were formerly available only to the Hollywood special-effect guru now run on desktop computers producing what is termed "virtual reality," i.e., photo-

realistic images and animations. These programs allow the user to create objects in space and assign surface qualities to them. Objects can be given paths and rotations to follow. Lights and cameras can be positioned anywhere in space. The software then creates the image by mapping the surfaces and calculating the cast shadows, yielding individual frames of an animated movie.

This new modeling software, combined with available two-dimensional animation packages, is rapidly evolving due to competition, giving authors easier-to-use, lower-cost, and more-integrated tools. As hardware further improves in the next decade, real-time complex three-dimensional rendering will become commonplace, adding a whole new facet to multimedia.

## 2.2. SUMMARY

We don't have to go back very far to see that the computer industry has moved along at an amazing pace. Computer capabilities that were unimaginable just a decade ago are now at our disposal. This increase in power has created multimedia, which promises to revolutionize the information world. The number of actors in this field goes far beyond the number involved in the computer industry, enlisting the giants of home entertainment and the professional media. The challenge that lies ahead for the engineering world is to determine what exactly we should do with all these new toys.

## 2.3. HARDWARE USED

### 2.3.1. Windows PC

The majority of the development of EDICS was done on an Intel 486DX2-66 VESA local bus computer. It had 64MB of RAM and 500MB of hard-disk space. The graphics board was a ATI MACH32 with 2MB of DRAM driving a 21-inch color monitor allowing for 24-bit color (16.4-million colors). Sound was handled by a 16-bit sound card that also

allowed for sound recording. Three other PCs were also used: two 486DX-50 machines and an IBM PS/2 Ultimedia M77 (486DX2-66). All machines were networked together to a server with a 750MB hard disk.

### 2.3.2. Apple Macintosh

We used two 900-series Macintosh Quadras, both with 13-inch color monitors. The Macintoshes served as temporary servers and for image editing. Also image scanning was all done on the Macintosh.

### 2.3.3. Video capture

Video capture was done using the capture utility that is part of Video for Windows and the Intel Smart Video Recorder card. The card has built in Indeo compression technology and allows for up to 320x240 pixels at 15 frames per second. It can also capture individual frames at 640x480 resolution.

### 2.3.4. Picture and slide scanning

We had the good fortune of having access to a Canon Color Laser Copier 500 with a Fiery computer interface and a separate slide-scanning unit. The system was capable of scanning 8 1/2" x 11" paper at 400dpi in full color.

## 2.4. SOFTWARE USED

### 2.4.1. AutoCAD

AutoCAD is a computer-aided design (CAD) program that does 2D and 3D drawing along with solid modeling. Initial design work of the hydrofoil was done using the DOS version of AutoCAD 11 that was later upgraded to release 12. The Windows version of release 12 was used to transfer drawings from AutoCAD to Photoshop to be added to the archive. The UNIX version, running on MIT's Athena system, was also used by undergraduate research students to update drawings.

### 2.4.2. 3DStudio

3DStudio is a photo-realistic rendering and animation package. Three-dimensional objects can be created directly or imported from AutoCAD. The workspace is set up like a filming studio where spot-lights, ambient-light and point-lights can be added anywhere in space. Materials are selected from a library and mapped onto the object for rendering. The object is viewed from a camera that can be placed anywhere in space and has any focal

length desired. The program can make animations using a key-frame system where objects, cameras, and/or lights are moved or rotated to different positions and the computer fills in the in-between frames. Frames are then rendered one-at-a-time for play back.

### 2.4.3. Photoshop

We used Photoshop on both the Macintosh and the PC for scanning and editing images. It has a powerful set of tools and filters that can be used to enhance images. It also includes a filter for de-interlacing video, which is useful for images picked from video. Photoshop was also used for making all the buttons and backgrounds for the program.

### 2.4.4. Adobe Premiere

Adobe Premiere is a digital video editing package. It allows for the splicing of video files with image files. A library of effect transitions provide transitions like fades. Up to three audio tracks can be played on top of each other. It also has a library of filters for special effects, cropping and touching up video. We mainly used Premiere for linking video sequences and placing a voice track on top.

### 2.4.5. Visio

A very easy-to-use program for making simple 2D vector drawings is Visio. Vector graphics are based on (and saved as) graphical primitives, i.e., points, lines, circles, arcs and fills, as opposed to bit-maps that are rasterized graphics. Visio also can import and export many different types of bit-map and vector files. This was one of the main features we used. It was also used for some text operations and simple diagrams.

### 2.4.6. Video for Windows

Video for Windows allows the Windows media player to play digital videos. It also includes applications for capturing and editing video and for capturing audio. Version 1.1 can use several different codecs, including Indeo and Cinepak. The video-capture program was used in conjunction with the Intel capture board for digitizing the video and capturing frames.

# CHAPTER 3. EDICS DATABASES

## 3.1. MULTIMEDIA DATABASES

Databases are systems that store and retrieve information. In a media-rich environment a database can include text, images, videos, animation, and sound. It is even possible to treat objects from other programs, such as finite-element models or spreadsheets, as media records. The idea is that information is not just text-based but can be of any form.

Different database tables can be linked using shared entries. This is called a relational database, and is a very efficient way to store information. The database is manipulated by means of queries—linking tables, performing calculations, or sorting records. Say you have tables 1 and 2 shown in Figure 3.1. Table 1 has fields A and B; table 2 has fields B and C. This would be useful if different values of A can be related to one value of C. A query function uses the B field as a relating code to create a virtual table combining A, B, and C. The savings become apparent, for example, if C contains large bit-map images, duplication is avoided. It also has advantages in editing—one change to C changes it in all instances.

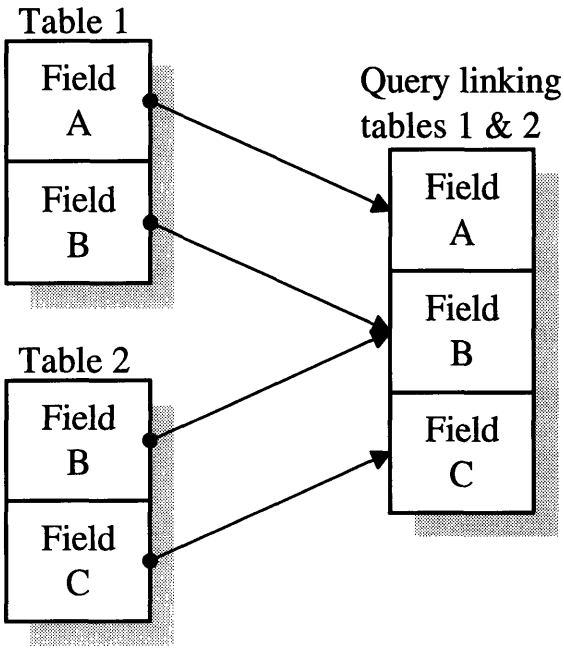



FIGURE 3.1: QUERY LINKING EXAMPLE

Query tools are for now just text-based devices that have carried over from text-based systems. New multimedia database tools are going to eventually develop that are similar to the one IBM is developing: a database query tool that can do searches on shapes, colors or textures that appear in images and videos [Brown 94]. Such a tool would allow you to search for images based on a simple sketch you enter.

## 3.2. THE EDICS ARCHIVE

The first step I took in preparing for the development of the new EDICS was to create an archive database to keep track of the various media objects, (see Figure 3.2). The database was initially created in Borland's Paradox for Windows and later moved to Microsoft's Access. The "archive code" field allows for linking to EDICS pages using coding system given in Table 3.1. For example S100 refers to the bit-map file S100.TGA, and V200 refers to the video file V200.AVI. The "Description" and "Archive source" fields provide the information required to search through the database and to locate originals (i.e., slides, drawings, videos, etc.). The actual archive field is an OLE (object

linking and embedding) object accepting anything in the Windows clipboard and retaining information about what program it was clipped from. For example, if a video is copied and clipped in to this field the current frame of the video is displayed in the window; then if double clicked it will play the video using the same media-player program that it originally came from. The other fields will be discussed later. The EDICS database currently contains some 620 still images, 270 videos, 119 audio files, and 15 vector images that can all be searched for and/or sorted.

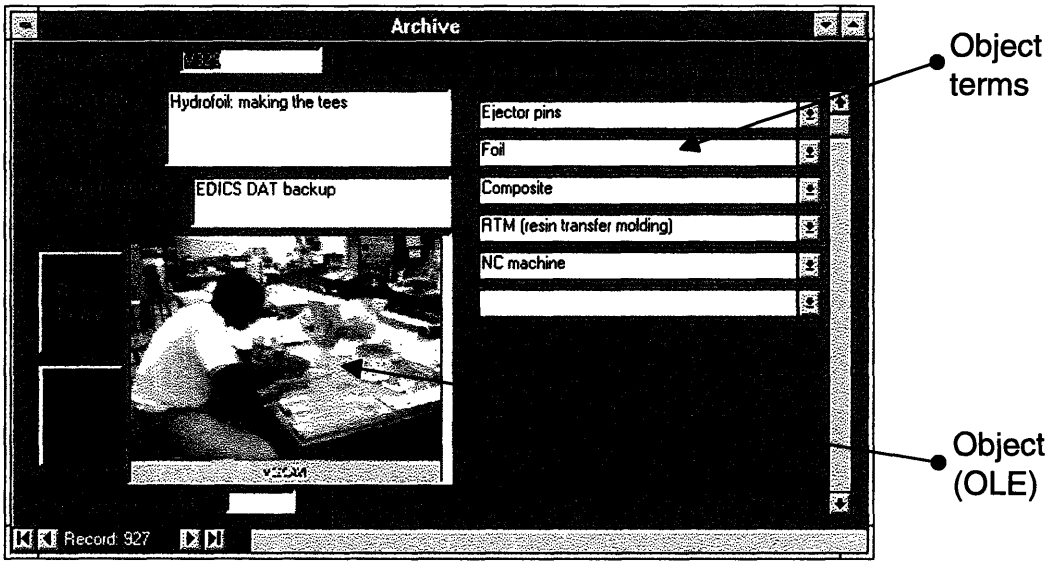



FIGURE 3.2: EDICS ARCHIVE TABLE FORM

TABLE 3.1: ARCHIVE CODES

| CODE | OBJECT TYPE | FILE TYPE |
|---|---|---|
| S | Still images files—bit-maps | targa (.tga) |
| A | Audio files | wave (.wav) |
| V | Video and animation files | AVI (.avi) |
| M | Vector graphics files | metafile (.wmf) |
| T | Text | Access database table (.mdb) |

The EDICS database tables provide the instructions (code) that the EDICS player uses to run the tutorial. The key advantage is that the same SQL (Standard Query Language) database and media files are platform-independent. Therefore players can be written in any language on any system, resulting in the same multimedia program for all platforms without having to redo the content work. A few commercially available authoring systems provide either a translator or player for cross-platform operation. The main problem with these systems is that upgrade and compatibility is completely under the control of the vendor. If for example we later decide to use a player written in a different language, or if Visual Basic languishes, we would have all of our standard databases and none of the content work would be lost.

Figure 3.3 shows the database structure of EDICS. At the top level is a tutorial table. Each chapter in EDICS is a tutorial, so bearing design, spring design and design process chapters each have their own record in this table. The page table is keyed to the tutorial table where each record is an object. The different types of objects and their codes are listed in Table 3.1. Each record in the Page table is sorted first by Tutorial codes and then by page codes. Objects with the same page codes are placed on the screen in the order of the sequence field and placed in the location indicated. We have provided for several transition types listed in Table 3.2. New types of transitions are easily accommodated by adding new codes.

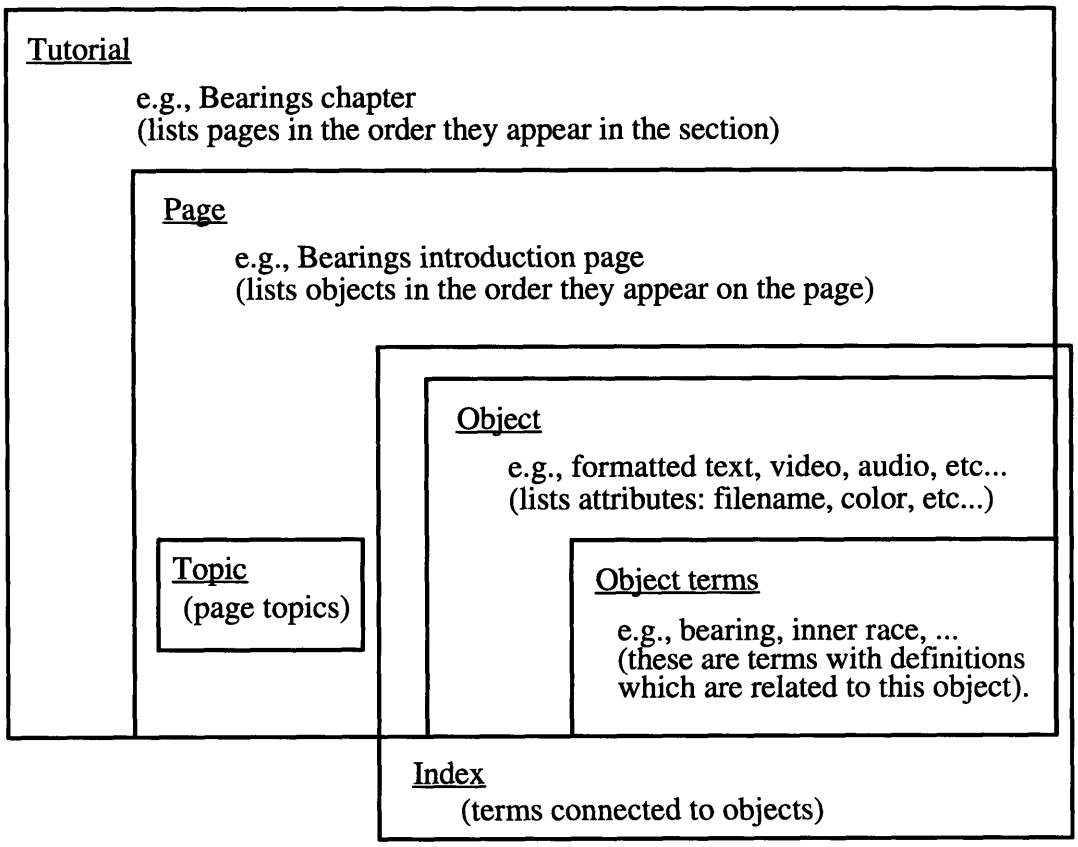



FIGURE 3.3: EDICS DATABASE STRUCTURE

TABLE 3.2: TRANSITION TYPES AND CODES

| Transition Code | Transition Description |
|---|---|
| 0 | none |
| M | move |
| V | move and drop |
| D | drop |
| W | wait |
| F{font specifications} | font |
| L | loop video |

The archive table is used only for development purposes and is not needed for delivery. All the objects are in the form of files (see Table 3.1) except for text codes that refer to a separate text table; this is more efficient than having separate text files. Finally

an object code that starts with an "L" refers to a title (the text for which follows the "L", e.g., "LDesign Review"). The transition code field is used to set font and style of the title.

Tied to every object in the archive is a series of object terms from the object-term table. These terms are defined in the Index table. Whenever that object is on the screen, all its object terms are listed in the list box at the bottom of the screen. For example if there is a term used in a video that a student may need defined, that term will be attached to that video object and the user can click on the list to bring up the definition. Also, terms connect to text objects will be highlighted in the text and can be clicked on directly (this feature is not yet fully functional). The index can have all available object types linked to it for display.

The index-set table allows groups of pages to be linked to form a sequence. Any object (except audio files) can be connected to an index-set, functioning like a button, that when clicked-on branches to that index-set. This feature has a second intended function: of allowing anyone to take a series of pages or objects and stitching them together into a sequence. The idea is that an instructor can go into a class with a pre-selected series of pages and objects to present in class or have students go through them for an assignment.

## 3.3. DATA ENTRY: BUILDING SCREENS

Eight tables make up the main EDICS database along with image, audio and video files in total making up the entire content of EDICS. The tables are designed for efficient information storage and retrieval. By themselves, however, they do not clearly present content information and screen layout. Editing a table directly does, however, allow for making mass changes. For example to change a particular image that appears on multiple screens would be laborious without direct editing.

Laying out each screen would be a daunting task without some sort of help from the system. It requires a full knowledge of all the different types of transition codes, scale of images, coordinate location (x, y) of all objects, and link code information. To simplify

this, two front-ends to the database were created. They represent two different levels of access to the database.

### 3.3.1. Queried Form

The first-level database front-end uses the query and form tools that are part of the Access database program. Querying all the tables together in one "Screen editor" form, shown in Figure 3.4, the data are somewhat more recognizable than in a raw table of codes. The query automatically takes care of the relational links between tables, whereas direct table-editing requires manual linking. The page-form works on a per object basis (an object is any image, audio, video, text, etc.). It does not, however, show what the screen will look like. This level of data entry was useful because development of the database and the next higher level front-end needed to occur simultaneously. It also allows quick access to the information and at the same time provides visual cues from the archive. The built-in text searching feature furnishes an efficient way to locate information in large tables.

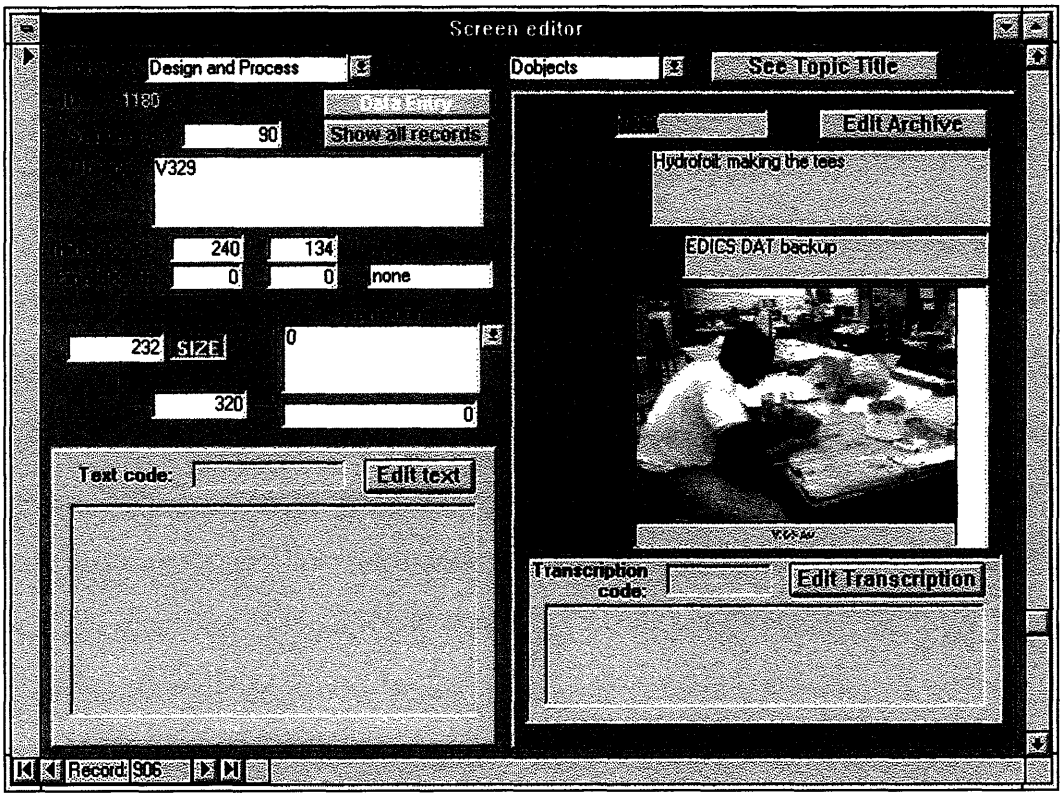



FIGURE 3.4: SCREEN EDITOR FORM

### 3.3.2. Screen-Arranger Application

The screen-arranger application, shown in Figure 3.5, is a much more elaborate front-end to the EDICS database. It is written in Visual Basic, as is the rest of EDICS. Visual Basic's built-in SQL calls are used to manipulate the eight tables that make up the EDICS content. Screen layout and behavior are the same as the EDICS player (that actually uses the same Visual Basic form), except branching does not work.

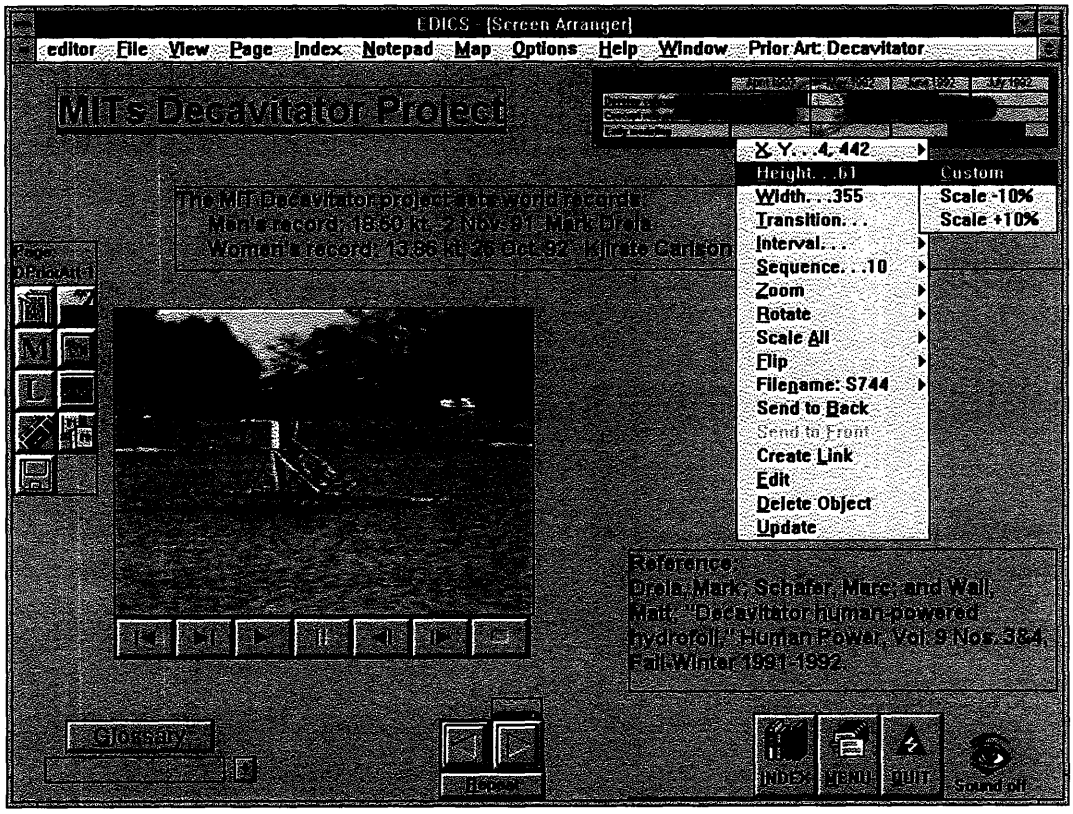



FIGURE 3.5: SCREEN ARRANGER

A floating object pallet, see Figure 3.6, provides access to all types of available objects. Clicking with the left mouse button on any icon creates an empty object box of the desired type. The objects can then be dragged to any part of the screen with the left mouse button.

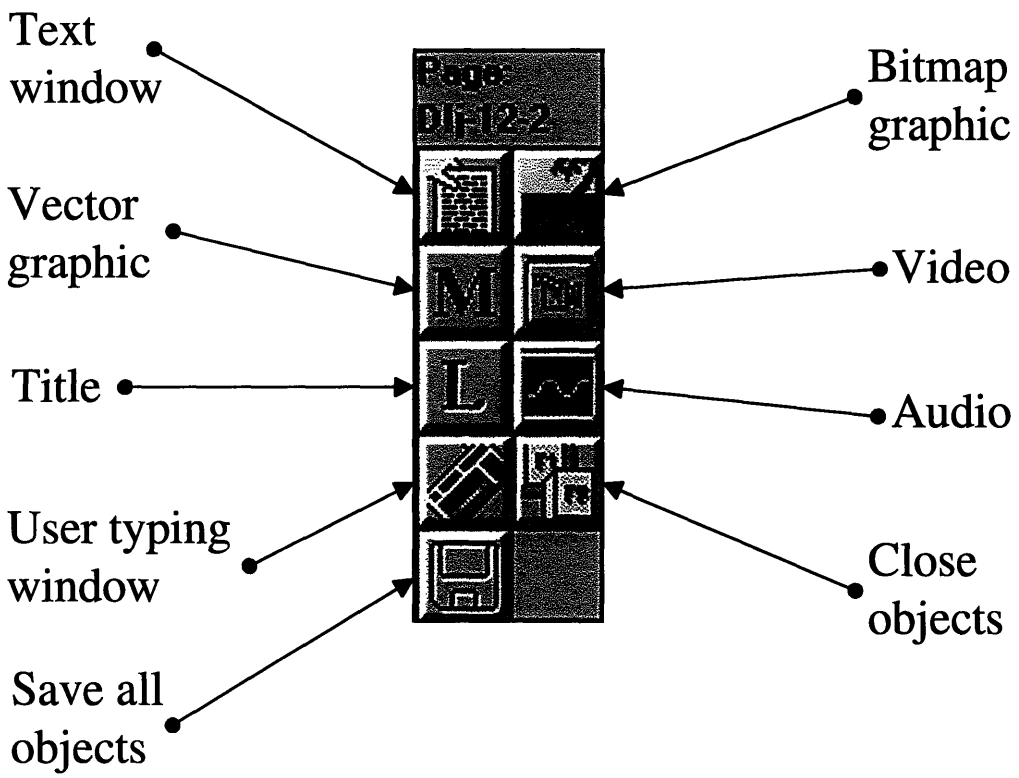



FIGURE 3.6: OBJECT PALLET

Pressing the right button of the mouse on any object brings up a menu of relevant properties, shown in Figure 3.7, that can be set for each object. Setting filename, location and size features has an immediately visible effect. To see the effect of setting other features like sequence, transition, and interval require the screen to be "played." This is done by pressing the "repeat" button on the bottom of the screen. Links function only in the player mode and do not function in the screen arranger. Figure 3.8 shows the link dialog-box. Any object except audio files can be linked to either an index-set (a set of pages connected together) or to another object on any page. Changes to objects on the screen are saved either one at a time with the "update" menu item or all at once by clicking on the disk icon on the object pallet.

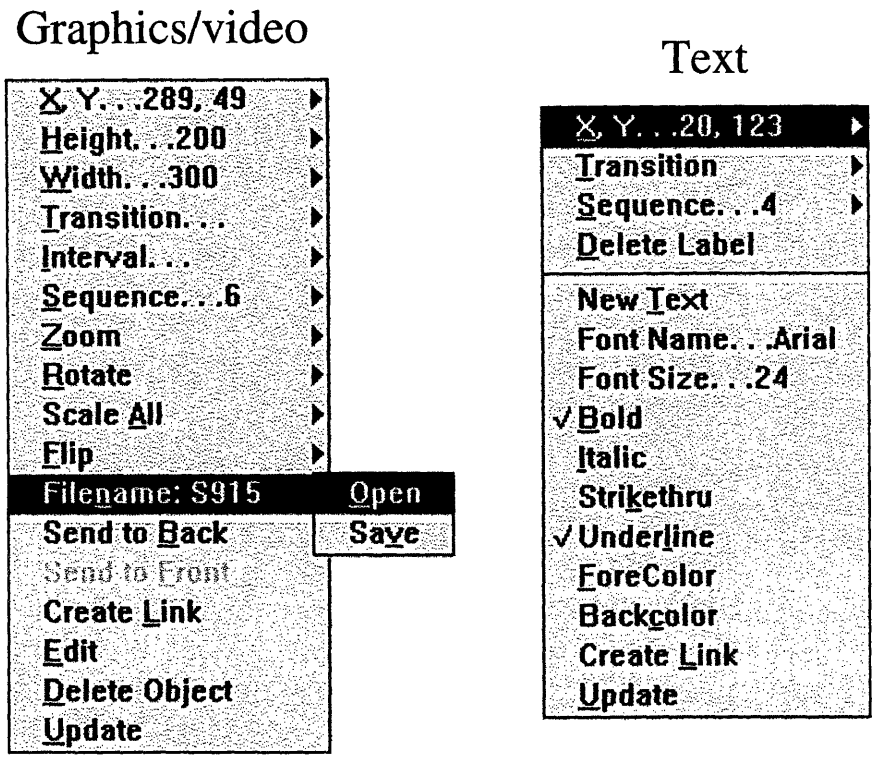



FIGURE 3.7: POP-UP MENU

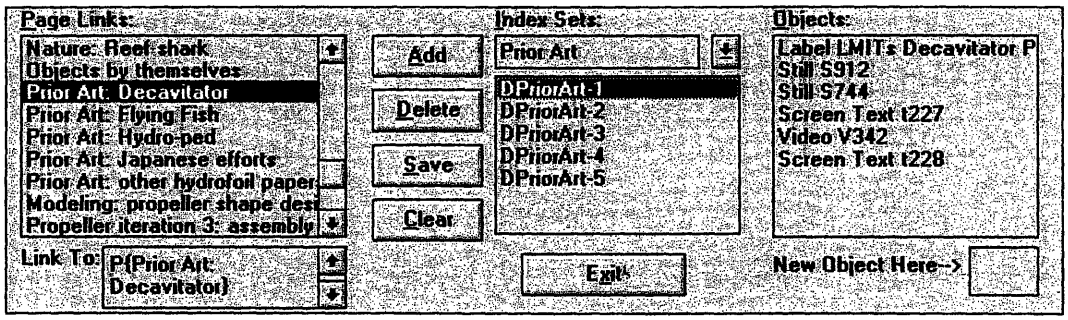



FIGURE 3.8: PAGE LINKS DIALOG BOX

The screen-arranger code represents the largest and most complex part of the EDICS software. It is also the place where there is the most room for improvement in use-ability. The current state of the program is such that adding new items directly to the database is much easier. The screen arranger, however, manipulates objects already on the screen easily. The ultimate goal would be to have the screen-arranger application robust enough to function without the need of directly manipulating the database tables. I would recommend the following changes and additions to reach this goal:

- Have a table listing of all the objects on the screen (the page table) accessible. It would list all the objects in sequential order, and all their features. The sequence could be changed by just dragging objects (or sets of objects) up or down the list. Features could also be edited directly.
- Add a properties window that is similar to the Visual Basic properties box. The properties of the highlighted object would appear in the window allowing them to be edited. This could even replace the pop-up menu.
- Have a question box ask if you are sure you want to leave the current page without having saved the changes.
- Allow objects to be scaled by dragging a corner. This is difficult to implement so we left it for future development. Currently object scaling is set by changing the actual width and height values. You should also be able to rescale the object to original size.
- Add some basic drawing tools, i.e., lines, circles, rectangles, fills, line types and colors. These would all be vector graphics and become part of a metafile (a Windows metafile combines both bit-map and vector graphics in one file).

# CHAPTER 4. DEVELOPING EDICS' SOFTWARE

## 4.1. AUTHORING SYSTEMS: SELECTION OF VISUAL BASIC

The search for an authoring environment for the new EDICS was started in the Fall of 1993 with the help of a study conducted by MIT's Center for Educational Computer Initiatives (CECI) [Ali-Ahamd, Bolduc, Trometer, & Webster 92]. Five authoring environments were considered: Claris HyperCard, Asymetrix's Toolbook, MacroMedia's Authorware Professional, MacroMedia Director, and Microsoft Visual Basic. The selection was based on speed, ability to present all media types (especially digital video), cross-platform compatibility (Apple Macintosh and IBM PC), ability to play simple 2D animations and to allow for hypertext.

HyperCard was very appealing because previous work was conducted in it, permitting us to lose the least amount of work. The problem was that it didn't readily support color graphics or QuickTime video. A different program would be required for the IBM PC that could convert HyperCard stacks. This was determined undesirable because translation is not very good and the IBM PC was to be our main delivery platform.

Asymetrix's Toolbook was a very strong contender for PC-based delivery because there was some translation between HyperCard and Toolbook. Toolbook also filled all the requirements except that of speed.

MacroMedia's Authorware Professional was evaluated purely based on CECI study [Ali-Ahamd, Bolduc, Trometer, & Webster 92] because we could not get an evaluation copy and it was beyond the project's budget to purchase one just for evaluation. It had a very appealing characteristic of being cross-platform. However, it was such a high level environment (as opposed to low level like C) that it was not as flexible as other options. It was also deemed a poor performer and therefore not selected.

MacroMedia Director was also cross-platform and had all the performance characteristics desired, including very strong animation and graphics features. Unfortunately it didn't support hypertext and became cumbersome with large programs such as this one.

Microsoft Visual Basic was initially not considered because it is not really considered a multimedia authoring environment, but rather a programing language, requiring much more coding than the other systems. It uses objects (forms and controls) that can control each other in a hierarchical relationship that includes built-in Windows system objects [Appleman 93]. We considered Visual Basic based on the suggestion of CECI, because it filled all the criteria except cross-platform capabilities (although a Mac version was rumored). We chose Visual Basic because it gave us much greater control over the system. It allowed us to design a database-driven program that would make cross-platform playback relatively easy.

## 4.2. DESIGN OF THE PROGRAM

One of the key advantages of Visual Basic that we exploited was its built-in SQL calls. It uses Microsoft Access database files as a native file format. I wanted to take advantage of this feature to create a system in which the content data is completely independent of the software. Prior experience upgrading old EDICS from HyperCard version one to version two showed that updating the system also required updating the content.

The basic database structure is discussed in the previous chapter. The program approach is shown in Figure 4.1. Archived objects are all stored as separate files and played by the Player according to the commands in the main Database. The main database is programmed using the Screen Arranger.

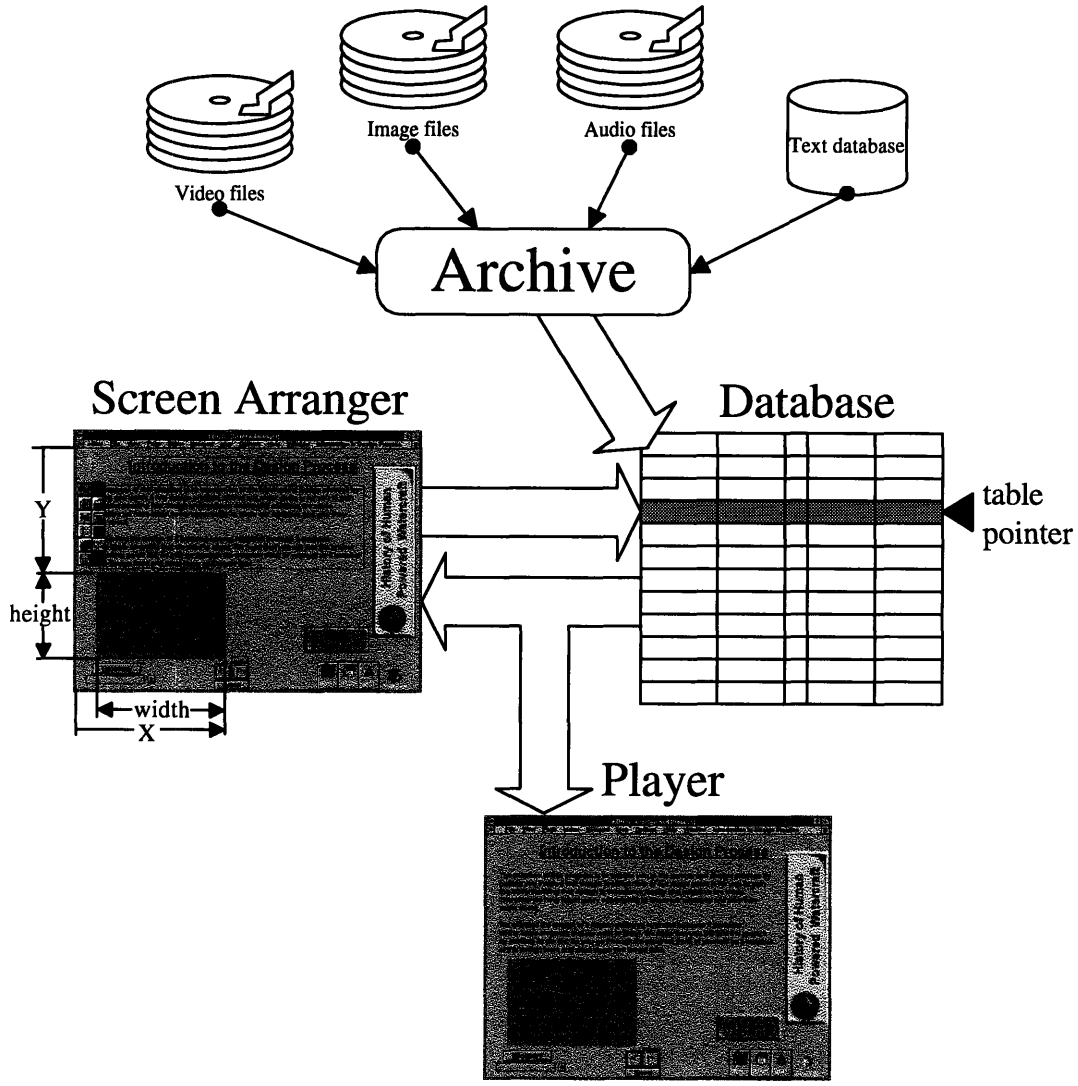



FIGURE 4.1: EDICS PROGRAM APPROACH

A main menu structure provides access to all the different chapters of EDICS, which currently includes Bearings and Design Process. The index, shown in Figure 4.2, is a separate floating window that has terms defined and linked to the pages with related objects (from the object term table) providing access to all those pages—this particular feature is not yet fully operational. Selecting a term brings up the glossary, shown in Figure 4.3, giving a written definition and potentially cross references to other objects. Navigation controls follow a basic left-to-right metaphor similar to old EDICS. Branches can be attached to any object (except audio files). Objects with branches attached act like

buttons that do one of two things when clicked on. If the object is attached to an index-set, the program heads down the series of pages in that index-set. A return button becomes available allowing the user to return to the branch point. If the object is linked to another object then the latter object will appear on top of the screen and can be put away by clicking on it.

For the complete program structure outline see Appendix A. The majority of the actual coding was done by undergraduate research student Acee Agoyo: a complete listing is in Appendix B.

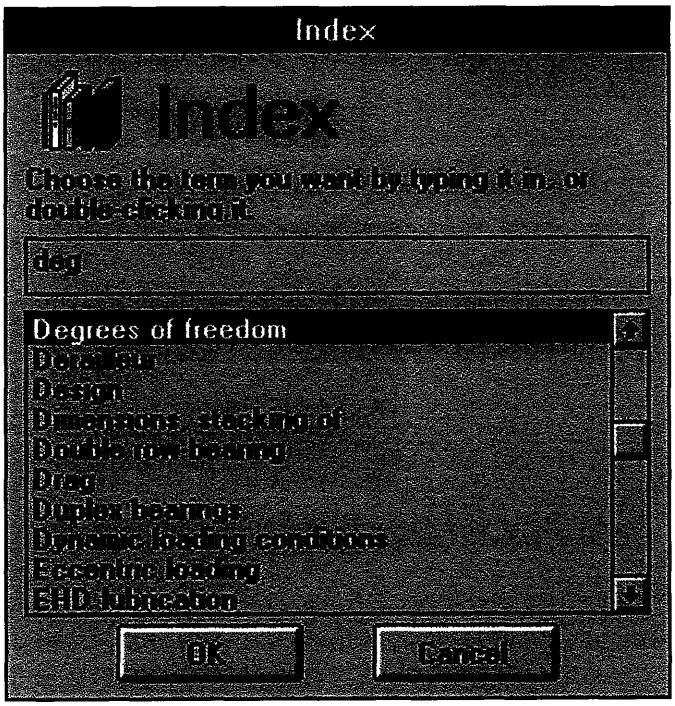



FIGURE 4.2: INDEX WINDOW

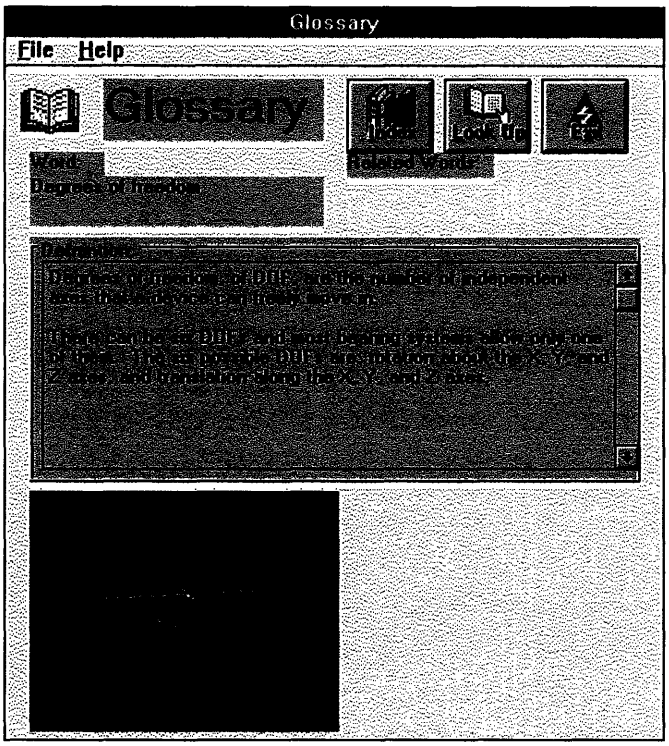



FIGURE 4.3: GLOSSARY WINDOW

4.2.1. Interactivity

Five types of interactive sessions were designed: locate the item, interactive design, typed definitions, estimation, and multiple-choice check box. The approach taken with these was a general one, in line with the rest of the program. Therefore the implementation required carefully thought-out code that would work for many different problems.

4.2.1.1. Locate the item. Locate the item is based on the "Find the Bearings" activity from the old version of EDICS. It allows hot spots to be attached to any part of any image and questions associated with these spots. When a student selects one of these points on the picture a series of questions about that location is asked. With the Find the Bearings problem the student is given a Rube Goldberg device, shown in Figure 4.4, and is asked to find all the bearings. A series of questions is then asked about each selected bearing.

Underlying the program is a set of database tables that allows any image and any number of hot spots to be associated with a problem.

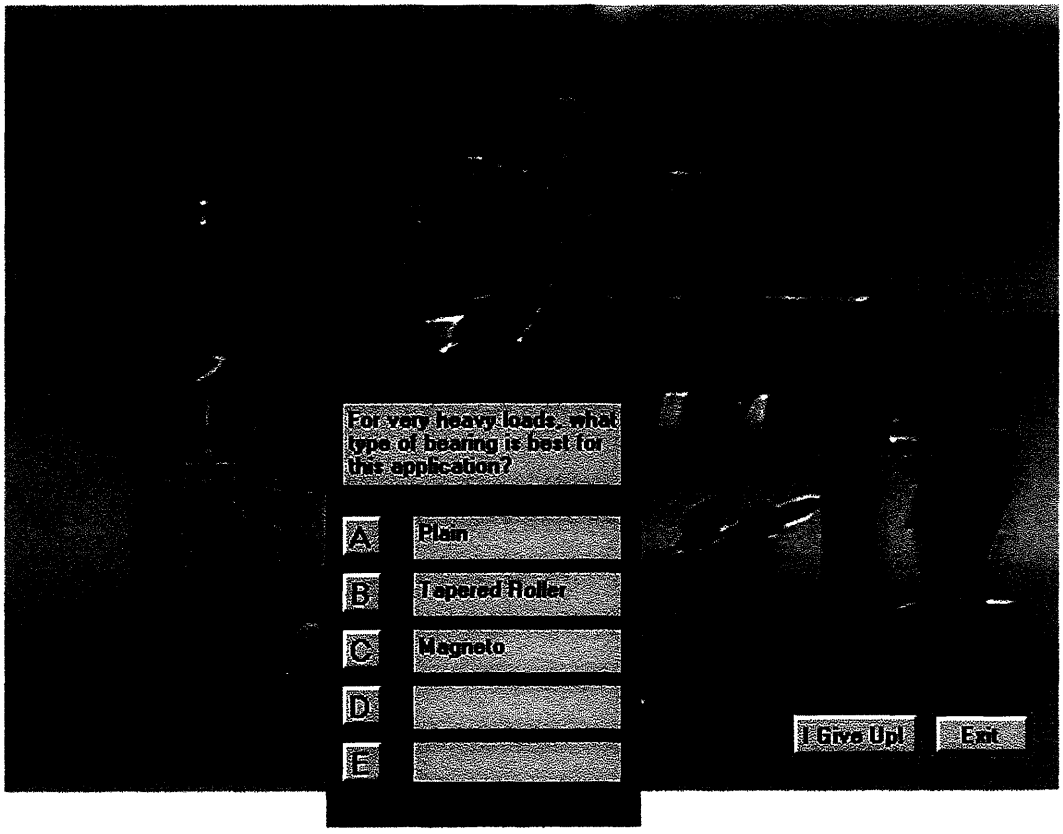



FIGURE 4.4: FIND THE BEARINGS ACTIVITY

4.2.1.2. Interactive Design. Interactive design is based on a program designed by an MIT undergraduate research student, Joe DeMare, using SuperCard on the Macintosh. In the Visual Basic version an image of an incomplete or incorrect design is displayed (see Figure 4.5). The user selects potential solutions from a pallet connected a particular area of the design. The original image is changed to match the selection. As many pallets as desired can be accommodated, corresponding to different parts of the design. The underlying database accesses metafiles. Solution images (up to three) can be requested once the user has selected items from the pallet.

