UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

DEVELOPING AND IMPLEMENTING WIRELESS PERSONAL DIGITAL ASSISTANT (PDAS) APPLICATIONS FOR TRAINING AND EDUCATION

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

degree of

MASTER OF SCIENCE

(Electrical Engineering)

By

SRUJAN REDDY PEDDY

Norman, Oklahoma

2002
DEVELOPING AND IMPLEMENTING WIRELESS PERSONAL DIGITAL ASSISTANT (PDAS) APPLICATIONS FOR TRAINING AND EDUCATION

A THESIS APPROVED FOR THE SCHOOL OF ELECTRICAL ENGINEERING

BY

Co-Chair: ______________________________________

Dr. Kurt Gramoll

Co-Chair: ______________________________________

Dr. John Cheung

Member: ______________________________________

Dr. Jerry Crain
ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to Dr. Kurt Gramoll for his guidance, support and motivation through out my Master’s Thesis. I am extremely thankful to Dr. John Cheung, Co-Chair and Dr. Jerry Crain, committee member for their suggestions and time.

I am grateful to all of the Engineering Media Lab members for their co-operation and friendship. I would like to thank all my friends who supported me throughout my Master’s education and made my stay at University of Oklahoma an enjoyable experience. I would also like to thank Sarma and Suresh for their encouragement and support throughout my stay in USA.

Above all I thank my parents and sister for their support and encouragement.
# TABLE OF CONTENTS

**LIST OF FIGURES** ................................................................................................................................. viii

**LIST OF TABLES** .................................................................................................................................... x

**ABSTRACT** ................................................................................................................................................ xi

**CHAPTER 1 INTRODUCTION** ..................................................................................................................... 1

  1.1 E-learning Usage ................................................................................................................................... 2
  1.2 Role of Multimedia in E-learning ............................................................................................................. 2
  1.3 Adapting technology and process ............................................................................................................. 3
    1.3.1 PDA .................................................................................................................................................. 3
  1.4 Objective Of the Research ....................................................................................................................... 7

**CHAPTER 2 LITERATURE SURVEY** .............................................................................................................. 9

  2.1 Online learning for education and training purposes ............................................................................... 9
  2.2 Changing trends in e-learning ............................................................................................................... 9
    2.2.1 PDAs role in e-learning ................................................................................................................... 10
    2.2.2 Features available in PDAs ............................................................................................................. 10
  2.3 Wireless devices in classroom .............................................................................................................. 11
  2.4 PDAs usage in industrial training ........................................................................................................ 12

**CHAPTER 3 PDA BASED EDUCATION** ........................................................................................................ 13

  3.1 Target Device ......................................................................................................................................... 13
  3.2 PDA based E-book ............................................................................................................................... 14

**CHAPTER 4 DESIGN OF PDA BASED TRAINING SYSTEM** ............................................................................. 18

  4.1 Navigation and Layout ........................................................................................................................... 18
  4.2 User Interface ....................................................................................................................................... 20
  4.3 Learning System Structure .................................................................................................................. 21
4.4 Design of Pocket PC based Training System .......................................................... 22
4.5 Tracking user progress .......................................................................................... 23
4.6 Implementation ..................................................................................................... 24

CHAPTER 5 CREATING ONLINE LECTURES .............................................................. 26
5.1 Online Lecture Formats ....................................................................................... 26
  5.1.1 Text-Based Online Content ........................................................................... 26
  5.1.2 Multimedia-Based Lecture ........................................................................... 27
  5.1.3 PowerPoint-Based Lecture .......................................................................... 27
  5.1.4 Full-Motion Video Lecture .......................................................................... 27
5.2 Wireless Chalkboard ......................................................................................... 28
5.3 Online Lecture Generation System ..................................................................... 30
  5.3.1 User System .................................................................................................. 31
  5.3.2 Server Side .................................................................................................... 34

CHAPTER 6 DESKTOP/PDA BASED FREE BODY DIAGRAM BOARD ..................... 36
6.1 Quiz Structure .................................................................................................... 38

CHAPTER 7 DESIGN TOOLS ..................................................................................... 43
7.1 Windows CE and Pocket PC ............................................................................. 43
7.2 Embedded Visual C++ ....................................................................................... 43
7.3 Microsoft Windows Platform SDK for Pocket PC ............................................ 44
7.4 Microsoft Visual Basic ....................................................................................... 44
7.5 ActiveX Control ................................................................................................. 45
7.6 COM ................................................................................................................. 45
7.7 Active Server Pages .......................................................................................... 46
7.8 ASP Flash Turbine ............................................................................................. 46
7.9 Microsoft Access .........................................................................................................47
7.10 Macromedia Flash .......................................................................................................47
7.11 Macromedia Dreamweaver ........................................................................................48

CHAPTER 8 CONCLUSIONS ........................................................................................... 49
8.1 Accomplishments ........................................................................................................50
8.2 Recommendations for Future Research ...................................................................50

REFERENCES ..................................................................................................................... 52

APPENDICES ..................................................................................................................... 55
Appendix A ASP Script for generating dynamic Flash movies ...................................... 56