The example shows an improperly designed shaft/bearing-mounting. The user is given a pallet of possible bearings and mountings to select from for each side of the shaft. When the mouse is over one of the items in the pallet a description appears at the bottom of the screen. When bearings have been replaced the user can see two solutions to the problem.

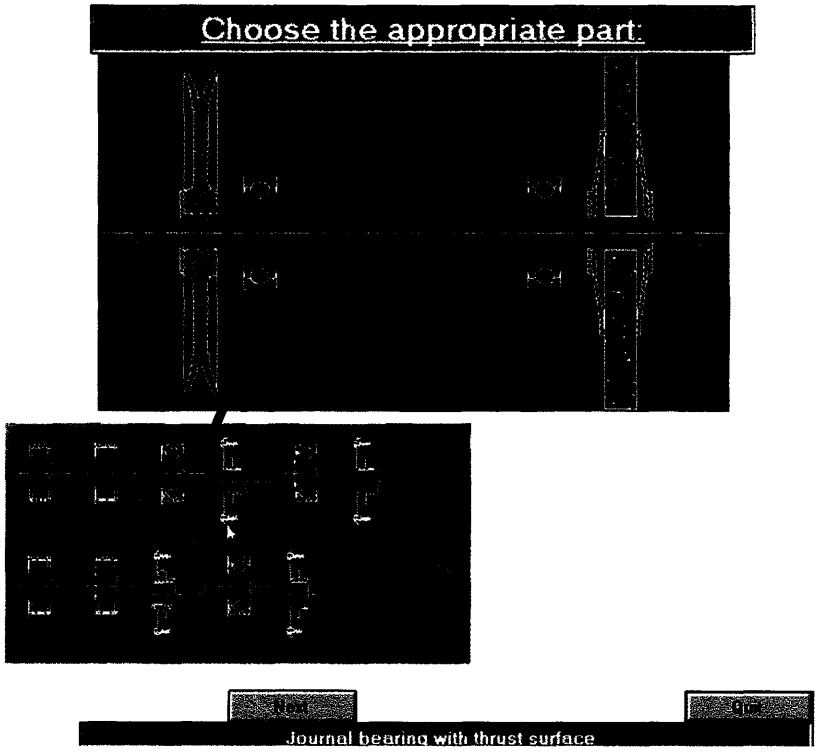



FIGURE 4.5: INTERACTIVE DESIGN ACTIVITY

4.2.1.3. Typed Definition Box. The typed-definition box is a simple scrollable text window that can be placed anywhere on the screen. The user is asked to type in a definition. The typed information can then be carried over to another screen to compare it with other possibilities.

4.2.1.4. Estimation. The estimation activity has not yet been implemented. It is based on the old EDICS example estimation questions. The user is asked to estimate the value of some system and to enter it on a slider bar. Then the user can click on the answer button to see if she or he was right. In one example the user is shown a video of a dentist drill and

is asked to estimate the speed at which it spins. Estimation skills like these are extremely important for a design student to develop.

4.2.1.5. Multiple-Choice Check Box. The multiple-choice check box has not yet been implemented. It allows a series of questions to be asked with multiple choice solutions offered in check boxes.

### 4.2.2. Notepad

A notepad is designed to allow the student to take notes from the program or to copy information and objects. Though not fully implemented it would link notes to particular pages for easy reference.

### 4.2.3. Sound-off

Finally the sound off or silent mode brings up a transcription window where all the audio is transcribed. This allows EDICS to be both run in situations where the sound would bother other people or on systems without sound capabilities.

## 4.3. DISADVANTAGES OF OUR APPROACH

When I set out to design the program I was expecting a much larger full-time programming staff. Ultimately one very skilled undergraduate research student wrote the lion's share of the code, with only the index, interactive-design, and "locate-the-item" being coded by two other undergraduates. As a result the content end of EDICS had to be partially put on hold while the software was being developed. In the end we have a very ambitious piece of software with two chapters almost complete.

One of the major problems with the of EDICS v2 was that the openness of HyperCard allowed for bad code to be hidden in all sorts of little corners, written by well-intentioned research students. By bad code is meant code that is not properly documented and is written in an undisciplined fashion. While it may initially function properly, it will probably

cause problems further down the road. Because of this, debugging EDICS v2 was a nearly impossible task. This version of EDICS is a partial response to the problems encountered with HyperCard. The problem is that we have replaced a highly decentralized system with a highly centralized one. In the old EDICS if anyone wanted to add a new type of interactivity, she or he could just go in and do it (of course you may end up with more bad code lying around). With the new design, adding anything new is a slower and more involved process that requires strong programing skills and a detailed understanding of the structure of the EDICS program.

There may be a compromise to be struck here, allowing custom forms to be objects themselves. This would allow new features to be added much more quickly and at the same time provide a buffer isolating the main program from potentially poorly written code.

The other major problem inherent to any program of this size is the need for maintenance. It requires a skilled programmer with a strong working knowledge of the software. This will become most apparent when upgrading to the next (32-bit) version of Visual Basic, expected in 1995, promising improved performance, new features, and many hassles. This will be reminiscent of the transition from HyperCard version one to two.

Ultimately programs of this size will need significant overhauls to overcome bugs that are built into the program structure. The only way to purge the program of these is to restructure the code. Major revisions of EDICS should coincide with operating-system upgrades, version changes, or platform changes. Note that the key advantage of having the content decoupled from the software is that the content is retained through a software upgrade.

# CHAPTER 5. FLYING ON WATER—A DESIGN-PROCESS CASE STUDY

## 5.1. THE NEED FOR THIS CHAPTER

On the current status of engineering students and curricula Wilson and Blanco write that there are three trends:

- The range of disciplines that design has to cover, or at least introduce to students, is continually expanding.
- The share of the curriculum that is devoted to teaching design has been significantly reduced over the last 50 years, while the emphasis on engineering physics has increased, so that the background of the younger design instructors often includes little design.
- The range of preparation of the undergraduate students entering mechanical engineering is much more varied now, and the mechanical experience is generally less than in the past [Wilson & Blanco 91].

The subjects developed thus far for EDICS are all component-based, using the power of the medium to familiarize students with various components and the design of elements such as bearings and shaft/rotor connections. This is very important for students who lack the experience of taking mechanical devices apart. From my experience, however, in working with students on various project competitions and courses, students have a difficult time understanding how all these elements come together into a complete device. Good design requires two things: experience and a "systemized" process [Sherwin 82]. Both experience and understanding of design come from dissecting mechanical devices (design anthropology) and from doing actual design. Dixon would argue, however, that there is a danger of confusing experience with education [Dixon 91]. My basic goal is to attempted to give students a guided design experience using the capabilities of multimedia. Specifically, I hope to get across the following four concepts.

### 5.1.1. Process

Many design textbooks cover process with simple hypothetical examples or in a purely symbolic fashion. Often I find students in the machine shop cutting material without having a solid idea of what they plan to do. Their plans are usually very vague and they may have a crude conceptual sketch. This comes from a public that thinks a pretty rendering of a new-car design means it is ready for production. At the same time these students have been taught design process steps in their design courses. Engineers need to understand not only the process, but experience it to have faith in it. The overriding question of this thesis is: can you give students some of this experience with an interactive media-rich example?

### 5.1.2. System and component levels

Students have difficulty understanding the difference between component and system level design. The concern here is that the cause-and-effect relationship between system and subsystem are not fully grasped by the new designer. Students learn to design and analyze at a component level well, while the system design is often neglected. This is especially true when design is a team effort, where design tasks are often issued at the subsystem level. The new designer often embarks on what I call "gimmick design." This is the tendency to expect a design to succeed because it included some novelty, like carbon-fiber frame tubes on a poorly designed bicycle. High-performance design in things like race cars force designers to be thorough. You may have the best carbon/carbon clutch ever designed, but if the wheels fall off half way through the race you have failed. A case study provides many examples of how the selection or design of components affects the system.

### 5.1.3. Geometric modeling

Geometric modeling is also greatly neglected in the current mechanical engineer's education. Drafting in itself is not what I mean, but the ability to develop models to simulate the look, fit and function of a system; and to be able to communicate that design, i.e., drafting. An element that has been squeezed out of the engineering curriculum in many schools is the drafting course. For some reason this topic is looked down on as not being a core engineering science. I generally agree with Slaughter that drawing (geometric modeling) comes first in order of importance in design and calculations come second [Slaughter 63]. This is not to say that engineering science isn't important: it is central to design. I maintain that the majority of mechanical design is a geometry problem. Whether you're a specialist in fluid dynamics or controls, where design is involved so is geometric modeling. I break this subject into three main areas: conceptualization, modeling, and communication. This is similar to Barr & Juricic's break up of engineering-design graphics (EDG) into design ideation and design communication [Barr & Juricic 91]. The goal here is for the student to understand that there is a need to develop the skills in all these levels, to do good design, using the best available techniques. The EDICS design chapter is rich with notebook sketches, color renderings, rendered solid models and plain-old dimensioned detail-drawings.

### 5.1.4. Swimming in the Charles River: design iteration

Too often design case studies are given with the edge taken off them: they show the problems and how the brilliant engineer solved them. There is a great deal more drama and unexpected results that makes design a rich and exciting experience. Design examples that resolve perfectly intimidate students because they show a problem, and its solution without showing the struggle and (sometimes embarrassing) mistakes. By showing the iterations, the mistakes that were made and their consequences, I hope to empower the student to not be afraid of those mistakes.

## 5.2. AN EXPERIENTIAL PEDAGOGICAL APPROACH

All these lessons I wish to relay in the context of a real design problem. By presenting design phases in an as-they-occur manner it is hoped that the student will learn from the experience without having necessarily been told what he or she is to learn. This is much like the mentor or apprentice model of learning, where lessons are hidden in tasks. Here the lesson is hidden in the design experience.

## 5.3. USING A HUMAN-POWERED HYDROFOIL AS A CASE STUDY

Why build a human-powered hydrofoil? First I wanted to select a real device as opposed to some purposeless or worse a hypothetical design. Students need to be excited and passionate about design, so selecting an exciting device was a major criterion. Secondly the selected device must have all the components that are or will be covered by EDICS, that is bearings, fastening and joining, springs, torque transmitting elements, power transmission, materials, and cam mechanisms. Finally it must satisfy scope, scale and financial limitations. I also wanted to draw on available human resources and experiences. Selecting a human-powered device seemed to fill all the above criteria. MIT has the world record for human-powered speed on water (1991 men's, 1992 women's), in a hydrofoil [Drela, Schafer & Wall 92].

Instead of a record-setting craft a practical human-powered hydrofoil was selected, later to be called the Skeeter. At first I wanted to take the project all the way to a beta or preproduction stage, but that was later trimmed back to an alpha prototype. The criterion for the boat was for it to be a commercial recreational craft that significantly out performs a single-person scull or an eight-person crew shell, to be in the price range of a very high-end bicycle, and to take a range of rider sizes and abilities. Initial design specifications are given in Table 5.1.

TABLE 5.1: INITIAL DESIGN SPECIFICATIONS

| | |
|---|---|
| Rider sizes | |
| heights | to 2 m (6'5") |
| weights | to 110 kg (250 LB) |
| Design specifications | |
| weight | 23kg (50 LB) |
| take off power | 250 watts |
| top speed (well trained) | 7.7 m/s (15 knots) |
| Manufacturing price range | to $3000 |

Design of the hydrofoil, shown in Figure 5.1, started in the summer of 1992 and was built during the summer and fall of 1993. It was named the Skeeter, the common name for a water bug.

FIGURE 5.1: SKEETER HYDROFOIL

The general approach to presenting this material is to give the theory or concept behind each phase in very general terms. If the student is then interested in how that phase was implemented in the design of the Skeeter he or she must click-on the related buttons on the screen. The sequence more or less follows the timeline of the design. The timeline

is continuously displayed so that the user can have some idea of where in time he or she is. The main sections of the chapter are:

introduction;

design process;

project planning;

concept research;

concept development;

layout development;

modeling;

design review;

concept/layout selection;

human factors study;

detail development; and

conclusions.

For a complete outline of the chapter see Appendix E.

Pains were taken to show every mistake made in the design work. For example when the Skeeter was first put in the water the outriggers were so undersized that the whole thing flipped over. I have included the video footage of one of the designers (me) taking a bath in the Charles River and the four subsequent iterations required to get the design right. "Murphy meant to remind us—the engineers, mechanics, fabricators, welds, machinists, drivers and pilots of the world (in short, those who deal intimately with the potentially deadly combination of man and machine in motion) that the price of human error and/or oversight in any branch of engineering can be (and often is) very high indeed" [Smith 84]. The consistent comment that I got from students looking at the program was that it was real to them. It was not some perfect solution to a contrived problem, but one based on the real struggle associated with design. Every element of the design of the

Skeeter can be accessed by the student on the detail-design-and-construction pages shown in Figure 5.2.

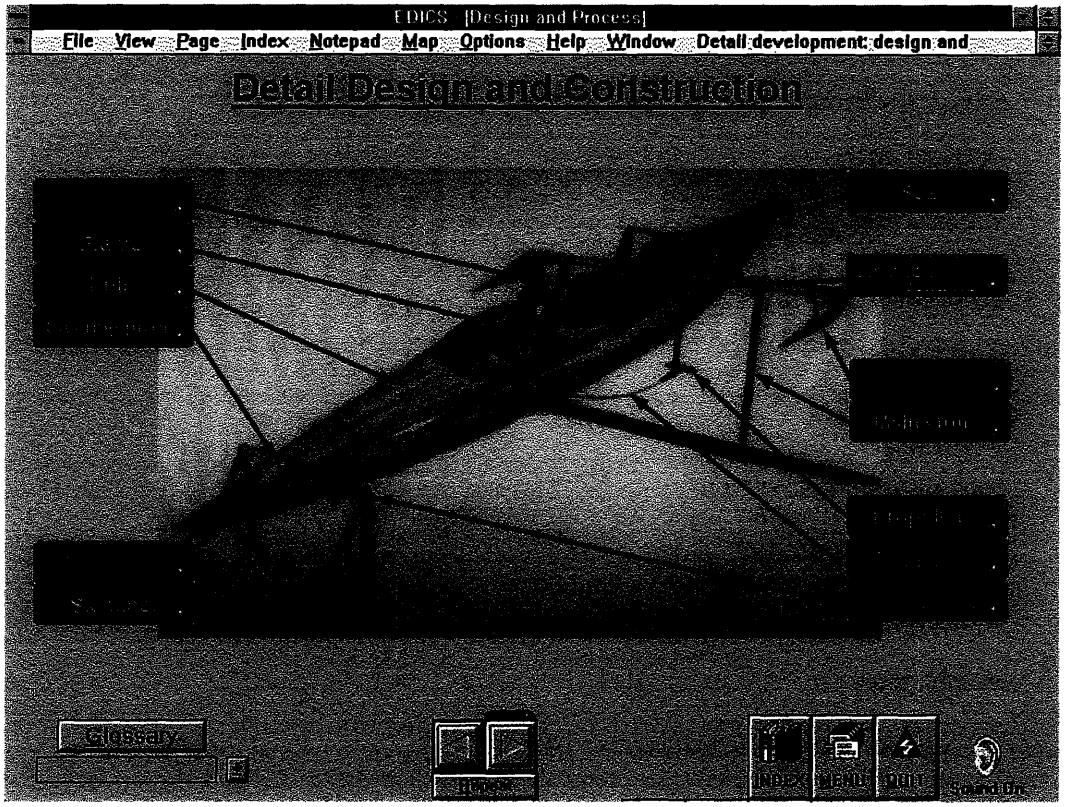



FIGURE 5.2: EDICS PAGE—DETAIL DESIGN AND CONSTRUCTION

Two other important areas that I wanted to make sure are clear to the student are that design is a team effort and that design needs to be scrutinized by peers and users. A "people" page shows the faces and contributions of the 16 people associated with the Skeeter project. Also, a design review was conducted, see Figure 5.3, illustrating the need for design review for success in design [Thompson 85].

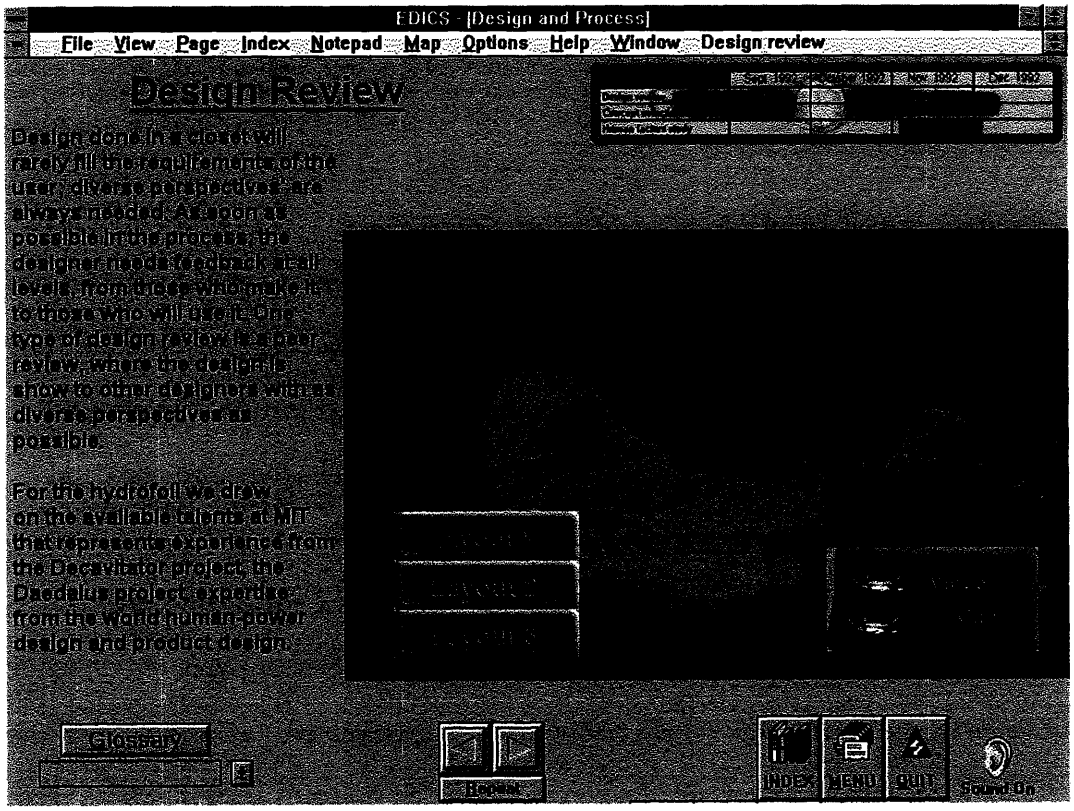



FIGURE 5.3: DESIGN REVIEW PAGE

In the end the tutorial covers as much of the details of the design case as possible. It does this with 35 video segments that account for 330MB of disk-space and 332 bit-map files (targa) accounting for 85MB of disk-space. The majority of video files and images are associated with the detail development page. There are some details left incomplete as of this writing including the page on the history of human-powered water-craft; pictures for people involved with the project; modeling pages for propeller design, foil design and dynamics/controls simulation; design decision-making page; and scans of notebook sketches associated with detail development pages.

# CHAPTER 6. EVALUATION AND STUDENT TESTING

## 6.1. TESTING APPROACH

There are two main goals in testing the tutorial at this point: to get overall feedback on the content and approach, and to get specific feedback on object design, program flow, and page layout. It is premature and beyond the scope of this thesis to conduct a full evaluation on the overall effectiveness of this tutorial. It is intended to use the data collected for continued development of the program and to adjust the material focus.

Second-and third-year mechanical engineering students are solicited for the evaluations by using the MIT email system to advertise. They are offered ten-dollar gift certificates for ice cream and told the test will take a maximum of two hours.

A pre-questionnaire was made up to try to assess the student's level in design. The first two questions are based on questions asked by Crismond, in a comparative study on the effectiveness of EDICS [Crismond 92], to categorize the students into novice, borderline and experienced. My intention is to try to get a feel for where these subjects are at in their understanding of design and design process. In the pretest I also asked for their general concept of what design is.

> Answers for questions 1 and 2: 0=none, 1=a little, 2=occasionally, 3=frequently
>
> 1. How much experience have you had taking things apart over the last few years?
>
> 2. How much experience have you had in repairing mechanical objects?
>
> 3. What does "design" mean to you?

The subject then goes through the program taking as much time as they wish. They were asked go through naturally and not to feel they must cover everything, that is to go through only the items that interest them. They were also encouraged to complain out-

loud so that the complaints, e.g., can not read text, could be recorded for the problem. Also the information that the subject covered on each page was noted.

Once the subject completed the tutorial, a post-questionnaire was given to try to understand what the he or she got out of the program, and recommendations to improve the program.

1. What does the design process mean to you now?
2. If you were part of a design team for a new type of Truck Transmission and you had to outline your design process in 5 steps what would they be?
3. What do you think you learned from the program?
4. Was there one particular element of the program that sticks out in your mind?
5. What suggestions could you make to improve the program?

## 6.2. TESTING

A total of ten students were tested. There were three females, seven males; four were second-year and six were third-year students. Testing occurred at the end of the year, so effectively all the subjects were one grade higher. They were volunteers and all were interested in design, multimedia and/or ice cream. This would bias them towards being more interested in the program than a random pool of students forced to use it as part of a course. Subjects spent between 25 and 120 minutes going through the program.

### 6.2.1. Problems with the program

There are about six sections of the program that were not complete and therefore blank. The topics were not critical to the overall effectiveness of the program. It should be noted that most students wanted to look at the history page; and system/control modeling, propeller design and airfoil design buttons on the modeling pages.

One bug in the software that we were unable to fix before testing was in loading images (specifically targa files). For some reason the plug-in that allowing us to load these images doesn't return all the memory resources when the object is unloaded. Consequently as you flip through screens memory is slowly consumed. Eventually objects on the screen start disappearing and the only solution is to quit the program and restart Windows. This is about a 2-minute process and breaks the rhythm of the user. Subjects had to quit from one to four times depending on which and how many pages they looked at.

Because we were running the program off a network server, sometimes the program would slow down and videos will pause for a disconcerting period of time. I would sometimes have to tell the subject that the video was not done playing even thought it had paused for an extended period of time.

### 6.2.2. Observations

It was interesting to note how different student concentrated on different areas of the program. Two students looked at every single concept while most only looked at a few. Some students read every bit of text on the screen, while others barely even glanced (one went as far as complaining).

## 6.3. RESULTS

Based on the pre-test questions and discussions with the subjects, they were scored on their understanding of design and design process. For example if a subject would use key terms in describing the design process, i.e., planning, need, etc., his/her scores would be higher. The answers to the first two questions were added to a subjective score of one-third weight based on the third question and discussion with the student. The total scores were then normalized on a percent scale, and plotted in Figure 6.1, showing that all had at least basic understanding of what design is.

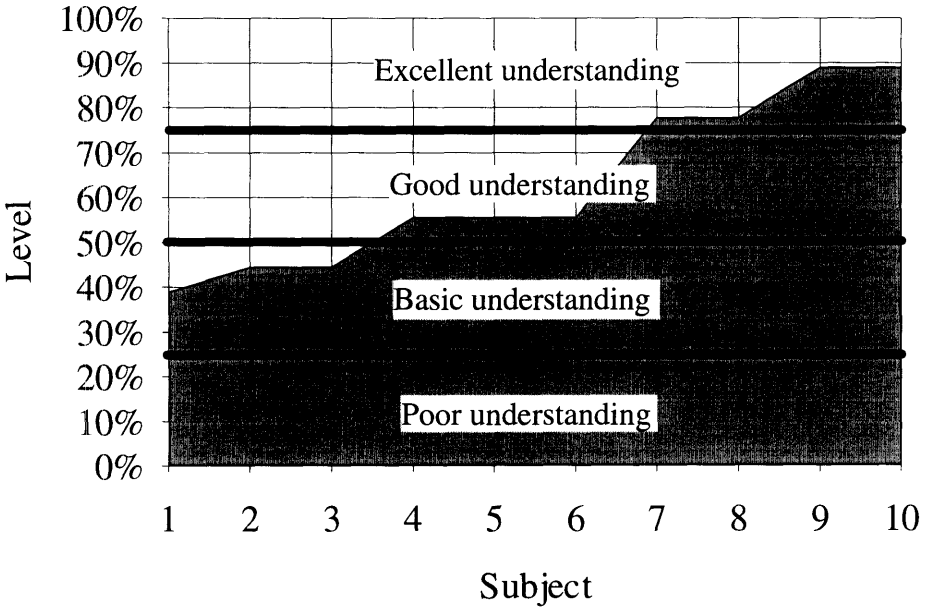



FIGURE 6.1: SUBJECTS LEVEL OF UNDERSTANDING OF DESIGN PROCESS

This would be expected because all the students have recently taken MIT's 2.70 course. This is the second-year mechanical-engineering-design course at MIT. Students learn the basics of design, including the process, and do a design-and-fabrication project. Students in Crismond's study were all in the process of taking 2.70 and he found that a sizable portion of them had little experience with mechanical objects, and even less with design [Crismond 92].

From observations made of the subjects as they went though the program, usage per-page was collected. Figure 6.2 shows the average percentage of each page covered by all the subjects and the standard deviation, along with the number of buttons on the page. Subjects tended to explore the whole page if the page was simple, i.e., the number of buttons on the page was small. The concept development page had 17 buttons, though it is not necessarily the purpose of this page for all the concepts to be looked at, (see Figure 6.3). The plot for this page has a high standard deviation, because two subjects went

through all the different concepts. If those two students are removed the average is 19% with a standard deviation of 10%. The other page with many buttons is the detail page, thirteen (see Figure 5.2). Again with this page it is not necessary for the student to go through every detail; however, each button has significantly more information than the concept page. A coverage on the average of 40% is acceptable. The responses to these pages were mixed. Students either really appreciated that there was so many choices and options to explore, or felt overwhelmed.

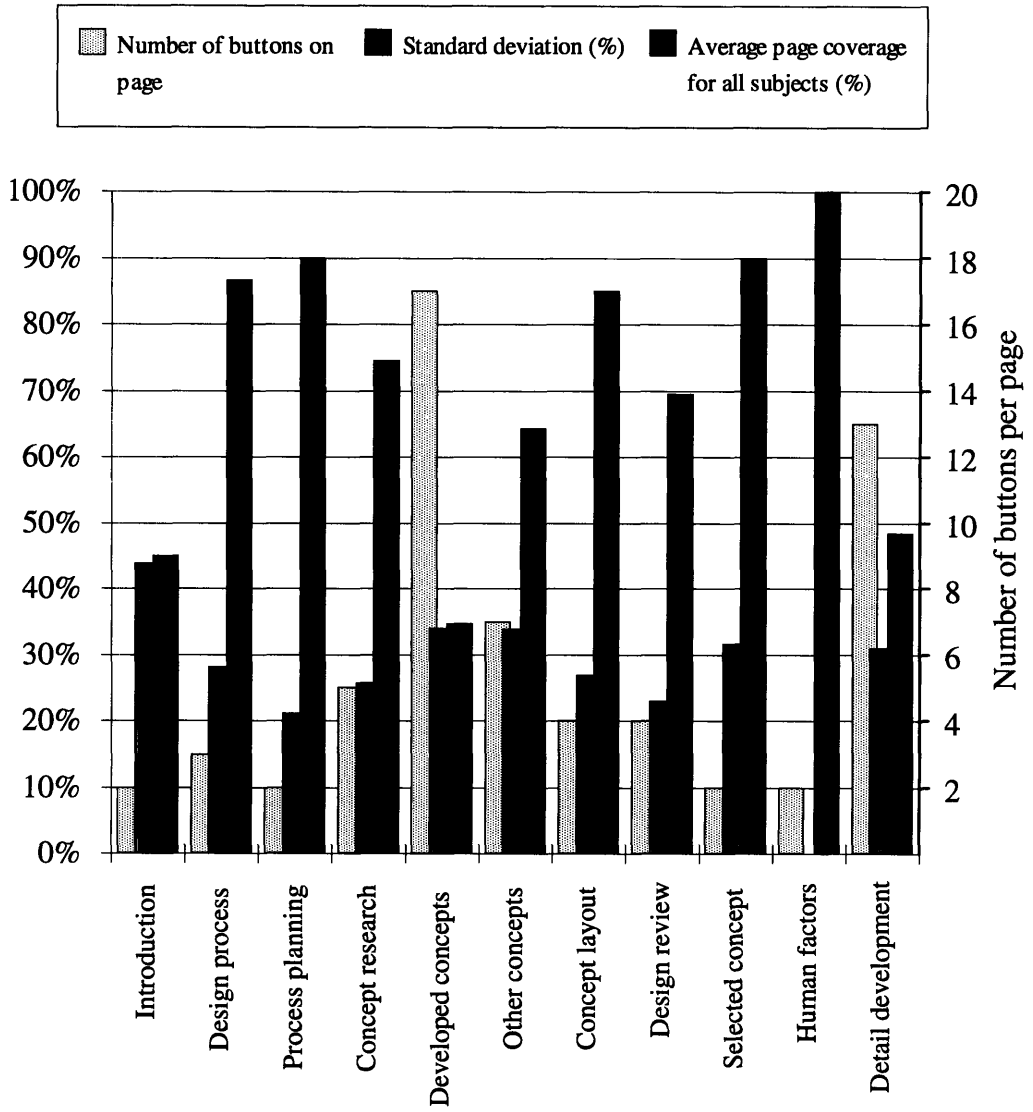



FIGURE 6.2: AVERAGE PERCENT OF MATERIAL COVERED ON EACH PAGE

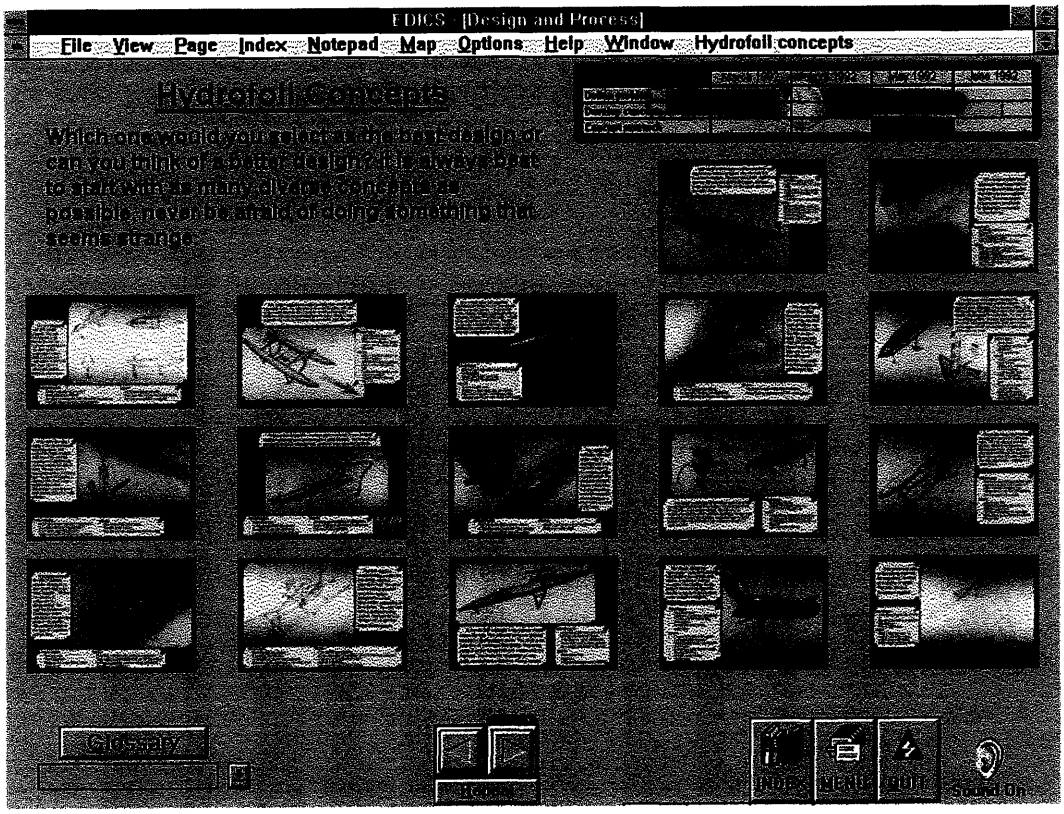



FIGURE 6.3: EDICS PAGE—HYDROFOIL CONCEPTS

Answers from the post questionnaire were categorized and plotted in Figure 6.4 for the number of subjects who felt that the topic was important to them or that they included it in their description of the design process. This is only a self-evaluative measure of content penetration.

- Importance of detail: Five subjects of the ten felt they gained an understanding of the quantity of detail work involved in design that they never had before, e.g., "An appreciation for the amount of detail."
- Iteration: This relates to evolution in component details. It was often noted by the students, e.g., "Some of the details (iterations, construction), I know about, but I didn't really consider them."

- Notebook: Throughout the program where appropriate notebook pages, scanned directly from the engineer's notebook, can be viewed. A couple students commented the value of seeing actual notebook pages.
- Process: Five subjects expressed an appreciation for the process presented on a real design, e.g., "It was really interesting to see the design process in action." All of these students had a good understanding of the process of design in general terms to start with.

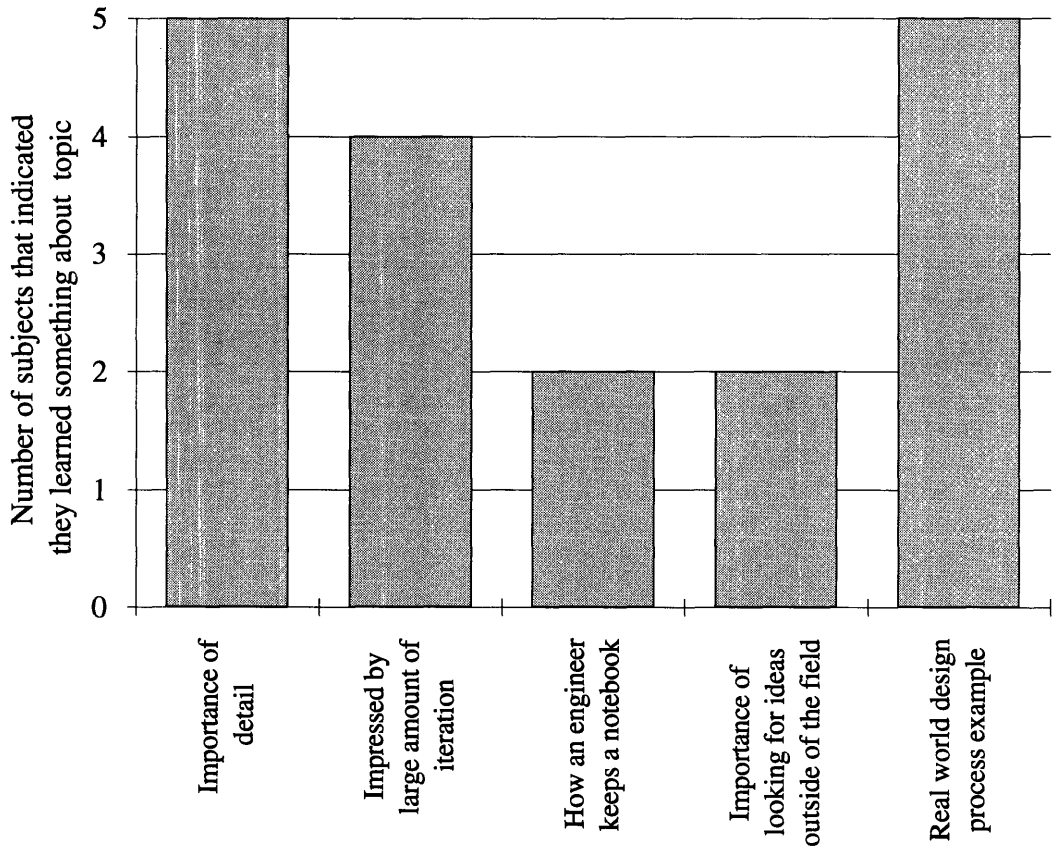



FIGURE 6.4: SUBJECT MATTER CONVEYED

In the end most of the subjects liked the material, but all had comments on the presentation. Many of the comments and suggestions were based on individual images and

videos while others were more general. Most felt that the first two-thirds of the material was slow and boring. They all suggested having more video earlier on. I was surprised at the number of students who were interested in the process plan and how it evolved; they suggested that some sort of overlay be used to show the changes. Many of the students wanted to know where they were in the program and suggested some sort of navigation assistance.

## 6.4. CONCLUSIONS

The goal of the this study was to get overall feedback on the content and approach, and to get specific feedback on the screen objects. Overall the subjects were positive about the program, but not extremely. The results show that there are some clear flaws making the presentation occasionally slow and unclear. The target audience for the program should be one grade level lower than the tested students. They had prior experience with the material, hence the educational goals were not met. This adjustment would require defining of technical terms more clearly.

The main learning objectives for the user are process, system and component interaction, geometric modeling, and understanding design as an iterative process. From these only process and design iteration were clearly experienced by the subjects. It is clear from the test that the coverage of component interaction and geometric modeling is covered too passively and needs to be brought more directly to the surface.

# CHAPTER 7. CONCLUSIONS

A database-driven multimedia program was created using Microsoft Visual Basic and Access database. Database provides code for the program to present the media. A human-powered hydrofoil was design and built as a design-process case study for the program. The EDICS chapter on the design process of the human-powered hydrofoil was put into the database including 35 video segments and 332 bit-map files (targa). Ten students were used to test the overall approach to the subject of design process and to test individual elements of the program. Results show that there is potential in the approach but some changes are need to make it more interesting and effective.

# CHAPTER 8. RECOMMENDATIONS

## 8.1. SIMULATION

One area that is neglected in EDICS is simulation. Computers provide an excellent environment for simulating natural phenomena. Simulation as opposed to animation would allow for a student to change parameters to a problem and the computer would provide a simulation as a result of the input.

Because design is such a broad subject, the potential types of simulation and modeling that could be incorporated are also diverse. They could include finite-element modeling for material mechanics, kinematics simulations, fluid dynamics, etc.

The goal set early on for developing EDICS was to provide students with a "hands-on" type experience, in the hope that they would gain some basic understanding of mechanical design and design components. Simulation of systems would provide just that. The main difficulty lies in developing the simulation. Simulation and modeling software is becoming advanced and readily available on inexpensive desktop computers (the same ones used for multimedia). It would not be prudent to develop our own simulation programs when so much good software already exists. The best approach here would be to link EDICS to these programs. The tools for doing this are now becoming standard on all the different operating systems. They allow a program to be controlled from within another program. In Microsoft Windows this is called object linking and embedding (OLE).

Say we were to give the student a spring-design problem allowing him or her to size a spring in a dynamic system. The dimensions and material of the spring that affect the dynamics of the system could be simulated in a window under Mat Lab (an interactive "matrix laboratory") at the same time a FEA (finite-element analysis) fatigue model could

be in another window. The student could explore the problem in analysis packages that are available on the system at the time.

The advantage to this cellular approach is that the development efforts are not duplicated. The simulation and modeling programs are also stand alone packages. This reaches another need of new engineers: experience in using new computer aided engineering (CAE) tools [Kitto 93].

## 8.2. EXPANDABILITY

### 8.2.1. Where we are now

Current lay-out of the EDICS program allows for other users to create "index-sets," which are pages and objects from the EDICS program linked together. For example if an instructor wanted students to go through a specific series of pages, videos and images he or she could create an index-set connecting them together and assign it for the students to review. This would also work for projecting in the classroom. The instructor could have pre-selected a series of videos or images to playback in the classroom. The next level of user modification of EDICS is basically functioning. Here the user actually expands and changes the database to include or exclude items. If the professor has a special bearing example he or she wants the student to see at some point in the bearing chapter it can be added using the Screen Arranger utility adding videos and images from the local hard disk.

### 8.2.2. Future potentials

Implicit in the long-term success of a program such as this is its ability to grow and expand to fit the needs of its users. The large-scale high-capacity networking that is currently evolving nationally and internationally opens the possibilities for having EDICS as a less-defined web of information located on various systems.

For example if someone at a west-coast university had developed a database of composite material properties. A student on the east coast could be looking at the

database linked into a composite design tutorial developed and running at some southern university. Software tools that allow this are just now becoming available on the Internet, such as Mosaic. Because systems like these are both highly integrated and very autonomous, they give authors a forum for putting out materials much easier than publishing.

The usefulness of the system will lie in its completeness. If companies list their products and services on the net with the proper links, EDICS could be accessed as a type of help in understanding the product. In the other direction EDICS could access the product catalogs and manufacture's product-selection programs.

For example if an engineer needed to design a transmission and was reviewing power transmission in EDICS he or she could quickly link to a gear catalog and maybe run an automated gear-selection routine provided by the manufacturer. Also, if that same designer was trying to pick out a clutch mechanism from a networked catalog and needed a refresher on clutch types he or she could go into EDICS from the catalog.

As you can see this becomes a very vast system quickly. Any centrally controlled and managed system would be doomed to failure because it would presuppose that all of that knowledge is centralized. By having standard software links and networked information, the quantity of information becomes boundless.

Setting up a network-served system will require both an authoring environment that is portable and a server that is available on the Internet. A project that is currently being developed on the Internet is the National Engineering Education Delivery System (NEEDS) which is the brainchild of the Synthesis Coalition supported by the National Science Foundation. NEEDS is a courseware multimedia database. The intention of the system is to be an open curriculum covering the entire engineering spectrum [Agogino 93]. If EDICS were to be migrated to a cross-platform environment compatible with NEEDS it could become a node on the system. If EDICS is intended to be a money-making operation, then being part of NEEDS may not fulfill this.

The net result for the future planning of EDICS is to go beyond designing a single-user-based system that is CD-ROM delivered. The CD-ROM is an interim delivery solution with maybe a 5 to 10 year life. Since academic settings are the main aim for EDICS and they are the first to be networked together, focusing a broad effort towards network delivery would be the wisest long term goal.

## 8.3. DEVELOPING A STANDARD

The costs involved with developing a multimedia program are currently very high, but within the next five years they will drop very significantly. This will make authoring interactive multimedia information a single-desktop task where authors can do all or most of the work themselves. A standard accepted quality level, style and format for technical papers have evolved to the form it is today. Engineers and scientists know and can follow these without any additional help thanks to advancement in word-processing software and inexpensive high-quality printing hardware. Computer tools evolved to match the accepted standard more than standards changed to meet the capabilities of the computer tools.

The general potentials of multimedia are clear. Accepted standard structures and styles, however, do not yet exist [Rojas-Fernandez 91]. Software developers are designing tools that they hope will satisfy user needs; at the same time users don't know what they want or need. Ultimately efforts as this one will eventually lead to standards that will allow for the content person to focus on the content.

## 8.4. REALITY OF HARDWARE NEEDS

One of the realities that I came across is that we were severely under-powered for the work we wanted to do. Digital video is an enormous resource consumer. For example to capture 5 minutes of video (at 320x240 pixels and 15fps), edit it down to 2 minutes, and compress it requires the following:

- hard-disk space for pre-edited files: 200 to 300 MB;

- recompression time on a 486DX-66 with 64MB of RAM: 160 to 200 minutes; and
- hard disk space for final movie: 18 to 22 MB.

These computer problems plagued this thesis, i.e., processor speed (video compression is very demanding on floating-point operations), disk-space and network performance.

At the time the DX-66 was the fastest PC class computer available; now with new systems performance is easily doubled or tripled. That means that compressing the 2-minute video will take 50 to 60 minutes—still pretty slow. Specially designed digital video-processing hardware is still expensive but offers a good solution.

Disk space is a high-priority item. We had a 750 MB server for EDICS that had just enough space for one chapter (Bearings, Design Process, etc.) at a time. The design process chapter has about 350 MB of video files. This required nearly four full DAT backup tapes (5 GBs used) to backup all the original capture files for future editing. What is needed is about 500 MB for each chapter of EDICS plus an additional 500 MB for support files. A dedicated capture machine with one to two GBs of storage and a tape backup (1 to 25 GB per tape) would be required.

The size of the overall undertaking will govern what system and network (if a local network is called for) is selected. One way to eliminate the huge requirement for storage is to go to slide shows instead of full-motion video. In many cases a slide sequence with voice over can be more effective than video and consumes significantly less space (the 2:41 minute 650x415 pixel 8-bit slide sequence on the propeller development used only 5.2 MB of disk space and was very well received by students).

## 8.5. AUTHORING: WHERE TO GO FROM HERE?

It is very easy to think that something on the other side of the fence is better than what you have now. We have problems with bugs and implementation of this software that are very annoying; at the same time it must be remembered that the HyperCard version of

EDICS had entrenched problems too. In deciding the next step for the development of EDICS the reality of changing to another system must be considered.

The future of EDICS staffing will really govern what is the best direction to go. If minimal or no programming staff is going to be available then a switch to a non-programmed authoring environment (Authorware or Icon Author) maybe the best thing to do. Transferring the information maybe partially automated, but mostly manual. Issues of cross-platform capabilities need to be fully tested. The main problem with this approach is that it will not (for now) be available to UNIX users (the exception is Silicon Graphics machines).

If some programming staff is available, I would recommend developing a networked version of EDICS that is running under Mosaic. The approach would be to "hard wire" it. This takes the minimum up-front programming, as opposed to the database approach that we have take thus far. Mosaic would make EDICS available on all platforms and completely networkable. Compatibility issues with file format should be worked out to make EDICS compatible with NEEDS.

If a capable staff of programs is available then the options are very many. Continued development of the Visual Basic program, which may include a complete software rewrite is one possibility. This would allow for players to be written for other platforms. Network issues will need to worked out in the software if network delivery is desired.

All these solutions have the potential of being delivered as a CD-ROM, networked or both (choosing Mosaic would not be a great choice for CD-ROM if networking wasn't considered). These suggestions should be weighed against the current tide of the multimedia industry and academia.

# REFERENCES


[Ali-Ahamd, Bolduc, Trometer, & Webster 92] Ali-Ahamd, Wissam; Bolduc, Lynne; Trometer, Ruth; Webster, Michael; An Evaluation of Low-cost Multimedia Authoring Environments, MIT Center for Educational Computer Initiatives in collaboration with the American University of Beirut, MIT/CECI 1992.

[Appleman 93] Appleman, Daniel, Visual Basic Programmer's Guide to the Windows API, Ziff Davis Press, Emeryville, CA © 1993.

[Barr & Juricic 91] Barr, Ronald E.; Juricic, Davor; "Development of a Modern Curriculum for Engineering Design Graphics," Engineering Education, American Society of Engineering Education, Washington DC, Vol. 81 No. 1, Jan/Feb 1991.

[Bell 93] Bell, Trudy; "The Specialties: Multimedia, Magnetics, Multidisciplinary Education," IEEE Spectrum, Institute of Electrical and Electronic Engineers, Inc., New York, NY Vol. 30 No. 1, Jan. 1993.

[Borkowski 92] Borkowski, Chris A.; Kulp, Paul T.; Luetzelschwab, Mark; Gile, Michael; DiCesare, Frank; "Role of a Hypermedia Interactive Environment in a Laboratory Course," Frontiers in Education: Proceedings from 22rd Annual Conference, Nov. 11-14, 1992, Nashville, Tennessee.

[Brown 94] Brown, Eric, "Query Technology Recognizes Color and Shapes," New Media, Hypermedia Communications Inc. Vol. 4 No. 6, June 94.

[Coburn 92] Coburn, W. Geoffrey; Collins, Robert L.; Lindauer, George C.; Mullin, Thomas E.; Hnate, William P.; "Development of Multifaceted Instructional Modules for Introductory Thermodynamics," Frontiers in Education: Proceedings from 22rd Annual Conference, Nov. 11-14, 1992, Nashville, Tennessee.

[Cornell 92] Cornell, Ed, "Who Pushes The Buttons? Pixel Dust Part III," AV Video, Montage Publishing, Inc. November 1992, Vol. 14, No. 11.

[Crismond 92] Crismond, David P., Evaluating EDICS: analyzing the use of an interactive multimedia system in engineering design education, Massachusetts Institute of Technology M.S. thesis, Feb. 92.

[Dixon 91] Dixon, John R.; "The State of Education," Mechanical Engineering, The American Society of Mechanical Engineers, Vol. 113, No. 2, Feb. 1991.

[Drela, Schafer & Wall 92] Drela, Mark; Schafer, Marc; Wall, Matt, "Decavitator human-powered hydrofoil," Human Power, Vol. 9 No. 3&4, Fall-Winter 92-92.

[Jaafari, Picard & Bessège 93] Jaafari, A.; Picard, J.P.; Bessège, R.; "Multimedia, an Application for Education in Power Electronics," Fifth European Conference on Power Electronics and Applications, Brighton Conference Centre, UK, September 13-16, 1993, The Institution of Electrical Engineers ©1993.

[Kitto 93] Kitto, Kathleen; "The Role of CAE Tools in Engineering Technology," Frontiers in Education: Proceedings from 23rd Annual Conference, Nov. 6-9, 1993, Washington DC.

[Kretz & Colaïtis 92] Kretz, Francis; Colaïtis, Françoise; "Standardizing Hypermedia Information Objects," IEEE Communications, Institute of Electrical and Electronic Engineers, Inc., New York, NY Vol. 30 No. 5, May 1992.

[Loeb 92] Loeb, Shoshana; "Delivering Interactive Multimedia Documents over Networks," IEEE Communications, Institute of Electrical and Electronic Engineers, Inc., New York, NY Vol. 30 No. 5, May 1992.

[Marsden 90] Marsden, Douglas W., Development of the Engineering Design Instructional Computer System (EDICS), Massachusetts Institute of Technology M.S.M.E thesis, June 90.

[Rojas-Fernandez 91] Rojas-Fernandez, Helena; "Online and Hypermedia Information Design," IEEE International Professional Communication Conference, Institute of Electrical and Electronic Engineers, Inc., New York, 1991.

[Sherwin 82] Sherwin, Keith, Engineering Design for Performance, Ellis Horwood Ltd., West Sussex, England, ©1982.

[Smay & Genalo 92] Smay, Terry A.; Genalo, Lawrence J.; "Creating Courseware for Engineering Education," Creativity: Educating World-Class Engineers, Conference Proceedings, American Society for Engineering Education, June 21-25, 1992, Toledo, Ohio.

[Smith 84] Smith, Carroll, Engineer to Win, Motorbooks International, Osceola, WI, ©1984.

[Spina & Mukund 93] Spina, Robert; Mukund, P.R.; "A Multimedia Approach to Teaching Design in Core Laboratories," Frontiers in Education: Proceedings from 23rd Annual Conference, Nov. 6-9, 1993, Washington DC.

[Thompson 85] Thompson, Graham; Design Review: The critical Analysis of the Design of Production Facilities; Mechanical Engineering Publications, London ©1985.

[Wilson & Blanco 91] Wilson, David Gordon, Blanco, Ernesto E., "EDICS: A Multimedia Tutor for Engineering Design," MECHANICAL ENGINEERING, The American Society of Mechanical Engineers, October 1991, Vol. 113, No. 10.

# APPENDIX A: EDICS PROGRAM DESIGN

Menu Structure

- Single menu system that will provide access to all levels and sections of EDICS. It will indicate what the user has covered and will be a flip up item from the main controller. Will include some sort of preview.
    *1. The Design Process
    *2. Bearings
    3. Fastening and Joining
    4. Springs
    5. Power Transmission
    6. Materials
- Map will be a single scrolling page that covers all of EDICS.

Index/dictionary

- *Separate floating window.
- *All terms will be defined with a written definition and one of the following: still image, video, or animation illustration.
- Double clicking on all relevant index items will connect all related pages to that term in a linear sequence. Also devices (e.g., the Black and Decker hammer drill) will play back all related media.
- User will be able to create an index item that will connect all selected pages and media in a linear sequence. User will add pages that user is currently on.

Developer tools

- *These would be created for in house use initially, with future consideration for others to add their material to EDICS.
- *The main tool here will be the "screen arranger." The purpose of this utility is the creation of the EDICS screens. Media based on the Paradox database script will be placed and manipulated in this editor including sequence information. Once desired settings are made the screen will be "printed" as a record to a SQL database.

Media and presentation

- The decisions on which media to use will come from the answer of the question, "How would a real world engineer communicate this idea?" These are the purposed basic rules for the presentation of information on the new EDICS.
    1. Renderings, i.e., color hand drawings: conceptual ideas.
    2. *Sketches*, i.e., black and white pencil sketches: convey more detailed/developed ideas.
    3. *CAD solids*, i.e., computer generated 3D images and animation's: fully developed design.
    4. *CAD drafted*, i.e., computer generated 2D line drawings: implementation level.
    5. *Photos and Video*: for showing actual parts.
    6. *Audio*:
    7. *Text*:

- *Video: one video compression standard will be used for all slide shows, video and full animations. It will be MCI (Multimedia Controller Interface) compatible (note: currently we are using Cinepak, final selection is not necessary until end of the project).
- *A video controller will be provided for movies.
- *Still images: these will all use the same file format, allowing for 16 bit or less color, depending on desired quality. At time of CD printing all images will be fully edited and backgrounds incorporated. Currently using targa format for fastest loads.
- *Vector images: Windows metafile format will be used.
- *Audio files: all audio files that are not interleaved with video will be store as .WAV files. The program will allow the sequencing of media, including sound files.

Controls

- (*The prototype of this has been completed) The main EDICS controller will be a physical controller device similar in function features to that done earlier (Sepehr's interface in MacroMind Director).
- Navigation
    1. *Left-right navigation metaphor
    2. Down will correspond to in-depth information, e.g., calculations and equations
    3. *Menu access
    4. *Map
    5. *Index
    6. *Quit EDICS
    7. Page location will be indicated and user will be able to move between pages
    8. Speaker volume
    9. *Voice off/on

Branching

- *Index-set branching
    1. Any object, except audio, can be linked to any index-set. When that object is clicked on program branches to first page in that index-set. Subsequent pages appear to the right.
    2. A return-to control-button appears allowing the user to return to the branch point.
- *Object branching
    1. Any object, except audio, can be linked to any other object on any other screen. When the former object is clicked on the latter object appears on top of the current screen.
    2. Clicking on the "opened" object puts it away.

Interactions (the types of user activities will be limited)

- *Locate the item

1. Define invisible areas over an image that will have pop-up menus attached to them allowing for selection from a list. This is based on the current "Find the Bearing" activity.
2. Scores will be tracked for number of incorrect selections and tallied for that user.

- *Interactive design (this is based on work done by EDICS UROP Joe DeMare)
  1. Incomplete or incorrect design is given to the student.
  2. User selects design corrections. Design is changed to the selection.
  3. Each option will have pros and cons that will be given after student is done selecting.
  4. Up to three "expert" designs can be given as the solution.
  5. Each design selection will have a score allocated to it that will be used to evaluate the student.
- *Typed definitions
  1. Provide text field for user to type in requested definition.
  2. Transfer text to notepad if requested.
- Check Box Multiple Choice
  1. Puts boxes next to terms and allows for user to select relevant items
  2. Scores will be tracked for number of incorrect selections and tallied for that user.
- Estimation (this is based on example estimation questions)
  1. Three objects are associated with this screen all "play" at the same time.
  2. A slider bar or dial indicates values to a question posed.
  3. Multiple problems are allowed for each problem.
  4. Score are tracked for user.

Note Pad

- *Allow user to make notes.
- Add current page to notes with hot button for easy location.
- *Print out.
- Add a piece of media to notes.

User tracking

- *Ask user to enter name and password
- *Create new password for new users
- Track user's progress through the program.
- Keep scores on activities for user feedback.

Voice off

- *A floating/scrolling text field will come up if voice is turned off.
- *This will be transcription of the narration.

* indicates that this is completed

# APPENDIX B: EDICS PROGRAM LISTING

(Program code was written mostly by Acee Agoyo, with assistance from Chang Suh and Ryan Ehlert all MIT undergraduate students)

## FRMDEFIN.FRM CODE

```
Dim Objects(3) As String

Sub CreateIndexSet ()

   Dim SQLStmt As String

   SQLStmt = "Select * from [Index Database] where [Term] LIKE '" & CurWord & "'"
   dtaIndex.RecordSource = SQLStmt
   dtaIndex.Refresh

End Sub

Sub DisplayMedia ()

   Dim ObjectType As String, Definition As String
   Dim Ctr As Integer

   txtDefinition.Text = Objects(2)
   For Ctr = 1 To 3
      ObjectType = LCase(Left$(Objects(Ctr), 1))
      If ObjectType <> "" Then
         Load picArchive(Ctr)
         Select Case ObjectType
            Case STILL_CODE
               'Debug.Print STILLPATH & Objects(Ctr) & FILE_STILL
               picArchive(Ctr).Image = STILLPATH & Objects(Ctr) & FILE_STILL
               picArchive(Ctr).AutoRedraw = True
               picArchive(Ctr).Visible = True
            Case SCREENTEXT_CODE
               'txtDefinition.Text = CurObject
               'txtDefinition.Text = ReadFile(TEXTPATH & Objects(Ctr) & FILE_TEXT)
            Case AUDIO_CODE
            Case VIDEO_CODE
         End Select
      End If
   Next Ctr

End Sub

Sub Form_Load ()

   lblWord.Caption = CurWord
   CreateIndexSet
   GetMedia
   DisplayMedia
```

```
End Sub

Sub GetMedia ()

   On Error GoTo ErrHandler:

   Objects(1) = dtaIndex.Recordset.Fields("Object 1")
   Objects(2) = dtaIndex.Recordset.Fields("Object 2")
   Objects(3) = dtaIndex.Recordset.Fields("Object 3")

ErrHandler:
   Exit Sub

End Sub
```

## FRMDIALO.FRM CODE

```
Option Explicit

Sub cmdOk_Click (Index As Integer)

   Dim Temp As Integer, Ctr As Integer

   Tag = 0
   If Index = 0 Then
      If frameDialog(DIALOG_ZOOM).Visible Then
         Tag = Val(txtZoom.Text)
      ElseIf frameDialog(DIALOG_SCALE).Visible Then
         Temp = Val(txtScale.Text)
         If optScale(0) Then
            Tag = -(1 - (Temp / 100))
         Else
            Tag = Temp / 100
         End If
      Else
         Tag = Val(txtDimension.Text)
      End If
   End If

   For Ctr = DIALOG_SCALE To DIALOG_HEIGHT
      frameDialog(Ctr).Visible = False
   Next Ctr

   Hide

End Sub

Sub spnDimension_SpinDown (Index As Integer)

   If txtDimension.Text > 0 Then
      txtDimension.Text = Val(txtDimension.Text) - 1
      txtDimension.Refresh
   End If
```

```
End Sub

Sub spnDimension_SpinUp (Index As Integer)

   txtDimension.Text = Val(txtDimension.Text) + 1
   txtDimension.Refresh

End Sub

Sub spnScale_SpinDown ()

   If txtScale.Text > 0 Then
      txtScale.Text = Val(txtScale.Text) - 1
      txtScale.Refresh
   End If

End Sub

Sub spnScale_SpinUp ()

   txtScale.Text = Val(txtScale.Text) + 1
   txtScale.Refresh

End Sub

Sub spnZoom_SpinDown ()

   If txtZoom.Text > 0 Then
      txtZoom.Text = Val(txtZoom.Text) - 1
      txtZoom.Refresh
   End If

End Sub

Sub spnZoom_SpinUp ()

   txtZoom.Text = Val(txtZoom.Text) + 1
   txtZoom.Refresh

End Sub

Sub txtDimension_KeyPress (KeyAscii As Integer)

   If KeyAscii = 13 Then
      KeyAscii = 0
      cmdOk_Click 0
   End If

End Sub

Sub txtScale_KeyPress (KeyAscii As Integer)

   If KeyAscii = 13 Then
```

```
    KeyAscii = 0
    cmdOk_Click 0
  End If

End Sub

Sub txtZoom_KeyPress (KeyAscii As Integer)

  If KeyAscii = 13 Then
    KeyAscii = 0
    cmdOk_Click 0
  End If

End Sub
```

## FRMDICT.FRM CODE

```
Option Explicit

Sub cmdExit_Click ()

  Unload Me

End Sub

Sub mnuWindowItem_Click (Index As Integer)

  Select Case Index
    Case 0                    ' Cascade
      frmMain.Arrange CASCADE
    Case 1                    ' Tile
      frmMain.Arrange TILE_HORIZONTAL
    Case 2                    ' Arrange
      frmMain.Arrange ARRANGE_ICONS
  End Select

End Sub
```

**FRMLOGIN.FRM code**

```
Option Explicit

Const EDICSPWDDB = "d:\share\edicsdat.a\access\pwd.mdb"
Const EDICSUSERS = "c:\edics\users"

Dim Users() As String
Dim PwdDb As Database
Dim NewUser As Integer

Sub AddPwd (SupplyTb As Table, ByVal UserName As String, ByVal Password As String)

  SupplyTb.Index = "PrimaryKey"
  SupplyTb.Seek "=", UserName
```

```
   SupplyTb.Edit
   SupplyTb.Fields("Password") = Password
   SupplyTb.Update

   SupplyTb.Close

End Sub

Sub AddUser (SupplyTb As Table, ByVal UserName As String)

   Dim Response As Integer

   Response = MsgBox("You are a new user, would you like to register for an account?",
MB_OKCANCEL + MB_ICONQUESTION + MB_APPLMODAL, "New User")

   If Response = IDOK Then
      SupplyTb.AddNew
      SupplyTb.Fields("UserName") = UserName
      SupplyTb.Fields("Files") = EDICSUSERS + UserName + "\"
      SupplyTb.Update
      SupplyTb.Close
      txtNewVerify.Text = ""
      picPasswd.Visible = True
      txtNewVerify.SetFocus
   Else
      cmdOk_Click 1
   End If

End Sub

Sub cmdOk_Click (Index As Integer)

   UserName = txtUserName.Text

   Select Case Index
      Case 0                        ' OK Button
         If Not ValidUser(PwdDb.OpenTable("Password Table"), UserName) Then
            NewUser = True
            AddUser PwdDb.OpenTable("Password Table"), txtUserName.Text
         Else
            If Not ValidPwd(PwdDb.OpenTable("Password Table"), UserName, txtPasswd.Text) Then
               MsgBox "Invalid Password, Please Try Again", MB_OK + MB_APPLMODAL
               txtUserName.Text = ""
               txtPasswd.Text = ""
               If NewUser Then
                  picPasswd.Visible = False
               End If
               txtUserName.SetFocus
            Else
               GetUserInfo PwdDb.OpenTable("Password Table"), UserName
               PwdDb.Close
               Unload frmLogin
               frmMainMenu.Move 2500, 1400
            End If
```

```
        End If
    Case 1                      ' Cancel Button
        QuitPgm
  End Select

End Sub

Sub Form_Load ()

  txtUserName.Text = ""
  txtPasswd.Text = ""

  NewUser = False

  OpenPwdDb
  'ReadUserNames


End Sub

Sub GetUserInfo (SupplyTb As Table, ByVal UserName As String)

  Debug.Print "Valid user, looking in directory"

  Dim UserFiles As String

  SupplyTb.Index = "PrimaryKey"
  SupplyTb.Seek "=", UserName
  UserFiles = SupplyTb.Fields("Files")

  Debug.Print UserFiles

End Sub

Sub OpenPwdDb ()

  'SetDefaultWorkspace "Admin", "gears"

  Set PwdDb = OpenDatabase(EDICSPWDDB)

End Sub

Sub ReadUserNames (SupplyTb As Table)



End Sub

Sub txtNewVerify_KeyPress (KeyAscii As Integer)

  If KeyAscii = 13 Then
    KeyAscii = 0
    If txtPasswd.Text = txtNewVerify.Text Then
      AddPwd PwdDb.OpenTable("Password Table"), txtUserName.Text, txtPasswd.Text
```

```
        cmdOk_Click 0
      Else
        Beep
      End If
    End If

End Sub

Sub txtPasswd_KeyPress (KeyAscii As Integer)

    If txtUserName <> "" Then
      cmdOk(0).Enabled = True
    End If

    If KeyAscii = 13 And txtPasswd.Text <> "" And txtUserName.Text <> "" Then
      KeyAscii = 0
      cmdOk_Click 0
    End If

End Sub

Sub txtUserName_KeyPress (KeyAscii As Integer)

    If txtPasswd.Text <> "" Then
      cmdOk(0).Enabled = True
    End If

    If KeyAscii = 13 And txtUserName.Text <> "" Then
      KeyAscii = 0
      txtPasswd.SetFocus
    End If

End Sub

Function ValidPwd (SupplyTb As Table, ByVal UserName As String, ByVal Password As String) As
Integer

    SupplyTb.Index = "PrimaryKey"
    SupplyTb.Seek "=", UserName

    If SupplyTb.Fields("Password") = Password Then
      ValidPwd = True
    Else
      ValidPwd = False
    End If

    SupplyTb.Close

End Function

Function ValidUser (SupplyTb As Table, ByVal UserName As String) As Integer

    SupplyTb.Index = "PrimaryKey"
    SupplyTb.Seek "=", UserName
```

```
    ValidUser = Not SupplyTb.NoMatch

    SupplyTb.Close

End Function
```

## FRMMN3.FRM CODE

```
Option Explicit

Const DROPINT = 0
Const FileName = 1
Const FinalX = 2
Const FinalY = 3
Const OBJHEIGHT = 4
Const INITX = 5
Const INITY = 6
Const MOVEINT = 7
Const TRANSTYPE = 8
Const OBJTIME = 9
Const OBJWIDTH = 10
Const MAXTITLES = 11
Const BRANCH_PIC = 4
Const BRANCH_JUMPTO = 5
Const BRANCH_LOADOBJ = 6
Const MAXCACHE = 10

Dim ListBoxTitles(MAXTITLES) As String
Dim ObjectText(MAXTITLES) As String
Dim CachedList(MAXCACHE) As CachedObj
Dim LinkIndex As Integer
Dim DragX As Single, DragY As Single
Dim CurX As Single, CurY As Single
Dim PageCtr As Integer
Dim LabelCtr As Integer, TextCtr As Integer, GraphicCtr As Integer
Dim InputCtr As Integer, VideoCtr As Integer, AudioCtr As Integer, CommandCtr As Integer
Dim MetafileCtr As Integer, BranchCtr As Integer
Dim LinkPtr As Integer
Dim CurText As Integer, CurGraphic As Integer
Dim NumObjects As Integer, NewObjects As Integer
Dim NextPage As Integer
Dim CurVoice As String
Dim Active As Control
Dim PageDb As Database
Dim SectionDs As Dynaset
Dim ObjectTermDb As Database, IndexDb As Database
Dim ObjectTermTb As Table, IndexTb As Table, TopicTb As Table
Dim TextTb As Table
Dim AllPages() As String
Dim AllTopics() As TopicInfo
Dim ObjectsOnPage() As ObjectInfo
Dim UserIndexList() As IndexList
Dim UserIndexItems() As String, UserCtr As Integer
```

```
Dim IndexPtr As Integer, UserIndex As Integer
Dim SoundDone As Integer, VideoDone As Integer
Dim WavePlayer As MMControl
Dim UserTool As Integer
Dim CurrentBookmark As String
Dim SavedText As String
Dim ObjectMap() As Integer
Dim PageCaption As String
Dim hCurOpen As Integer, hCurClose As Integer

Sub AddNewItem (ByVal ObjectName As String, OutputCombo As ComboBox)

   Dim ObjType As String

   ObjType = Left$(ObjectName, 1)
   OutputCombo.AddItem UnparseObjName(ObjType) + "   " + ObjectName

End Sub

Sub BuildMap (AllTopics() As TopicInfo)

   ' modifies  frmMap
   ' effects   Builds a map in frmMap, based on AllTopics

   Dim Ctr As Integer
   Dim Page As String

   For Ctr = 1 To UBound(AllTopics)
         Page = AllTopics(Ctr).Topic
         frmMap!outlineMap.AddItem Page
   Next Ctr

End Sub

Sub CacheObject (CacheList() As CachedObj, Pointer As Integer, ByVal ObjectName As String, ByVal
LinkIndex As Integer, ByVal SourceType As String, ByVal SourceIndex As Integer)

   If Pointer > MAXCACHE Then
         Pointer = 0
   End If

   CacheList(Pointer).Name = ObjectName
   CacheList(Pointer).Index = LinkIndex
   CacheList(Pointer).SourceType = SourceType
   CacheList(Pointer).SourceIndex = SourceIndex

End Sub

Sub cboIndexNames_Click ()

   Dim IndexItems As String

   IndexItems = ItemList(UserIndexList(), cboIndexNames.Text)
```

```
    lstIndexSet.Clear
    PutItems lstIndexSet, IndexItems, INDEXDELIMITER

End Sub

Sub cboIndexNames_DblClick ()

    txtLink.Text = cboIndexNames.Text

End Sub

Sub cboObjects_Click ()

    lstObjectProp(cboObjects.ListIndex).Visible = True

End Sub

Sub cboObjectTerms_DblClick ()

    ' effects   Clicks the definiton button.

    cmdDefinition_Click

End Sub

Sub ClearAllObjects (ByVal OldLabels As Integer, ByVal OldGraphics As Integer, ByVal OldText As
Integer, ByVal OldVideo As Integer, ByVal OldAudio As Integer, ByVal OldInput As Integer, ByVal
OldCommand As Integer, ByVal OldMetafile As Integer)


    ' effects   Unloads all labels, graphics, text, video, audio, and user input objects.
    '         A 'NotLoaded' Error occurs if an object is not present on screen.  This error
    '         occurs if the user presses and navigating buttons before the entire screen is
    '         drawn.

    ' An "Object Not Loaded" error will occur when and Unload statement is called on
    ' an object that is not present on the screen.
    On Error GoTo NotLoaded

    Dim Ctr As Integer
    Dim ObjName As String

    ' Clear all label objects
    For Ctr = 1 To OldLabels
        Unload lblPage(Ctr)
        ObjName = "l"
    Next Ctr

    ' Clear all graphics objects
    For Ctr = 1 To OldGraphics
        Unload picAny(Ctr)
        ObjName = "s"
    Next Ctr
```

```
    ' Clear all text objects
    For Ctr = 1 To OldText
        Unload lblAny(Ctr)
        ObjName = "t"
    Next Ctr

    ' Clear all video players and video windows
    For Ctr = 1 To OldVideo
        Unload mmcPlayer(Ctr)
        Unload picVideo(Ctr)
        ObjName = "v"
    Next Ctr

    ' Clear all audio and command pictures, if in Editor
    If UserTool = EDITOR Then
        For Ctr = 1 To OldAudio
          Unload picAudio(Ctr)
          ObjName = "a"
        Next Ctr
        For Ctr = 1 To OldCommand
          Unload picCommand(Ctr)
          ObjName = "c"
        Next Ctr
    End If

    For Ctr = 1 To OldInput
        Unload txtUserInput(Ctr)
        ObjName = "u"
    Next Ctr

    For Ctr = 1 To OldMetafile
        Unload picMetafile(Ctr)
        ObjName = "m"
    Next Ctr

NotLoaded:

    Debug.Print "Not Loaded: " & ObjName & "--" & Ctr & " on page " & lblCurPage.Caption
    Debug.Print Error
    Resume Next

End Sub

Sub cmdArranger_Click (Index As Integer)

    ' modifies  NewObjects
    ' effects   Peforms one of several commands availble in the Editor:
    '           NewText -- creates a new text object,
    '           NewGraphic -- creates a new graphic (still) object.
    '           NewMetafile -- creates a new metafile object.
    '           NewCommand -- loads the special command object.
    '           LoadBackground -- loads a picture into the background.
    '           NewVideo -- creates a new video object.
    '           NewAudio -- creates a new audio object.
```

```
'       NewLabel -- creates a new label object.
'       NewInput -- creates a new input object.
'       SavePage -- saves all the objects on the current page to the database.

  If Index <> PAGESAVE Then
        NewObjects = NewObjects + 1
  End If

  Select Case Index
        Case TEXTTOOL                   ' Text Button
          NewText
        Case GRAPHIC                    ' Graphic Button
          NewGraphic
        Case METAFILE                   ' Load Metafile
          NewMetafile
        Case COMMANDBUTTON                      ' Special Command
          NewCommand
        Case VIDEO                      ' Video Button
          NewVideo
        Case AUDIO                      ' Audio Button
          NewAudio
        Case Label                      ' Label Button
          NewLabel
        Case USERINPUT                  ' User Input Button
          NewInput
        Case PAGESAVE                   ' Save Page Button
          SavePage
  End Select

End Sub

Sub cmdCreateIndex_Click (Index As Integer)

  Dim Response As Integer

  Dim AppName As String, Section As String, Entry As String, FileName As String
  Dim ParentPtr As Integer, ChildPtr As Integer
  Static EntryPtr As Integer
  Static InBranch As Integer

  Select Case Index
        Case 0
          picPage(0).Width = 449
          cmdCreateIndex(7).Enabled = True
        Case 1
          If lstPages.Text <> "" Then
                lstIndexSet.AddItem lstPages.Text
          End If
        Case 2
          If lstIndexSet.ListIndex >= 0 Then
                lstIndexSet.RemoveItem lstIndexSet.ListIndex
          End If
        Case 3
          If cboIndexNames.Text = "" Then
```

```
                MsgBox "Enter a Name for this Index Set", MB_OK + MB_APPLMODAL, "Create
Index Set"
            Else
                CreateIndexSet UserIndexList(), cboIndexNames.Text, lstIndexSet
                If Not (Member(cboIndexNames.Text, cboIndexNames)) Then
                  cboIndexNames.AddItem cboIndexNames.Text
                End If
                AppName = "Index Items"
                Section = UserIndexList(IndexPtr).Name
                Entry = UserIndexList(IndexPtr).List
                FileName = "c:\edics\users\" + UserName + "\" + UserName + ".ini"
                Response = WritePrivateProfileString(AppName, ByVal Section, ByVal Entry,
FileName)
                If Response Then
                  'MsgBox UserName + ".ini updated successfully.", MB_OK + MB_APPLMODAL,
"Index Set"
                Else
                  'MsgBox UserName + ".ini update failed.", MB_OK + MB_APPLMODAL, "Index
Set"
                End If
            End If
          Case 4
            lstIndexSet.Clear
          Case 5
            'If UserIndex Then
                ' an index set is already being run: change re-entry point
            '  InBranch = True
                'PageCtr = FindPageNo(AllPages, UserIndexItems(UserCtr))
            'Else
            '  EntryPtr = IndexPtr
            'End If
            UserCtr = 0
            UserIndex = True
            mnuPageItem(6).Enabled = UserIndex
            cmdNavigate(3).Visible = True
            RunIndexSet IndexPtr
            KeyPreview = True
          Case 6
            cmdCreateIndex(7).Enabled = False
            picPage(0).Visible = False
            lstObjects.Visible = False
            lblObjects.Visible = False
            cmdCreateIndex(0).Visible = True
            cmdCreateIndex(5).Visible = True
            txtLink.Visible = False
            lblNewObj.Visible = False
            lblLinkLabel.Visible = False
            picDropNew.Visible = False
            lstPages.Tag = 0
            KeyPreview = True
          Case 7
            'If InBranch Then
                ' return to branch point, not trunk
            '  IndexPtr = IndexPtr - 1
```

```
            '   If IndexPtr <= EntryPtr Then
            '      InBranch = False
            '      cmdCreateIndex_Click 7
            '   Else
            '      cmdCreateIndex_Click 5
            '   End If
            'Else
                  ' return to trunk
                  UserIndex = False
                  InBranch = False
                  mnuPageItem(6).Enabled = UserIndex
                  picPage(0).Visible = False
                  Caption = DatabaseSection
                  KeyPreview = True
                  cmdNavigate_Click 2
            'End If
   End Select

End Sub

Sub cmdCtlBox_Click (Index As Integer)

   Select Case Index
         Case 0
            PopupMenu mnuCtlBox, POPUPMENU_LEFTALIGN, picProperties(0).Left,
picProperties(0).Top + cmdCtlBox(0).Height
         Case 1
            PopupMenu mnuCtlBox, POPUPMENU_LEFTALIGN, picVolume(0).Left,
picVolume(0).Top + cmdCtlBox(1).Height
         Case 2
            PopupMenu mnuCtlBox, POPUPMENU_LEFTALIGN, picSpecialObj(0).Left,
picSpecialObj(0).Top + cmdCtlBox(2).Height
   End Select

End Sub

Sub cmdDefinition_Click ()

   ' effects   Shows the definition window, if the text
   '         in the ObjectTerm combo box is not empty.

   If cboObjectTerms.Text <> "" Then
         CurWord = cboObjectTerms.Text
         Load frmDict
         If CurWord = "0" Then frmDict.Show 1
         If CurWord = "1" Then Unload frmDict
   End If

End Sub

Sub cmdInterface_Click (Index As Integer)

   ' effects   Performs one of several actions, based on the value of Index.
   '         0 -- Starts the Edics Index
```

```
'       1 -- Shows the map
'       2 -- Opens the Notepad
'       3 -- Opens Edics Help
'       4 -- Request to return to main menu
'       5 -- Request to quit program.

Dim Temp As Integer, Response As Integer
Dim Message As String

Select Case Index
        Case 0                  ' Starts Index
          frmDicSearch.lstSearch.ListIndex = 0
          frmDicSearch.txtSearch.Text = ""
          frmDicSearch.Show 1
          If CurWord = "" Then
                Exit Sub
          End If
          Load frmDict
          If CurWord = "0" Then
                frmDict.Show
          End If
          If CurWord = "1" Then
                Unload frmDict
          End If
        Case 1                  ' Opens the Map
          frmMap.Show
        Case 2                  ' Starts the Notepad
          mnuNotepadItem_Click 0
          'Interactive = "i1"
          'frmInter.Show
        Case 3                  ' Opens the Edics Help
          Temp = WinHelp(hWnd, "c:\windows\winhelp.hlp", HELP_HELPONHELP, "help")
        Case 4
          If UserTool = EDITOR Then
                Message = MSG_MAIN_EDITOR
          Else
                Message = MSG_MAIN_PLAYBACK
          End If
          Response = MsgBox(Message, MB_OKCANCEL + MB_ICONQUESTION +
MB_APPLMODAL, TITLE_MAIN)
          If Response = IDOK Then
                ClearAllObjects LabelCtr, GraphicCtr, TextCtr, VideoCtr, AudioCtr, InputCtr,
CommandCtr, MetafileCtr
          ' Hide
          frmMainMenu.Show
          End If
        Case 5
          ' Display Quit message box, returns value of button pressed
          Response = MsgBox(MSG_QUIT, MB_OKCANCEL + MB_ICONQUESTION +
MB_APPLMODAL, TITLE_QUIT)

          If Response = IDOK Then
                Unload Me
                End
```

```
            End If
    End Select

End Sub

Sub cmdMin_Click (Index As Integer)

    Select Case Index
            Case 0
                picProperties(0).Visible = False
            Case 1
                picVolume(0).Visible = False
            Case 2
                picSpecialObj(0).Visible = False
    End Select

End Sub

Sub cmdNavigate_Click (Index As Integer)

    ' modifies  frmPlayer
    ' effects   Performs navigation of screens, based on the value of Index.
    '         0 -- Moves back to previous page in Page Database
    '              and prints the page.
    '         1 -- Moves to the next page in the Page Database and
    '              prints the page.
    '         2 -- Prints the current page again.

    Select Case Index
            Case 0                  ' Back
                If UserIndex Then
                        If UserCtr > 1 Then
                            'ClearAllObjects LabelCtr, GraphicCtr, TextCtr, VideoCtr, AudioCtr, InputCtr,
CommandCtr, MetafileCtr
                            UserCtr = UserCtr - 1
                            MainScreenPrint UserIndexItems(UserCtr)
                        End If
                ElseIf PageCtr > 1 Then
                        'ClearAllObjects LabelCtr, GraphicCtr, TextCtr, VideoCtr, AudioCtr, InputCtr,
CommandCtr, MetafileCtr
                        PageCtr = PageCtr - 1
                        MainScreenPrint AllPages(PageCtr)
                End If
            Case 1                  ' Forward
                If UserIndex Then
                        If UserCtr < UBound(UserIndexItems) Then
                            'ClearAllObjects LabelCtr, GraphicCtr, TextCtr, VideoCtr, AudioCtr, InputCtr,
CommandCtr, MetafileCtr
                            UserCtr = UserCtr + 1
                            MainScreenPrint UserIndexItems(UserCtr)
                        End If
                ElseIf PageCtr < UBound(AllPages) Then
                        'ClearAllObjects LabelCtr, GraphicCtr, TextCtr, VideoCtr, AudioCtr, InputCtr,
CommandCtr, MetafileCtr
```

```
                PageCtr = PageCtr + 1
                MainScreenPrint AllPages(PageCtr)
            End If
        Case 2              ' Repeat
            'ClearAllObjects LabelCtr, GraphicCtr, TextCtr, VideoCtr, AudioCtr, InputCtr, CommandCtr,
MetafileCtr
            MainScreenPrint AllPages(PageCtr)
        Case 3
            cmdCreateIndex_Click 7
            cmdNavigate(3).Visible = False
    End Select

End Sub

Sub cmdObj_Click (Index As Integer)

    Dim Ctr As Integer

    Select Case Index
        Case 0                  ' Add
            If lstAllObjects.Text <> "" Then
                lstClearObjects.AddItem lstAllObjects.Text
            End If
        Case 1                  ' Add All
            For Ctr = 0 To lstAllObjects.ListCount - 1
                lstClearObjects.AddItem lstAllObjects.List(Ctr)
            Next Ctr
            cmdObj(Index).Enabled = False
        Case 2                  ' Remove
            If lstClearObjects.ListIndex > -1 Then
                lstClearObjects.RemoveItem lstClearObjects.ListIndex
            End If
        Case 3                  ' Accept
            picObj.Visible = False
            If cmdObj(1).Enabled Then
                ObjectsOnPage(Active.Tag).Transition = OBJECT_CLEARSOME
                ParseCommandAttribute lstClearObjects, ObjectsOnPage(Active.Tag).Transition
            Else
                ObjectsOnPage(Active.Tag).Transition = OBJECT_CLEARALL
            End If
        Case 4                  ' Close
            picObj.Visible = False
    End Select

End Sub

Sub cmdSound_Click ()

    ' effects   Turns the sound on or off, depending on the icon -- if eye, sound is off,
    '           if ear, sound is on.

    Dim Sound As Integer

    If cmdSound.Value = FRAME_EYE Then
```

```
        mnuOptionsItem(0).Checked = False
        cmdSound.Caption = "Sound Off"
        mmcSound.Command = "Stop"
        frmTranscription.Show
    Else
        mnuOptionsItem(0).Checked = True
        cmdSound.Caption = "Sound On"
        mmcSound.Command = "Play"
        SoundDone = False
        frmTranscription.Hide
    End If

End Sub

Sub cmdSpecial_Click (Index As Integer)

    Select Case Index
        Case 0                  ' Browse database
        Case 1                  ' Close Window
            picSpecialObj(0).Visible = True
    End Select

End Sub

Sub cmdVolume_Click (Index As Integer)

    Dim NullVal As Integer

    Select Case Index
        Case 1
            mmcSound.Command = "Close"
            mmcSound.Command = "Open"
            mmcSound.FileName = "tada.wav"
            mmcSound.Command = "Play"
            NullVal = sndPlaySound("tada.wav", 1)
        Case 2
            cmdMin_Click 2
    End Select

End Sub

Sub CreateIndexItems (OutputArray() As String, ByVal IndexList As String)

    ' modifies  OutputArray
    ' effects   Creates an list of the items in IndexList and places it in the array
    '           OutputArray.

    Dim Ctr As Integer
    Dim In As String

    Ctr = 0

    ReDim OutputArray(Ctr)
```

```
    Do Until Len(IndexList) < 2
        In = Pop(IndexList, INDEXDELIMITER)
        Ctr = Ctr + 1
        ReDim Preserve OutputArray(Ctr)
        OutputArray(Ctr) = Left$(In, Len(In) - 1)
    Loop

End Sub

Sub CreateIndexSet (OutputArray() As IndexList, ByVal ItemName As String, InputList As ListBox)

    ' modifies  OutputArray
    ' effects   Creates an index list consisting of the items in the InputList ListBox.
    '           The index list is given the name ItemName added to the array OutputArray.

    ' Increase OutputArray by one
    IndexPtr = IndexPtr + 1
    ReDim Preserve OutputArray(IndexPtr)          ' keep all previous index lists

    ' Add the items to the OutputArray
    OutputArray(IndexPtr).Name = ItemName
    OutputArray(IndexPtr).List = GetItems(InputList, INDEXDELIMITER)

End Sub

Sub CreateListBoxTitles ()

  ListBoxTitles(DROPINT) = "Drop Interval" + Chr$(9) + Chr$(9)
  ListBoxTitles(FileName) = "Filename" + Chr$(9) + Chr$(9)
  ListBoxTitles(FinalX) = "Final X" + Chr$(9) + Chr$(9)
  ListBoxTitles(FinalY) = "Final Y" + Chr$(9) + Chr$(9)
  ListBoxTitles(OBJHEIGHT) = "Height" + Chr$(9) + Chr$(9)
  ListBoxTitles(INITX) = "Initial X" + Chr$(9) + Chr$(9)
  ListBoxTitles(INITY) = "Initial Y" + Chr$(9) + Chr$(9)
  ListBoxTitles(MOVEINT) = "Move Interval" + Chr$(9) + Chr$(9)
  ListBoxTitles(TRANSTYPE) = "Transition Type" + Chr$(9) + Chr$(9)
  ListBoxTitles(OBJTIME) = "Time" + Chr$(9) + Chr$(9)
  ListBoxTitles(OBJWIDTH) = "Width" + Chr$(9) + Chr$(9)

End Sub

Sub CreateObjectsArray (PageDs As Dynaset, Page As String)

    ' modifies  NumObjects, ObjectsOnPage, ObjectMap
    ' effects   For each Record in the PageDs, creates an object and creates a mapping to the
    '           object.  Obtains all object terms for each object.

    Dim ID As Integer, TimeOnScreen As Integer, Sequence As Integer
    Dim CurObject As String, Link As String
    Dim ObjectHeight As Integer, ObjectWidth As Integer
    Dim InitialX As Integer, InitialY As Integer, FinalX As Integer, FinalY As Integer
    Dim TransitionCode As String, TransitionType As String, TransitionInfo As Integer
    Dim PageObject As ObjectInfo
```

```
  Do Until PageDs.EOF          ' do until no more objects associated with Page

        ' Increase current page array for each object
        NumObjects = NumObjects + 1
        ReDim Preserve ObjectsOnPage(NumObjects)
        ReDim Preserve ObjectMap(NumObjects)

        ' get object information
        ID = PageDs.Fields("ID")
        CurObject = PageDs.Fields("Object")
        ObjectHeight = PageDs.Fields("Height")
        ObjectWidth = PageDs.Fields("Width")
        InitialX = PageDs.Fields("Initial X")
        InitialY = PageDs.Fields("Initial Y")
        FinalX = PageDs.Fields("Final X")
        FinalY = PageDs.Fields("Final Y")
        Sequence = PageDs.Fields("Sequence")
        TransitionCode = PageDs.Fields("Transition Code")
        TimeOnScreen = PageDs.Fields("Transition Attribute")
        Link = PageDs.Fields("Link")

        ' make an object of type ObjectInfo and add it to the current page array
        MakeObject PageObject, ID, DatabaseSection, Page, InitialX, InitialY, FinalX, FinalY,
Sequence, TransitionCode, TimeOnScreen, ObjectHeight, ObjectWidth, "", CurObject, "", Link
        ObjectsOnPage(NumObjects) = PageObject

        ' get all object terms
        GetObjectTerms CurObject

        PageDs.MoveNext

  Loop

End Sub

Sub DelayOff ()

  ' modifies  tmrWait
  ' effects   Disables the timer if the prescribed delay period is over, i.e. ReturnValue
  '           is true.

  If ReturnValue Then
        tmrWait.Enabled = False
  End If

End Sub

Sub DelayOn (ByVal DelayTime)

  ' modifies  tmrWait
  ' effects   Enables tmrWait to the interval DelayTime

  tmrWait.Enabled = False
  tmrWait.Interval = DelayTime
```

```
    tmrWait.Enabled = True

End Sub

Sub DeleteObject (Source As Control)

    ' modifies  Me, PageDb
    ' effects   Removes the object from the database corresponding to Source.

    ObjectsOnPage(Source.Tag).Contents = DELETED
    UpdateObject ObjectsOnPage(Source.Tag), PageDb.OpenTable("Page Database")
    Unload picAny(Source.Index)
    GraphicCtr = GraphicCtr - 1
    NumObjects = NumObjects - 1

End Sub

Sub DialogBox (ByVal DialogType As Integer)

    ' effects   Displays a dialog box, based on DialogType, where DIALOG_WIDTH and
DIALOG_HEIGHT
    '         denote setting the dimensions of the Active object, DIALOG_SETTIME denotes
    '         setting the time for the Active object, and DIALOG_SCALE and DIALOG_ZOOM
    '         denote setting visual effects for the Active Object (stills only).


    Select Case DialogType
        Case DIALOG_HEIGHT
            frmDialog.lblDialog.Caption = "Height"
            frmDialog.txtDimension = Active.Height
        Case DIALOG_WIDTH
            frmDialog.lblDialog.Caption = "Width"
            frmDialog.txtDimension = Active.Width
            DialogType = DialogType - 1
        Case DIALOG_SETTIME
            frmDialog.lblDialog.Caption = "Time"
            frmDialog.txtDimension = ObjectsOnPage(Active.Tag).TimeOnScreen
            DialogType = DialogType - 2
        Case DIALOG_SETJUMP
            frmDialog.lblDialog.Caption = "Jump Interval"
            frmDialog.txtDimension = Sensitivity
            DialogType = DialogType - 3
    End Select

    ' Display the DialogBox
    frmDialog.frameDialog(DialogType).Visible = True
    frmDialog.Show MODAL

End Sub

Function DisplayObject (PageObject As ObjectInfo, ByVal ObjectCtr As Integer)

    On Error GoTo PrintError:
```

```
Dim ID As Integer, RetIndex As Integer
Dim Initial As Coords, Final As Coords
Dim InitialX As Integer, InitialY As Integer, FinalX As Integer, FinalY As Integer
Dim IntervalX As Integer, IntervalY As Integer, Position As Integer
Dim deltaX As Integer, deltaY As Integer
Dim ObjectHeight As Integer, ObjectWidth As Integer
Dim TransitionCode As String, TransitionType As String, TransitionInfo As Integer
Dim ObjectType, CurObject As String, Link As String, FileName As String
Dim Sequence As Integer, TimeOnScreen As Integer
Dim Object As Control, Player As MMControl
Dim TransFile As String

ID = PageObject.ID
CurObject = PageObject.Contents
ObjectHeight = PageObject.Height
ObjectWidth = PageObject.Width
InitialX = PageObject.Initial.Left
InitialY = PageObject.Initial.Top
FinalX = PageObject.Final.Left
FinalY = PageObject.Final.Top
Sequence = PageObject.Sequence
TransitionCode = PageObject.Transition
TimeOnScreen = PageObject.TimeOnScreen
Link = PageObject.Link

ObjectType = UCase(Left$(CurObject, 1))

Select Case ObjectType

    Case LABEL_CODE

        LabelCtr = LabelCtr + 1
        RetIndex = LabelCtr
        ObjectMap(ObjectCtr) = LabelCtr

        Set Object = lblPage(LabelCtr)
        Load Object

        Object.Tag = ObjectCtr

        Screen.MousePointer = 11
        FormatLabel lblPage(LabelCtr), TransitionCode, TimeOnScreen
        Object.Caption = Right$(CurObject, Len(CurObject) - 1)
        Object.Top = InitialY
        Object.Left = InitialX
        Object.Visible = True
        DoEvents
        Screen.MousePointer = 0

Case COMMAND_CODE

        CommandCtr = CommandCtr + 1
        RetIndex = CommandCtr
        ObjectMap(ObjectCtr) = CommandCtr
```

```
        If UserTool = EDITOR Then
            Set Object = picCommand(CommandCtr)
            Load Object
            Object.Top = 64 + 50 * (CommandCtr - 1)
            Object.Left = 16
            Object.Visible = True
            Object.Tag = ObjectCtr
        End If

        Object.Visible = True
        If Not SoundDone Or Not VideoDone Then
            Do
              DoEvents
            Loop Until SoundDone And VideoDone
        End If

        Select Case UCase(Left$(TransitionCode, 1))
              Case OBJECT_CLEARALL
                    If UserTool = EDITOR Then
                      HideAllObjects LabelCtr, GraphicCtr, TextCtr, VideoCtr, 0, InputCtr,
MetafileCtr
                    Else
                      ClearAllObjects LabelCtr, GraphicCtr, TextCtr, VideoCtr, 0, InputCtr,
CommandCtr, MetafileCtr
                    End If
              Case OBJECT_CLEARSOME
                    HideSomeObjects Right$(TransitionCode, Len(TransitionCode) - 1)
              Case OBJECT_NEXTPAGE
                    cmdNavigate_Click 1
              Case OBJECT_SPECIAL
                    CurObject = Right$(CurObject, Len(CurObject) - 1)
                    Select Case UCase(Left$(CurObject, 1))
                      Case OBJ_INTERACTIVE
                            'Dim Interact As New frmInter
                            Interactive = CurObject
                            'Load frmInter
                            WindowState = 1
                            frmInter.Show
                            frmInter.SetFocus
                      Case OBJ_FIND
                            'Dim Find As New frmFindThe_
                            Problem = CurObject
                            WindowState = 1
                            'Find.Tag = CurObject
                            Load frmFindThe_
                            'frmFindThe_.Show
                    End Select
        End Select

        Case AUDIO_CODE

            AudioCtr = AudioCtr + 1
            RetIndex = AudioCtr
```

```
        ObjectMap(ObjectCtr) = AudioCtr

        If Not SoundDone Then
          Do
                DoEvents
          Loop Until SoundDone
        End If

        'Set WavePlayer = mmcSound

        If UserTool = EDITOR Then
          Set Object = picAudio(AudioCtr)
          Load Object
          Object.Top = 40
          Object.Left = 16 + 50 * (AudioCtr - 1)
          Object.Visible = True
          Object.Tag = ObjectCtr
        End If

        Screen.MousePointer = 11
        CurVoice = AUDIOPATH & CurObject & FILE_WAVE
        ObjectsOnPage(ObjectCtr).Path = AUDIOPATH
        ObjectsOnPage(ObjectCtr).Extension = FILE_WAVE
        ObjectsOnPage(ObjectCtr).Contents = CurObject

        'Load Player
        mmcSound.FileName = AUDIOPATH & CurObject & FILE_WAVE
        mmcSound.Notify = True
        mmcSound.Command = "Close"
        mmcSound.Notify = True
        mmcSound.Command = "Open"
        Screen.MousePointer = 0

        If cmdSound = FRAME_EAR Then
          SoundDone = False
          mmcSound.Command = "Play"
        Else
          SoundDone = True
        End If

        TransFile = Mid$(CurObject, 2, Len(CurObject))
        TransFileReadAndPrint TransFile, TextTb
        NextPage = False

        ' Display Transcription Window if necessary
        If cmdSound = FRAME_EYE And mnuViewItem(0).Checked Then
          frmTranscription.Show
        End If

Case STILL_CODE

        GraphicCtr = GraphicCtr + 1
        RetIndex = GraphicCtr
        ObjectMap(ObjectCtr) = GraphicCtr
```

```
        Set Object = picAny(GraphicCtr)
        Load Object

        Screen.MousePointer = 11
        Object.Image = STILLPATH & CurObject & FILE_STILL
        Object.Tag = ObjectCtr
        ObjectsOnPage(ObjectCtr).Path = STILLPATH
        ObjectsOnPage(ObjectCtr).Extension = FILE_STILL
        ObjectsOnPage(ObjectCtr).Contents = CurObject

        If ObjectHeight = 0 Then
           Object.Height = Object.ImageHeight
        Else
           Object.Height = ObjectHeight
        End If

        If ObjectWidth = 0 Then
           Object.Width = Object.ImageWidth
        Else
           Object.Width = ObjectWidth
        End If

        Object.Top = InitialY
        Object.Left = InitialX
        Object.Visible = True
        Screen.MousePointer = 0

Case META_CODE

        MetafileCtr = MetafileCtr + 1
        RetIndex = MetafileCtr
        ObjectMap(ObjectCtr) = MetafileCtr

        Set Object = picMetafile(MetafileCtr)
        Load Object

        Screen.MousePointer = 11
        Object.Picture = LoadPicture(STILLPATH & CurObject & FILE_METAFILE)
        Object.Tag = ObjectCtr
        ObjectsOnPage(ObjectCtr).Path = STILLPATH
        ObjectsOnPage(ObjectCtr).Extension = FILE_METAFILE
        ObjectsOnPage(ObjectCtr).Contents = CurObject

        Object.Top = InitialY
        Object.Left = InitialX
        Object.Visible = True
        Screen.MousePointer = 0

Case SCREENTEXT_CODE, TRANSCRIPTION_CODE

        TextCtr = TextCtr + 1
        RetIndex = TextCtr
        ObjectMap(ObjectCtr) = TextCtr
```

```
        Set Object = lblAny(TextCtr)
        Load Object

        Object.Height = ObjectHeight
        Object.Width = ObjectWidth
        Object.Tag = ObjectCtr
        ObjectsOnPage(ObjectCtr).Path = TEXTPATH
        ObjectsOnPage(ObjectCtr).Extension = FILE_TEXT
        ObjectsOnPage(ObjectCtr).Contents = CurObject

        Object.Top = InitialY
        Object.Left = InitialX
        Object.Visible = True

        FileName$ = TEXTPATH & CurObject & FILE_TEXT
        PrettyPrint lblAny(TextCtr), cboObjectTerms, CurObject, TextTb

Case VIDEO_CODE

        VideoCtr = VideoCtr + 1
        RetIndex = VideoCtr
        ObjectMap(ObjectCtr) = VideoCtr

        If Not VideoDone Then
          Do
                DoEvents
          Loop Until VideoDone
        End If

        Set Player = mmcPlayer(VideoCtr)
        Load Player
        Set Object = picVideo(VideoCtr)
        Load Object

        Screen.MousePointer = 11
        Object.Height = ObjectHeight
        Object.Width = ObjectWidth
        Object.Tag = ObjectCtr
        ObjectsOnPage(ObjectCtr).Path = VIDEOPATH
        ObjectsOnPage(ObjectCtr).Extension = FILE_VIDEO
        ObjectsOnPage(ObjectCtr).Contents = CurObject

        'If UserTool = EDITOR Then
          Object.Top = InitialY
          Object.Left = InitialX
          Object.Visible = True
        'End If

        Player.Notify = True
        Player.Top = Object.Top + ObjectHeight
        Player.Left = Object.Left
        Player.Width = Object.Width
```

```
                Player.Command = "Close"

                'If TransitionCode = OBJECT_DROP Then
                '   Player.Wait = True
                'Else
                '   Player.Wait = False
                'End If

                Player.hWndDisplay = Object.hWnd
                Player.FileName = VIDEOPATH & CurObject & FILE_VIDEO
                Player.Command = "Open"

                Screen.MousePointer = 0
                If cmdSound = FRAME_EAR Then
                  Player.Silent = False
                Else
                  Player.Silent = True
                End If

                Player.Notify = True
                TransitionCode = Left$(TransitionCode, 1)

                If TransitionCode = OBJECT_LOOP Then
                  Player.HelpContextID = MNUINFOTRANS_LOOP
                End If

                If TransitionCode = OBJECT_WAIT Or TransitionCode = OBJECT_DROP Then
                  VideoDone = False
                Else
                  VideoDone = True
                End If

                'mmcPlayer_StepClick VideoCtr, True
                'mmcPlayer_StepClick VideoCtr, True

                Player.Command = "Play"
                Player.Visible = True

                'If UserTool = PLAYBACK Then
                '   VideohWnd = FindWindowByString(ByVal "AviWnd", ByVal (CurObject + ".avi"))
                '   NewVal = SetWindowLong(VideohWnd, GWL_STYLE, ByVal WS_Video)
                '   NullVal = SetWindowPos(VideohWnd, HWND_TOPMOST, InitialY, InitialX +
OFFSETTOP, ObjectWidth, ObjectHeight, SWP_SHOWWINDOW)
                'End If

        Case INPUT_CODE

                InputCtr = InputCtr + 1
                RetIndex = InputCtr
                ObjectMap(ObjectCtr) = InputCtr

                Load txtUserInput(InputCtr)
                Set Object = txtUserInput(InputCtr)
```

```
        ' CurObject = right$(CurObject, Len(CurObject - 1))
        Object.Tag = ObjectCtr
        If ObjectsOnPage(ObjectCtr).Transition = "S" Then
          Object.Text = SavedText
        ElseIf ObjectsOnPage(ObjectCtr).Transition = "C" Then
          SavedText = ""
        End If
        'Object.Text = Right$(CurObject, Len(CurObject) - 1)
        Object.Top = InitialY
        Object.Left = InitialX
        Object.Visible = True

End Select

If ObjectCtr = LINKEDOBJ Then
  Object.ZOrder
  If ObjectType = VIDEO_CODE Then
      Player.ZOrder
  End If
End If

If Link = "none" And ObjectType <> STILL_CODE Then
  Object.MousePointer = 0
ElseIf ObjectType = STILL_CODE And Link = "none" Then
  Object.PrintSize = NOLINK
Else

End If

TransitionType = UCase(Left$(TransitionCode, 1))

If TransitionType = OBJECT_MOVE Or TransitionType = OBJECT_MOVEANDDROP Then

  TransitionInfo = Val(Right$(TransitionCode, Len(TransitionCode) - 1))

  MakeCoords Initial, InitialX, InitialY
  MakeCoords Final, FinalX, FinalY

  deltaX = Final.Left - Initial.Left
  deltaY = Final.Top - Initial.Top

  If deltaX <> 0 Then
      IntervalX = deltaX / TransitionInfo
      For Position = Initial.Left To Final.Left Step IntervalX
        Object.Left = Position
      Next Position
  End If

  If deltaY <> 0 Then
      IntervalY = deltaY / TransitionInfo
      For Position = Initial.Top To Final.Top Step IntervalY
        Object.Top = Position
      Next Position
  End If
```

```
        End If

        If TransitionType = OBJECT_DROP Or TransitionType = OBJECT_MOVEANDDROP Then
          If ObjectType = VIDEO_CODE Then
                Do
                  DoEvents
                  If Player.Mode = 524 Then
                        MsgBox "Video Could Not Play", MB_OK
                        VideoDone = True
                  End If
                Loop Until VideoDone
                Player.Visible = False
                Object.Visible = False
          Else
                DelayOn TimeOnScreen * 1000          ' convert to milliseconds
                Do
                  DoEvents
                Loop Until ReturnValue
                Object.Visible = False
                DelayOff
          End If
        End If

  DisplayObject = RetIndex

PrintError:
  'MsgBox "There is an error " & Err, MB_OK
  'If mmcPlayer(VideoCtr).Error Then
        'MsgBox "MCI Error: " & mmcPlayer(VideoCtr).ErrorMessage & " " &
mmcPlayer(VideoCtr).Error, MB_OK
  'End If
  Debug.Print Error
  Resume Next

End Function

Sub DisplayTransitionCode (ByVal ObjectType As String, ByVal Transition As String, OutputCombo As
ComboBox)

  ' modifies  OutputCombo
  ' effects   Takes the Transition Code

  OutputCombo.Text = UnparseTransCode(Transition)

End Sub

Sub FillLinks (Db As Database)

  Dim Tb As Table

  Set Tb = Db.OpenTable("Page Database")

  Tb.MoveFirst
```

```
    Do Until Tb.EOF
        Tb.Edit
        Tb.Fields("Link") = "none"
        Tb.MoveNext
    Loop
    'tb.Update
End Sub

Sub FillListBoxProperties (Index As Integer)

    Dim Ctr As Integer

    'For Ctr = 0 To MAXTITLES
    '   lstObjectProp(Index).AddItem ListBoxTitles(Ctr)
    'Next Ctr

End Sub

Sub FillListBoxValues (Index As Integer, CurObject As ObjectInfo)

    Dim Ctr As Integer

    ParseObjectInfo CurObject

    lstObjectProp(Index).Clear

    For Ctr = 0 To MAXTITLES
        lstObjectProp(Index).AddItem ListBoxTitles(Ctr) + ObjectText(Ctr), Ctr
    Next Ctr

End Sub

Function FindIndex (ObjectsArray() As ObjectInfo, ByVal ObjToClear As String) As Integer

    Dim Ctr As Integer

    For Ctr = 1 To UBound(ObjectsArray)
        If ObjectsArray(Ctr).Contents = ObjToClear Then
            FindIndex = Ctr
            Exit Function
        End If
    Next Ctr

    FindIndex = -1

End Function

Function FindIndexPtr (ByVal IndexName As String, InputCombo As ComboBox)

    Dim Ctr As Integer

    For Ctr = 0 To InputCombo.ListCount
        If IndexName = InputCombo.List(Ctr) Then
```

```
            FindIndexPtr = Ctr + 1
            Exit Function
        End If
    Next Ctr

End Function

Sub Form_DragDrop (Source As Control, X As Single, Y As Single)

    ' effects   Moves the Source control to the position denoted by X - DragX, Y - DragY.
    '           If Source is a video object, also move the player object to the new position.

    Source.Move (X - DragX), (Y - DragY)
    If Source Is picVideo(Source.Index) Then
        mmcPlayer(Source.Index).Top = Source.Top + Source.Height
        mmcPlayer(Source.Index).Left = Source.Left
    End If

End Sub

Sub Form_DragOver (Source As Control, X As Single, Y As Single, State As Integer)

    'lblLoc.Caption = X & ", " & Y

End Sub

Sub Form_KeyDown (KeyCode As Integer, Shift As Integer)

    Select Case KeyCode
        Case KEY_UP
            Active.Move Active.Left, Active.Top - Sensitivity
        Case KEY_DOWN
            Active.Move Active.Left, Active.Top + Sensitivity
        Case KEY_RIGHT
            Active.Move Active.Left + Sensitivity
        Case KEY_LEFT
            Active.Move Active.Left - Sensitivity
    End Select

End Sub

Sub Form_Load ()

    GraphicCtr = 0
    TextCtr = 0
    VideoCtr = 0
    LabelCtr = 0
    InputCtr = 0
    AudioCtr = 0
    MetafileCtr = 0
    CommandCtr = 0
    NewObjects = 0
    LinkPtr = -1
    PageCtr = 1
```

```
    SoundDone = True
    NextPage = True
    VideoDone = True
    UserIndex = False
    UserTool = TOOL
    Sensitivity = 10
    SavedText = ""

    Caption = DatabaseSection
    mnuUserName.Caption = UserName

    Show
    DoEvents

    'This part loads the search textbox with the term
    Load_SearchText

    OpenPageDb
    OpenObjectTermDb
    OpenIndexDb

    GetAllPages PageDb, AllPages(), AllTopics(), DatabaseSection, lstPages
    'GetAllPages PageDb, AllPages(), AllTopics(), DatabaseSection, frmDialogs!lstPages
    ReadIndexSets PageDb

    GetAllTopics PageDb, AllTopics()
    'BuildMap AllTopics()

    LoadCursorModule

    RaiseWindow frmTranscription.hWnd

    If UserTool = EDITOR Then
        StartEditor
    Else
        StartPlayback
    End If

End Sub

Sub Form_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    MousePointer = 0
    'lblShowLink.Visible = False
    'mnuPageLabel.Caption = PageCaption

End Sub

Sub Form_Unload (Cancel As Integer)

    SaveBookmark
    SectionDs.Close
    'PageDb.Close
    IndexTb.Close
```

```
    ObjectTermTb.Close
    frmTranscription.Hide

End Sub

Function FormatLink (LinkType As String, PageName As String, ObjectName As String, LinkName As
String)

    Dim Temp As String

    If LinkType = NAMELINK Then
        Temp = LinkType & CODEDELIMITERLEFT & LinkName & CODEDELIMITERRIGHT
    Else
        Temp = PAGELINK & CODEDELIMITERLEFT & PageName & CODEDELIMITERRIGHT
        If LinkType = ObjLink Then
            Temp = Temp & CODEDELIMITERLEFT & ObjectName & CODEDELIMITERRIGHT
        End If
    End If

    FormatLink = Temp

End Function

Sub GetAllObjects (ByVal Page As String, PageDs As Dynaset, SectionDs As Dynaset)

    SectionDs.Filter = "[Page Code] LIKE '" & Page & "'"
    SectionDs.Sort = "[Sequence]"
    'Set PageDs = SectionDs.CreateDynaset()
    PageDs.MoveFirst

End Sub

Sub GetAllPages (SupplyDb As Database, AllPages() As String, AllTopics() As TopicInfo,
DatabaseSection As String, lstPages As ListBox)

    ' modifies  AllPagess
    ' effects   Gets all unique Page Code items from SupplyDb and places them in
    '           AllPages.

    Dim UniquePageDs As Dynaset, TopicTb As Table
    Dim Page As String, Topic As String
    Dim Ctr As Integer

    Ctr = 0

    If TOOL <> EDITOR Then
        Set UniquePageDs = SupplyDb.CreateDynaset("Select DISTINCT [Page Code] from [Page
Database] where [Tutorial Code] = '" & DatabaseSection & "'")
    Else
        Set UniquePageDs = SupplyDb.CreateDynaset("Select DISTINCT [Page Code] from [Page
Database]")
    End If

    Set TopicTb = SupplyDb.OpenTable("Topic Database")
```

```
    ' Set Index
    TopicTb.Index = "Page Code"

    ' add all records in the UniquePageDs into the AllPages dynamic array
    Do
        Page = UniquePageDs.Fields("Page Code")
        TopicTb.Seek "=", Page
        If TopicTb.NoMatch Then
          Topic = Page
        Else
          Topic = TopicTb.Fields("Topic")
        End If
        Ctr = Ctr + 1
        ReDim Preserve AllPages(Ctr)
        ReDim Preserve AllTopics(Ctr)
        AllPages(Ctr) = UniquePageDs.Fields("Page Code")
        AllTopics(Ctr).Topic = Topic
        lstPages.AddItem Topic
        'lstPages.AddItem AllPages(Ctr)
        UniquePageDs.MoveNext
    Loop Until UniquePageDs.EOF

End Sub

Sub GetAllTopics (SupplyDb As Database, AllTopics() As TopicInfo)

    ' modifies  AllPages
    ' effects   Gets all Topics from SupplyDb as places them in AllTopics.

    Dim Page As String, Topic As String
    Dim Ctr As Integer
    Dim TopicDs As Dynaset

    Set TopicDs = SupplyDb.CreateDynaset("Select * from [Topic Database]")

    Ctr = 0
    ReDim AllTopics(Ctr)

    Do Until TopicDs.EOF
        Page = TopicDs.Fields("Page Code")
        Topic = TopicDs.Fields("Topic")
        If Topic <> "none" Then
          AllTopics(Ctr).Parent = Page
          AllTopics(Ctr).Topic = Topic
          Ctr = Ctr + 1
          ReDim Preserve AllTopics(Ctr)
        End If

        TopicDs.MoveNext

    Loop

End Sub
```

```
Function GetBookmark () As Integer

   Dim SectionName, KeyName As String, DefaultVal As Integer, FileName As String
   Dim Response As Integer

   SectionName = "Bookmark"
   KeyName = "Section"
   DefaultVal = 51
   FileName = USERPATH + UserName + "\" + UserName + ".ini"

   Response = GetPrivateProfileInt(SectionName, KeyName, DefaultVal, FileName)

   GetBookmark = Response

End Function

Function GetItems (InputList As ListBox, ByVal DELIMITER As String) As String

   ' effects   Takes the items of InputList and returns a string consisting of each of
   '           the items, followed by Delimiter.

   Dim Ctr As Integer
   Dim Out As String

   Out = ""

   For Ctr = 0 To InputList.ListCount
         Out = Out + InputList.List(Ctr) + DELIMITER
   Next Ctr

   GetItems = Left$(Out, Len(Out) - 1)

End Function

Sub GetObjectTerms (ByVal ObjectCode As String)

   ' effects   Creates a dynaset from the Object Term Database consisting of records where
   '           the Object Code field is equal to ObjectCode.  Moves through the dynaset and
   '           places the information the Object Term field into the ObjectTerm combo box
   '           on the Screen form.

   Dim ObjectTermDs As Dynaset
   Dim ObjectTerm As String

   ' Create a dynaset consisting of all records where the Object Code is equal to ObjectCode.
   ' The dynaset will contain any number of records, listing the object terms associated with
   ' the ObjectCode
   Set ObjectTermDs = PageDb.CreateDynaset("Select * from [Object Term Database] where [Object
Code] = '" & ObjectCode & "'")

   ' Put object terms in the Definition Combo Box
   Do Until ObjectTermDs.EOF
         ObjectTerm = ObjectTermDs.Fields("Object Term")
```

```
        ' don't add duplicates
        If (Not Member(ObjectTerm, cboObjectTerms)) Then
           cboObjectTerms.AddItem ObjectTerm
        End If

        ObjectTermDs.MoveNext
    Loop ' ObjectTermDs.EOF

End Sub

Sub GetOneObject (ByVal Page As String, ByVal ObjectName As String, SectionDs As Dynaset,
ObjectDs As Dynaset)

    SectionDs.Filter = "[Page Code] = '" & Page & "'" & ", [Object] = '" & ObjectName & "'"
    'Set ObjectDs = SectionDs.CreateDynaset()

End Sub

Sub GetOneObjfromPage (ByVal ObjectName As String, PageDs As Dynaset, ObjectDs As Dynaset)

    PageDs.Filter = "[Object] = '" & ObjectName & "'"
    'Set ObjectDs = PageDs.CreateDynaset()

End Sub

Sub HideAllObjects (ByVal OldLabels As Integer, ByVal OldGraphics As Integer, ByVal OldText As
Integer, ByVal OldVideo As Integer, ByVal OldAudio As Integer, ByVal OldInput As Integer, ByVal
OldMetafile As Integer)

    Dim Ctr As Integer

    For Ctr = 1 To OldLabels
        lblPage(Ctr).Visible = False
    Next Ctr

    For Ctr = 1 To OldGraphics
        picAny(Ctr).Visible = False
    Next Ctr

    For Ctr = 1 To OldText
        lblAny(Ctr).Visible = False
    Next Ctr

    For Ctr = 1 To OldVideo
        picVideo(Ctr).Visible = False
        mmcPlayer(Ctr).Visible = False
    Next Ctr

    For Ctr = 1 To OldAudio
        picAudio(Ctr).Visible = False
    Next Ctr

    For Ctr = 1 To OldInput
```

```
        txtUserInput(Ctr).Visible = False
    Next Ctr

    For Ctr = 1 To OldMetafile
        picMetafile(Ctr).Visible = False
    Next Ctr

End Sub

Sub HideSomeObjects (ByVal ObjectList As String)

    Dim ObjToClear As String, ObjType As String
    Dim Object As Control
    Dim ObjNum As Integer, ObjIndex As Integer

    Do While Len(ObjectList) > 1
        ObjToClear = Pop(ObjectList, ",")
        ObjType = Left$(ObjToClear, 1)
        ObjIndex = FindIndex(ObjectsOnPage(), Left$(ObjToClear, Len(ObjToClear) - 1))
        MapAndClearObject ObjIndex, ObjType
    Loop

End Sub

Sub imgTrash_DragDrop (Source As Control, X As Single, Y As Single)

    Set Active = Source
    mnuInfoItem_Click MNUINFO_DELETE

End Sub

Sub imgTrash_DragOver (Source As Control, X As Single, Y As Single, State As Integer)

    If State = ENTER Or State = OVER Then
        Source.DragIcon = imgTrash.Picture
    Else
        Source.DragIcon = LoadPicture("")
    End If

End Sub

Function ItemList (InputArray() As IndexList, ByVal ItemName As String) As String

    ' effects   Returns the item list associated with ItemName in the array InputArray.

    Dim Ctr As Integer

    For Ctr = 1 To UBound(InputArray)
        ' if found, return the list and exit the function
        If InputArray(Ctr).Name = ItemName Then
            ItemList = InputArray(Ctr).List
            Exit Function
        End If
    Next Ctr
```

```
    ' not found, return an empty value
    ItemList = ""

End Function

Sub lblAny_Click (Index As Integer)

    Dim Link As String

    If UserTool = PLAYBACK Then
        If lblAny(Index).Tag = LINKEDOBJ Then
            lblAny(Index).Visible = False
        Else 'If (lblAny(Index).HelpContextID <> LINKUP) Then
            Link = ObjectsOnPage(lblAny(Index).Tag).Link
            If Link <> "none" Then
                lblAny(Index).HelpContextID = LINKUP
                ProcessLink Link, SCREENTEXT_CODE, Index
                If LinkIndex <> UNCACHE Then
                    lblAny(Index).LinkTimeout = LinkIndex
                End If
            End If
        End If
    End If

End Sub

Sub lblAny_KeyPress (Index As Integer, KeyAscii As Integer)

    If UserTool = EDITOR Then
        If KeyAscii = KEY_DELETE Or KeyAscii = Asc("d") Or KeyAscii = Asc("D") Then
            ObjectsOnPage(Active.Tag).Contents = DELETED
            UpdateObject ObjectsOnPage(Active.Tag), PageDb.OpenTable("Page Database")
            Unload lblAny(Index)
            TextCtr = TextCtr - 1
            NumObjects = NumObjects - 1
        End If
    End If

End Sub

Sub lblAny_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
Single)

    Dim CurObject As ObjectInfo
    Dim Transition As Integer

    Set Active = lblAny(Index)

    If UserTool = EDITOR Then
        If Button = LEFT_BUTTON Then
            lblAny(Index).Drag BEGIN_DRAG
            DragX = X
            DragY = Y
```

```
        ElseIf Button = RIGHT_BUTTON Then
            CurObject = ObjectsOnPage(Active.Tag)
            mnuInfoItem(MNU_XY).Caption = "&X, Y. . ." & Active.Top & ", " & Active.Left
            mnuInfoLocItem(MNU_LOCINITIAL).Caption = "As &Initial: " &
UnparseCoords(CurObject.Initial)
            mnuInfoLocItem(MNU_LOCFINAL).Caption = "As &Final: " &
UnparseCoords(CurObject.Final)
            mnuInfoItem(MNU_HEIGHT).Caption = "&Height. . ." & CurObject.Height
            mnuInfoItem(MNU_WIDTH).Caption = "&Width. . ." & CurObject.Width
            mnuInfoItem(MNUINFO_SEQUENCE).Caption = "&Sequence. . ." & CurObject.Sequence
            Transition = ParseTransitionCode(CurObject)
            TransitionCheck Transition
            If Transition = MNUINFOTRANS_MOVEANDDROP Or Transition =
MNUINFOTRANS_DROP Then
                mnuInfoItem(MNUINFO_TIME).Caption = "Drop &Interval. . ." &
CurObject.TimeOnScreen
            ElseIf Transition <> MNUINFOTRANS_NONE Then
                mnuInfoItem(MNUINFO_TIME).Caption = "Move &Interval. . ." &
Right$(CurObject.Transition, Len(CurObject.Transition) - 1)
            End If
            mnuInfoItem(MNUINFO_ZOOM).Enabled = False
            mnuInfoItem(MNUINFO_ROTATE).Enabled = False
            'MNUINFOTRANSITEM(MNUINFOTRANS_ROTATE).Enabled = True
            mnuInfoItem(MNUINFO_SCALE).Enabled = True
            mnuInfoItem(MNUINFO_FLIP).Enabled = False
            dlgGraphic.Filter = "Text Files|*.txt|All Files|*.*"
            dlgGraphic.InitDir = Left$(TEXTPATH, Len(TEXTPATH) - 1)
            If CurObject.Contents <> "" Then
                mnuInfoItem(MNUINFO_FILE).Caption = "File&name. . ." & CurObject.Contents
            Else
                mnuInfoItem(MNUINFO_FILE).Caption = "File&name. . ."
            End If
            PopupMenu mnuInfo, POPUPMENU_LEFTALIGN
        End If
    End If

End Sub

Sub lblAny_MouseMove (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
Single)

    'lblLoc.Caption = CurX + X & ", " & CurY + Y

End Sub

Sub lblAny_MouseUp (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

    lblAny(Index).Drag END_DRAG

End Sub

Sub lblCurPage_Change ()

    'frmTool!lblCurPage.Caption = lblCurPage.Caption
```

```
End Sub

Sub lblCurPage_DblClick ()

   lblToolPage_DblClick

End Sub

Sub lblFromOutline_Change ()

   PageCtr = Val(lblFromOutline.Caption)
   cmdNavigate_Click 1

End Sub

Sub lblFromOutline_Click ()

   PageCtr = Val(lblFromOutline.Caption)
   cmdNavigate_Click 1

End Sub

Sub lblPage_Click (Index As Integer)

   Dim Link As String

   If UserTool = PLAYBACK Then
         If lblPage(Index).Tag = LINKEDOBJ Then
            lblPage(Index).Visible = False
         Else 'If (lblPage(Index).LinkMode <> LINKUP) Then
            Link = ObjectsOnPage(lblPage(Index).Tag).Link
            If Link <> "none" Then
                  lblPage(Index).LinkMode = LINKUP
                  ProcessLink Link, LABEL_CODE, Index
                  If LinkIndex <> UNCACHE Then
                     lblPage(Index).LinkTimeout = LinkIndex
                  End If
            End If
         End If
   End If

End Sub

Sub lblPage_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
Single)

   Dim CurObject As ObjectInfo
   Dim Transition As Integer

   Set Active = lblPage(Index)

   If UserTool = EDITOR Then
         If Button = LEFT_BUTTON Then
```

```
            lblPage(Index).Drag BEGIN_DRAG
            DragX = X / Screen.TwipsPerPixelX
            DragY = Y / Screen.TwipsPerPixelY
          ElseIf Button = RIGHT_BUTTON Then
            CurObject = ObjectsOnPage(Active.Tag)
            mnuLabelItem(MNU_XY).Caption = "&X, Y. . ." & Active.Top & ", " & Active.Left
            mnuLabelLocItem(MNU_LOCINITIAL).Caption = "As &Initial: " &
UnparseCoords(CurObject.Initial)
            mnuLabelLocItem(MNU_LOCFINAL).Caption = "As &Final: " &
UnparseCoords(CurObject.Final)
            mnuLabelItem(MNULBL_SEQUENCE).Caption = "&Sequence. . ." & CurObject.Sequence
            mnuLabelItem(MNULBL_FONTNAME).Caption = "Font Name. . ." & Active.FontName
            mnuLabelItem(MNULBL_FONTSIZE).Caption = "Font Size. . ." & Active.FontSize
            mnuLabelItem(MNULBL_FONTBOLD).Checked = Active.FontBold
            mnuLabelItem(MNULBL_FONTITALIC).Checked = Active.FontItalic
            mnuLabelItem(MNULBL_FONTSTRIKE).Checked = Active.FontStrikeThru
            mnuLabelItem(MNULBL_FONTUNDERLINE).Checked = Active.FontUnderline
            'Transition = ParseTran
            PopupMenu mnuLabel
          End If
    End If

End Sub

Sub lblPage_MouseUp (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

    lblPage(Index).Drag END_DRAG

End Sub

Sub lblProperties_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)

    picProperties_MouseDown 0, Button, Shift, X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY

End Sub

Sub lblProperties_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)

    picProperties_MouseUp 0, Button, Shift, X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY

End Sub

Sub lblTextHot_Click (Index As Integer)

    MsgBox "word", MB_OK

End Sub

Sub lblToolPage_DblClick ()

    KeyPreview = False
    picPage(0).Top = 240
    picPage(0).Left = 72
    picPage(0).Width = 217
```

```
    lstPages.ListIndex = (PageCtr - 1)
    picPage(0).Visible = True
    picPage(0).ZOrder

End Sub

Sub Load_SearchText ()

    Dim I As Integer

    MousePointer = 11
    I = 0
    frmDicSearch.dtaSearch.Refresh
    Do
        frmDicSearch.lstSearch.List(I) = frmDicSearch.txtWords.Text
        I = I + 1
        frmDicSearch.dtaSearch.Recordset.MoveNext
    Loop Until frmDicSearch.dtaSearch.Recordset.EOF = True

    frmDicSearch.dtaSearch.Recordset.MoveFirst
    MousePointer = 0
End Sub

Sub LoadBackground ()

End Sub

Sub LoadCursorModule ()

    Dim LibName As String
    Dim hInstance As Integer
    Dim lpCursorName As String

    LibName = "d:\share\edicsdat.a\agoyo1\cursor.rc"

    hInstance = LoadLibrary(LibName)
    'MsgBox "h " & hInstance

    lpCursorName = "IDC_CURSOR2"
    hCurOpen = LoadCursor(hInstance, lpCursorName)
    'MsgBox "ho " & hCurOpen
    lpCursorName = "IDC_CURSOR1"
    hCurClose = LoadCursor(hInstance, lpCursorName)
    'MsgBox "hc " & hCurClose

End Sub

Sub lstObjectProp_Click (Index As Integer)

    Select Case lstObjectProp(Index).ListIndex
        Case DROPINT
        Case FileName
        Case FinalX
        Case FinalY
```

```
        Case OBJHEIGHT
        Case INITX
        Case INITY
        Case MOVEINT
        Case TRANSTYPE
        Case OBJTIME
        Case OBJWIDTH
    End Select

End Sub

Sub lstObjects_Click ()

    txtLink.Text = FormatLink(ObjLink, (lstPages.Text), Str$(lstObjects.ItemData(lstObjects.ListIndex)),
"")

End Sub

Sub lstObjects_DblClick ()

    txtLink.Text = FormatLink(ObjLink, (lstPages.Text), Str$(lstObjects.ItemData(lstObjects.ListIndex)),
"")

End Sub

Sub lstPages_Click ()

    Dim PageDs As Dynaset
    Dim Page As String
    Dim Contents As String

    If lstPages.Tag = Str$(Link) Then
        txtLink.Text = FormatLink(PAGELINK, (lstPages.Text), "", "")
        Page = AllPages(lstPages.ListIndex + 1)
        SectionDs.Filter = "[Page Code] LIKE '" & Page & "'"
        SectionDs.Sort = "[Sequence]"
        Set PageDs = SectionDs.CreateDynaset()
        PageDs.MoveFirst

        MousePointer = 11
        lstObjects.Clear

        Do Until PageDs.EOF
            Contents = PageDs.Fields("Object")
            lstObjects.AddItem UnparseObjName(Left$(Contents, 1)) & " " & Contents
            lstObjects.ItemData(lstObjects.NewIndex) = Val(Right$(Contents, Len(Contents) - 1))
            PageDs.MoveNext
        Loop

        MousePointer = 0
    End If

End Sub
```

```
Sub lstPages_DblClick ()

    ' modifies  frmPlayer
    ' effects   Goes to the page selected by the index of lstPages.

    If lstPages.Tag = Str$(Link) Then
        txtLink.Text = FormatLink(PAGELINK, (lstPages.Text), "", "")
    Else
        picPage(0).Visible = False
        PageCtr = (lstPages.ListIndex + 1)
        MainScreenPrint AllPages(PageCtr)
    End If

End Sub

Sub MainScreenPrint (ByVal Page As String)

    ' effects   Prints to the current instance of frmPlayer the database objects linked to
    '           Page.

    Dim NullVal As Integer
    Dim ObjectCtr As Integer
    Dim PageDs As Dynaset
    Dim PageObject As ObjectInfo

    ' Unload old graphics and text
    ClearAllObjects LabelCtr, GraphicCtr, TextCtr, VideoCtr, AudioCtr, InputCtr, CommandCtr,
MetafileCtr

    LabelCtr = 0
    GraphicCtr = 0
    TextCtr = 0
    VideoCtr = 0
    AudioCtr = 0
    InputCtr = 0
    CommandCtr = 0
    NumObjects = 0
    NewObjects = 0
    VideoDone = True
    SoundDone = True
    NextPage = True

    ' Clear current page array of objects
    ReDim ObjectMap(NumObjects)
    ReDim ObjectsOnPage(NumObjects)

    ' Clear the object term list
    cboObjectTerms.Clear

    'MsgBox SectionDs.Fields("Tutorial Code"), MB_OK

    'CurSection = SectionDs.Fields("Tutorial Code")
    lblCurPage.Caption = Page    ' appears only in EDITOR mode
    If UserIndex Then
```

```
        TopicTb.Index = "Topic"
        TopicTb.Seek "=", Page
        PageCaption = Page
        If Len(PageCaption) <= 25 Then
          mnuPageLabel.Caption = PageCaption
        Else
          mnuPageLabel.Caption = Left$(PageCaption, 25)
        End If

        If Not (TopicTb.NoMatch) Then
          Page = TopicTb.Fields("Page Code")
        End If

    Else
        TopicTb.Index = "Page Code"
        TopicTb.Seek "=", Page
        If Not (TopicTb.NoMatch) Then
          PageCaption = TopicTb.Fields("Topic")
          If Len(PageCaption) <= 25 Then
                mnuPageLabel.Caption = PageCaption
          Else
                mnuPageLabel.Caption = Left$(PageCaption, 25)
          End If
        End If
    End If


    ' Select only those records where the Page Code field is equal to Page, and sort
    ' by sequence, in ascending order
    SectionDs.Filter = "[Page Code] LIKE '" & Page & "'"
    SectionDs.Sort = "[Sequence]"
    Set PageDs = SectionDs.CreateDynaset()
    PageDs.MoveFirst

    CurrentBookmark = PageDs.Bookmark

    'If Tool = EDITOR Then
    '   frmEditor.dtaEditor.RecordSource = SQL1 & Page & SQL2
    '   frmEditor.dtaEditor.Refresh
    'End If

    CreateObjectsArray PageDs, Page

    For ObjectCtr = 1 To NumObjects

        PageObject = ObjectsOnPage(ObjectCtr)
        NullVal = DisplayObject(PageObject, ObjectCtr)

    Next ObjectCtr

End Sub

Sub MakeNewDbObject (Object As Control, ByVal Pointer As Integer)
```

```
    ' modifies  ObjectsOnPage
    ' effects   Creates a new object, of type ObjectInfo, based on the information provided by
    '           Object, such as Object.Width, Object.Height, Object.Top, etc.
    '           Places the new object into the ObjectOnPage global array, in the position denoted
    '           by Pointer.

    Dim DbObject As ObjectInfo
    Dim Nullname As String

    If TypeOf Object Is ACCUSOFT Then
        Nullname = "s0"
    ElseIf TypeOf Object Is HEVBLayer Then
        Nullname = "t0"
    ElseIf TypeOf Object Is SSCommand Then
        Nullname = "b0"
    Else
        Nullname = Object.LinkItem
    End If

    MakeObject DbObject, 0, DatabaseSection, AllPages(PageCtr), Object.Left, Object.Top, Object.Left,
Object.Top, Pointer, "0", 0, Object.Height, Object.Width, "", Nullname, "", "none"

    ObjectsOnPage(Pointer) = DbObject

    Set Active = Object

End Sub

Sub MakeObject (Object As ObjectInfo, ByVal ID As Integer, ByVal CurSection As String, ByVal
CurPage As String, ByVal InitialX As Integer, ByVal InitialY As Integer, ByVal FinalX As Integer,
ByVal FinalY As Integer, ByVal Sequence As Integer, ByVal Transition As String, ByVal
TimeOnScreen As Integer, ByVal ObjectHeight As Integer, ByVal ObjectWidth As Integer, ByVal Path
As String, ByVal CurObject As String, ByVal Extension As String, Link)

    ' modifies  Object
    ' effects   Creates an object of type ObjectInfo, based on information provided by ID,
    '           CurSection, CurPage, etc.

    ' A new Object doesn't have an ID until it is sent to the database
    ' A negative flag is sent to this procedure when a new object is created
     If ID > 0 Then
        Object.ID = ID
    End If

    Object.TutorialCode = CurSection
    Object.PageCode = CurPage
    Object.Initial.Left = InitialX
    Object.Initial.Top = InitialY
    Object.Final.Left = FinalX
    Object.Final.Top = FinalY
    Object.Sequence = Sequence
    Object.Transition = Transition
    Object.TimeOnScreen = TimeOnScreen
    Object.Height = ObjectHeight
```

```
    Object.Width = ObjectWidth
    Object.Path = Path
    Object.Contents = CurObject
    Object.Extension = Extension
    Object.Link = Link

    ParseObjectInfo Object
    'Load lstObjectProp(NumObjects)
    FillListBoxProperties NumObjects
    AddNewItem CurObject, cboObjects
    'MsgBox "prop box created Obj: " & CurObject & " Obj #: " & NumObjects, MB_OK
End Sub

Sub MapAndClearObject (ByVal Index As Integer, ByVal ObjectType As String)

    Dim ObjNum As Integer
    Dim Object As Control

    ObjNum = ObjectMap(Index)

    Select Case UCase(ObjectType)
        Case STILL_CODE
            Set Object = picAny(ObjNum)
        Case VIDEO_CODE
            Set Object = picVideo(ObjNum)
            mmcPlayer(ObjNum).Command = "Stop"
            mmcPlayer(ObjNum).Visible = False
        Case SCREENTEXT_CODE
            Set Object = lblAny(ObjNum)
        Case INPUT_CODE
            Set Object = txtUserInput(ObjNum)
        Case LABEL_CODE
            Set Object = lblPage(ObjNum)
    End Select

    Object.Visible = False

End Sub

Sub mmcPlayer_Done (Index As Integer, NotifyCode As Integer)

    VideoDone = True

    If mmcPlayer(Index).HelpContextID = MNUINFOTRANS_LOOP Then
        mmcPlayer(Index).Command = "Prev"
        mmcPlayer(Index).Command = "Play"
    End If

End Sub

Sub mmcPlayer_PlayCompleted (Index As Integer, ErrorCode As Long)

    VideoDone = True
```

```
End Sub

Sub mmcSound_Done (NotifyCode As Integer)

   'Print NotifyCode
   'SoundDone = True


End Sub

Sub mmcSound_PlayCompleted (ErrorCode As Long)

   SoundDone = True

End Sub

Sub mmcSound_StatusUpdate ()

   If mmcSound.Mode <> 526 Then
         SoundDone = True
   Else
         SoundDone = False
   End If

End Sub

Sub mnuAudioFileItem_Click (Index As Integer)

   Select Case Index
         Case 0                        ' New File
            dlgGraphic.Filter = "Audio Files|*.wav|All Files|*.*"
            mnuInfoFileItem_Click Index
   End Select

End Sub

Sub mnuAudioItem_Click (Index As Integer)

   Dim Sound As Integer, SoundFile As String

   Select Case Index
         Case 0                        ' Play File
            SoundFile = ObjectsOnPage(Active.Tag).Path & ObjectsOnPage(Active.Tag).Contents &
".wav"
            If SoundFile <> "" Then
                  Sound = sndPlaySound(SoundFile, SND_ASYNC)
            End If
         Case 4                        ' Delete Audio
            picAudio_KeyPress (Active.Index), Asc("d")
   End Select

End Sub

Sub mnuAudioSeqItem_Click (Index As Integer)
```

```
    mnuInfoSeqItem_Click Index

End Sub

Sub mnuAudioTransItem_Click (Index As Integer)

    mnuAudioTransItem(0).Checked = Not (mnuAudioTransItem(0).Checked)
    mnuAudioTransItem(1).Checked = Not (mnuAudioTransItem(1).Checked)

    If Index = 0 Then
        ObjectsOnPage(Active.Tag).Transition = AUDIO_SYNC
    Else
        ObjectsOnPage(Active.Tag).Transition = AUDIO_NOSYNC
    End If

End Sub

Sub mnuBranchHeightItem_Click (Index As Integer)

    Dim CurObject As ObjectInfo
    Dim NewH As Integer
    Dim Custom As Integer

    CurObject = ObjectsOnPage(Active.Tag)

    Select Case Index
        Case 0                          ' Scale Custom
            DialogBox DIALOG_HEIGHT
            Custom = frmDialog.Tag
            If Custom <> 0 Then
                NewH = Custom
            Else
                NewH = Active.Height
            End If
        Case 1                          ' Scale -10%
            NewH = Active.Height * .9
        Case 2                          ' Scale +10%
            NewH = Active.Height + (Active.Height * .1)
    End Select

    ObjectsOnPage(Active.Tag).Height = NewH
    Active.Height = NewH
    mnuInfoItem(1).Caption = "Height. . ." & NewH

End Sub

Sub mnuBranchItem_Click (Index As Integer)

    Dim FileName As String, ActiveFileName As String
    Dim Extension As String, CurPlayer As MMControl
    Dim TempIndex As Integer

    Select Case Index
```

```
        Case BRANCH_PIC  ' picture

          ActiveFileName = ObjectsOnPage(Active.Tag).Contents
          dlgGraphic.FilterIndex = 1
          dlgGraphic.Action = 1
          FileName = dlgGraphic.Filename

          If FileName <> ActiveFileName Then
                Extension = LCase(Mid$(FileName, Len(FileName) - 3, Len(FileName)))
                ObjectsOnPage(Active.Tag).Extension = Extension

                Select Case Extension
                  Case FILE_BITMAP, FILE_ICON, FILE_DIB
                        Active.Picture = LoadPicture(FileName)
                        FileName = Mid$(FileName, Len(STILLPATH) + 1)
                        FileName = Mid$(FileName, 1, Len(FileName) - 4)
                        ObjectsOnPage(Active.Tag).Height = Active.Height
                        ObjectsOnPage(Active.Tag).Width = Active.Width
                        ObjectsOnPage(Active.Tag).Path = STILLPATH
                        ObjectsOnPage(Active.Tag).Contents = FileName
                End Select
          End If

        Case BRANCH_JUMPTO  ' jump to
        Case BRANCH_LOADOBJ  ' load object
    End Select

End Sub

Sub mnuBranchScaleItem_Click (Index As Integer)

    Dim NewH As Integer, NewW As Integer
    Dim Custom As Integer

    Select Case Index
        Case 0                  ' - 10%
          NewH = Active.Height * .9
          NewW = Active.Width * .9
        Case 1                  ' + 10%
          NewH = Active.Height + (Active.Height * .1)
          NewW = Active.Width + (Active.Width * .1)
        Case 2                  ' Custom
          DialogBox DIALOG_SCALE
          Custom = frmDialog.Tag
          If frmDialog.Tag < 0 Then
                NewH = Active.Height * Abs(frmDialog.Tag)
                NewW = Active.Width * Abs(frmDialog.Tag)
          ElseIf frmDialog.Tag > 0 Then
                NewH = Active.Height + (Active.Height * frmDialog.Tag)
                NewW = Active.Width + (Active.Width * frmDialog.Tag)
          Else
                NewH = Active.Height
                NewW = Active.Width
          End If
```

```
    End Select

    mnuInfoItem(1).Caption = "Height. . ." & NewH
    mnuInfoItem(2).Caption = "Width. . ." & NewW

    Active.Height = NewH
    Active.Width = NewW
    ObjectsOnPage(Active.Tag).Height = NewH
    ObjectsOnPage(Active.Tag).Width = NewW

End Sub

Sub mnuBranchWidthItem_Click (Index As Integer)

    Dim CurObject As ObjectInfo
    Dim NewW As Integer
    Dim Custom As Integer

    CurObject = ObjectsOnPage(Active.Tag)

    Select Case Index
        Case 0                          ' Custom
            DialogBox DIALOG_WIDTH
            Custom = frmDialog.Tag
            If Custom <> 0 Then
                NewW = Custom
            Else
                NewW = Active.Width
            End If
        Case 1                          ' Scale - 10%
            NewW = Active.Width * .9
        Case 2                          ' Scale + 10%
            NewW = Active.Width + (Active.Width * .1)
    End Select

    ObjectsOnPage(Active.Tag).Width = NewW
    Active.Width = NewW
    mnuInfoItem(2).Caption = "Width. . ." & NewW

End Sub

Sub mnuCmdAttrItem_Click (Index As Integer)

    Select Case Index
        Case CLEAR_ALL
            ObjectsOnPage(Active.Tag).Transition = OBJECT_CLEARALL
        Case CLEAR_SOME
            picObj.Top = 72
            picObj.Left = 80
            If ObjectsOnPage(Active.Tag).Transition = OBJECT_CLEARALL Then
                cmdObj(1).Enabled = False
            Else
                cmdObj(1).Enabled = True
            End If
```

```
            picObj.ZOrder
            picObj.Visible = True
            PrintObjects lstAllObjects
            UnParseClearList lstClearObjects, ObjectsOnPage(Active.Tag).Transition
        Case NEXT_PAGE
            ObjectsOnPage(Active.Tag).Transition = OBJECT_NEXTPAGE
        Case SPECIAL_OBJ
            picSpecialObj(0).Visible = True
    End Select

End Sub

Sub mnuCmdSeqItem_Click (Index As Integer)

    mnuInfoSeqItem_Click Index

End Sub

Sub mnuCommandItem_Click (Index As Integer)

    Select Case Index
        Case 3              ' Delete Command
    End Select

End Sub

Sub mnuFileItem_Click (Index As Integer)

    Select Case Index
        Case 0
            PrintForm
        Case 1              ' Select Quit
            cmdInterface_Click 5
    End Select

End Sub

Sub mnuHelpItem_Click (Index As Integer)

    Select Case Index
        Case 0

        Case 1

        Case 2

        Case 3
            MsgBox MSG_ABOUT, MB_OK + MB_ICONINFORMATION, TITLE_ABOUT
    End Select

End Sub

Sub mnuIndexItem_Click (Index As Integer)
```

```
    cmdInterface_Click 0

End Sub

Sub mnuInfoFileItem_Click (Index As Integer)

    On Error GoTo ErrHandler

    Dim FileName As String, ActiveFileName As String
    Dim Extension As String, CurPlayer As MMControl
    Dim TempIndex As Integer

    ActiveFileName = ObjectsOnPage(Active.Tag).Contents

    Select Case Index

        Case 0                          ' Load in file
            dlgGraphic.FilterIndex = 1
            dlgGraphic.Action = 1
            FileName = dlgGraphic.Filename

            If FileName <> ActiveFileName Then
                Extension = LCase(Mid$(FileName, Len(FileName) - 3, Len(FileName)))
                ObjectsOnPage(Active.Tag).Extension = Extension

                Select Case Extension
                  Case FILE_BITMAP, FILE_ICON, FILE_STILL, FILE_DIB
                        Active.Image = FileName
                        Active.Height = Active.ImageHeight
                        Active.Width = Active.ImageWidth
                        FileName = Mid$(FileName, Len(STILLPATH) + 1)
                        FileName = Mid$(FileName, 1, Len(FileName) - 4)
                        ObjectsOnPage(Active.Tag).Height = Active.Height
                        ObjectsOnPage(Active.Tag).Width = Active.Width
                        ObjectsOnPage(Active.Tag).Path = STILLPATH
                        ObjectsOnPage(Active.Tag).Contents = FileName
                  Case FILE_METAFILE
                        Active.Picture = LoadPicture(FileName)
                        FileName = Mid$(FileName, Len(STILLPATH) + 1)
                        FileName = Mid$(FileName, 1, Len(FileName) - 4)
                        ObjectsOnPage(Active.Tag).Height = Active.Height
                        ObjectsOnPage(Active.Tag).Width = Active.Width
                        ObjectsOnPage(Active.Tag).Path = STILLPATH
                        ObjectsOnPage(Active.Tag).Contents = FileName
                  Case FILE_VIDEO
                        Set CurPlayer = mmcPlayer(Active.Index)
                        CurPlayer.FileName = FileName
                        FileName = Mid$(FileName, Len(VIDEOPATH) + 1)
                        FileName = Mid$(FileName, 1, Len(FileName) - 4)
                        ObjectsOnPage(Active.Tag).Path = VIDEOPATH
                        ObjectsOnPage(Active.Tag).Contents = FileName
                        CurPlayer.hWndDisplay = Active.hWnd
                        CurPlayer.Command = "Close"
                        CurPlayer.Command = "Open"
```

```
                    CurPlayer.Command = "Play"
                Case FILE_WAVE
                    FileName = Mid$(FileName, Len(AUDIOPATH) + 1)
                    FileName = Mid$(FileName, 1, Len(FileName) - 4)
                    ObjectsOnPage(Active.Tag).Path = AUDIOPATH
                    ObjectsOnPage(Active.Tag).Contents = FileName
                Case FILE_TEXT
                    FileName = Mid$(FileName, Len(TEXTPATH) + 1)
                    FileName = Mid$(FileName, 1, Len(FileName) - 4)
                    ObjectsOnPage(Active.Tag).Path = TEXTPATH
                    ObjectsOnPage(Active.Tag).Contents = FileName
                    'HESetFileName lblAny(Active.Index).hWnd, TEXTPATH & FileName &
Extension
                    PrettyPrint lblAny(Active.Index), cboObjectTerms, FileName, TextTb
            End Select

        End If

    Case 1                  ' Save File
        Extension = LCase(Mid$(ActiveFileName, Len(ActiveFileName) - 3, Len(ActiveFileName)))
        Select Case Extension
            Case FILE_STILL
                Active.Save = ObjectsOnPage(Active.Tag).Path &
ObjectsOnPage(Active.Tag).Contents & Extension
        End Select
End Select

ErrHandler:
  Exit Sub

End Sub

Sub mnuInfoFlipItem_Click (Index As Integer)

  Select Case Index
    Case 0                  ' Flip Horizontally
      Active.FlipX = True
    Case 1                  ' Flip Vertically
      Active.FlipY = True
  End Select

End Sub

Sub mnuInfoHeightItem_Click (Index As Integer)

  Dim CurObject As ObjectInfo
  Dim NewH As Integer
  Dim Custom As Integer

  CurObject = ObjectsOnPage(Active.Tag)

  Select Case Index
    Case 0                  ' Scale Custom
      DialogBox DIALOG_HEIGHT
```

```
            Custom = frmDialog.Tag
            If Custom <> 0 Then
                  NewH = Custom
            Else
                  NewH = Active.Height
            End If
         Case 1                        ' Scale -10%
            NewH = Active.Height * .9
         Case 2                        ' Scale +10%
            NewH = Active.Height + (Active.Height * .1)
   End Select

   ObjectsOnPage(Active.Tag).Height = NewH
   Active.Height = NewH
   If CurObject.Extension = FILE_TEXT Then
         'HESetFileName lblAny(Active.Index).hWnd, TEXTPATH & CurObject.Contents &
FILE_TEXT
         PrettyPrint lblAny(Active.Index), cboObjectTerms, CurObject.Contents, TextTb
   End If

   mnuInfoItem(1).Caption = "Height. . ." & NewH

End Sub

Sub mnuInfoItem_Click (Index As Integer)

   Dim Extension As String
   Dim Ctr As Integer
   Dim Contents As String

   Select Case Index
         Case MNUINFO_BACK                     ' Send to back
            Active.ZOrder 1
            mnuInfoItem(MNUINFO_BACK).Enabled = False
            mnuInfoItem(MNUINFO_FRONT).Enabled = True
         Case MNUINFO_FRONT                    ' Send to Front
            Active.ZOrder
            mnuInfoItem(MNUINFO_FRONT).Enabled = False
            mnuInfoItem(MNUINFO_BACK).Enabled = True
         Case MNUINFO_LINK
            cmdCreateIndex(0).Visible = False
            cmdCreateIndex(5).Visible = False
            lstPages.ListIndex = (PageCtr - 1)
            lblGotoPage(1).Caption = "&Index Sets:"
            picPage(0).Width = 625
            picPage(0).Visible = True
            lstObjects.Visible = True
            lstPages.Tag = Str$(Link)
            lblPageLinks.Visible = True
            lblObjects.Visible = True
            txtLink.Visible = True
            lblNewObj.Visible = True
            lblLinkLabel.Visible = True
            txtLink.Text = FormatLink(PAGELINK, AllPages(PageCtr), "", "")
```

```
            picDropNew.Visible = True
            'frmDialogs!cmdCreateIndex(0).Visible = False
            'frmDialogs!cmdCreateIndex(5).Visible = False
            'frmDialogs!lstPages.ListIndex = (PageCtr - 1)
            'frmDialogs!lblGotoPage(1).Caption = "&Index Sets:"
            'frmDialogs!picPage.Visible = True
            'frmDialogs!lstObjects.Visible = True
            'frmDialogs!lstPages.Tag = Str$(Link)
            'frmDialogs!lblPageLinks.Visible = True
            'frmDialogs!lblObjects.Visible = True
            'frmDialogs!txtLink.Visible = True
            'frmDialogs!lblNewObj.Visible = True
            'frmDialogs!lblLinkLabel.Visible = True
            'frmDialogs!txtLink.Text = AllPages(PageCtr)
            'frmDialogs!picDropNew.Visible = True
            For Ctr = 1 To NumObjects
                  Contents = ObjectsOnPage(Ctr).Contents
                  lstObjects.AddItem UnparseObjName(Left$(Contents, 1)) & " " & Contents
                  'frmDialogs!lstObjects.AddItem UnparseObjName(Left$(Contents, 1)) & " " &
Contents
            Next Ctr
            'frmDialogs.Show
         Case MNUINFO_DELETE
            Extension = ObjectsOnPage(Active.Tag).Extension
            Select Case Extension
                  Case FILE_STILL
                     picAny_KeyPress (Active.Index), Asc("d")
                  Case FILE_TEXT
                     lblAny_KeyPress (Active.Index), Asc("d")
                  Case FILE_VIDEO
                     picVideo_KeyPress (Active.Index), Asc("d")
            End Select
         Case MNUINFO_UPDATE
            UpdateObject ObjectsOnPage(Active.Tag), PageDb.OpenTable("Page Database")
   End Select

End Sub

Sub mnuInfoLocItem_Click (Index As Integer)

   Select Case Index
         Case MNU_LOCINITIAL                    ' Set as initial
            MakeCoords ObjectsOnPage(Active.Tag).Initial, Active.Top, Active.Left
         Case MNU_LOCFINAL                      ' Set as final
            MakeCoords ObjectsOnPage(Active.Tag).Final, Active.Top, Active.Left
   End Select


End Sub

Sub mnuInfoRotateItem_Click (Index As Integer)

   Dim Temp As Integer
```

```
    Temp = Active.Height
    Select Case Index
        Case 0                  ' 90 CCW
            Active.Rotate = 90
        Case 1                  ' 90 CW
            Active.Rotate = 270
    End Select

    Active.Height = Active.Width
    Active.Width = Temp

End Sub

Sub mnuInfoScaleItem_Click (Index As Integer)

    Dim NewH As Integer, NewW As Integer
    Dim Custom As Integer

    Select Case Index
        Case 0                  ' - 10%
            NewH = Active.Height * .9
            NewW = Active.Width * .9
        Case 1                  ' + 10%
            NewH = Active.Height + (Active.Height * .1)
            NewW = Active.Width + (Active.Width * .1)
        Case 2                  ' Custom
            DialogBox DIALOG_SCALE
            Custom = frmDialog.Tag
            If frmDialog.Tag < 0 Then
                NewH = Active.Height * Abs(frmDialog.Tag)
                NewW = Active.Width * Abs(frmDialog.Tag)
            ElseIf frmDialog.Tag > 0 Then
                NewH = Active.Height + (Active.Height * frmDialog.Tag)
                NewW = Active.Width + (Active.Width * frmDialog.Tag)
            Else
                NewH = Active.Height
                NewW = Active.Width
            End If
    End Select

    mnuInfoItem(1).Caption = "Height. . ." & NewH
    mnuInfoItem(2).Caption = "Width. . ." & NewW

    Active.Height = NewH
    Active.Width = NewW
    ObjectsOnPage(Active.Tag).Height = NewH
    ObjectsOnPage(Active.Tag).Width = NewW
    If ObjectsOnPage(Active.Tag).Extension = FILE_TEXT Then
        'HESetFileName lblAny(Active.Index).hWnd, TEXTPATH &
ObjectsOnPage(Active.Tag).Contents & FILE_TEXT
        PrettyPrint lblAny(Active.Index), cboObjectTerms, ObjectsOnPage(Active.Tag).Contents,
TextTb
    End If
```

```
End Sub

Sub mnuInfoSeqItem_Click (Index As Integer)

   Dim Sequence As Integer

   Sequence = ObjectsOnPage(Active.Tag).Sequence
   Select Case Index
         Case 0                        ' Higher sequence
            Sequence = Sequence + 1
         Case 1                        ' Lower Sequence
            Sequence = Sequence - 1
   End Select

   ObjectsOnPage(Active.Tag).Sequence = Sequence

End Sub

Sub mnuInfoTimeItem_Click (Index As Integer)

   DialogBox DIALOG_SETTIME
   If frmDialog.Tag <> 0 Then
         ObjectsOnPage(Active.Tag).TimeOnScreen = frmDialog.Tag
   End If

End Sub

Sub mnuInfoTransItem_Click (Index As Integer)

   Dim TransitionValue As Integer

   TransitionValue = 0
   TransitionCheck Index

   Select Case Index
         Case MNUINFOTRANS_MOVE, MNUINFOTRANS_MOVEANDDROP
            DialogBox DIALOG_SETTIME
            If frmDialog.Tag <> 0 Then
                  TransitionValue = frmDialog.Tag
            Else
                  TransitionValue = TIME_DEFAULT
            End If
            mnuInfoItem(MNUINFO_TIME).Caption = "Move &Interval. . ." & TransitionValue
   End Select

   ObjectsOnPage(Active.Tag).Transition = UnparseTransitionCode(Index, TransitionValue)

   If Index = MNUINFOTRANS_MOVEANDDROP Or Index = MNUINFOTRANS_DROP Then
         DialogBox DIALOG_SETTIME
         If frmDialog.Tag <> 0 Then
            TransitionValue = frmDialog.Tag
         Else
            TransitionValue = DROP_DEFAULT
         End If
```

```
        mnuInfoItem(MNUINFO_TIME).Caption = "Drop &Interval. . ." & TransitionValue
        ObjectsOnPage(Active.Tag).TimeOnScreen = TransitionValue
    End If

End Sub

Sub mnuInfoWidthItem_Click (Index As Integer)

    Dim CurObject As ObjectInfo
    Dim NewW As Integer
    Dim Custom As Integer

    CurObject = ObjectsOnPage(Active.Tag)

    Select Case Index
        Case 0                          ' Custom
            DialogBox DIALOG_WIDTH
            Custom = frmDialog.Tag
            If Custom <> 0 Then
                NewW = Custom
            Else
                NewW = Active.Width
            End If
        Case 1                          ' Scale - 10%
            NewW = Active.Width * .9
        Case 2                          ' Scale + 10%
            NewW = Active.Width + (Active.Width * .1)
    End Select

    ObjectsOnPage(Active.Tag).Width = NewW
    Active.Width = NewW
    If CurObject.Extension = FILE_TEXT Then
        'HESetFileName lblAny(Active.Index).hWnd, TEXTPATH & CurObject.Contents &
CurObject.Extension
        PrettyPrint lblAny(Active.Index), cboObjectTerms, CurObject.Contents, TextTb
    End If

    mnuInfoItem(2).Caption = "Width. . ." & NewW

End Sub

Sub mnuInfoZoomItem_Click (Index As Integer)

    Select Case Index
        Case 0                          ' Zoom in 50%
            Active.Zoom = 150
        Case 1                          ' Zoom Out 50%
            Active.Zoom = 50
        Case 2                          ' Zoom Custom
            DialogBox DIALOG_ZOOM
            If frmDialog.Tag <> 0 Then
                Active.Zoom = frmDialog.Tag
            End If
    End Select
```

```
    Active.ScrollBars = False

End Sub

Sub mnuInputItem_Click (Index As Integer)

    Select Case Index
        Case MNUINPUT_DELETE
            ObjectsOnPage(Active.Tag).Contents = DELETED
            UpdateObject ObjectsOnPage(Active.Tag), PageDb.OpenTable("Page Database")
            Unload Active
            InputCtr = InputCtr - 1
            NumObjects = NumObjects - 1
        Case MNUINPUT_UPDATE
            UpdateObject ObjectsOnPage(Active.Tag), PageDb.OpenTable("Page Database")
    End Select

End Sub

Sub mnuInputLocItem_Click (Index As Integer)

    mnuInfoLocItem_Click Index

End Sub

Sub mnuInputSeqItem_Click (Index As Integer)

    mnuInfoSeqItem_Click Index

End Sub

Sub mnuInputTimeItem_Click (Index As Integer)

    DialogBox DIALOG_SETTIME
    If frmDialog.Tag <> 0 Then
        ObjectsOnPage(Active.Tag).TimeOnScreen = frmDialog.Tag
    End If

End Sub

Sub mnuInputTransItem_Click (Index As Integer)

    mnuInfoTransItem_Click Index

End Sub

Sub mnuLabelItem_Click (Index As Integer)

    Dim TempText As String
    Dim TempTrans As String

    Select Case Index
        Case MNULBL_DELETE
```

```
    ObjectsOnPage(Active.Tag).Contents = DELETED
    UpdateObject ObjectsOnPage(Active.Tag), PageDb.OpenTable("Page Database")
    Unload Active
    LabelCtr = LabelCtr - 1
    NumObjects = NumObjects - 1
Case MNULBL_NEW
    TempText = InputBox$("Enter New Text:", "New Text", Active.Caption)
    If TempText <> "" Then
        ObjectsOnPage(Active.Tag).Contents = LABEL_CODE + TempText
        Active.Caption = TempText
    End If
Case MNULBL_FONTNAME, MNULBL_FONTSIZE
    dlgGraphic.CancelError = True
    On Error GoTo ErrCancel
    dlgGraphic.Flags = &H1& Or &H100&
    dlgGraphic.FontName = Active.FontName
    dlgGraphic.FontSize = Active.FontSize
    dlgGraphic.FontBold = Active.FontBold
    dlgGraphic.FontUnderLine = Active.FontUnderline
    dlgGraphic.FontStrikeThru = Active.FontStrikeThru
    dlgGraphic.Color = Active.ForeColor
    dlgGraphic.Action = 4
    Active.FontName = dlgGraphic.FontName
    Active.FontSize = dlgGraphic.FontSize
    Active.FontBold = dlgGraphic.FontBold
    Active.FontUnderline = dlgGraphic.FontUnderLine
    Active.FontStrikeThru = dlgGraphic.FontStrikeThru
    Active.ForeColor = dlgGraphic.Color
Case MNULBL_FONTBOLD
    mnuLabelItem(Index).Checked = Not (mnuLabelItem(Index).Checked)
    Active.FontBold = mnuLabelItem(Index).Checked
Case MNULBL_FONTITALIC
    mnuLabelItem(Index).Checked = Not (mnuLabelItem(Index).Checked)
    Active.FontItalic = mnuLabelItem(Index).Checked
Case MNULBL_FONTUNDERLINE
    mnuLabelItem(Index).Checked = Not (mnuLabelItem(Index).Checked)
    Active.FontUnderline = mnuLabelItem(Index).Checked
Case MNULBL_FONTCOLORFORE
    dlgGraphic.CancelError = True
    On Error GoTo ErrCancel
    dlgGraphic.Flags = &H1&
    dlgGraphic.Color = Active.ForeColor
    dlgGraphic.Action = 3
    Active.ForeColor = dlgGraphic.Color
Case MNULBL_FONTCOLORBACK
    dlgGraphic.CancelError = True
    On Error GoTo ErrCancel
    dlgGraphic.Flags = &H1&
    dlgGraphic.Color = Active.BackColor
    dlgGraphic.Action = 3
    Active.BackStyle = TRANSPARENT
    Active.BackColor = dlgGraphic.Color
Case MNULBL_LINK
    mnuInfoItem_Click MNUINFO_LINK
```

```
        Case MNULBL_UPDATE
          UpdateObject ObjectsOnPage(Active.Tag), PageDb.OpenTable("Page Database")
    End Select

    If Index <> MNULBL_NEW And Index <> MNULBL_DELETE Then
        ObjectsOnPage(Active.Tag).Transition = ParseTrans(lblPage(Active.Index))
    End If

ErrCancel:
    Exit Sub

End Sub

Sub mnuLabelLocItem_Click (Index As Integer)

    mnuInfoLocItem_Click Index

End Sub

Sub mnuLabelSeqItem_Click (Index As Integer)

    mnuInfoSeqItem_Click Index

End Sub

Sub mnuLblTransItem_Click (Index As Integer)

    Dim Ctr As Integer, TransitionValue As Integer

    For Ctr = MNULBLTRANS_NONE To MNULBLTRANS_MOVEANDDROP
        If Ctr = Index Then
          mnuLblTransItem(Ctr).Checked = True
        Else
          mnuLblTransItem(Ctr).Checked = False
        End If
    Next Ctr

    Select Case Index
        Case MNULBLTRANS_MOVE, MNULBLTRANS_MOVEANDDROP
          DialogBox DIALOG_SETTIME
          If frmDialog.Tag <> 0 Then
              TransitionValue = frmDialog.Tag
          Else
              TransitionValue = TIME_DEFAULT
          End If
    End Select

End Sub

Sub mnuMapItem_Click (Index As Integer)

    Select Case Index
        Case 0
        Case 1
```

```
        Case 3
        Case 4
    End Select

End Sub

Sub mnuNotepadItem_Click (Index As Integer)

    Dim hWnd As Integer

    Select Case Index
        Case MNUNOTE_OPEN                    ' Open Notepad
            DocCtr = DocCtr + 1
            ReDim Preserve Documents(DocCtr)
            Load Documents(DocCtr)
            mnuWindowItem_Click 0
            mnuNotepadItem(MNUNOTE_CLOSE).Enabled = True
        Case MNUNOTE_CLOSE                   ' Close Notepad
            Unload frmNotepad
        Case MNUNOTE_PASTEFROM
        Case MNUNOTE_PASTETO
    End Select

End Sub

Sub mnuOptionsItem_Click (Index As Integer)

    Dim Custom As Integer

    Select Case Index
        Case 0                       ' Sound Option
            cmdSound_Click
        Case 1
            picVolume(0).Visible = True
        Case 2                       ' Jump Interval
            DialogBox DIALOG_SETJUMP
            Custom = frmDialog.Tag
            If Custom <> 0 Then
                Sensitivity = Custom
            End If
            mnuOptionsItem(Index).Caption = "&Jump Interval: " & Sensitivity
    End Select

End Sub

Sub mnuPageItem_Click (Index As Integer)

    Select Case Index
        Case 0
            cmdNavigate_Click 1
        Case 1
            cmdNavigate_Click 0
        Case 2
            cmdNavigate_Click 2
```

```
        Case 4
          lblToolPage_DblClick
        Case 5
          lblToolPage_DblClick
          cmdCreateIndex_Click 0
          cmdCreateIndex(7).Visible = True
        Case 6
          cmdCreateIndex_Click 7
    End Select

End Sub

Sub mnuViewItem_Click (Index As Integer)

    mnuViewItem(Index).Checked = Not mnuViewItem(Index).Checked

    Select Case Index
        Case 0
          If mnuViewItem(Index).Checked Then
                frmTranscription.Show
          Else
                frmTranscription.Hide
          End If
        Case 1
          picProperties(0).Visible = mnuViewItem(Index).Checked
          picProperties(0).ZOrder
        Case 2
          picToolbar(0).Visible = mnuViewItem(Index).Checked
          picToolbar(0).ZOrder
        Case 3
          ShowAllObjects LabelCtr, GraphicCtr, TextCtr, VideoCtr, InputCtr
        Case 4
          'frmObjectInfo.Show
    End Select

End Sub

Sub mnuWindowItem_Click (Index As Integer)

    Select Case Index
        Case 0                          ' Cascade
          frmMain.Arrange CASCADE
        Case 1                          ' Tile
          frmMain.Arrange TILE_HORIZONTAL
        Case 2                          ' Arrange
          frmMain.Arrange ARRANGE_ICONS
    End Select

End Sub

Sub NewAudio ()

    ' modifies  picAudio Control Array
    '         AudioCtr, NumObjects, ObjectsOnPage
```

```
    ' effects   Loads and places a new Audio picture on the Screen from.  Creates a new database
    '           object based on the information provided by picAudio

    Dim Object As PictureBox

    AudioCtr = AudioCtr + 1
    NumObjects = NumObjects + 1
    ReDim Preserve ObjectsOnPage(NumObjects)

    Set Object = picAudio(AudioCtr)
    Load Object
    Object.Top = 304 + 5 * (AudioCtr - 1)
    Object.Left = 296 + 5 * (AudioCtr - 1)
    Object.Visible = True
    Object.LinkItem = AUDIO_CODE & "0"
    Object.ZOrder
    Object.Tag = NumObjects

    MakeNewDbObject Object, NumObjects

End Sub

Sub NewCommand ()

    ' modifies  picCommand Control Array
    '           CommandCtr, NumObjects, ObjectsOnPage
    ' effects   Loads and places a new Command picture on the Screen form.  Creates a new
    '           database object based on the information by picCommand.

    Dim Object As PictureBox

    CommandCtr = CommandCtr + 1
    NumObjects = NumObjects + 1
    ReDim Preserve ObjectsOnPage(NumObjects)

    Set Object = picCommand(CommandCtr)
    Load Object

    Object.Top = 80 + 5 * (CommandCtr - 1)
    Object.Left = 16 + 5 * (CommandCtr - 1)
    Object.Visible = True
    Object.LinkItem = COMMAND_CODE & "0"
    Object.ZOrder
    Object.Tag = NumObjects

    MakeNewDbObject Object, NumObjects

End Sub

Sub NewGraphic ()

    ' modifies   picAny Control Array
    '           GraphicCtr, NumObjects, ObjectsOnPage
    ' effects   Loads and places a new graphic picture on the Screen from.  Creates a new database
```

```
    '       object based on the information provided by  picAny

    Dim Object As ACCUSOFT
    'Dim Object2 As PictureBox

    GraphicCtr = GraphicCtr + 1
    NumObjects = NumObjects + 1
    ReDim Preserve ObjectsOnPage(NumObjects)

    Set Object = picAny(GraphicCtr)
    Load Object
    'Set Object2 = picAny2(GraphicCtr)
    'Load Object2
    Object.Top = 200 + 8 * (GraphicCtr - 1)
    Object.Left = 360 + 5 * (GraphicCtr - 1)
    'Object2.Top = Object.Top
    'Object2.Left = Object.Left
    Object.Visible = True
    'Object.Image = ""
    'Object.PrintSize = "0"
    Object.ZOrder
    Object.Tag = NumObjects
    'Object.Print "Graphic " & GraphicCtr

    MakeNewDbObject Object, NumObjects

End Sub

Sub NewInput ()

    ' modifies  txtUserInput Control Array
    '       InputCtr, NumObjects, ObjectsOnPage
    ' effects   Loads and place a new text box on the Screen form.  Creates a new database
    '       object based on the information provided by  picAny

    Dim Object As TextBox

    InputCtr = InputCtr + 1
    NumObjects = NumObjects + 1
    ReDim Preserve ObjectsOnPage(NumObjects)

    Set Object = txtUserInput(InputCtr)
    Load Object
    Object.Top = 320 + 10 * (InputCtr - 1)
    Object.Left = 296 + 10 * (InputCtr - 1)
    Object.Text = "Input " & InputCtr
    Object.Tag = NumObjects
    Object.LinkItem = INPUT_CODE & "0"
    Object.ZOrder
    Object.Visible = True

    MakeNewDbObject Object, NumObjects

End Sub
```

```
Sub NewLabel ()

    ' modifies   lblPage Control Array
    '         LabelCtr, NumObjects, ObjectsOnPage
    ' effects   Loads and places a new label on the Screen from.  Creates a new database
    '         object based on the information provided by  lblPage

    Dim Object As Label

    LabelCtr = LabelCtr + 1
    NumObjects = NumObjects + 1
    ReDim Preserve ObjectsOnPage(NumObjects)

    Set Object = lblPage(LabelCtr)
    Load Object
    Object.Top = 312 + 10 * (LabelCtr - 1)
    Object.Left = 296 + 5 * (LabelCtr - 1)
    Object.Caption = "Label " & LabelCtr
    Object.Visible = True
    Object.Tag = NumObjects
    Object.LinkItem = LABEL_CODE & 0
    Object.ZOrder

    MakeNewDbObject Object, NumObjects

End Sub

Sub NewMetafile ()

    ' modifies  picMetafile Control Array
    '         MetafileCtr, NumObjects, ObjectsOnPage
    ' effects   Loads and places a new metafile picture on the Screen from.  Creates a new database
    '         object based on the information provided by picMetafile.

    Dim Object As PictureBox

    MetafileCtr = MetafileCtr + 1
    NumObjects = NumObjects + 1
    ReDim Preserve ObjectsOnPage(NumObjects)

    Set Object = picMetafile(MetafileCtr)
    Load Object
    Object.Top = 210 + 8 * (MetafileCtr - 1)
    Object.Left = 370 + 5 * (MetafileCtr - 1)
    Object.LinkItem = META_CODE & 0
    Object.Visible = True
    Object.ZOrder
    Object.Tag = NumObjects
    Object.Print "Metafile " & MetafileCtr

    MakeNewDbObject Object, NumObjects

End Sub
```

```
Sub NewTestSub ()

   ' this is a test of windows on the network

   ' another comment.

End Sub

Sub NewText ()

   ' modifies   lblAny Control Array
   '          TextCtr, NumObjects, ObjectsOnPage
   ' effects   Loads and places a new text picture on the Screen from.  Creates a new database
   '          object based on the information provided by  lblAny

   'Dim Object As HEVBLayer
   Dim Object As PictureBox

   TextCtr = TextCtr + 1
   NumObjects = NumObjects + 1
   ReDim Preserve ObjectsOnPage(NumObjects)

   Set Object = lblAny(TextCtr)
   Load Object
   Object.Top = 280 + 8 * (TextCtr - 1)
   Object.Left = 360 + 5 * (TextCtr - 1)
   Object.Visible = True
   Object.Tag = NumObjects
   Object.LinkItem = SCREENTEXT_CODE & 0
   Object.ZOrder
   'HEInitNewDoc Object.hWnd
   'Object.Text = "Text " & TextCtr
   MakeNewDbObject Object, NumObjects

End Sub

Sub NewVideo ()

   ' modifies   picVideo Control Array
   '          VideoCtr, NumObjects, ObjectsOnPage
   ' effects   Loads and places a new Video picture and MCI player on the Screen from.  Creates
   '          a new database object based on the information provided by
   '           picAudio and  mmcPlayer.

   Dim Object As PictureBox, Player As MMControl

   VideoCtr = VideoCtr + 1
   NumObjects = NumObjects + 1
   ReDim Preserve ObjectsOnPage(NumObjects)

   Set Player = mmcPlayer(VideoCtr)
   Set Object = picVideo(VideoCtr)
   Load Player
```

```
    Load Object
    Player.Visible = True
    Player.ZOrder
    Object.Visible = True
    Object.LinkItem = VIDEO_CODE & 0
    Object.ZOrder
    Object.Tag = NumObjects

    MakeNewDbObject Object, NumObjects

End Sub

Sub OpenIndexDb ()

    ' effects   Opens the Edics Database, for use as Index Table.

    'Set IndexDb = OpenDatabase(EDICSDB)
    Set IndexTb = PageDb.OpenTable("Index Database")
    Set TextTb = PageDb.OpenTable("Text Database")

End Sub

Sub OpenObjectTermDb ()

    ' effects   Opens the Edics database, for use as Object Term Table.

    'Set ObjectTermDb = OpenDatabase(EDICSDB)
    Set ObjectTermTb = PageDb.OpenTable("Object Term Database")

End Sub

Sub OpenPageDb ()

    ' modifies  PageDb
    ' effects   Opens the Edics database

    Set PageDb = OpenDatabase(EDICSDB)
    Set TopicTb = PageDb.OpenTable("Topic Database")
    TopicTb.Index = "Page Code"

End Sub

Sub ParseCommandAttribute (InfoBox As ListBox, OutputString As String)

    ' modifies  OutputString
    ' effects   Creates a list of objects to clear, based on the information in InfoBox

    Dim Ctr As Integer

    OutputString = OBJECT_CLEARALL

    For Ctr = 0 To InfoBox.ListCount - 1
        OutputString = OutputString + InfoBox.List(Ctr) + ","
    Next Ctr
```

```
   MsgBox "Clear Update Succesful: " & OutputString, MB_OK

End Sub

Sub ParseObjectInfo (CurObject As ObjectInfo)

   ObjectText(OBJHEIGHT) = Str$(CurObject.Height)
   ObjectText(OBJWIDTH) = Str$(CurObject.Width)
   ObjectText(OBJTIME) = Str$(CurObject.TimeOnScreen)
   ObjectText(INITY) = Str$(CurObject.Initial.Top)
   ObjectText(INITX) = Str$(CurObject.Initial.Left)
   ObjectText(FinalY) = Str$(CurObject.Final.Top)
   ObjectText(FinalX) = Str$(CurObject.Final.Left)
   'lblInfo(9).Caption = CurObject.Contents & " at " & Active.Top & ", " & Active.Left
   'DisplayTransitionCode ObjectType, Left$(CurObject.Transition, 1), cboTransitions

End Sub

Function ParseTrans (CurLbl As Label) As String

   ' effects   Returns a string consisting of the formatting attributes of CurLbl.
   '         The string is in the form of Code1 AttributeListn . . . Coden AttributeListn,
   '         where Coden = {TEXT_FONT, TEXT_COLOR, TEXT_FORMAT}, AttributeListn =
   '         {Attribute1, . . ., Attributen}' and Attribute = {UNDERLINE, UNBOLD,
   '         STRIKETHRU, ITALIC}

   Dim TempTrans As String
   Dim TempFormat As String

   TempTrans = TEXT_FONT + CODEDELIMITERLEFT + Active.FontName +
FORMATDELIMITER & Active.FontSize & CODEDELIMITERRIGHT
   TempTrans = TempTrans + TEXT_COLOR + CODEDELIMITERLEFT + Active.ForeColor +
CODEDELIMITERRIGHT

   TempFormat = TEXT_FORMAT + CODEDELIMITERLEFT
   If Active.FontUnderline Then
         TempFormat = TempFormat + UNDERLINE + FORMATDELIMITER
   End If

   If Not Active.FontBold Then
         TempFormat = TempFormat + UNBOLD + FORMATDELIMITER
   End If

   If Active.FontItalic Then
         TempFormat = TempFormat + ITALIC + FORMATDELIMITER
   End If

   If Active.FontStrikeThru Then
         TempFormat = TempFormat + STRIKETHRU
   End If

   ParseTrans = TempTrans + TempFormat + CODEDELIMITERRIGHT
```

```
End Function

Function ParseTransitionCode (CurObject As ObjectInfo) As Integer

   ' effects   Returns the transition code for the CurObject

   Dim TransCode As String

   TransCode = Left$(CurObject.Transition, 1)

   Select Case TransCode
         Case OBJECT_NOCODE
            ParseTransitionCode = MNUINFOTRANS_NONE
         Case OBJECT_MOVE
            ParseTransitionCode = MNUINFOTRANS_MOVE
         Case OBJECT_DROP
            ParseTransitionCode = MNUINFOTRANS_DROP
         Case OBJECT_ENLARGE
            'ParseTransitionCode = MNUINFOTRANS_ENLARGE
         Case OBJECT_MOVEANDDROP
            ParseTransitionCode = MNUINFOTRANS_MOVEANDDROP
         Case OBJECT_WAIT
            ParseTransitionCode = MNUINFOTRANS_WAIT
         Case OBJECT_LOOP
            ParseTransitionCode = MNUINFOTRANS_LOOP
         Case TEXT_FONT
         'Case TEXT_ROTATE
         '   ParseTransitionCode = 3
   End Select

End Function

Sub picAny_Click (Index As Integer)

   Dim Link As String

   If UserTool = PLAYBACK Then
         If picAny(Index).Tag = LINKEDOBJ Then
            picAny(Index).Visible = False
         Else 'If (picAny(Index).SaveQuality <> LINKUP) Then
            Link = ObjectsOnPage(picAny(Index).Tag).Link
            If Link <> "none" Then
                  picAny(Index).SaveQuality = LINKUP
                  ProcessLink Link, STILL_CODE, Index
                  If LinkIndex <> UNCACHE Then
                     picAny(Index).PrintXs = LinkIndex
                  End If
            End If
         End If
   End If

End Sub

Sub picAny_DblClick (Index As Integer)
```

```
    Dim Link As String

    Set Active = picAny(Index)

    'If UserTool = EDITOR Then
    '   ShowObjectInfo STILL_CODE
    'End If

End Sub

Sub picAny_KeyPress (Index As Integer, KeyAscii As Integer)

    If UserTool = EDITOR Then
        If KeyAscii = KEY_DELETE Or KeyAscii = Asc("d") Or KeyAscii = Asc("D") Then
            ObjectsOnPage(Active.Tag).Contents = DELETED
            UpdateObject ObjectsOnPage(Active.Tag), PageDb.OpenTable("Page Database")
            Unload picAny(Index)
            GraphicCtr = GraphicCtr - 1
            NumObjects = NumObjects - 1
        End If
    End If

End Sub

Sub picAny_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
Single)

    Dim CurObject As ObjectInfo
    Dim Transition As Integer

    Set Active = picAny(Index)

    If UserTool = EDITOR Then
        If Button = LEFT_BUTTON Then
            picAny(Index).Drag BEGIN_DRAG
            DragX = X / Screen.TwipsPerPixelX
            DragY = Y / Screen.TwipsPerPixelY
        ElseIf Button = RIGHT_BUTTON Then
            CurObject = ObjectsOnPage(Active.Tag)
            mnuInfoItem(MNU_XY).Caption = "&X, Y. . ." & Active.Top & ", " & Active.Left
            mnuInfoLocItem(MNU_LOCINITIAL).Caption = "As &Initial: " &
UnparseCoords(CurObject.Initial)
            mnuInfoLocItem(MNU_LOCFINAL).Caption = "As &Final: " &
UnparseCoords(CurObject.Final)
            mnuInfoItem(MNU_HEIGHT).Caption = "&Height. . ." & CurObject.Height
            mnuInfoItem(MNU_WIDTH).Caption = "&Width. . ." & CurObject.Width
            mnuInfoItem(MNUINFO_SEQUENCE).Caption = "&Sequence. . ." & CurObject.Sequence
            Transition = ParseTransitionCode(CurObject)
            TransitionCheck Transition
            If Transition = MNUINFOTRANS_MOVEANDDROP Or Transition =
MNUINFOTRANS_DROP Then
                mnuInfoItem(MNUINFO_TIME).Caption = "Drop &Interval. . ." &
CurObject.TimeOnScreen
```

```
            ElseIf Transition <> MNUINFOTRANS_NONE Then
                mnuInfoItem(MNUINFO_TIME).Caption = "Move &Interval. . ." &
Right$(CurObject.Transition, Len(CurObject.Transition) - 1)
            End If
            mnuInfoItem(MNUINFO_FILE).Caption = "File&name: " & CurObject.Contents
            mnuInfoItem(MNUINFO_ZOOM).Enabled = True
            mnuInfoItem(MNUINFO_ROTATE).Enabled = True
            mnuInfoItem(MNUINFO_SCALE).Enabled = True
            mnuInfoItem(MNUINFO_FLIP).Enabled = True
            dlgGraphic.Filter = "Still Files|*.tga|All Files (*.*)|*.*"
            dlgGraphic.InitDir = Left$(STILLPATH, Len(STILLPATH) - 1)
            PopupMenu mnuInfo, POPUPMENU_LEFTALIGN
        End If
    End If

End Sub

Sub picAny_MouseMove (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
Single)

    Dim NullVal As Integer

    If picAny(Index).PrintSize <> NOLINK Then
        'NullVal = SetCursor(hCurOpen)
        MousePointer = 2
        'ShowLink picAny(Index).Tag, picAny(Index).Left, picAny(Index).Top
    Else
        MousePointer = 0
    End If

End Sub

Sub picAny_MouseUp (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

    picAny(Index).Drag END_DRAG

End Sub

Sub picAudio_DblClick (Index As Integer)

    mnuAudioItem_Click 0

End Sub

Sub picAudio_KeyPress (Index As Integer, KeyAscii As Integer)

    If UserTool = EDITOR Then
        If KeyAscii = KEY_DELETE Or KeyAscii = Asc("d") Or KeyAscii = Asc("D") Then
            ObjectsOnPage(Active.Tag).Contents = DELETED
            UpdateObject ObjectsOnPage(Active.Tag), PageDb.OpenTable("Page Database")
            Unload picAudio(Index)
            AudioCtr = AudioCtr - 1
            NumObjects = NumObjects - 1
        End If
```

```
  End If

End Sub

Sub picAudio_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
Single)

  Dim CurObject As ObjectInfo

  Set Active = picAudio(Index)

  If UserTool = EDITOR Then
      If Button = LEFT_BUTTON Then
        picAudio(Index).Drag BEGIN_DRAG
        DragX = X
        DragY = Y
      ElseIf Button = RIGHT_BUTTON Then
        CurObject = ObjectsOnPage(Active.Tag)
        If CurObject.Transition = AUDIO_SYNC Then
            mnuAudioTransItem(0).Checked = True
            mnuAudioTransItem(1).Checked = False
        Else
            mnuAudioTransItem(0).Checked = False
            mnuAudioTransItem(1).Checked = True
        End If
        mnuAudioItem(MNUAUDIO_SEQUENCE).Caption = "&Sequence. . ." &
CurObject.Sequence
        mnuAudioItem(3).Caption = "&Filename: " & CurObject.Contents
        dlgGraphic.Filter = "Audio Files|*.wav|All Files|*.*"
        dlgGraphic.InitDir = Left$(AUDIOPATH, Len(AUDIOPATH) - 1)
        PopupMenu mnuAudio
      End If
  End If

End Sub

Sub picAudio_MouseUp (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
Single)

  picAudio(Index).Drag END_DRAG

End Sub

Sub picCommand_KeyPress (Index As Integer, KeyAscii As Integer)

  If UserTool = EDITOR Then
      If KeyAscii = KEY_DELETE Or KeyAscii = Asc("d") Or KeyAscii = Asc("D") Then
        ObjectsOnPage(Active.Tag).Contents = DELETED
        UpdateObject ObjectsOnPage(Active.Tag), PageDb.OpenTable("Page Database")
        Unload picAudio(Index)
        CommandCtr = CommandCtr - 1
        NumObjects = NumObjects - 1
      End If
  End If
```

```
End Sub

Sub picCommand_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y
As Single)

   Dim CurObject As ObjectInfo

   Set Active = picCommand(Index)

   If UserTool = EDITOR Then
         If Button = LEFT_BUTTON Then
            picCommand(Index).Drag BEGIN_DRAG
            DragX = X
            DragY = Y
         ElseIf Button = RIGHT_BUTTON Then
            CurObject = ObjectsOnPage(Active.Tag)
            UnParseCommandAttribute CurObject
            mnuCommandItem(MNUCOMMAND_SEQUENCE).Caption = "&Sequence. . ." &
CurObject.Sequence
            PopupMenu mnuCommand
         End If
   End If

End Sub

Sub picMetafile_Click (Index As Integer)

   Dim Link As String

   If UserTool = PLAYBACK Then
         If picMetafile(Index).Tag = LINKEDOBJ Then
            picMetafile(Index).Visible = False
         Else 'If (picMetafile(Index).HelpContextID <> LINKUP) Then
            Link = ObjectsOnPage(picMetafile(Index).Tag).Link
            If Link <> "none" Then
                  picMetafile(Index).HelpContextID = LINKUP
                  ProcessLink Link, META_CODE, Index
                  If LinkIndex <> UNCACHE Then
                     lblAny(Index).LinkTimeout = LinkIndex
                  End If
            End If
         End If
   End If

End Sub

Sub picMetafile_DblClick (Index As Integer)

   Dim Link As String

   Set Active = picMetafile(Index)
```

```
    'If UserTool = EDITOR Then
    '   ShowObjectInfo META_CODE
    'End If


End Sub

Sub picMetafile_KeyPress (Index As Integer, KeyAscii As Integer)

    If UserTool = EDITOR Then
        If KeyAscii = KEY_DELETE Or KeyAscii = Asc("d") Or KeyAscii = Asc("D") Then
            ObjectsOnPage(Active.Tag).Contents = DELETED
            UpdateObject ObjectsOnPage(Active.Tag), PageDb.OpenTable("Page Database")
            Unload picMetafile(Index)
            MetafileCtr = MetafileCtr - 1
            NumObjects = NumObjects - 1
        End If
    End If


End Sub

Sub picMetafile_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
Single)

    Dim CurObject As ObjectInfo
    Dim Transition As Integer

    Set Active = picMetafile(Index)

    If UserTool = EDITOR Then
        If Button = LEFT_BUTTON Then
            picMetafile(Index).Drag BEGIN_DRAG
            DragX = X
            DragY = Y
        ElseIf Button = RIGHT_BUTTON Then
            CurObject = ObjectsOnPage(Active.Tag)
            mnuInfoItem(MNU_XY).Caption = "&X, Y. . ." & Active.Top & ", " & Active.Left
            mnuInfoLocItem(MNU_LOCINITIAL).Caption = "As &Initial: " &
UnparseCoords(CurObject.Initial)
            mnuInfoLocItem(MNU_LOCFINAL).Caption = "As &Final: " &
UnparseCoords(CurObject.Final)
            mnuInfoItem(MNU_HEIGHT).Caption = "&Height. . ." & CurObject.Height
            mnuInfoItem(MNU_WIDTH).Caption = "&Width. . ." & CurObject.Width
            mnuInfoItem(MNUINFO_SEQUENCE).Caption = "&Sequence. . ." & CurObject.Sequence
            'MNUINFOTRANSITEM(MNUINFOTRANS_ROTATE).Enabled = False
            Transition = ParseTransitionCode(CurObject)
            TransitionCheck Transition
            If Transition = MNUINFOTRANS_MOVEANDDROP Or Transition =
MNUINFOTRANS_DROP Then
                mnuInfoItem(MNUINFO_TIME).Caption = "Drop &Interval. . ." &
CurObject.TimeOnScreen
            ElseIf Transition <> MNUINFOTRANS_NONE Then
```

```
                mnuInfoItem(MNUINFO_TIME).Caption = "Move &Interval. . ." &
Right$(CurObject.Transition, Len(CurObject.Transition) - 1)
            End If
            mnuInfoItem(MNUINFO_FILE).Caption = "File&name: " & CurObject.Contents
            mnuInfoItem(MNUINFO_ZOOM).Enabled = True
            mnuInfoItem(MNUINFO_ROTATE).Enabled = True
            mnuInfoItem(MNUINFO_SCALE).Enabled = True
            mnuInfoItem(MNUINFO_FLIP).Enabled = True
            dlgGraphic.Filter = "Meta Files (*.WMF)|*.wmf|Bitmaps (*.BMP)|*.bmp|Dib Files
(*.DIB)|*.dib|Icon Files (*.ICO)|*.ico|All Files (*.*)|*.*"
            dlgGraphic.InitDir = Left$(STILLPATH, Len(STILLPATH) - 1)
            PopupMenu mnuInfo, POPUPMENU_LEFTALIGN
        End If
    End If


End Sub

Sub picMetafile_MouseMove (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
Single)

    Dim NullVal As Integer

    'ShowLink picMetaFile(Index).Tag, picAny(Index).Left, picAny(Index).Top
    'NullVal = SetCursor(hCurOpen)

End Sub

Sub picMetafile_MouseUp (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
Single)

    picMetafile(Index).Drag END_DRAG

End Sub

Sub picProperties_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
Single)

    If UserTool = EDITOR Then
        If Button = LEFT_BUTTON Then
            picProperties(Index).Drag BEGIN_DRAG
            DragX = X
            DragY = Y
        End If
    End If

End Sub

Sub picProperties_MouseUp (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
Single)

    picProperties(Index).Drag END_DRAG

End Sub
```

```
Sub picToolbar_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
Single)

   If UserTool = EDITOR Then
         If Button = LEFT_BUTTON Then
            picToolbar(Index).Drag BEGIN_DRAG
            DragX = X
            DragY = Y
         End If
   End If

End Sub

Sub picToolbar_MouseUp (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
Single)

   picToolbar(Index).Drag END_DRAG

End Sub

Sub picVideo_Click (Index As Integer)

   Dim Link As String

   If UserTool = PLAYBACK Then
         If picVideo(Index).Tag = LINKEDOBJ Then
            picVideo(Index).Visible = False
            mmcPlayer(Index).Visible = False
         Else 'If (picVideo(Index).HelpContextID <> LINKUP) Then
            Link = ObjectsOnPage(picVideo(Index).Tag).Link
            If Link <> "none" Then
                  picVideo(Index).HelpContextID = LINKUP
                  ProcessLink Link, VIDEO_CODE, Index
                  If LinkIndex <> UNCACHE Then
                     lblAny(Index).LinkTimeout = LinkIndex
                  End If
            End If
         End If
   End If

End Sub

Sub picVideo_DblClick (Index As Integer)

   Dim Link As String

   Set Active = picVideo(Index)

   'If UserTool = EDITOR Then
   '   ShowObjectInfo VIDEO_CODE
   'End If

End Sub
```

```
Sub picVideo_KeyPress (Index As Integer, KeyAscii As Integer)

   If UserTool = EDITOR Then
         If KeyAscii = KEY_DELETE Or KeyAscii = Asc("d") Or KeyAscii = Asc("D") Then
            ObjectsOnPage(Active.Tag).Contents = DELETED
            UpdateObject ObjectsOnPage(Active.Tag), PageDb.OpenTable("Page Database")
            Unload picVideo(Index)
            Unload mmcPlayer(Index)
            VideoCtr = VideoCtr - 1
            NumObjects = NumObjects - 1
         End If
   End If

End Sub

Sub picVideo_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
Single)

   Dim CurObject As ObjectInfo
   Dim Transition As Integer

   Set Active = picVideo(Index)

   If UserTool = EDITOR Then
         If Button = LEFT_BUTTON Then
            picVideo(Index).Drag BEGIN_DRAG
            DragX = X
            DragY = Y
         ElseIf Button = RIGHT_BUTTON Then
            CurObject = ObjectsOnPage(Active.Tag)
            'ShowObjectInfo VIDEO_CODE
            mnuInfoItem(MNU_XY).Caption = "&X, Y. . ." & Active.Top & ", " & Active.Left
            mnuInfoLocItem(MNU_LOCINITIAL).Caption = "As &Initial: " &
UnparseCoords(CurObject.Initial)
            mnuInfoLocItem(MNU_LOCFINAL).Caption = "As &Final: " &
UnparseCoords(CurObject.Final)
            mnuInfoItem(MNU_HEIGHT).Caption = "&Height. . ." & CurObject.Height
            mnuInfoItem(MNU_WIDTH).Caption = "&Width. . ." & CurObject.Width
            'mnuInfoItem(MNUINFOTRANS_ROTATE).Enabled = False
            Transition = ParseTransitionCode(CurObject)
            TransitionCheck Transition
            If Transition = MNUINFOTRANS_MOVEANDDROP Or Transition =
MNUINFOTRANS_DROP Then
                  mnuInfoItem(MNUINFO_TIME).Caption = "Drop &Interval. . ." &
CurObject.TimeOnScreen
            ElseIf Transition <> MNUINFOTRANS_NONE Then
                  mnuInfoItem(MNUINFO_TIME).Caption = "Jump &Interval. . ." &
Right$(CurObject.Transition, Len(CurObject.Transition) - 1)
            End If
            mnuInfoItem(MNUINFO_SEQUENCE).Caption = "&Sequence. . ." & CurObject.Sequence
            mnuInfoItem(MNUINFO_ZOOM).Enabled = False
            mnuInfoItem(MNUINFO_ROTATE).Enabled = False
            mnuInfoItem(MNUINFO_SCALE).Enabled = True
```

```
            mnuInfoItem(MNUINFO_FLIP).Enabled = False
            If CurObject.Contents <> "" Then
                mnuInfoItem(MNUINFO_FILE).Caption = "File&name: " & CurObject.Contents
            Else
                mnuInfoItem(MNUINFO_FILE).Caption = "File&name: "
            End If
            dlgGraphic.Filter = "Video Files|*.avi|All Files (*.*)|*.*"
            dlgGraphic.InitDir = Left$(VIDEOPATH, Len(VIDEOPATH) - 1)
            PopupMenu mnuInfo, POPUPMENU_LEFTALIGN
        End If
    End If

End Sub

Sub picVideo_MouseMove (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
Single)

    'ShowLink picVideo(Index).Tag, picAny(Index).Left, picAny(Index).Top

End Sub

Sub picVideo_MouseUp (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
Single)

    picVideo(Index).Drag END_DRAG

End Sub

Sub picVideo_Resize (Index As Integer)

    mmcPlayer(Index).Top = picVideo(Index).Top + picVideo(Index).Height
    mmcPlayer(Index).Left = picVideo(Index).Left
    mmcPlayer(Index).Width = picVideo(Index).Width

End Sub

Sub PrettyPrint (CurLbl As PictureBox, ObjectTerms As ComboBox, ByVal In As String, TextTb As
Table)

    ' modifies  CurLbl
    ' effects   Prints the contents of In in the CurLbl PictureBox.  If a word in the file
    '           In is an object term (Member of Object Term combo box), then the word is
    '           printed in green.  Any formatting codes are also printed.

    Dim CurWord As String, Char As String
    Dim WidthCtr As Integer, HeightCtr As Integer, Start As Integer
    Dim Text As String, TextOutput As Integer
    Dim Result As Integer
    Dim HotSpotCtr As Integer

    Text = ReadFile(In, TextTb)

    HotSpotCtr = 0
    WidthCtr = 0
```

```
    HeightCtr = 1
    Start = 0
    CurLbl.Cls

    Do Until Len(Text) <= 1

        CurWord = PopWord(Text)

        If Len(CurWord) > 1 Then
          'If Member(Left$(CurWord, Len(CurWord) - 1), ObjectTerms) Or (CurWord = "/h ") Then
          '  CurLbl.ForeColor = GREEN
          '  CurWord = PopWord(Text)
          '  If CurWord = "\h " Then
          '     CurLbl.ForeColor = &H0&
          '  Else
          '     HotSpotCtr = HotSpotCtr + 1
          '     'Load lblTextHot(HotSpotCtr)
          '     'lblTextHot(HotSpotCtr).Top = CurLbl.CurrentY - lblTextHot(HotSpotCtr).Height
          '     'lblTextHot(HotSpotCtr).Left = CurLbl.CurrentX
          '     'lblTextHot(HotSpotCtr).Caption = CurWord
          '     'lblTextHot(HotSpotCtr).Visible = True
          '  End If
          'ElseIf CurWord = "\h " Then
          '  CurLbl.ForeColor = &H0&
          '  CurWord = PopWord(Text)
          'End If
        End If

        'CurWord = FormatWord(CurLbl, CurWord)
        WidthCtr = WidthCtr + (CurLbl.TextWidth(CurWord))
        'WidthCtr = WidthCtr + (CurLbl.TextWidth(CurWord) * Len(CurWord))

        If WidthCtr < CurLbl.Width Then
          CurLbl.Print CurWord;
        Else
          HeightCtr = HeightCtr + 1
          'If HeightCtr > CurLbl.ScaleHeight Then
          '  CurLbl.Height = CurLbl.Height + (4 * CurLbl.ScaleHeight)
          'End If
          If CurWord = " " Then
                CurWord = ""
          End If
          CurLbl.Print Chr(13) & CurWord;
          WidthCtr = CurLbl.TextWidth(CurWord)
        End If
    Loop

End Sub

Sub PrintObjects (OutputBox As ListBox)

    ' effects   Prints a list of the objects on the current page in OutputBox, according
    '           to sequence and object contents.
```

```
    Dim Ctr As Integer
    Dim Contents As String, CurObject As ObjectInfo

    OutputBox.Clear

    For Ctr = 1 To NumObjects
        CurObject = ObjectsOnPage(Ctr)
        Contents = CurObject.Contents
        If Left$(Contents, 1) <> COMMAND_CODE Then
          If Left$(Contents, 1) = LABEL_CODE Then
              Contents = Right$(Contents, Len(Contents) - 1)
          End If
          OutputBox.AddItem Contents
        End If
    Next Ctr

End Sub

Sub ProcessLink (ByVal Link As String, SourceType As String, SourceIndex As Integer)

    Dim PagLink As String, ObjLink As String, NamLink As String
    Dim PageName As String, ObjName As String
    Dim ObjDs As Dynaset

    UnparseLink Link, PagLink, ObjLink, NamLink
    LinkIndex = UNCACHE

    If NamLink <> "" Then
        ' process name link
        IndexPtr = FindIndexPtr(Mid$(NamLink, 2, Len(NamLink) - 2), cboIndexNames)
        cmdCreateIndex_Click 5
    Else
        If ObjLink <> "" Then
          ' process object link
          PageName = Mid$(PagLink, 2, Len(PagLink) - 2)
          ObjName = Mid$(ObjLink, 3, Len(ObjLink) - 3)
          SectionDs.Filter = "[Page Code] = '" & PageName & "'" & " and [Object] = '" & ObjName &
"'"
          Set ObjDs = SectionDs.CreateDynaset()
          CreateObjectsArray ObjDs, "LINKEDOBJ"
          LinkPtr = LinkPtr + 1
          LinkIndex = DisplayObject(ObjectsOnPage(NumObjects), LINKEDOBJ)
          CacheObject CachedList(), LinkPtr, ObjName, LinkIndex, SourceType, SourceIndex
        ElseIf PagLink <> "" Then
          ' process page link
          lstIndexSet.Clear
          lstIndexSet.AddItem Mid$(PagLink, 2, Len(PagLink) - 2)
          CreateIndexSet UserIndexList(), "Link " & Link, lstIndexSet
          cmdCreateIndex_Click 5
          ReDim Preserve UserIndexList(UBound(UserIndexList) - 1)
        End If
    End If

End Sub
```

```
Sub PutItems (ListIndex As ListBox, ByVal Items As String, DELIMITER As String)

   ' modifies  ListIndex
   ' effects   Places the "elements" of Items, separated by Delimiter, and adds them to
   '          the ListIndex ListBox.

   Dim In As String

   Do Until Len(Items) < 2
         In = Pop(Items, DELIMITER)
         ListIndex.AddItem Left$(In, Len(In) - 1)
   Loop

End Sub

Sub ReadIndexSets (InputDb As Database)

   ' modifies  IndexList
   ' effects   Reads the Index sets from InputDb.

   Dim IndexSetDs As Dynaset
   Dim IndexName As String, PageItem As String

   IndexName = ""
   lstIndexSet.Clear

   Set IndexSetDs = InputDb.CreateDynaset("Select * from [Index Sets]")

   IndexSetDs.MoveFirst

   Do Until IndexSetDs.EOF
         If IndexSetDs.Fields("Index Name") <> IndexName Then
            If IndexName <> "" Then
                  cmdCreateIndex_Click 3
            End If
            IndexName = IndexSetDs.Fields("Index Name")
            cboIndexNames.Text = IndexName
            lstIndexSet.Clear
         End If
         PageItem = IndexSetDs.Fields("Page Item")
         lstIndexSet.AddItem PageItem
         IndexSetDs.MoveNext
   Loop

End Sub

Sub ResizeControls (ByVal WinH As Integer, ByVal WinW As Integer)

   ' modifies  frmPlayer
   ' effects   Resizes controls, based on WinH and WinW,
   '          which are the maximum possible height and
   '          width of a window in the resolution under
   '          which the program is running.
```

```
    Dim ResizeH As Long, ResizeW As Long, ResizeT As Long, ResizeL As Long
    Dim Ctr As Integer

    'For Ctr = 0 To Controls.Count - 1
    '   If TypeOf Controls(Ctr) Is SSCommand Then
    '      If Controls(Ctr).Top <> 0 Then Controls(Ctr).Top = WinH / (DEFAULTH / Controls(Ctr).Top)
    '      If Controls(Ctr).Left <> 0 Then Controls(Ctr).Left = WinW / (DEFAULTW / Controls(Ctr).Left)
    '      If Controls(Ctr).Height <> 0 Then Controls(Ctr).Height = WinH / (DEFAULTH /
Controls(Ctr).Height)
    '      If Controls(Ctr).Width <> 0 Then Controls(Ctr).Width = WinW / (DEFAULTW /
Controls(Ctr).Width)
    '   End If
    'Next Ctr

    ' Resize the height of the controls, unless WinH is
    ' of the default size.
    If WinH <> DEFAULTH Then
          ResizeH = WinH / RATIOH
          ResizeW = WinW / RATIOW
          ResizeT = WinH / RATIOT
          ResizeL = WinW / RATIOL
          For Ctr = 0 To 5
             cmdInterface(Ctr).Top = ResizeT
             cmdInterface(Ctr).Left = ResizeL + Ctr * ResizeW
             cmdInterface(Ctr).Height = ResizeH
             cmdInterface(Ctr).Width = ResizeW
          Next Ctr

          For Ctr = 0 To 2
             cmdNavigate(Ctr).Top = WinH / (DEFAULTH / cmdNavigate(Ctr).Top) + 1
             cmdNavigate(Ctr).Left = WinW / (DEFAULTW / cmdNavigate(Ctr).Left) + 1
             cmdNavigate(Ctr).Height = WinH / (DEFAULTH / cmdNavigate(Ctr).Height)
             cmdNavigate(Ctr).Width = WinW / (DEFAULTW / cmdNavigate(Ctr).Width)
          Next Ctr
    End If

End Sub

Sub RunIndexSet (ByVal IndexPtr As Integer)

    ' effects   Runs the index set, specified by the IndexPtr position in the UserIndexList
    '           array.

    Caption = "Index Set: " + UserIndexList(IndexPtr).Name
    CreateIndexItems UserIndexItems(), UserIndexList(IndexPtr).List
    picPage(0).Visible = False
    cmdNavigate_Click 1

End Sub

Sub SaveBookmark ()

    Dim Response As Integer
```

```
    Dim AppName As String, Section As String, Entry As Variant, FileName As String


    AppName = "BookMark"
    Section = "Section"
    Entry = PageCtr
    FileName = "c:\edics\users\" + UserName + "\" + UserName + ".ini"
    Response = WritePrivateProfileString(AppName, ByVal "Section", ByVal Str$(PageCtr), FileName)
    If Response Then
        'MsgBox UserName + ".ini updated successfully:  " & CurrentBookmark, MB_OK +
MB_APPLMODAL, "Bookmark"
    Else
        'MsgBox UserName + ".ini update failed.", MB_OK + MB_APPLMODAL, "Bookmark"
    End If

End Sub

Sub SavePage ()

    ' modifies  EDICSDB
    ' effects   Updates the Edics Page Database to include the objects and information on the
    '           current screen.

    Dim Ctr As Integer

    For Ctr = 1 To NumObjects
        UpdateObject ObjectsOnPage(Ctr), PageDb.OpenTable("Page Database")
    Next Ctr

End Sub

Sub ShowAllObjects (ByVal Labels As Integer, ByVal Graphics As Integer, ByVal Texts As Integer,
ByVal Videos As Integer, ByVal Inputs As Integer)

    ' modifies  Me
    ' effects   Reveals all hidden objects

    On Error GoTo ErrShow

    Dim Ctr As Integer

    ' Show all label objects
    For Ctr = 1 To Labels
        lblPage(Ctr).Visible = True
    Next Ctr

    ' Show all graphics objects
    For Ctr = 1 To Graphics
        picAny(Ctr).Visible = True
    Next Ctr

    ' Show all text objects
    For Ctr = 1 To Texts
        lblAny(Ctr).Visible = True
```

```
    Next Ctr

    ' Show all video players and video windows
    For Ctr = 1 To Videos
        mmcPlayer(Ctr).Visible = True
        picVideo(Ctr).Visible = True
    Next Ctr

    For Ctr = 1 To Inputs
        txtUserInput(Ctr).Visible = True
    Next Ctr

ErrShow:
    Exit Sub

End Sub

Sub ShowLink (ByVal ObjectPtr As Integer, ByVal X As Integer, ByVal Y As Integer)

    Dim NameOfLink As String

    'PageCaption = mnuPageLabel.Caption
    'NameOfLink = ObjectsOnPage(ObjectPtr).Link
    'mnuPageLabel.Caption = NameOfLink

    'lblShowLink.Top = Y
    'lblShowLink.Left = X
    'lblShowLink.Cls
    'lblShowLink.Print NameOfLink, 0, 0
    'lblShowLink.Visible = True
    'lblShowLink.ZOrder

End Sub

Sub ShowObjectInfo (ByVal ObjectType As String)

    ' modifies  ObjectsOnPage(Active.Tag)
    ' effects   Displays the Object Info picture box, and places the information associated
    '           with Active in the relevant fields.

    Dim CurObject As ObjectInfo

    CurObject = ObjectsOnPage(Active.Tag)

    frmDialogs!txtInfo(0).Text = CurObject.Height
    frmDialogs!txtInfo(1).Text = CurObject.Width
    frmDialogs!txtInfo(2).Text = CurObject.TimeOnScreen
    frmDialogs!txtInfo(3).Text = CurObject.Initial.Top
    frmDialogs!txtInfo(4).Text = CurObject.Initial.Left
    frmDialogs!txtInfo(5).Text = CurObject.Final.Top
    frmDialogs!txtInfo(6).Text = CurObject.Final.Left
    frmDialogs!lblInfo(9).Caption = CurObject.Contents & " at " & Active.Top & ", " & Active.Left
    DisplayTransitionCode ObjectType, Left$(CurObject.Transition, 1), frmDialogs!cboTransitions
```

```
    'picObjectInfo.Top = Active.Top + 10
    'picObjectInfo.Left = Active.Left + 10
    'picObjectInfo.Visible = True
    'picObjectInfo.ZOrder
    FillListBoxValues NumObjects, CurObject
    picProperties(0).Visible = True
    lstObjectProp(Active.Tag).Visible = True

End Sub

Sub spnVolume_SpinDown ()

    ' modifies  txtVolume
    ' effects   decreases the value of txtVolume.Text by 1
    '           if it is greater than 0.

    If Val(txtVolume.Text) > 0 Then
        txtVolume.Text = Val(txtVolume.Text) - 1
    End If

End Sub

Sub spnVolume_SpinUp ()

    ' modifies  txtVolume
    ' effects   increase the value of txtVolume.Text by 1
    '           if it is less than MAXVOL.

    If Val(txtVolume.Text) < MAXVOL Then
        txtVolume.Text = Val(txtVolume.Text) + 1
    End If

End Sub

Sub StartEditor ()

    ' effects   Starts the Screen Arranger.

    UserName = "editor"
    mnuUserName.Caption = UserName

    ' sound is off for editor
    cmdSound = FRAME_EYE
    cmdSound.Caption = "Sound off"

    mnuViewItem_Click 2
    mnuViewItem(0).Checked = False
    mnuOptionsItem(1).Caption = mnuOptionsItem(1).Caption & Sensitivity
    lblAny(0).BorderStyle = 1
    picAny(0).BorderStyle = 1
    picToolbar(0).Visible = True
    lblCurPage.Visible = True
    lblPage(0).BorderStyle = 1
    picVideo(0).BorderStyle = 1
```

```
    picMetafile(0).BorderStyle = 1
    'cmdBranch(0).BorderStyle = 1
    'imgTrash.Visible = True
    'imgHide.Visible = True
    frmDialogs!cboTransitions.AddItem "None"
    frmDialogs!cboTransitions.AddItem "Move"
    frmDialogs!cboTransitions.AddItem "Drop"
    frmDialogs!cboTransitions.AddItem "Move and Drop"
    cboObjectValue(TRANSTYPE).AddItem "None"
    cboObjectValue(TRANSTYPE).AddItem "Move"
    cboObjectValue(TRANSTYPE).AddItem "Drop"
    cboObjectValue(TRANSTYPE).AddItem "Move and Drop"

    'Show

    DoEvents
    'RaiseWindow frmMainDb
    'frmMainDb.Show
    'RaiseWindow frmTool.hWnd
    'frmTool.Show
    CreateListBoxTitles
    'FillLinks PageDb
    Set SectionDs = PageDb.CreateDynaset(SQLSTMTEDIT)
    PageCtr = GetBookmark()
    MainScreenPrint AllPages(PageCtr)

End Sub

Sub StartPlayback ()

    ' effects   Starts Playback mode.

    Dim Ctr As Integer

    For Ctr = MNUVIEW_TOOLBAR To MNUVIEW_ALLOBJ
        mnuViewItem(Ctr).Visible = False
    Next Ctr

    lblCurPage.Visible = False
    'Show

    DoEvents

    Set SectionDs = PageDb.CreateDynaset(SQLSTMT & DatabaseSection & SQLSTMT2)

    MainScreenPrint AllPages(PageCtr)

End Sub

Sub tmrWait_Timer ()

    ReturnValue = True

    Debug.Print "timer " & ReturnValue
```

```
End Sub

Sub TransFileReadAndPrint (ByVal FileName As String, TextTb As Table)

   ' effects   Reads and prints the information in the transcription file to the
   '           Transcription text box in the Transcription Form.  If the current page
   '           has more than one transcription file, then the previous text is reprinted,
   '           along with the new text.

   Dim LinesFromFile As String
   Dim OldText As String

   If NextPage Then
         OldText = ""
   Else
         OldText = frmTranscription.txtTranscription.Text + Chr(13) + Chr(10)
   End If

   LinesFromFile = ReadFile(FileName, TextTb)

   frmTranscription.txtTranscription.Text = OldText & LinesFromFile

End Sub

Sub TransitionCheck (ByVal TransitionItem As Integer)

   ' modifies  mnuInfoItem Menu Array
   ' effects   Removes checks in mnuInfoItem Menu and places check in position
   '           TransitionItem

   'On Error GoTo skip

   Dim Ctr As Integer

   For Ctr = MNUINFOTRANS_NONE To MNUINFOTRANS_LOOP
         If Ctr = TransitionItem Then
            MNUINFOTRANSITEM(Ctr).Checked = True
         Else
            MNUINFOTRANSITEM(Ctr).Checked = False
         End If
   Next Ctr

skip:
   Exit Sub

End Sub

Sub txtLink_KeyPress (KeyAscii As Integer)

   KeyAscii = 0

End Sub
```

```
Sub txtUserInput_Click (Index As Integer)

   Dim Link As String

   If UserTool = PLAYBACK Then
         If txtUserInput(Index).Tag = LINKEDOBJ Then
            txtUserInput(Index).Visible = False
         Else 'If (txtUserInput(Index).HelpContextID <> LINKUP) Then
            Link = ObjectsOnPage(picVideo(Index).Tag).Link
            If Link <> "none" Then
                  txtUserInput(Index).HelpContextID = LINKUP
                  ProcessLink Link, INPUT_CODE, Index
                  If LinkIndex <> UNCACHE Then
                     txtUserInput(Index).LinkTimeout = LinkIndex
                  End If
            End If
         End If
   End If


End Sub

Sub txtUserInput_KeyPress (Index As Integer, KeyAscii As Integer)

   ObjectsOnPage(txtUserInput(Index).Tag).Contents = INPUT_CODE & txtUserInput(Index).Text
   SavedText = SavedText + Chr(KeyAscii)

End Sub

Sub txtUserInput_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
Single)

   Set Active = txtUserInput(Index)

   If UserTool = EDITOR Then
         If Button = RIGHT_BUTTON Then
            txtUserInput(Index).Drag BEGIN_DRAG
            DragX = X / Screen.TwipsPerPixelX
            DragY = Y / Screen.TwipsPerPixelY
         End If
   End If

End Sub

Sub txtUserInput_MouseUp (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
Single)

   Dim CurObject As ObjectInfo
   Dim Transition As Integer

   txtUserInput(Index).Drag END_DRAG
   If Button = RIGHT_BUTTON Then
         Set Active = txtUserInput(Index)
         CurObject = ObjectsOnPage(Active.Tag)
```

```
        mnuInputItem(MNU_XY).Caption = "&X, Y. . ." & Active.Top & ", " & Active.Left
        mnuInputLocItem(MNU_LOCINITIAL).Caption = "As &Initial: " &
UnparseCoords(CurObject.Initial)
        mnuInputItem(MNUINPUT_SEQUENCE).Caption = "&Sequence. . ." & CurObject.Sequence
        Transition = ParseTransitionCode(CurObject)
        TransitionCheck Transition
        mnuInfoItem(MNUINFO_TIME).Caption = "&Interval. . ." & CurObject.TimeOnScreen
        PopupMenu mnuInput
    End If

End Sub

Sub txtVolume_Change ()

    Dim NullVal As Integer
    Dim Value As String

    Value = "&H" + txtVolume.Text + "0000"
    'Value = Value + Value

    MsgBox Str$(Val(Value)), MB_OK

    NullVal = waveOutSetVolume(mmcSound.hWnd, Val(Value))

    MsgBox Str$(NullVal), MB_OK
End Sub

Sub txtVolume_KeyPress (KeyAscii As Integer)

    If KeyAscii = 13 Then
        cmdVolume_Click 1
    End If

End Sub

Sub UnParseClearList (OutputBox As ListBox, ByVal InputList As String)

    ' effects   Parses and prints the command clear list for the current command object.

    Dim Item As String

    InputList = Right$(InputList, Len(InputList) - 1)

    Do While Len(InputList) > 1
        Item = Pop(InputList, INDEXDELIMITER)
        OutputBox.AddItem Left$(Item, Len(Item) - 1)
    Loop

End Sub

Sub UnParseCommandAttribute (CurObject As ObjectInfo)

    Dim Value As Integer, Ctr As Integer
```

```
    Select Case Left$(CurObject.Transition, 1)
        Case OBJECT_CLEARALL
            Value = 0
        Case OBJECT_CLEARSOME
            Value = 1
        Case OBJECT_NEXTPAGE
            Value = 2
    End Select

    For Ctr = 0 To 2
        mnuCmdAttrItem(Ctr).Checked = (Value = Ctr)
    Next Ctr

End Sub

Sub UnparseLink (Link As String, PagLink As String, ObjectLink As String, NamLink As String)

    Dim LinkType As String
    Dim Temp1 As String, Temp As String

    PagLink = ""
    ObjectLink = ""
    NamLink = ""

    LinkType = Left$(Link, 1)
    If LinkType = NAMELINK Then
        NamLink = Mid$(Link, 2, Len(Link) - 1)
    Else
        PagLink = Mid$(Link, 2, Len(Link) - 1)
        Temp1 = PagLink
        Temp = Pop(Temp1, CODEDELIMITERRIGHT)
        'Temp = ObjectLink
    End If

    If Left$(Temp1, 1) = ObjLink Then
        'ObjectLink = Pop(Link, CODEDELIMITERRIGHT)
        ObjectLink = "O" + Mid$(Temp1, 2, Len(Temp1) - 1)
        PagLink = Temp
    End If

End Sub

Function UnparseObjName (ByVal ObjectType As String) As String

    Dim Temp As String

    Select Case UCase(ObjectType)
        Case VIDEO_CODE
            Temp = "Video"
        Case SCREENTEXT_CODE
            Temp = "Screen Text"
        Case AUDIO_CODE
            Temp = "Audio Clip"
        Case STILL_CODE
```

```
            Temp = "Still"
        Case LABEL_CODE
            Temp = "Label"
        Case INPUT_CODE
            Temp = "Input Box"
        Case COMMAND_CODE
            Temp = "Command Code"
        Case META_CODE
            Temp = "Metafile"
        Case BRANCH_CODE
            Temp = "Branch Code"
    End Select

    UnparseObjName = Temp

End Function

Function UnparseTransCode (ByVal TransitionCode As String) As String

    Dim TempVal As String

    Select Case TransitionCode
        Case OBJECT_NOCODE, "0"
            TempVal = "None"
        Case OBJECT_DROP
            TempVal = "Drop"
        Case OBJECT_MOVE
            TempVal = "Move"
        Case OBJECT_MOVEANDDROP
            TempVal = "Move and Drop"
        Case OBJECT_WAIT
            TempVal = "Wait"
    End Select

    UnparseTransCode = TempVal

End Function

Function UnparseTransitionCode (ByVal TransitionCode As Integer, ByVal TransitionVal As Integer) As
String

    ' effects   returns an unparsed transition code

    Select Case TransitionCode
        Case MNUINFOTRANS_NONE
            UnparseTransitionCode = OBJECT_NOCODE
        Case MNUINFOTRANS_MOVE
            UnparseTransitionCode = OBJECT_MOVE & TransitionVal
        Case MNUINFOTRANS_MOVEANDDROP
            UnparseTransitionCode = OBJECT_MOVEANDDROP & TransitionVal
        Case MNUINFOTRANS_DROP
            UnparseTransitionCode = OBJECT_DROP & TransitionVal
        'Case MNUINFOTRANS_ENLARGE
            UnparseTransitionCode = OBJECT_ENLARGE
```

```
        'Case MNUINFOTRANS_ROTATE
          UnparseTransitionCode = TEXT_ROTATE
        Case MNUINFOTRANS_WAIT
          UnparseTransitionCode = OBJECT_WAIT
        Case MNUINFOTRANS_LOOP
          UnparseTransitionCode = OBJECT_LOOP
    End Select

End Function

Sub UpdateObject (Object As ObjectInfo, OutputTable As Table)

    ' modifies  EDICSDB
    ' effects   Updates the Edics Page Database to include the informaton specified by
    '           Object.  If Object.ID is not found in the database, then a new record is
    '           created to reflect the new Object.  Otherwise, the previous information
    '           is updated.  If Object.Contents = DELETED, the record is deleted from the
    '           database

    Screen.MousePointer = 11

    Dim ID As Integer, NewObject As Integer, CurrentRecord As Variant
    Dim NewDs As Dynaset, CurDs As Dynaset

    NewObject = False
    ID = Object.ID

    'OutputTable.LockEdits = False
    OutputTable.Index = "PrimaryKey"
    OutputTable.Seek "=", ID

    If Object.Contents = DELETED Then
        ' Remove record from database, if the ID doesn't exist, no records are deleted.
        If Not OutputTable.NoMatch Then
          OutputTable.Delete
        End If
    Else
        If OutputTable.NoMatch Then
          OutputTable.AddNew
          NewObject = True
        Else
          OutputTable.Edit
        End If

        OutputTable.Fields("Tutorial Code") = Object.TutorialCode
        OutputTable.Fields("Page Code") = Object.PageCode
        OutputTable.Fields("Sequence") = Object.Sequence
        OutputTable.Fields("Object") = Object.Contents
        OutputTable.Fields("Initial X") = Object.Initial.Left
        OutputTable.Fields("Initial Y") = Object.Initial.Top
        OutputTable.Fields("Final X") = Object.Final.Left
        OutputTable.Fields("Final Y") = Object.Final.Top
        OutputTable.Fields("Height") = Object.Height
        OutputTable.Fields("Width") = Object.Width
```

```
        OutputTable.Fields("Transition Code") = Object.Transition
        OutputTable.Fields("Transition Attribute") = Object.TimeOnScreen
        OutputTable.Fields("Link") = Object.Link
        OutputTable.Update
    End If

    If NewObject Then
        OutputTable.MoveLast
        Object.ID = OutputTable.Fields("ID")
    End If

    OutputTable.Close

    Screen.MousePointer = 0

End Sub

Sub WriteIndexSet (OutputTb As Table, InputArray() As IndexList)

    ' modifies  OutputTb
    ' effects   Writes the Index Set to the table OutputTb

End Sub
```

## FRMMAINM.FRM CODE

```
Option Explicit

Const SQLSTMT = "Select * from [Page Database] where [Tutorial Code] = '"
Const SQLSTMT2 = "' Order by [Page Code] Asc, [Sequence] Asc"

Dim TutorialDb As Database
Dim TutorialDs As Dynaset

Sub cmdQuit_Click (Index As Integer)

    QuitPgm

End Sub

Sub cmdSection_Click (Index As Integer)

    'frmScreen.Image1.Picture = LoadPicture(BKGRNDPATH & lblBgrndObject(Index).Caption)
    DatabaseSection = lblSection(Index).Caption

    Select Case Index
        Case 0, 1, 2
            Tool = PLAYBACK          ' Starts the player
        Case 3
            Tool = EDITOR            ' Starts screen arranger development Tool
    End Select

    Unload frmMainMenu
```

```
    ScreenCtr = ScreenCtr + 1
    ReDim Screens(ScreenCtr)

    Load Screens(ScreenCtr)
    Screens(ScreenCtr).Show

End Sub

Sub Form_Load ()

    ' Load all forms into memory
    'Load frmTranscription
    'Load frmScreen
    'Load frmDefinition
    'Load frmMap
    'Load frmText
    'Load frmTopic
    'load frmLogin

    ' Show Login Screen
    'frmLogin.Show MODAL

    OpenTutorialDb
    PrintLabels

End Sub

Sub OpenTutorialDb ()

    Set TutorialDb = OpenDatabase(EDICSDB)
    Set TutorialDs = TutorialDb.CreateDynaset("Tutorial Database")

End Sub

Sub PrintLabels ()

    Dim Ctr As Integer

    Ctr = 0

    Do Until TutorialDs.EOF
        'Load lblSection(Ctr)
        'Load lblBgrndObject(Ctr)
        'lblSection(Ctr).Top = lblSection(Ctr - 1).Top + 50
        lblSection(Ctr).Caption = TutorialDs.Fields("Tutorial Code").Value
        'lblBgrndObject(Ctr).Caption = TutorialDs.Fields("Background Object").Value
        Ctr = Ctr + 1
        TutorialDs.MoveNext
    Loop

End Sub
```

## FRMMAP.FRM CODE

```
Option Explicit

Sub cmdPrevious_Click ()

   frmMap.Hide

End Sub
```

## FRMMDI.FRM CODE

```
Option Explicit

Sub MDIForm_Load ()

   Load frmBlank
   WS_Video = GetWindowLong(frmBlank.hWnd, GWL_STYLE)

   DocCtr = 0
   ScreenCtr = 1

   ReDim frmState(0)
   ReDim Documents(0)
   ReDim Screens(0)

   ' Put login form on screen.
   'frmLogin.Move Screen.Width / 3.5, Screen.Height / 4.5
   frmMainMenu.Move Screen.Width / 4.5, Screen.Height / 4.5

End Sub
```

## FRMTRANS.FRM CODE

```
   ' If the button is pressed, display the up bitmap if the
   ' mouse is dragged outside the button's area, otherwise
   ' display the up bitmap
   Select Case Button
   Case 1
      If X <= 0 Or X > imgCopyButton.Width Or Y < 0 Or Y > imgCopyButton.Height Then
         imgCopyButton.Picture = imgCopyButtonUp.Picture
      Else
         imgCopyButton.Picture = imgCopyButtonDn.Picture
      End If
   End Select
End Sub

Sub imgCopyButton_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)
   imgCopyButton.Picture = imgCopyButtonUp.Picture
End Sub

Sub imgCutButton_Click ()
   imgCutButton.Refresh
   EditCutProc
End Sub
```

```
Sub imgCutButton_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
  imgCutButton.Picture = imgCutButtonDn.Picture
End Sub

Sub imgCutButton_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)
  ' If the button is pressed, display the up bitmap if the
  ' mouse is dragged outside the button's area, otherwise
  ' display the up bitmap
  Select Case Button
  Case 1
    If X <= 0 Or X > imgCutButton.Width Or Y < 0 Or Y > imgCutButton.Height Then
      imgCutButton.Picture = imgCutButtonUp.Picture
    Else
      imgCutButton.Picture = imgCutButtonDn.Picture
    End If
  End Select
End Sub

Sub imgCutButton_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)
  imgCutButton.Picture = imgCutButtonUp.Picture
End Sub

Sub imgFileNewButton_Click ()
  imgFileNewButton.Refresh
  FileNew
End Sub

Sub imgFileNewButton_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
  imgFileNewButton.Picture = imgFileNewButtonDn.Picture
End Sub

Sub imgFileNewButton_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)
  ' If the button is pressed, display the up bitmap if the
  ' mouse is dragged outside the button's area, otherwise
  ' display the up bitmap
  Select Case Button
  Case 1
    If X <= 0 Or X > imgFileNewButton.Width Or Y < 0 Or Y > imgFileNewButton.Height Then
      imgFileNewButton.Picture = imgFileNewButtonUp.Picture
    Else
      imgFileNewButton.Picture = imgFileNewButtonDn.Picture
    End If
  End Select
End Sub

Sub imgFileNewButton_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)
  imgFileNewButton.Picture = imgFileNewButtonUp.Picture
End Sub

Sub imgFileOpenButton_Click ()
  imgFileOpenButton.Refresh
  FOpenProc
End Sub
```

```
Sub imgFileOpenButton_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
   imgFileOpenButton.Picture = imgFileOpenButtonDn.Picture
End Sub

Sub imgFileOpenButton_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)
   ' If the button is pressed, display the up bitmap if the
   ' mouse is dragged outside the button's area, otherwise
   ' display the up bitmap
   Select Case Button
   Case 1
      If X <= 0 Or X > imgFileOpenButton.Width Or Y < 0 Or Y > imgFileOpenButton.Height Then
         imgFileOpenButton.Picture = imgFileOpenButtonUp.Picture
      Else
         imgFileOpenButton.Picture = imgFileOpenButtonDn.Picture
      End If
   End Select
End Sub

Sub imgFileOpenButton_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)
   imgFileOpenButton.Picture = imgFileOpenButtonUp.Picture
End Sub

Sub imgPasteButton_Click ()
   imgPasteButton.Refresh
   EditPasteProc
End Sub

Sub imgPasteButton_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
   imgPasteButton.Picture = imgPasteButtonDn.Picture
End Sub

Sub imgPasteButton_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)
   ' If the button is pressed, display the up bitmap if the
   ' mouse is dragged outside the button's area, otherwise
   ' display the up bitmap
   Select Case Button
   Case 1
      If X <= 0 Or X > imgPasteButton.Width Or Y < 0 Or Y > imgPasteButton.Height Then
         imgPasteButton.Picture = imgPasteButtonUp.Picture
      Else
         imgPasteButton.Picture = imgPasteButtonDn.Picture
      End If
   End Select
End Sub

Sub imgPasteButton_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)
   imgPasteButton.Picture = imgPasteButtonUp.Picture
End Sub

Sub MDIForm_Load ()
   ' Application starts here (Load event of Startup form).
   Show
   ' Always set working directory to directory containing the application.
```

```
    ChDir App.Path

    'Initialize document form arrays, and show first document.
    ReDim Document(1)
    ReDim FState(1)
    Document(1).Tag = 1
    FState(1).Dirty = False
    Document(1).Show

    ' Read MDINOTE.INI and set recent file menu items appropriately
    GetRecentFiles
End Sub

Sub MDIForm_Unload (Cancel As Integer)
    ' If the Unload was not canceled (in the QueryUnload events for the Notepad forms)
    ' there will be no document windows left, so go ahead and end the application.

    If Not AnyPadsLeft() Then
        End
    End If

End Sub

Sub mnuFExit_Click ()
    End
End Sub

Sub mnuFNew_Click ()
    FileNew
End Sub

Sub mnuFOpen_Click ()
    FOpenProc
End Sub

Sub mnuOptions_Click ()
    mnuOToolbar.Checked = frmMDI!picToolbar.Visible
End Sub

Sub mnuOToolbar_Click ()
    OptionsToolbarProc Me
End Sub

Sub mnuRecentFile_Click (index As Integer)
  OpenFile (mnuRecentFile(index).Caption)
  ' Update recent files list.
  GetRecentFiles
End Sub
```

## NOTEPAD.FRM CODE

```
Option Explicit
```

```
'Type FormState
'   Deleted As Integer
'   Dirty As Integer
'   Color As Long
'End Type

Dim gFindString, gFindCase As Integer, gFindDirection As Integer
Dim gCurPos As Integer, gFirstTime As Integer
Dim TextChanged As Integer



Sub EditCopyProc ()
   ' Copy selected text to Clipboard.
   ClipBoard.SetText txtEdit.SelText
End Sub

Sub EditCutProc ()

   ' Copy selected text to Clipboard.
   ClipBoard.SetText txtEdit.SelText

   ' Delete selected text.
   txtEdit.SelText = ""

End Sub

Sub EditPasteProc ()

   ' Place text from Clipboard into active control.
   txtEdit.SelText = ClipBoard.GetText()

End Sub

Sub FileNew ()

   Dim Msg As String, Filename As String, NL As String
   Dim Response As Integer

   If TextChanged Then
         Filename = Caption
         NL = Chr$(10) & Chr$(13)
         Msg = "The text in [" & Filename & "] has changed." & NL & "Do you want to save the
changes?"
         Response = MsgBox(Msg, MB_YESNOCANCEL, Filename)
         If Response = IDYES Then
            Do
                  Filename = GetFileName()
            Loop Until Filename <> ""
            SaveFileAs (Filename)
         ElseIf Response = IDCANCEL Then
            Exit Sub
         End If
   End If
```

```
    txtEdit.Text = ""
    TextChanged = False
    Caption = "Untitled"

End Sub

Sub FindIt ()
    Dim start, pos, findstring, sourcestring, Msg, Response, Offset

    If (gCurPos = txtEdit.SelStart) Then
        Offset = 1
    Else
        Offset = 0
    End If

    If gFirstTime Then Offset = 0

    start = txtEdit.SelStart + Offset

    If gFindCase Then
        findstring = gFindString
        sourcestring = txtEdit.Text
    Else
        findstring = UCase(gFindString)
        sourcestring = UCase(txtEdit.Text)
    End If

    If gFindDirection = 1 Then
        pos = InStr(start + 1, sourcestring, findstring)
    Else
        For pos = start - 1 To 0 Step -1
            If pos = 0 Then Exit For
            If Mid(sourcestring, pos, Len(findstring)) = findstring Then Exit For
        Next
    End If

    ' If string is found
    If pos Then
        txtEdit.SelStart = pos - 1
        txtEdit.SelLength = Len(findstring)
    Else
        Msg = "Cannot find " & Chr(34) & gFindString & Chr(34)
        Response = MsgBox(Msg, 0, App.Title)
    End If

    gCurPos = txtEdit.SelStart
    gFirstTime = False

End Sub

Sub FOpenProc ()

    Dim RetVal
```

```
    On Error Resume Next
    Dim OpenFileName As String

    dlgTextEdit.Filename = ""
    dlgTextEdit.Action = 1
    dlgTextEdit.Filter = "Text Files|*.txt|All Files|*.*"
    dlgTextEdit.InitDir = Left$(TEXTPATH, Len(TEXTPATH) - 1)

    If Err <> 32755 Then 'user pressed cancel
        OpenFileName = dlgTextEdit.Filename
        OpenFile (OpenFileName)
    End If

End Sub

Sub Form_Load ()

    TextChanged = False

End Sub

Sub Form_QueryUnload (Cancel As Integer, UnloadMode As Integer)

    Dim Msg, Filename, NL
    Dim Response As Integer

    If TextChanged Then
        Filename = Caption
        NL = Chr$(10) & Chr$(13)
        Msg = "The text in [" & Filename & "] has changed."
        Msg = Msg & NL
        Msg = Msg & "Do you want to save the changes?"
        Response = MsgBox(Msg, 51, Caption)
        Select Case Response
        ' User selects Yes
        Case 6
            'Get the filename to save the file
            Filename = GetFileName()
            'If the user did notspecify a file name,
            'cancel the unload; otherwise, save it.
            If Filename = "" Then
                Cancel = True
            Else
                SaveFileAs (Filename)
            End If


        ' User selects No
        ' Ok to unload
        Case 7
            Cancel = False
        ' User selects Cancel
        ' Cancel the unload
        Case 2
```

```
            Cancel = True
        End Select
    End If
End Sub

Sub Form_Resize ()
    If windowstate <> 1 And ScaleHeight <> 0 Then
        txtEdit.Visible = False
        txtEdit.Height = ScaleHeight
        txtEdit.Width = ScaleWidth
        txtEdit.Visible = True
    End If
End Sub

Function GetFileName ()

    'Displays a Save As dialog and returns a file name
    'or an empty string if the user cancels

    On Error Resume Next

    dlgTextEdit.Filename = ""
    dlgTextEdit.Filter = "Text Files|*.txt|All Files|*.*"
    dlgTextEdit.InitDir = Left$(TEXTPATH, Len(TEXTPATH) - 1)
    dlgTextEdit.Action = 2

    If Err <> 32755 Then      'User cancelled dialog
        GetFileName = dlgTextEdit.Filename
    Else
        GetFileName = ""
    End If

End Function

Sub imgBoldButton_Click ()

    imgBoldButton.Refresh

End Sub

Sub imgBoldButton_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)

    imgBoldButton.Picture = imgBoldDn.Picture

End Sub

Sub imgBoldButton_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    Select Case Button
        Case 1
            If X <= 0 Or X > imgBoldButton.Width Or Y < 0 Or Y > imgBoldButton.Height Then
                imgBoldButton.Picture = imgBoldUp.Picture
            Else
                imgBoldButton.Picture = imgBoldDn.Picture
```

```
            End If
    End Select

End Sub

Sub imgBoldButton_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)

    imgBoldButton.Picture = imgBoldUp.Picture

End Sub

Sub imgCopyButton_Click ()
    imgCopyButton.Refresh
    EditCopyProc
End Sub

Sub imgCopyButton_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
    imgCopyButton.Picture = imgCopyButtonDn.Picture
End Sub

Sub imgCopyButton_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)
    ' If the button is pressed, display the up bitmap if the
    ' mouse is dragged outside the button's area, otherwise
    ' display the up bitmap
    Select Case Button
    Case 1
        If X <= 0 Or X > imgCopyButton.Width Or Y < 0 Or Y > imgCopyButton.Height Then
            imgCopyButton.Picture = imgCopyButtonUp.Picture
        Else
            imgCopyButton.Picture = imgCopyButtonDn.Picture
        End If
    End Select
End Sub

Sub imgCopyButton_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)
    imgCopyButton.Picture = imgCopyButtonUp.Picture
End Sub

Sub imgCutButton_Click ()
    imgCutButton.Refresh
    EditCutProc
End Sub

Sub imgCutButton_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
    imgCutButton.Picture = imgCutButtonDn.Picture
End Sub

Sub imgCutButton_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)
    ' If the button is pressed, display the up bitmap if the
    ' mouse is dragged outside the button's area, otherwise
    ' display the up bitmap
    Select Case Button
    Case 1
        If X <= 0 Or X > imgCutButton.Width Or Y < 0 Or Y > imgCutButton.Height Then
```

```
            imgCutButton.Picture = imgCutButtonUp.Picture
        Else
            imgCutButton.Picture = imgCutButtonDn.Picture
        End If
    End Select
End Sub

Sub imgCutButton_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)
    imgCutButton.Picture = imgCutButtonUp.Picture
End Sub

Sub imgFileNewButton_Click ()

    imgFileNewButton.Refresh
    FileNew

End Sub

Sub imgFileNewButton_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)

    imgFileNewButton.Picture = imgFileNewButtonDn.Picture

End Sub

Sub imgFileNewButton_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    ' If the button is pressed, display the up bitmap if the
    ' mouse is dragged outside the button's area, otherwise
    ' display the up bitmap

    Select Case Button
        Case 1
            If X <= 0 Or X > imgFileNewButton.Width Or Y < 0 Or Y > imgFileNewButton.Height Then
                imgFileNewButton.Picture = imgFileNewButtonUp.Picture
            Else
                imgFileNewButton.Picture = imgFileNewButtonDn.Picture
            End If
    End Select

End Sub

Sub imgFileNewButton_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)

    imgFileNewButton.Picture = imgFileNewButtonUp.Picture

End Sub

Sub imgFileOpenButton_Click ()

    imgFileOpenButton.Refresh
    FOpenProc

End Sub
```

```
Sub imgFileOpenButton_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
   imgFileOpenButton.Picture = imgFileOpenButtonDn.Picture
End Sub

Sub imgFileOpenButton_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)
   ' If the button is pressed, display the up bitmap if the
   ' mouse is dragged outside the button's area, otherwise
   ' display the up bitmap
   Select Case Button
   Case 1
         If X <= 0 Or X > imgFileOpenButton.Width Or Y < 0 Or Y > imgFileOpenButton.Height Then
            imgFileOpenButton.Picture = imgFileOpenButtonUp.Picture
         Else
            imgFileOpenButton.Picture = imgFileOpenButtonDn.Picture
         End If
   End Select
End Sub

Sub imgFileOpenButton_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)
   imgFileOpenButton.Picture = imgFileOpenButtonUp.Picture
End Sub

Sub imgItalicButton_Click ()

   imgItalicButton.Refresh

End Sub

Sub imgItalicButton_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)

   imgItalicButton.Picture = imgItalicDn.Picture

End Sub

Sub imgItalicButton_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

   Select Case Button
         Case 1
            If X <= 0 Or X > imgItalicButton.Width Or Y < 0 Or Y > imgItalicButton.Height Then
                  imgItalicButton.Picture = imgItalicUp.Picture
            Else
                  imgItalicButton.Picture = imgItalicDn.Picture
            End If
   End Select

End Sub

Sub imgItalicButton_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)

   imgItalicButton.Picture = imgItalicUp.Picture

End Sub

Sub imgPasteButton_Click ()
```

```
  imgPasteButton.Refresh
  EditPasteProc

End Sub

Sub imgPasteButton_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
  imgPasteButton.Picture = imgPasteButtonDn.Picture
End Sub

Sub imgPasteButton_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)
  ' If the button is pressed, display the up bitmap if the
  ' mouse is dragged outside the button's area, otherwise
  ' display the up bitmap
  Select Case Button
  Case 1
        If X <= 0 Or X > imgPasteButton.Width Or Y < 0 Or Y > imgPasteButton.Height Then
          imgPasteButton.Picture = imgPasteButtonUp.Picture
        Else
          imgPasteButton.Picture = imgPasteButtonDn.Picture
        End If
  End Select
End Sub

Sub imgPasteButton_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)
  imgPasteButton.Picture = imgPasteButtonUp.Picture
End Sub

Sub imgUnderlineButton_Click ()

  imgUnderlineButton.Refresh

End Sub

Sub imgUnderlineButton_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)

  imgUnderlineButton.Picture = imgUnderlineDn.Picture

End Sub

Sub imgUnderlineButton_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

  Select Case Button
        Case 1
          If X <= 0 Or X > imgUnderlineButton.Width Or Y < 0 Or Y > imgUnderlineButton.Height
Then
                imgUnderlineButton.Picture = imgUnderlineUp.Picture
          Else
                imgUnderlineButton.Picture = imgUnderlineDn.Picture
          End If
  End Select

End Sub
```

```
Sub imgUnderlineButton_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)

   imgUnderlineButton.Picture = imgUnderlineUp.Picture

End Sub

Sub mnuCharItem_Click (Index As Integer)

   Select Case Index
         Case 0
         Case 1
         Case 2
         Case 3
         Case 5
         Case 6
   End Select

End Sub

Sub mnuECopy_Click ()
   EditCopyProc
End Sub

Sub mnuECut_Click ()
   EditCutProc
End Sub

Sub mnuEDelete_Click ()
  ' If cursor is not at the end of the notepad.
  If screen.ActiveControl.SelStart <> Len(screen.ActiveControl.Text) Then
    ' If nothing is selected, extend selection by one.
    If screen.ActiveControl.SelLength = 0 Then
      screen.ActiveControl.SelLength = 1
      ' If cursor is on a blank line, extend selection by two.
      If Asc(screen.ActiveControl.SelText) = 13 Then
            screen.ActiveControl.SelLength = 2
      End If
    End If
    ' Delete selected text.
    screen.ActiveControl.SelText = ""
  End If
End Sub

Sub mnuEPaste_Click ()
   EditPasteProc
End Sub

Sub mnuESelectAll_Click ()
   txtEdit.SelStart = 0
   txtEdit.SelLength = Len(txtEdit.Text)
End Sub

Sub mnuETime_Click ()
   Dim TimeStr As String, DateStr As String
```

```
    txtEdit.SelText = Now
End Sub

Sub mnuFileItem_Click (Index As Integer)

    Select Case Index
        Case 0
            FileNew
        Case 1
            FOpenProc
        Case 2
            Dim Filename As String

            If Left(Me.Caption, 8) = "Untitled" Then
                ' The file hasn't been saved yet,
                ' get the filename, then call the
                ' save procedure
                Filename = GetFileName()
            Else
                ' The caption contains the name of the open file
                Filename = Me.Caption
            End If
            ' call the save procedure, if Filename = Empty then
            ' the user selected Cancel in the Save As dialog, otherwise
            ' save the file
            If Filename <> "" Then
                SaveFileAs Filename
            End If
        Case 3
            Dim SaveFileName As String

            SaveFileName = GetFileName()
            If SaveFileName <> "" Then SaveFileAs (SaveFileName)
            ' Update the recent files menu
        Case 5
            Unload Me
    End Select

End Sub

Sub mnuOptionsItem_Click (Index As Integer)

    mnuOptionsItem(Index).Checked = Not (mnuOptionsItem(Index).Checked)

    Select Case Index
        Case 0
            picToolbar.Visible = mnuOptionsItem(Index).Checked
        Case 1
            picPreview.Visible = mnuOptionsItem(Index).Checked
    End Select

End Sub
```

```
Sub mnuSFind_Click ()

   If txtEdit.SelText <> "" Then
         frmFind!txtFind.Text = Me!txtEdit.SelText
   Else
         'frmFind!txtFind.Text = FindString
   End If
   gFirstTime = True
   frmFind.Show
End Sub

Sub mnuSFindNext_Click ()
   If Len(gFindString) > 0 Then
         FindIt
   Else
         mnuSFind_Click
   End If
End Sub

Sub mnuWArrange_Click ()
   frmMain.Arrange ARRANGE_ICONS
End Sub

Sub mnuWCascade_Click ()
   frmMain.Arrange CASCADE
End Sub

Sub mnuWTile_Click ()
   frmMain.Arrange TILE_HORIZONTAL
End Sub

Sub OpenFile (Filename)

   Dim TextIn As String

   On Error Resume Next
   ' open the selected file
   Open Filename For Input As #1
   If Err Then
         MsgBox "Can't open file: " + Filename
         Exit Sub
   End If
   Close #1

   ' change mousepointer to an hourglass
   screen.MousePointer = 11
   TextIn = ReadFile(Filename)

   ' change form's caption and display new text
   Caption = UCase$(Filename)
   txtEdit.Text = TextIn
   TextChanged = False

   ' reset mouse pointer
```

```
    screen.MousePointer = 0

End Sub

Sub SaveFileAs (Filename)

    On Error Resume Next

    Dim Contents As String

    ' open the file
    Open Filename For Output As #1
    ' put contents of the notepad into a variable
    Contents = frmNotePad.txtEdit.Text
    ' display hourglass
    screen.MousePointer = 11
    ' write variable contents to saved file
    Print #1, Contents
    Close #1
    ' reset the mousepointer
    screen.MousePointer = 0
    ' set the Notepad's caption

    If Err Then
        MsgBox Error, 48, App.Title
    Else
        frmNotePad.Caption = UCase$(Filename)
        ' reset the dirty flag
        TextChanged = False
    End If

End Sub

Sub txtEdit_Change ()

    TextChanged = True

End Sub
```

# APPENDIX C: DATABASES STRUCTURE

Archive table structure

| Field name | Data type |
|---|---|
| Archive | Text |
| Description | Text |
| Archive source | Text |
| Archive object | OLE Object |
| Transcription code | Text |

Index table structure

| Field name | Data type |
|---|---|
| Term | Text |
| Object 1 | Memo |
| Object 2 | Memo |
| Object 3 | Memo |

Index-set table structure

| Field name | Data type |
|---|---|
| Index Name | Text |
| Page Item | Text |

Object-term table structure

| Field name | Data type |
|---|---|
| ID | Counter |
| Object Term | Text |
| Object Code | Text |

Page table structure

| Field name | Data type |
|---|---|
| ID | Counter |
| Tutorial Code | Text |
| Page Code | Text |
| Sequence | Number |
| Object | Text |
| Initial X | Number |
| Initial Y | Number |
| Final X | Number |
| Final Y | Number |
| Height | Number |
| Width | Number |
| Transition Code | Text |
| Transition Attribute | Number |
| Link | Text |

Text table structure

| Field name | Data type |
|---|---|
| Text code | Text |
| Contents | Memo |

Topic table structure

| Field name | Data type |
|---|---|
| Page Code | Text |
| Topic | Text |

Tutorial table structure

| Field name | Data type |
|---|---|
| Tutorial Code | Text |
| Background Object | Text |

# APPENDIX D: DESIGN PROCESS CHAPTER OUTLINE

**Legend**

Basic outline level 1 (⇒ go to next section)
- basic outline level 2
  - *basic outline level 3*
    - *basic outline level 4*
      - references
      - media
      - text or audio information
      - notebook pages
      - ☒ signifies a button to access something
      - "{}" represents an object code

Introduction
- screen:
  - *need symbol/icon for the hydrofoil or the design process*
  - *intro. text/problem statement*
- Introduction statement

  {t215} The purpose of this program is to expose you to the mechanical design process by putting you inside a real design problem. One of our major goals is for you to gain the understanding that design is sometimes messy and mistakes are made. Therefore we have taken every opportunity to show our mistakes and why they came about.

  We selected the design of a human-powered hydrofoil because it contains all the basic elements that we want to show. We hope that you will find the project as exciting as we did. As you go through the program think of yourself as a member of the design team and what steps you would take.
- Statement that hydrofoil was built, works and still needs work. Mention the boats name: the Skeeter.
- Statement on building a team and the need for diversity
- ☒ Introduce all people involved
  - *small picture of each person*
    - *use existing video whenever possible*
      - ☒{S883} Ernesto Blanco {S896}
        - Adjunct Professor of Mechanical Engineering at MIT
        - design advice and informal design review
      - ☒{S884} Mark Drela {S897}
        - Professor of Aeronautical Engineering at MIT
        - main design team
        - worked the Decavitator project
        - worked on the Daedalus project
        - set men's world speed record for human-powered travel on water from 11/2/91.
        - responsible for the foil and propeller design.
        - composite work
        - machining
        - mechanical design

participated in design review

☒{S885} Sepehr Kiani {S898}
MIT M.S. student in Mechanical Engineering
main design team
part of MS thesis
conceptual research and work
layout, assembly and detail design work
composite and machining
participated in design review

☒{S886} Matthew Wall {S899}
MIT Ph.D. student in Mechanical Engineering
main design team
worked the Decaitator project
composite work
machining
mechanical design
participated in design review

☒{S909} David Wilson {S900}
Professor of Mechanical Engineering at MIT
faculty overall project advisor
originated work on human powered hydrofoils in 1979
participated in design review

☒{S882} Julie Yang {S881}
MIT M.S. student in Mechanical Engineering
helped with composite work and testing

☒{S888} Li Shu {S901}
MIT Ph.D. student in Mechanical Engineering
helped with composite work

☒{S879} Ben Lindes {S880}
MIT Ph.D. student in Mechanical Engineering
helped with composite work and testing

☒{S889} Ian Kaye {S902}
Northeastern University B.S. student in Mechanical Engineering
helped with fabrication and machining

☒{S890} Fred Ackerman {S903}
MIT B.S. student in Mechanical Engineering
UROP (undergraduate research opportunity program) student
machining

☒{S891} Tom Washington {S904}
MIT Ph.D. student in Aeronautical Engineering
worked the Decavitator project
composite work
designed made skimmer molds

☒{S892} Harold "Guppy" Youngren {S905}
MIT M.S. in Aeronautical Engineering 1990
late night help and encouragement

☒{S893} Woodie Flowers {S906}
Professor of Mechanical Engineering at MIT
participated in design review

☒{S894} Marc Schafer {S907}
MIT M.S. student in Mechanical Engineering
worked the Decavitator project

participated in design review

☒{S895} Ted VanDusen / Composite Engineering {S908}
used men's Olympic kayak mold and expertise

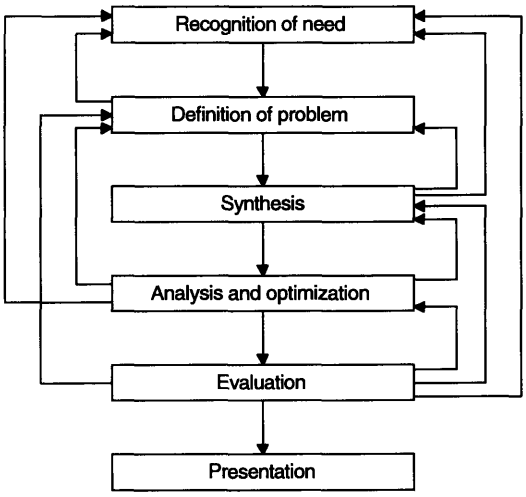
☒{S910} Shin Choi {S909}
MIT B.S. student in Mechanical Engineering, 1994
helped with hull fabrication

☒ {S833} Side bar on the history of boating and hydrofoils, similar to that in Scientific American article.

Brooks, Alec N.; Abbott, Allan V.; Wilson, David Gordon; "Human-powered Watercraft," Scientific American, Volume 255, Number 6, New York, NY, © 1986.

Define problem and plan project (3/28/92)

☒ {S843}

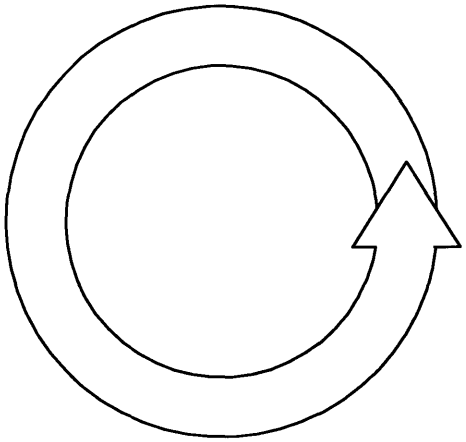



Phases of design from Shigley

{S837}

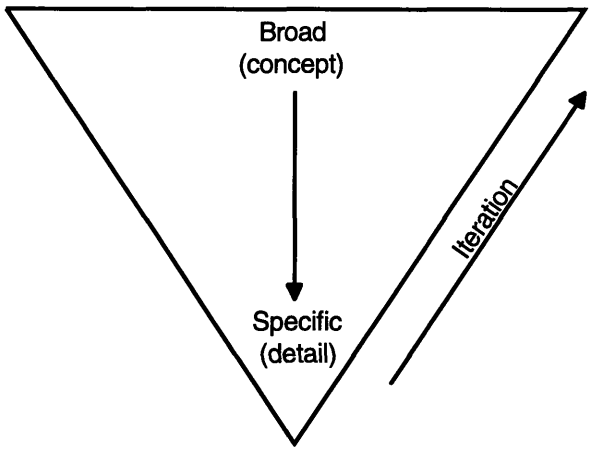
*Shigley, Joseph Edward and Mischke, Charles R.; Mechanical Engineering Design, McGraw-Hill Book Company, New York © 1989.*

☒ {S844}



{S835}

☒ {S845}




{S836}

*On design process:*

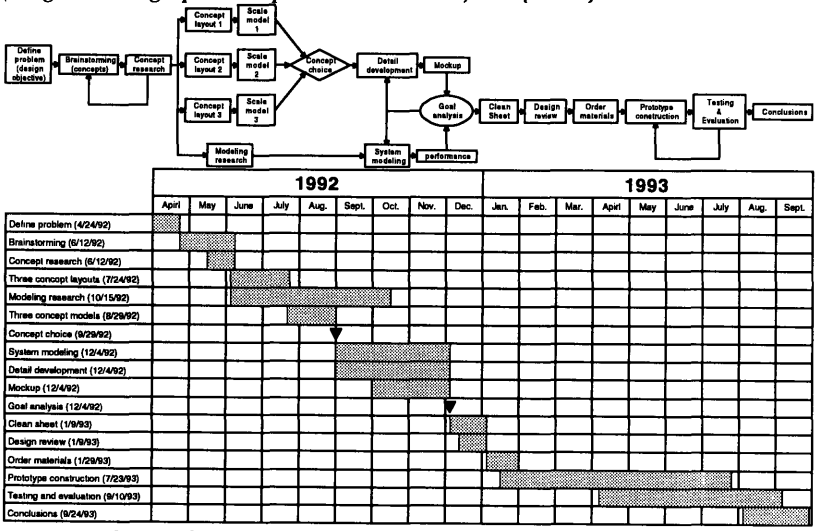
{t216} There are as many approaches to the process of design as there are design engineers and design problems. Consider these three general process flow diagrams. A key element of all these diagrams is iteration, i.e., backing up and trying another design path. No matter how much knowledge and experience one has, design is a large set of assumptions. The only way to know an assumption will work is to test it, either by building or modeling (usually both are needed). Sometimes a bad assumption will not show itself until the design becomes somewhat refined. The difficulty lies in figuring out how far to go before giving up and going "back to the drawing board" for another iteration.

Professor Ernesto Blanco always tells his students not to use a sharp pencil when sketching an idea, "because your thoughts are not sharp yet." Keep your thoughts free and open until your design is ready for detailing.

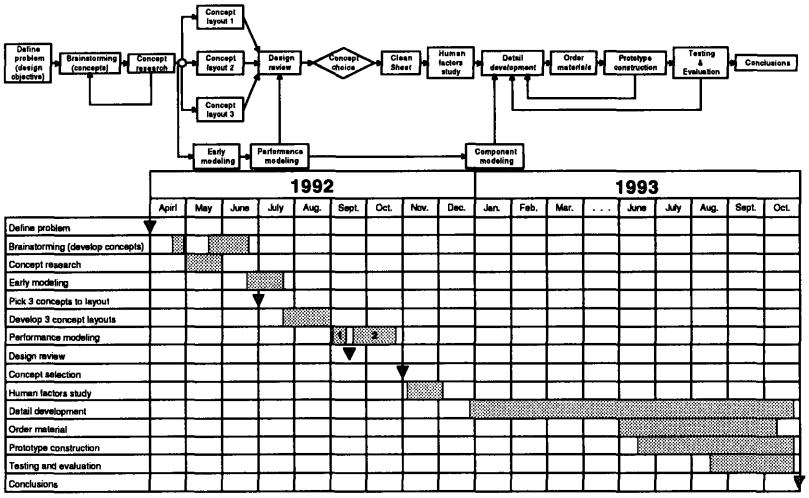
*On hydrofoil design process.*

{t217} A design engineer at Apple once said about development of the Apple PowerBook, "make lots of plans and change them often." This is the original plan for the hydrofoil project; notice how we go from broad concept to refined detail. As you follow along in this program note the departures from the original process plan.

*{S841} (original design process plan and timeline) => {S838}*




*{S840} process plan and timeline actually followed => {S839}*




*Defining the project:*

| | March 1992 | April 1992 | May 1992 | June 1992 |
|---|---|---|---|---|
| **Define problem** | | | | |
| **Develop concepts** | | | | |
| **Concept research** | | | | |

(S743)

*Definition of "clean sheet"*

{t218} To design and build an alpha-level prototype of a human-powered hydrofoil. We aimed the design of this craft at a high end recreational market; therefore it should be practical and relatively easy to use.

☒ *Prototype phase terminology*

{t219} An "alpha-phase" prototype is a term often associated with proof-of-concept work. Sometimes referred to as a "mule," an alpha prototype usually functions as a test bed for debugging design concepts. Beta-phase prototypes, often referred to as "preproduction" prototypes, are usually manufactured and assembled in a method as close to the production as possible. Often more than one beta is produced for user testing. This sort of terminology is very dependent on your local design culture.

## Concept research

| | April 1992 | May 1992 | June 1992 | July 1992 |
|---|---|---|---|---|
| **Develop concepts** | | | | |
| **Concept research** | | | | |
| **Early modeling** | | | | |

(S744)

{t200} A fresh approach to a problem sometimes provides revolutionary insights, i.e., not limiting your mind to previous conceived solutions. In this spirit, initial ideas and brainstorms for the hydrofoil were gathered prior to any research.

{t201} Research for developing concepts came from prior art, nature, and related things. Where else might you look for ideas and inspiration?

### ☒ Prior Art

*Decavitator (5/1/92)*

{t227} The MIT Decavitator project set the world records

Men's record: 18.50 kt, 2 Nov. 91 Mark Drela

Women's record: 13.86 kt, 26 Oct. 92 Kjirste Carlson

{V342} video of Decavitator

{t228} Drela, Mark; Schafer, Marc; and Wall, Matt; "Decavitator human-powered hydrofoil," Human Power, Vol. 9 Nos. 3&4, Fall-Winter 1991-1992.

*Japanese (5/10/92)*

video from Dave

On & Off Yamaha

*papers on hp hydrofoils (5/21/92), (5/26/92), (5/28/92)*

{S658} notebook sketch

{t229} Smitt, Lei Wagner; "Conceptual Design of Human Powered 'Hopping' Hydrofoil Boat"; Small Craft Group International Conference on Human Powered Marine Vehicles; Oars, propellers, paddles at the Weir Lecture Hall, London. Nov. 8, 1984.

Owers, D.J.; "Development of a Human-Powered Racing Hydrofoil"; Small Craft Group International Conference on Human Powered Marine Vehicles; Oars, propellers, paddles at the Weir Lecture Hall, London. Nov. 8, 1984.

Brewster, M. B., "The Design and Development of a Man-Powered Hydrofoil," Thesis for Bachelor of Science, Mechanical Engineering Dept., MIT, May 1979.

*Sid Shutt's stuff (5/30/92)*

{t230} Shutt, Sid; "Some ideas on Hydro-ped -- a hydrofoil pedal boat", Human Power 7(4), Summer 1989.

Shutt, Sid, "A practical Human Powered Hydrofoil"

Mark's video

*Flying Fish (5/28/92)*

{t231} Brooks, Alec N. "The Flying Fish Hydrofoil," Human Power, Vol. 3, No. 2, Winter 1984.

{V343} video

☒ Nature (5/3/92)

Looking for inspiration in nature.

{t232} Grzimeks, Bernhard; Grzimek's Animal Life Encyclopedia, Van Norstrand Reinhold Co. N.Y.; ©.

Volume 2: Insects © 1972 -- ☒ {S646}wasp

Volume 4: Fishes I © 1973 -- ☒ {S649}shark, ☒ {S647}flying fish, ☒ {S651}ray

Volume 7: Birds I © 1975 -- bird tails

{t233} McCafferty, Patrick W.; Aquatic Entomology, Science Books International, Inc. Boston; ©1981.

☒ {S648} Water Bugs (order hemiptera)

{t234} Katona, Steven K.; A Field Guide to the Whales, Porpoises and Seals of the Gulf of Maine and Eastern Canada; Charles Scribner's Sons, NY © 1983.

☒ {S645} Minke whale

☒ {S644} dolphin

☒ {S650} Humpback whale flukes

☒ Related

*commercial hydrofoils (5/28/92), (5/30/92)*

{t235} Trillo, Rober L.; Jane's High-Speed Marine Craft 1990, 23rd Edition, Jane's Information Group Limited, ©1990.

Italian build Sparviero hydrofoil missile craft

*picture in the book*

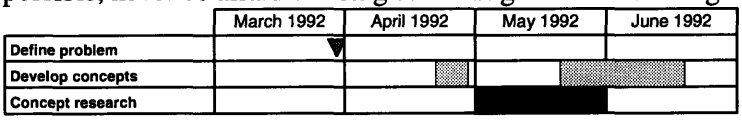
*{S654} sketch in notebook*
Russian made Cyclone
{S655} sketch in notebook

*The 40 knot sailboat (5/23/92)*
{t236} Smith, Bernard; The 40 Knot Sailboat, Crosset & Dunlap Inc., NY © 1963.
{S656} sketches in my notebook

*aero library research (5/26/92)*
{t237} Stinton, Darrol; The Anatomy of The Aeroplane; Granada Publishing Limited, NY; ©1966
{S657} sketches in notebook

☒ {S763} Develop concepts (4/25/92)

{t202} Which one would you select as the best design or can you think of a better design? It is always best to start with as many diverse concepts as possible, never be afraid of doing something that seems strange.



{S745}

☒{S653} drawing and sketching

{t203} Drawings and sketches have two basic functions, (1) geometric modeling and (2) communication. A sketch or drawing can serve as a model of a system or component, answering both functional and asthetic questions. Design engineering has developed drafting standards as a language to comunicate precise design information. Note the different types of visual modeling and communication we use in designing the Skeeter.

*"...there's something that's very intense about the experience of sitting down and having to look at something in the way that you do in order to make a drawing or a painting of it. By the time you've done that you feel that you've really understood what you were looking at... and somehow it becomes a method of possessing the experience in a unique way." --Robert Bechtle*

☒{S759} The engineers notebook

{t204} Good practice when doing any design work is to use a bound notebook. The notebook is a good way to keep track of the progress of a project and ideas. Also, a notebook is a legal document often used to defend patents. Some good rules to follow:

- ✓ use a bound notebook (not spiral bound);
- ✓ record you original data;
- ✓ record all sketches, ideas and thoughts;
- ✓ number, date and initial all pages; and
- ✓ use ink (go over any pencil work with pen).

☒{S627i} concept 1 (4/25/92)

☒ {S659} notebook sketch
{S627b} rendering

This design incorporates a surface-piercing main vee-foil in a catamaran layout. The rider sits in a recumbent position. The boat uses direct-shaft-drive

that has the exposed shaft in the water, driving a water propeller. The hulls are fiberglass and provide the main structure.

Advantages

- ✓ Catamaran layout provides higher takeoff speeds and is very stable
- ✓ Simple frame and drive system.
- ✓ Surface-piercing vee-foil is self-stabilizing.
- ✓ Hull integrity never compromised.

Issues

- ✓ Molding dual hulls is expensive.
- ✓ Weight of dual hulls may be high.
- ✓ Surface-piercing vee-foils have ventilation problems.
- ✓ Exposed propeller shaft has high drag.

☒ *definition of ventilation*

☒{S621i} concept 2 (4/26/92)

☒ {S660} notebook sketch

{S621b} rendering

This design is a fully faired catamaran that incorporates a main surface-piercing vee-foil. The rider sits in a recumbent position. The drive system incorporates a direct shaft or twisted-belt mechanism on the front foil.

Advantages

- ✓ Aesthetically pleasing design.
- ✓ Aerodynamic.
- ✓ Catamaran layout provides higher takeoff speeds.
- ✓ It has a simple drive system.
- ✓ Surface-piercing vee-foil is self-stabilizing.

Issues

- ✓ Safety in a capsize is a major problem.
- ✓ Increased weight may offset any aerodynamic advantages of a fairing.
- ✓ Molding of a complex fairing is very expensive.
- ✓ Surface-piercing vee-foils have ventilation problems.
- ✓ Main foil is in the propeller wash.
- ✓ Transportability.

☒{S617i} concept 3 (4/25/92)

☒ {S661} notebook sketch

{S617b} rendering

This is a single-hull craft with a water propeller. The hull incorporates a tail behind the rider for improved aerodynamics. The drive system incorporates a direct shaft or twisted-belt mechanism on the front foil. Rider sits in a recumbent position. Both foils are tee-type.

Advantages

- ✓ Simple frame and drive system.
- ✓ Tee-type foils are efficient.
- ✓ Has good aerodynamics with tail and low frontal area.
- ✓ Compact design improves transportability.

Issues

- ✓ No roll stabilization provided.
- ✓ Short and wide hull has low takeoff speed.
- ✓ Tail may be expensive complexity.

☒{S625i} concept 4 (4/25/92)

☒ {S662} notebook sketch

{S625b} rendering

This design incorporates a surface-piercing main vee-foil in a catamaran layout. The rider sits in a recumbent position and rides in the reverse direction, like a rowboat. It incorporates direct-shaft-drive that has the shaft exposed to the water, driving a water propeller. The hulls are fiberglass and provide the main structure.

Advantages

- ✓ Catamaran layout provides higher takeoff speeds.
- ✓ Simple frame and drive system.
- ✓ Surface piercing vee-foil is self-stabilizing.

Issues

- ✓ Safety concerns of riding in the reverse direction.
- ✓ Molding dual hulls is expensive.
- ✓ Weight of dual hulls may be high.
- ✓ Surface-piercing vee-foils have ventilation problems.
- ✓ Exposed propeller shaft has high drag.

☒{S628i} concept 5 (5/20/92)

☒ {S663} notebook sketch

{S628b} rendering

This design has a direct-drive water propeller on the front foil. Setting rear foils apart gets them out of the propeller wash. The foils are tee-type. The rear foils have roll-stabilizing control surfaces. The rider sits in a the hull in a recumbent position. The single hull incorporates an aerodynamic tail.

Advantages

- ✓ It has a simple drive system.
- ✓ Incorporates dynamic roll stabilization.
- ✓ Low frontal area and tail for aerodynamics.

Issues

- ✓ Molding of complex hull will be expensive.
- ✓ Takeoff will be a problem with the control system.

☒{S632i} concept 6 (5/20/92)

☒ {S664} notebook sketch

{S632b} rendering

This design is a fully faired mono-hull that incorporates a dynamically controlled main-foil. The main-foil takes advantage of the pressure differential at the foil tips due the difference in depth from roll pitching, thereby adjusting flaps to compensate. The rider sits in a recumbent position. The drive incorporates a standard bike chain using a derailleur/free-wheel transmission connecting to a shaft running down to a water propeller on the main foil (rear).

Advantages

- ✓ Aerodynamic.
- ✓ Incorporates dynamic roll stabilization.
- ✓ Clean appearance.

Issues

- ✓ Safety in a capsize is a major problem.
- ✓ Molding of a complex fairing is very expensive.
- ✓ Large hydrofoil cross-section required for mechanism, therefore drag cost.
- ✓ Active control system is complex and has questionable physics, requiring a great deal of development investment.

☒{S629i} concept 7 (5/21/92)

☒ {S665} notebook sketch
{S629b} rendering

A key element of this design is the mechanical roll-stabilizing system that controls the angle of attack each side of the main-foil. Coupling this system to the steering provides for banked turns. The rider sits in a recumbent position in a mono-hull. The drive incorporates a standard bike chain using a derailleur/free wheel transmission connecting to a shaft running down to a water-propeller.

Advantages

- ✓ Mono-hull is simpler and more transportable.
- ✓ Incorporates dynamic roll stabilization.
- ✓ Long hull has high take-off speed.
- ✓ Height out of water is completely adjustable.

Issues

- ✓ High loads in foil-control shafts.
- ✓ Complex drive is heavy.
- ✓ Main foil strut has complex mechanism inside.

☒ {S630i} concept 8 (5/24/92)

☒ {S666} notebook sketch
{S630b} rendering

This is a reverse-direction (rowboat style) mono-hull. It uses a direct-shaft or twisted-belt drive connecting to a water propeller. The forward foils are surface-piercing triangular foils, similar to those proposed by Smith.

Smith, Bernard; The 40 Knot Sailboat, Crosset & Dunlap Inc., NY © 1963.

Advantages

- ✓ Short drive-train.
- ✓ It has aerodynamic shaped bow.
- ✓ Forward foils are self stabilizing.
- ✓ Long hull has higher take off speed.

Issues

- ✓ Efficiency and ventilation of forward foils.
- ✓ Safety of going backwards at high speeds.
- ✓ Complex hull is expensive.

☒ {S620i} concept 9 (5/24/92)

☒ {S667} notebook sketch
☒ lessons on ladder foils from the Decavitator (video)
{S620b} rendering

This design uses a ladder foil with lowest-speed foil on the top and highest-speed foil on the bottom. The lower-speed foils lifts out of the water at higher speeds. The rider sits in a recumbent position in a short open hull. Supporting the rear-foils to the sides provides for roll stability and keeps them out of the propeller wash. The direct drive runs down the main foil to a water propeller.

Advantages

- ✓ Short simple drive-train.
- ✓ Low take-off speed with ladder-foils.
- ✓ It has a simple, inexpensive hull.

Issues

- ✓ Problem of the reverse ground effect as the runs of the ladder-foils lift out of the water.
- ✓ It is difficult to get center of lift at the Cg, with this layout.
- ✓ Flight-height can be very high.

☒{S624i} concept 10 (5/25/92)

☒ {S668} notebook sketch

{S624b} rendering

This design incorporates a surface-piercing main vee-foil in a catamaran layout. The rider sits in a recumbent position. The drive incorporates a standard bike chain using a derailleur/free-wheel transmission connecting to a shaft running down to a water propeller. The hulls are inflatable connected to a tubular frame.

Advantages

- ✓ Catamaran layout has higher takeoff speed.
- ✓ Inflatables can be less expensive and of lighter weight than fiberglass hulls.
- ✓ Surface-piercing vee-foil is self stabilizing.
- ✓ Deflating makes it transportable.

Issues

- ✓ Long complex drive.
- ✓ Inflatables have higher drag than non-inflatable counter parts.
- ✓ "Hard spot" load points may be problem.
- ✓ Surface-piercing vee-foil are less efficient than tee-type.

☒{S633i} concept 11 (5/25/92)

☒ {S669} notebook sketch

☒ {S650} Whale fluke sketch

{S633b} rendering

This design places the rider in a prone position, allowing for a short direct-drive to the water-propeller. The bow incorporates a windscreen for aerodynamics. The main-foil (forward) has a mechanical control system that changes flap angle for roll stability. The proposed shape of the main foil resembles a whale fluke.

Advantages

- ✓ Aerodynamics.
- ✓ Short drivetrain.
- ✓ Flaps allow for larger speed range.
- ✓ Stable in roll.

Issues

- ✓ Prone rider position is not usually as comfortable.
- ✓ Wide vertical support required for rear foil due to drive and flap mechanism, while not needed structurally.
- ✓ Windscreen is expensive.

☒{S615i} concept 12 (5/25/92)

☒ {S670} notebook sketch

{S615b} rendering

The rider sits in a standard riding position for this design, above a catamaran boat. The drive incorporates a standard bike chain using a derailleur/free-wheel transmission connecting an air-propeller. The main rear foil is a surface-piercing vee-foil. The front foil is a tee-type with a skimmer-based control system.

Advantages

- ✓ It has a familiar rider position.
- ✓ Air-prop has high efficiency.
- ✓ Surface-piercing vee-foil is self stabilizing.
- ✓ Its takeoff speed is higher due to catamaran layout.
- ✓ It uses a simple off-the-shelf drive.

Issues

- ✓ It has high aerodynamic drag.
- ✓ Its large size makes transportation difficult unless made collapsible.
- ✓ Large air prop is expensive and fragile.
- ✓ Dual hull is expensive.
- ✓ High center of mass makes it unstable.

☒{S622i} concept 13 (5/26/92)

☒ {S671} notebook sketch

{S622b} rendering

This design, dubbed the "hydrocopter," uses counter-rotating rotor blades to prevent rider from rotating. IT requires a minimal recumbent rider platform. As the platform lifts out of the water the hub tilts to provide forward motion, similar to that of helicopters.

Advantages

- ✓ Zero takeoff speed.
- ✓ It allows for a light simple rider platform.
- ✓ It has the potential of being highly maneuverable.
- ✓ Its small size makes it easily transportable.

Issues

- ✓ It requires a complex control system.
- ✓ It has questionable efficiency.
- ✓ Counter-rotating blades requires a very complex drive.
- ✓ Wide vertical support for hub required.
- ✓ Drivetrain efficiency.
- ✓ Cavitation at the rotor tips.
- ✓ Low top speed.

☒ concept 14 (5/26/92)

☒ {S672} notebook sketch

rendering similar to concept {S629}

This mono-hull uses two skimmers on each side of the hull linked to the nose and plain flaps of the main foil to provide roll stability. A skimmer at the bow provides active pitch control. The main foil strut encloses the drive.

Advantages

- ✓ Dynamic roll and pitch stabilization
- ✓ Clean package
- ✓ Mono hull is simpler

Issues

- ✓ Complex control mechanism yields a packaging problem.
- ✓ Mechanism running through the main strut makes the strut large.
- ✓ Complex foil structure need for all the flaps

☒{S614i} concept 15 (5/30/92)

☒ {S673} notebook sketch

{S614b} rendering

Here a small narrow hull supports a recumbent rider above the hull. Two external pontoons provide stability in displacement mode. Pivoting outrigger

supports give them a secondary function in hydrofoil mode -- as part of the rol-stability control system. The drive incorporates a standard bike chain using a derailleur/free-wheel transmission connecting to a shaft running down to a water-propeller.

Advantages

- ✓ Narrow main hull provides higher displacement mode speeds.
- ✓ Simple light-weight structures.
- ✓ It has dynamic roll stabilization.

Issues

- ✓ It has a complex drive.
- ✓ Pontoons in flight mode may "bite" waves.
- ✓ High loads on pontoon arms when landing.
- ✓ Coupling systems can make things very complex.

☒{S631i} concept 16 (5/30/92)

☒ {S674} notebook sketch

☒ {S654} Italian gun-boat sketch

{S631b} rendering

This design has hydrofoils lifted out of the water in displacement mode. The rider sits in a recumbent position in a mono-hull. The drive is a standard bicycle chain-drive and derailleur/free-wheel transmission that connects to a shaft drive exposed to the water.

Advantages

- ✓ Foils are protected in shallow water.
- ✓ In displacement mode foils don't provide drag, yielding higher take off speeds.

Issues

- ✓ Weight and complexity of mechanism.
- ✓ Drag of propeller shaft.

☒{S626i} concept 17 (5/30/92)

☒ {S675} notebook sketch

{S626b} rendering

A narrow hull supports a recumbent rider above the hull. Two external pontoons provide stability in displacement mode. The drive is an axial-flow or centrfugal type pump, that takes water in at the forward foil and outlets after the main-foil. The main-foil, supported on both ends, incorporates a dynamic control system.

Advantages

- ✓ Long narrow hull is inexpensive and has high displacement speed.
- ✓ It has dynamic lateral stabilization.
- ✓ Frame tube used as water passage.
- ✓ No propeller underwater to damage.

Issues

- ✓ Pump efficiencies.
- ✓ Head loss.
- ✓ Expense of outriggers and structures.

☒{S616i} concept 18 (6/18/92)

☒ {S676} notebook sketch

☒ {S644} dolphin drawing

{S616b} rendering

This mono-hull uses an oscillating main-foil for propulsion, like that of a dolphin or whale. The rider sits in a recumbent position.

Advantages

It eliminates propeller and problems of.
Oscillating hydrofoil has potential of high efficiency.
It has a simple frame.

Issues

There is little information, i.e., research on this type of drive.
It does not have roll stability.
It has a highly loaded pivot on drive.

☒{S764} drive-train concepts (4/25/92)

{S677} notebook sketch

5 types

☒{S765} swept-wing concept ((6/2/92)

{S681} sketch from notebook

☒{S766} wing-molding concept (5/21/92)

{S678} sketch from notebook

☒{S767} frame-design concepts (5/21/92)

{S679} sketch from notebook

☒{S768} dual-gain steering concept (5/23/92)

{S282} sketch from notebook

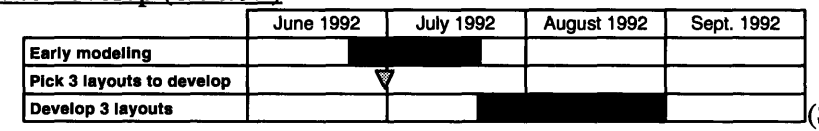
☒{S769} diecast-wing concept (w/ and w/o control surface) (5/26/92)

{S680} sketch from notebook

☒{S770} seat-clamping concepts

{S684, S683} sketch from notebook

Choose 3 models to develop (6/30/92)



(S748)

☒{S859} Design decision making. Tools available to the engineer

*decision matrix*

...

*House of Quality*

...

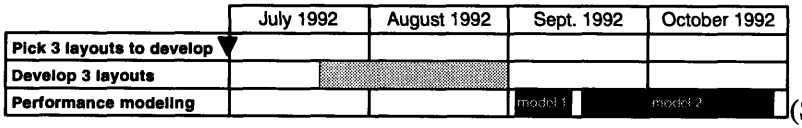
Hauser, J.R., and Clausing, D., "The House of Quality," Harvard Business Review, 63-73, May - June, 1988.

*axiomatic approach*

...

Suh, Nam; The Principles of Design, Oxford University Press, © 1990.

Develop 3 concepts



(S749)

{t220} Traditionally the design engineer would size systems in a layout drawing. These drawings would then be used by the draftsmen to create details and assemblies. With the advent of CAD (computer-aided design) 2D and 3D

modeling has replaced the traditional design layout. The term model has very appropriately crept into the drafting terminology. The idea that a drawing is really a geometric model of a system that predicts how the system will look and fit together. Sometimes prototype work happens directly from the layout drawing or the layout provides the information needed for generating detail (machine) drawings.

☒{S853} Choosing the Concepts

rider position

{S854} standard, prone, and recumbent rider positions

{t221} We chose recumbent rider position because its clear advantages for this case. A standard position has such a high CG that it has great stability problems. The prone position is rather uncomfortable.

riding direction

{S855} diagram

{t222} Though two of the proposed concepts had the traditional going-backwards direction of a row boat this is undesirable for a high speed boat. The only advantage to going backwards is that the riders' feet are close to the propeller, making the drivetrain shorter. This does not outweigh the safety advantage and satisfaction from seeing where you're going. Therefore we limited the final three concepts to forward-direction crafts.

hull layout

{S856} catamaran, wide mono hull (rider sits in the hull), and narrow mono-hull (rider sits on top of the hull)

{t223} All three of these designs are feasible and are worth more study. We decided to consider all three types using one in each layout.

propulsion (scans or diagram)

{S857} water propeller, air propeller, water pump, i.e., ducted jet, and articulating foil

{t224} Of the four propulsion systems proposed we considered the water propeller and water pump for the three concepts developed. The problem with the air prop is its size (10ft diameter), making it impractical. The articulating foil has little or no prior development; therefore we would be working from at very low level. This would consume too many of our resources.

hydrofoil layouts

{S858} diagram

{t225} dual-surface-piercing vee foils
front tee w/drive, dual rear w/controls
front fixed tee, single rear foil w/control system and drive
front tee w/pitch control, rear w/control system and drive
front vee foils (separate), rear tee w/drive
front controlled tee w/drive, rear fixed tee
front pitch control tee, rear surface piercing vee foil
front pitch control tee, rear controlled "u" type foil
ladder foil.

Designs selected considered both surfacing-piercing and non-surface-piercing designs.

☒{S851} concept 5 (7/22 to 8/12/92) -- {Pallet: S849}

The aesthetics of this layout, based on concept 5, invoked the most positive response from those reviewing the concepts. The drive system uses bevel a gear set at the cranks and propeller. To adjust for different rider sizes

the crank assembly moves towards the rider; to accomplish this a telescoping drive shaft and universal joints accommodate for the shift.
sketches
{S846} LAYOUT1.DWG

☒{S852} concept 17 (8/12 to 8/22/92)

This layout, based on concept 17, takes advantage of a narrow hull and "u" foils. The proposed water pump was originally to be used in this layout, but the team was unable to find any reliable information on the design and performance of a this sort of pump; the time penalty involved with seriously considering this design seemed too great. In changing the drive the foil layout was switched from the original concept, putting the rider toward the bow. Also, to improve the top-speed performance we added a two-stage foil system. The constant-section foil allows for extrusion or pultrusion construction.

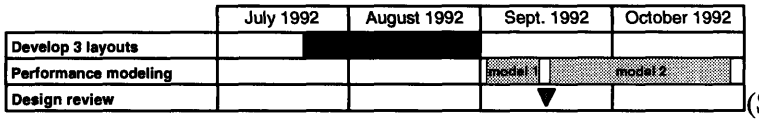
*Define extrusion and pultrusion*

sketches
{S848} LAYOUT2.DWG

☒{S850} concept 10 w/water prop (8/30 to 8/30/92)

Based on concept 10, layout 3 considers the lower-cost inflatable hulls in a catamaran design, and surface-piercing vee foils.
sketches
{S847} LAYOUT3.DWG

Modeling



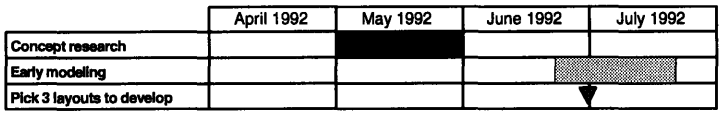
(S750)

{t238} Engineers have some of the basic tools needed creating mathematical, geometric and physical models that give insight into the workings of machines. The complexity of a model does not necessarily indicate how useful it is. Usually complex models require greater available resources, i.e., time, equipment, methods, etc., all of which affect modeling decisions.

{t239} For example if you have to size an aluminum tube for a structure, how would you go about it? You could do a complete FEA (finite-element analysis) model and size the tube with a minimal factor of safety. If you must select a standard-size tube, in the end it increases your factor of safety anyways, and you've wasted time. It may be better, if the item is not weight-critical, to do a simple hand calculation and heap on a bigger factor of safety to account for the greater uncertainty. Also, you should always have some way to make sure your computer models are in the right ball-park, so a simple hand calculation should always accompany your computer model.

{t240} Modeling occurs at different levels of systems and subsystems. If you consider the tube example again, the tube is a subsystem or component of a larger structure that in turn maybe part of another system. The tube size will have some effect on all these systems.

☒{S872} Early modeling: used Brooks for basic foil-sizing model for preliminary concept sizing (7/21/92)




(S747)

{t241} Brooks, Alec, "The 20-knot Human-Powered Water Craft," Human Power, Volume 6, Number 1, Spring 1987.

{t243} We used simple analysis and assumption in this paper for initial foil sizing for the three concept layouts.

{t242} Whitt, Frank Rowland; Wilson, David Gordon; Bicycling Science; The MIT Press; Cambridge, Massachusetts © 1982.

{S652} Figure 2.13

☒{S873} System Model: 1st model is relatively simple and predicts power requirements and speed (9/2 to 9/12/92)

*used to model two of the three layouts. The vee foil would require a different model.*

{t244} To understand the performance of these designs better we create a basic static model in a spreadsheet. Using this model on the Flying Fish produced a speed of 4.3 m/s (8.4 knots) at 224 watts (0.30 hp), 20% different from published numbers. Using this model on the Decavitator at 597 watts (0.8 hp) (this is an assumed maximum power from the rider) yielded a top speed of 12.0 m/s (23.4 knots), this is 23.5% higher than the world record set by this hydrofoil.

{t228} Decavitator article reference

{t241} Brooks article reference

{t245} Unfortunately, the validity of this model extends only to non-surface piercing foils; therefore layout 3 can not be modeled with it. The first two layouts were modeled yielding the following:

{t246} layout 1 3.7 m/s (7.2 knots) take off at 261 watts (0.35 hp)
5.4 m/s (10.5 knots) at 373 watts (0.5 hp)
7.3 m/s (14.2 knots) at 746 watts (1.0 hp)
layout 2 2.8 m/s (5.5 knots) takeoff
5.1 m/s (10.0 knots) at 276 watts (0.37 hp) (low speed foil lift out)
7.1 m/s (13.8 knots) at 373 watts (0.5 hp)
9.6 m/s (18.7 knots) at 746 watts (1.0 hp)

☒{S874} Control and Stability: 3rd model was done by Mark later, a controls/stability model

☒{S875} Foil Design: Xfoil

Drela, M., "XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils," Low-Reynolds Number Aerodynamics -- editor Muellar, T.J.; June 1989 lecture note in Engineering, No. 54.

Some plots

☒{S876} Propeller Design: Xrotor

Some plots

NOT included:

*2nd model is rather complex, tries to consider the dynamics and control geometry. Modeling Decavitator and Flying Fish showed good correlation to actual performance. (9/21 to 10/28/92)*

The next level model that was tried took into consideration the dynamics and control geometry ... finish

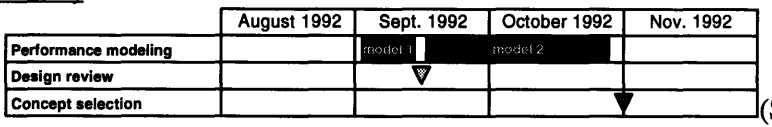
plot of Decavitator's expected performance

*spent a whole lot of time trying some of the ship hull shapes and numbers were off the charts (6/21/92)*

Harvald, SV. AA.; Resistance and Propulsion of Ships; John Wiley & Sons Inc., © 1983.

We needed a correlation between drag and speed for the boat hull, so using the method shown in Harvald a spreadsheet was setup. We found, after wasting two days work, that the charts and tables provided in Harvald do not support small-displacement hulls.

Design review (9/15/92)



(S751)

{t226} Design done in a closet will rarely fill the requirements of the user; diverse perspectives' are always needed. As soon as possible in the process, the designer needs feedback at all levels, from those who make it, to those who will use it. One type of design review is a peer review, where the design is show to other designers with as diverse perspectives as possible.

For the hydrofoil we drew on the available talents at MIT that represents experience from the Decavitator project, the Daedalus project, expertise from the world human-power design and product design.

video taped

*people*

*David Wilson*
*Woodie Flowers*
*Mark Drela*
*Matt Wall*
*Marc Schafer*
*Sepehr Kiani*
*(Tracy taking notes)*

{S866} background

☒{S867} Video clip {V341}

☒*{S863} comments on layout 1*

{S860}

✓ concept would work

✓ the wing sizes seem questionable

✓ flaps are too complex and the bearings would be very highly loaded

✓ finding bevel gears small enough is impossible, end up with a huge a gear box under water.

✓ flexible shaft is suggested here 1st.

☒*{S864} comments on layout 2*

{S862}

✓ universal joint: only one is shown, usually two are needed to be balance

✓ it is possible that because the shaft is so long that the only one joint would work, but may have vibs problem

✓ having a high speed and low speed wing arrangement is much complexity for a recreational boat and makes it difficult to fly.

✓ Sid Shutt's pop out wings were looked at.

✓ marketing the complexity of the pop out may be difficult

✓ flaps again are too complex

"want to keep the goodies out the water"

✓ main foil needs a large aspect ratio due to the free surface effect.

✓ 20% penalty for not having a tapered wing

✓ flexible shaft suggested again, vibrations problem discussed

✓ twisted chain discussed

✓ twisted belt suggested

☒*{S865} comments on layout 3*

{S861}

✓ this would also work with flexible shaft

✓ rider position can be more laid back

✓ vee foils ventilate

*deciding on what is needed for final layout*

✓ *drive is always a problem*

*chains always coming off in the competitions because of tension problem.*

*quality of gear boxes*

*keep comes back to the flexible shaft*

✓ *reverse layout 2? -- use a puller prop*

✓ *1:1 gearboxes are easier to implement*

✓ *seems the we are heading towards yellow layout with a flexible shaft and need outriggers*

✓ *mono-hull versus catamaran*

Concept selection

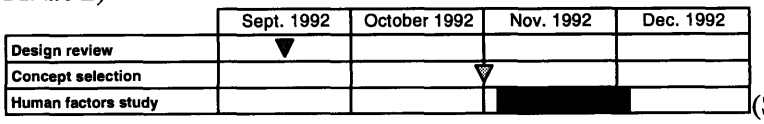
{S685} notebook sketch

{S623} rendering

*Design selection>*

{t205} The selected design derived from the design review and the developed concepts is shown here. Economic factors and transportability weighed in to the selection of a mono hull. The in-flight steering system ties into a central arm that pivots on an angled shaft to counter turning roll forces. To get the first-stage gear reduction and 90° direction change we employ a twisted chain. A spur gear set provides the second stage. All this connects to a flexible shaft exposed to the water. Outriggers provide roll stability in displacement mode. The pilot will control the angle of attack of the main foil manually, much like the Decavitator.

use model 2 for foil sizing, ends up very close to Decavitator in foil geometry and mechanism. (11/1/92)



(S752)

key element is the flexible shaft -- initial calcs to verify the viability (11/3/92).

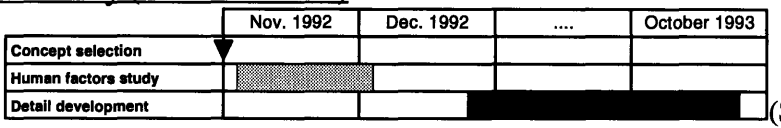
☒{S774} MathCAD calcs for flexible shaft {S777}.

8.5.1. also front steering arm

layouts

{S685} final concept notebook sketch
☒ LAYOUT02.DWG, layout

## 8.5.2. Human factors study (11/3 to 12/2/92)



(S753)

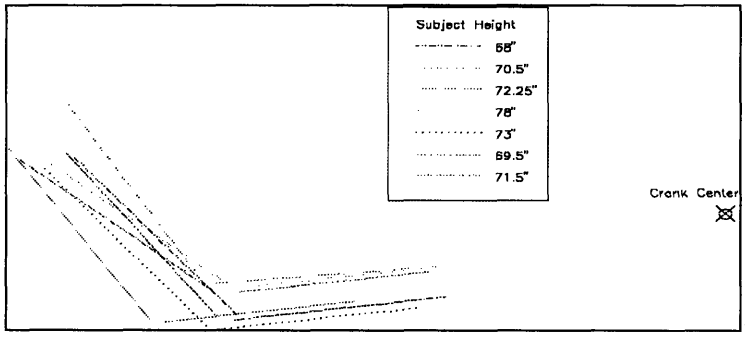
*Humans and their environment>*

{t206} The field of human factors is the physiological and psychological study and design of the human machine interface. Consider the different people that have to use or interact with any device: those who use it, those who buy it, those who make it, those who maintain it and those who dispose or recycle it. Often different users voice competing values. For example, if you design something with manufacturability in mind it may not be easy to maintain.
{t207} Sanders, M.S. and McCormick, E.J., Human Factors in Engineering and Design, McGraw-Hill Publishing Company (c) 1987.

☒{S774} establish rider position

*Video?> Seat geometry.*

{t208} We constructed a four-way adjustable mockup of the seat to facilitate the collection of seat-geometry data. The seat angle is adjusted by rotating the slide surface on a pivot in the back, adjusting this angle also affects the seat-back angle. This surface allows the main seat assembly to slide in the horizontal direction and locked in place with a lever. The seat-back adjusts on a series of 14 pegs. Finally height of the cranks can be adjusted up and down.



{S772}

*Seating geometry conclusions*

{t209} The results of the test indicated that the seat slide will be a fairly linear function of the rider height. Extrapolating the horizontal position data out for the 5th percentile female, 1.50m (59") tall (from Sanders and McCormick), it would require a horizontal seat position of 0.35m (13.8"). To fit my 1.98m (6'6") rider the craft would need 0.38m (15.1") of adjustability, which is quite a bit, but not impossible.

The vertical seat position and backrest angle are somewhat more illusive. To fix these two angles on the boat the position and angle would be at about 8.5" below the crank center and 129° respectively. Experience in bicycle racing has shown that seat or handle bar position maybe comfortable in a static situation -- it is the very subtle adjustments that make a 50 mile race comfortable.

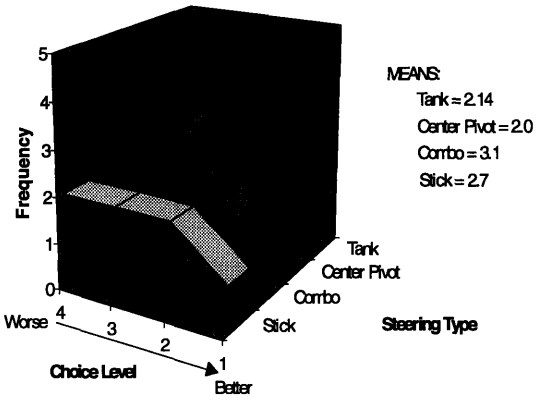
{t210} Zwikker, Bernd, "Riding position and speed in unfaired recumbents," Human Power, Vol. 8 No. 2 Spring 1990.

☒{S775} establish steering type

*Video?>* *Controls.*

{t211} We tested four control scenarios by having different removable control levers. The levers fit into a mechanism under the seat that had a "tank" steering mechanize on the top that forces the levers to rotate in opposite directions, and a vertically pivoted pair on the bottom. The whole mechanism slid forward and back for adjustment.

{t212} The coupled control scenarios were, by far, the most popular, but those who like the others like them because you have your hands on the controls at all times. We should test a couple of compromises, like combining the vertical pivot steering with height control and have stick control in both hands (linked together). Development of the design details (which are so important) waits for the control scenario to solidify. These details include the shape of the handles, the angle they are at, the force required to control, and whether to have a centering system for the steering.




{S773}

Detail development / Construction and construction failures

| | Nov. 1992 | Dec. 1992 | .... | October 1993 |
|---|---|---|---|---|
| Concept selection | | | | |
| Human factors study | | | | |
| Detail development | | | ...10 months... | |

(S754)

{t213} A common mistake that new design engineers often make is to base and entire design around a detail or material. There is this misconception that using an exotic material will in its self be the winning element. This is a fallacy; I have seen many a carbon fiber kluges at student competitions. An overall sound concept can fail miserably due to poorly developed details.

A good example is in human-powered-vehicle competitions. Observing one of these racing you will find many of these efforts poorly executed, while the concepts have the potential for overall better performance than a traditional road bike. At the same time, a highly developed steel-frame racing bicycle restrained by traditional rules performs superiorly. This goes to show that even a concept that is imperfect (as all concepts contain compromises) can function very well with carefully developed details.

*Factor of Safety>*

{t214} Sometimes referred to as a safety factor or margin of safety, the factor of safety (FS) is a device that allows for uncertainty in design. Uncertainties in design and analysis arise from assumption made. This is a very important part of engineering judgment and can sometimes be the difference between life and death. It is usually just a constant multiplier times a load or material strength.

For example if you have a 100 Newton load on a beam you may apply a FS directly to the that load, i.e., 100 x FS. Choosing the FS depends the answer to questions like. How well do we know that the load is 100 Newtons? Can anyone be injured if this beam fails? How badly? How much damage will the failure cause to the device? How detailed is the analysis of the beam structure? Etc. Sometimes the engineer applies a minimum FS mandated by code in certain applications.

☒{S794} Detail Development Success & Failure

{S623} rendering of the hydrofoil or a picture with flags to the buttons.

✓ *LAYOUT3.DWG*
✓ *LAYOUT4.DWG, twisted belt seat w/2nd support*
✓ *LAYOUT5,6.DWG, frame2*
✓ *LAYOUT7.DWG, trying screwed up frame to see if it could be made to work. bummer.*
✓ *LAYOUT8.DWG, uses frame4 -- not interesting*
✓ *LAYOUT9.DWG, uses frame9, still using push rods for steering.*
✓ *LAYOUTP.DWG, ??*

☒{S780} front steering pivot

*{S754} engineers discussing design*
*{S759}=>{S744} knock out mechanism (12/23/92)*
*quick release (12/23/92)*
*{S819} adjustable angle (3/21/93), (4/11/93)*
{S686} FFPIVOT.DWG
Matt's drawings
*rudder at the bow (3/21/93)*
☒ *{S810}=>{V331} Pivot fabrication. Audio:*

The wood bulk-head installed at the bow of the boat during the fabrication of the hull provides the mounting point for steering pivot. Thin plastic sheeting acts as a mold release for the carbon-epoxy composite. The two thin pieces of white string you see here, align with the bearing journal: a stainless-steel tube brazed to a stainless plate. We use a little trick here to hold the plate firmly in place for the lay-up. We wet the center portion of the cloth with a fast-setting epoxy and locate the plate. Two-phase adhesive bonding is sometimes employed in production too, a fast-setting glue -- like an ultra-violet set -- holds parts together strongly enough allowing for removal from the holding fixture. Then a secondary glue is then cured -- like a heat set. What advantage could you see in this?

Composites always provide an excellent illustration of the propagation of material stresses. In our case the loads transfer from the bearing to the screws, hence we run fibers with this in mind; if this part was a casting or forging the optimum shape would be similar to the section of this part.

Vacuum bagging this layup would provide the best control of resin content and fiber compaction, but pulling a vacuum on this hull wouldn't be very effective. We improvise by blotting out extra resin and taping it

down firmly with masking tape. The part, cured and cleaned, is light weight and mates perfectly with the hull.

*{S813}=>{V334} Steering arm fabrication. Audio:*

This angle-o-meter is used to orient all the wooden spacer blocks on the steering arm. Eighth-inch pins hold the bearing shaft in place and prevent rotation. The marine-grade plywood is painted with epoxy before lashing. The spacing of the front tees from the center line of the boat affects the roll-control time constant. That is if the tees are put wide apart, the roll stability of the boat is better. On the other hand a narrow spacing is lighter weight and less cumbersome. We were unable to choose a width from our analysis. There are several ways to deal with this issue; one is to make things adjustable, the other is to have several different size parts that can be interchanged during testing. The design team wrestled with this for a while. This is the compromise we came up with, giving a foot of adjustability on each side. After testing we decided to lower the pivot point, which is easily accomplished by cutting the Kevlar lashings and putting on new blocks.

*main cross bar design calcs (3/26/93)*

☒{S782} skimmer pivots and stops

*{S759} Engineer's notebook*

*break away skimmer arm (12/23/92)*

*angle adjustment (12/23/92)*

spring loaded w/single screw adjust (3/29/93)

*stop location (4/10/93), (4/12/93)*

*return springs*

☒{S776} Torsion spring MathCAD calculation (design never used)

*{S820} Skimmer pivot: final design (6/28/93)*

{S687} CLAMP.DWG (6/30/93)

{S688} SKMBEARN.DWG

{S689} STOPBRKT.DWG

*Skimmer pivots*

☒{S818}=>{V339} skimmer mechanism

*machined parts*

*springs*

*assembly*

*adjustability*

☒{S781} front strut design (7/19/93)

*Tooling*

☒{S800}=>{V317} Mold making. Audio:

Using a numerically controlled milling machine we are able to precisely machine the contours of the strut molds. We verify the code by running it on a piece of foam. It is much cheaper to make mistakes in a block of foam than a piece of metal. If a cutter takes too deep of a cut it will break in the metal but not in the foam. Once our code was verified, which took two tries, aluminum stock was cut to produce four mold halves. The surfaces are then hand finished and polished. Another way to have made this type of mold would have been to make male molds or plugs by hand, using templates as guides. This could have been done with wood, foam or clay as in the case of the propeller. Even with all this careful preparation we found that this groove didn't line up. This was caused by taking too deep of a cut, deflecting the cutter.

*Making the struts*

☒{S808}=>{V329} Front tees. Audio:

The general Murphy's law for glue is: if you want it to stick it won't, if you don't it will. These NC machined aluminum tee molds are prepped for the lay-up with a release wax. If these were epoxy based molds we would also spray a release agent like PVA (poly vinyl alcohol) on top of the wax. Clear templates aid in cutting of the cloth. The type of molding we are doing yields parts similar to those from RTM (resin transfer molding). The quantity and location the of carbon must be controlled precisely to insure a higher quality part. Can you think of another way to make these parts? We have a few early idea sketches that you can look at. These parts are designed very close to the maximum material strength; hand-laid carbon composite is pretty hard to match in this area. You may have asked why the commercial aircraft industry hasn't been using these materials, that military aircraft have been using for years? As you can see this is a very labor-intensive process and automation for composites is still in an infancy. Carbon composite prices, in the early 1990 are about $40.00 per pound once you consider processing and material costs.

Once the bottom halves of the mold are assembled the foil portion attaches on. Notice how the fibers are continuous from the strut to the tee; remember the fibers carry the load. The whole assembly is then allowed to cured. Careful planning and detail preperation pay big dividends in the quality of the final part. One thing we didn't give much thought to (on the first tee we pulled out of this mold) was how we were going to get this deep part out of the mold. There are few things more frustrating than a part stuck in the a mold, especially after you've spent many hours on the lay-up. Fortunately for us we were able to get this one out with repairable damage. After this ordeal we added ejector pins to the mold and the next part came out very easily.

☒{S821} Front strut =>{S690} STRUT2AS.DWG (7/23/93)

☒{S783} main foil angle adjustment/holding

*{S759} Engineer's Notebook*

push rod with racket mechanism (12/24/92)
combined with steering (12/24/92)
rotational motion transfer, so seat can slide w/o readjusting (12/24/92)
racket mechanism combined in steering (12/24/92)
consider a separate mechanism from steering (3/10/93)
cam mechanism (3/10/93), (3/21/93)
rotational motion transfer
trigger controlled clutch
use friction to hold foil in place (3/13/93)
detailed development of tight packaged system

*{S822} First iteration: cam mechanism below pivot (3/24/93 TO 7/25/93)*

{S691} CLAMP_L.DWG
{S692} CLAMP_R.DWG
{S693} ARM.DWG
{S694} LEVER.DWG
{S695} FOILCLP2.DWG

*{S823} Fourth iteration: cam mechanism above pivot (8/2/93 TO 8/3/93)*

{S696} CAM4.DWG
{S697} CLAMP4_L.DWG
{S698} CLAMP4_R.DWG
{S699} LEVER5.DWG

{S700} FOILCLP4.DWG

*{S816} Main foil control*

{V337} Main-foil control. Audio:

Here is the main foil-control cam mechanism. You'll notice the little aluminum bracket we had to make, because the seat interfered with the handle. We should have caught this is the drawing but we didn't. Accurate assembly drawings are so important; for every interference we did have we avoided 10 by doing proper drawings.

*redesign*
*machining mistakes*
*seat interferes*
*works real nice*
*pretty ugly*

☒{S784} seat

*{S831}fixed seat-back angle (12/25/92)*

{V310} How the seat design evolved

*{S759} Engineer's Notebook*

adjustable seat back angle (12/25/92)
tube-sizing calcs (4/24/93)
seat and hull have to be able to take loads of being upside down (1/17/93)
seat and steering attached (8/5/93)
concept of a two-part molded seat (9/2/93)

*{S824} Seat Assembly: redesign seat mount without third mount point (8/8/93)*

{S702} CHAIR6.DWG (8/25/93)
{S703} BRACKET.DWG (8/24/93)
{S704} CROSSTBE.DWG
{S705} FRNTTBE.DWG
{S706} REARSPPT.DWG
{S707} SIDETUBE.DWG
LAYOUT9.DWG

*{S799} Seat fabrication*

{V316} making the seat

*laying out tube to cad drawings*
*why tubes don't line up perfectly?*
*lashed and welded*
*meshed seat*

☒{S785} main-strut

*{S759}*

pinned (1/13/93), (7/19/93)
threaded pin through the center (7/4/93)

*Main struts*

☒{S809}=>{V330}. Audio:

The main struts are constructed in a manner similar to the front tees and propeller. Carbon fiber cloth is cut to size using acrylic templates. The cloth provides for both the aesthetical appearance and the torsional stiffness of the strut. Uni-directional carbon provides the compressive stiffness, which in this case is buckling-dominated. Mold inserts, like this one, provide for parts that couldn't be machined with the mold or for relief in directions out the parting-plane.

Because this strut sees almost purely compressive loads, its sizing is buckling dominated. Which means materials with high elastic modules

are required. The only non-composite materials that could possibly compete with the carbon or boron composites would be steel or titanium. Besides machining these materials could be made to net shape using lost-wax or investment-casting processes; similar to the way turbine blades are made.

The foil alignment hole broke here when we took out the insert. This happened because we neglected to put fibers parallel to the pin. This part would see these loads only in manufacturing.

☒{S825}=>{S708} RFOILAM.DWG (7/22/93)

☒{S786} propeller

*Interactive design problem on the prop bearings*
*adjustable angle of prop-blades (1/16/93)*
*bearing designs*

☒{S832}=>{V311} Design iterations. Audio:

Early on we thought to make the prop pitch adjustable, by having threaded shafts on the end of the propeller blades. To get the smallest package and lowest friction, needle bearings were first considered. This would require water tight seals running on hardened steal races. The prop shaft would be soldered, bonded or shrink fit in.

It was then decided that needle bearings would too complex. Using water lubricated journal bearings eliminates the need for seals. Also bearing loads are relatively small and are at a small diameter, making the friction penalty negligible.

The blades of the propeller are sandwiched between two cast AL pieces.

Putting a wrench down the hole at the middle of the spinner to loosen the bolt allowing the propeller blades to be pitched. This bolt also holds the entire bearing assembly together. Notice that these castings are the same, so the same part can be used in both places.

After some consideration we decided that the complexity of having a pitchable prop was not worth it. Having a fixed pitch prop would simplify the assembly greatly. Unfortunately to change the performance a selection of different propellers would be needed.

Turning the spinner bolt opens the whole assembly. The composite propeller hub mates with the stainless-steel bearing shaft on matched tapered flats to transmit the torque. The propeller acts as the thrust surface on this side and the bearing shaft provider thrust on this side. The detail and assembly drawings for this design are available for you to look at. After doing a material search we realized that we could have a stainless-steel propeller shaft that performs as well as the music wire originally planned.

This allowed for another iteration. Because this shaft doesn't require a coating and it is already pretty hard already we can run the bearings directly on the shaft.

Torque is transmitted from the propeller hub to the shaft via the mating tapers. The reverse thrust force is handled by this stainless steel thrust cone that is bonded to the shaft. The propeller hub provides the forward thrust surface. The assembly is held together with a spring pin that goes through the spinner and the shaft. An elastic spacer acts as a preload spring. Take a look at the assembly and detail drawings for this design.

This design sequence represents a six-month period of design and redesign. The first iteration has a hub diameter of over one inch in diameter and some 20 parts.

By the fourth iteration we are down to nine parts and a hub diameter of 0.38 inches. You'll notice in the video that the last prop made integrated the spinner and spacer into the propeller hub reducing part count to 7.

propeller tooling parts -- Matt's drawings

PROP2.DWG

PROP2EXP.DWG

*bushed with fixed prop (4 iterations) (1/28/93), -- the analysis on 1/29/93 is very interesting.*

*{S826} Third iteration*

PROP3.DWG

PROP3EXP.DWG

{S709} PROPASEM.DWG (6/6/93)

{S710} PROPBEARING.DWG

{S711} PROPBOLT.DWG

{S712} PROPSLEV.DWG

{S713} PROPHOUS.DWG (6/9/93)

*press fit of drive shaft into propeller calculations (1/23 to 1/24/94), (1/28/94) -- there are three iteration here.*

*{S823} Fourth iteration: redesign prop bearing so that bearing surface is the drive shaft (6/27/93)*

{S714} PROP4ASD.DWG (7/2/93)

{S715} BEARING4.DWG

{S716} PROP4.DWG

{S717} SHAFT5.DWG

{S718} SLEEVE4.DWG

{S719} SPACERS4.DWG

{S720} TAILCONE.DWG

{S721} THRUST4.DWG

*pinned spinner (7/2/93)*

{S805}=>{V323} putting prop on

*Propeller 1*

{S795}=>{V312} Propeller fabrication. Audio:

Once the propeller mold is prepared the composite materials must be cut to size. The outer layers of carbon-fiber cloth are carefully cut using a template; they provide the torsional stiffness. Surface-coat or gel-coat epoxy is then applied to the molds. The layers of cloth are carefully wetted out to eliminate voids ... Carbon tows, which are strands of carbon fiber, provide most of the bending stiffness. The amount of material is precisely measured to match the design requirements. An insert is used to shape the inside of the prop hub. Carefully planning and design are required at every stage from design to tooling to manufacturing. A weak link at any stage can be the difference between a good or bad part.

{S796}=>{V313} Load test. Audio:

This is a test prop with the left side made solely of a chopped carbon-fiber-epoxy composite. The right side is the same, but has a layer of carbon-cloth on the outside. We know what load the propeller will see, for the given power. With 15 lb applied at three-quarter of chord, this is how much this prop will deflect, clearly too much. Here's what happens at 20 lb; a strong rider wouldn't get very far this prop. We compare this the

propeller with the unidirectional fiber one; here with 25 lb at 3/4 chord. This is why advanced composites are so expensive, high performance is achieved only with precise fiber placement.

☒{S787} hull

*visit Composite Engineering (2/4/93)*

*decided to use their men's kayak mold*

*Hull*

*two days of lay-up work at CE*

{V344}

*adding ribs and bulkhead*

☒{S807}=>{V328}. Audio:

Ribs and bulk-head are added to the hull, because there is very little structure where the deck has been cut away. The ribs align with the rail screws, providing material for the screws to grab and stiffens the open section. A tough epoxy glue, as opposed to a laminating resin, attaches the ribs and bulk-head to the hull. The bulk-head is the back mounting point for the frame and made of marine-grade plywood, sealed with epoxy. Additional fiberglass strips reinforce the bulk-head.

*weight 75.2N (16.9 lbf)*

☒{S788} drive

*decide that twisted chain will not work geometrically (3/9/93)*

*size gears for right angle drive (3/9/93)*

*☒ coating test for the drive shaft*

*Fred Ackerman's tests and report*

- ✓ objective
- ✓ equipment
- ✓ test plan
- ✓ apparatus
- ✓ data sheet
- ✓ actual procedure
- ✓ observations
- ✓ conclusion

*☒{S759} Engineer's Notebook*

*More complete analysis or drive shaft*

vibs analysis establish natural frequency (2/3/93)

considering fatigue (4/26/93)

*Belt sizing and selection for the twisted belt scheme (6/10/93), (7/6/93)*

*tooling for making shaft end (7/2/93)*

*☒{S827} Bearing assembly: design bearing housing for small pulley (7/4/93 to 7/7/93)*

{S722} SPASSEM.DWG

{S723} HOUSING.DWG

{S724} FSPACER.DWG

{S725} RRSPACER.DWG

{S726} SHTSLEVE.DWG

{S727} SMPULLEY.DWG

*modes (7/14/93)*

*tooling for the pulley ring (7/21/93)*

*belt failed, gear box and 1/4" chain to be used (9/12/93)*

*Drive*

*Drive 1*

☒{S797}=>{V314} Pulley fabrication (7/30/93). Audio:

The crank-set pulley was cast in a wood mold; clay is used here to provide a smooth transition. We used a belt with the same pitch as the drive belt, as the mold for the teeth. This is a prototyping technique, how do you think is would be done in production? Before casting the pulley we need to test different materials; we settle for a mixture of cotton fibers and glass balloons for the best combination of surface finish, flow characteristics and weight. The mold assembles with a bicycle chain ring in the middle, becoming an integral part of the final pulley. We then fill the mold with our mixture. In casting plastics it is always best to get all the bubbles out of the mixture before pouring, this is done by placing the mixture in a vacuum bell -- something we neglected to do here. In any molding process whether it be injection molded plastics or cast metals, thought must be put into how the part is going to come out of the mold. You'll find later on that we didn't always do that. Fortunately here we thought ahead and provided ejector holes. After a bit of clean up this part looks pretty good ... too bad we don't end up using it ... you'll find out about that later on too.

☒{S801}=>{V318} Testing. Audio:

This drive train design uses a 1/5" pitch timing belt and achieves a 10.8 to 1 speed increase. Mark .... Notice this test idler pulley is crowned. Matt... The belt is twisted 90 degrees, eliminating the need for any gear box, but causes problems in reverse rotation. By tweaking the geometry of the idler we are able to get it functioning at almost a satisfactory level. Drive trains are inherently problematic and should always be tested under load. Mark ... Increasing the tension on the belt will help to alleviate this skipping problem. As belt tension increases the efficiency of the drive system drops off quickly. Another solution here would be a larger-pitch belt, but the pulley sizes would be much larger. Unfortunately at this pitch, even a highly tensioned belt .... skips.

☒{S802}=>{V319} mounting for bearing housing. Audio:

We attach the pulley bracket to the frame tube with a carbon epoxy composite. There is an isolating layer of fiberglass so the carbon doesn't react with the aluminum. Shrink tape is used to compact the lay-up while it is curing. The bearing housing was waxed so that it could be removed after the resin cured. We tried everything including liquid nitrogen to break it free. We finally succeeded though, another unplanned night of machining. This bracket worked out great; too bad the this drive train didn't. Well, chalk up another part to experience.

*Drive 2*

☒{S803}=>{V320} gearbox mount. Audio:

Ditching a belt-drive system we settle on a 1/4" pitch chain-drive connected to a right-angle gearbox. The gearbox mounting brackets are lashed to the frame tube with Kevlar fibers. We made the bracket out of aircraft-grade plywood. Whenever you have wood exposed to water you always need to seal it; this is why we paint the whole surface with epoxy. We are using a commercially available gearbox and have to use its mount holes. How would you have made this bracket?

☒{S804}=>{V321} gluing shaft hub. Audio:

Mark speaks ... We glue the drive shaft on the hub that attaches it to the gearbox. Both parts are stainless steel and an anaerobic glue called Loktite is used. We slightly roughed up both parts and thoroughly cleaned them before glue is applied. The amount of required glue area is determined by the maximum shear stress of the glue and the expected

torque. We multiplied in a safety factor of three above the worst-case scenario. Notice the smooth tapered transition for this part: this drive shaft is a very tightly size. Even a small stress riser would prove catastrophic.

☒{S805}=>{V323} attaching propeller. Audio:

The propeller slides on to the end of the drive shaft mating with the taper flats of the shaft. For testing of the drive train, we don't have the spinner. The spring pin goes through a small hole in the end of the shaft to hold the assembly together. Look at development the sequence for the propeller design to better understand this assembly.

☒{S811}=>{V332} chain poping. Audio:

The skimmer mold shape, based on this prototype skimmer, is formed using a modeling clay. Intensive research being undertaken to eliminate the hard model phase in things like the design of a car body. Today even the most advanced computer model doesn't provide the look and feel given by a real model -- this however, given time will change. Once satisfied with the shape we seal the clay and apply a release agent before making a carbon female mold. A set of carbon molds like these are good for maybe 50 to 200 parts depending on how they are handled. This is a real problem in at all but the smallest volumes. While the initial mold cost may be quite low they don't last very long. If you bite the bullet and pay for a very expensive steel tool it may last the entire product life, without any recurring costs.

1/4 inch chain and right angle drive
problems with commercial right angle drive.
fixing the oil problem
frame bending and solution
idler
Gearbox

☒{S806}=>{V326, S756} temperature test. Audio:

*Using a thermocouple attached to the right-angle drive we are able to track the temperature rise due to a constant input. The power comes from a cyclist wearing a heart-rate monitor to maintain constant aerobic output. The data and results are here for you to look at; they show that about 10% of the power goes to heat. Looks like we need a better gearbox.*

☒{S817}=>{V338} noisy gearbox
why so much noise?
how do you get this thing apart?
STP oil treatment, what a mess

*d-bore shaft mounting (6/28/93)*
*shear pins (6/28/93), (7/2/93)*

☒{S789} steering

☒*{S828} First iteration: center pivot selected*

{S728} ST_ASSEM.DWG (6/9/93)
{S729} ST_BASE.DWG
{S730} ST_BEARN.DWG
{S731} ST_PIN.DWG
{S732} ST_WASH.DWG
{S733} BTMRAIL2.DWG

☒*{S759} Engineer's Notebook*

transfer motion with cables

transfer motion in rotation

decouple mounting from the seat design quick-release mechanism (4/24/93)
design with quick-release mechanism and adjustable push rods (6/11/93)
steering attached to steering rail (6/11/93)

*{S826} Third iteration: steering part of the seat (8/5/93), (9/3/93)*

{S734} ST_CLMP3.DWG (9/10/93)
{S735} TBBRACK3.DWG
{S736} STEER3.DWG

*steering ratios (8/11/93)*
*using cables (9/15/93), (9/16/93)*
*{S815}=>{V336} Working system*
*Steering*
*mock up {V336}*
*why wasn't this designed?*
*lots of cable, real ugly*

☒{S790} skimmer

*arm calculations (4/22/93)*
*Skimmers*
*wood mistakes*
*temporary foam ones*
*making molds for final ones*
{S812}=>{V333} making the molds

☒{S791} frame design

*space frame (7/2/93)*
sketch of space frame
*tubular (7/16/93)*
*frame evolution*
{S738} FRAME.DWG (7/12/93)
*{S830} "square peg in a round hole"*
{S739} FRAME4.DWG (7/19/93) -- the design
{S740} FRAME6.DWG (8/5/93) -- what we made
*{S829} Eighth iteration*
{S741} FRAME8.DWG (8/16/93)
{S737} BOTBRACK,DWG (7/16/94)
{S742} SEATRAIL (7/30/93)
*simplified 7/28/93)*
*jigging (7/20/93)*
*{S798} Frame fabrication*
{V315} Audio:

Before lashing the tubes of the frame we epoxy glue them together. Whenever making a structure of this sort a carefully designed holding jig guarantees that things fit together. This is especially important when welding a frame: because things warp when welded, the jig better be strong enough to overcome the warping forces. Notice how the tubes are cut so that they mate perfectly: this is called mitering. When lashing, most fibers run in the direction of the load. Fibers running perpendicular, hold and tension the load fibers. The concept behind composite structures is beautifully illustrated in natural composites. Next time you look at a knot in a piece of wood, notice the grain. See if you can tell which side of the knot was the top. Before the resin cures, if heated it will flow better, wetting out the fibers thoroughly: this is very important for a good

composite. I forgot to mention one very important thing about the jigging: if you lay it out wrong you may have to make more than one frame. This, of course, can give you an opportunity to do some redesign: look at how the crank bracket is mounted on this frame, as opposed to the one earlier in this segment. Take a look at the drawing marked: "square peg in a round hole;" this is our attempt to see if we could still use the bad frame.

☒{S792} Outriggers

Note about coupling or uncoupling>

{S814}=>{V335} Swimming lessons: outrigger evolution. Audio:

Outriggers on either side of the rider provide rollover stability in the preflight condition. Their size and location were based on some basic rollover and buoyancy assumptions. Here is the first launch of the Skeeter. It is not always possible for the design engineer to have the opportunity to test his or her design. Think of the designers of the Space Shuttle; most of them never even get a chance to even sit in it. If you do get such the chance, the flaws in the design become readily apparent.

OK, so the outriggers are too small to start with. A little trial and error on the dock is a great way to figure this one out... Still too small... Great, seems like just enough to float. Aa, steering anyone? Floating and turning seem hard enough: are we sure we can get this thing to fly?

Taking what we learn on the dock, gives us the volume of foam for the next generation. Shape, come on. In hindsight the shape of these things are really wrong, but at the time, well... These things are more like water plows.

Finally, we start thinking; with this design we are able to fly! The only problem with it is in rough water, it bites the waves. I must say in our defense we figured when we started that these things were going to take some tinkering and they did.

With all that under our belts, some proper CAD work and with the help of a numerically controlled foam cutter the final outriggers are glassed on. They don't look half bad either.

*notebook sketches*

☒{S793} Rudder

*cable break*
*too small*
*still too small*
*nice and big*

Testing

outrigger problem -- getting it to float
chain-slipping mystery
gearbox

video of busted-up gearbox

propeller pitch

Conclusions

were goals achieved?
a good flight of the hydrofoil
recommended books
☒ Tech Talk article
☒ Dutch film

☒ CNN
☒ Beyond 2000
☒ National Geographic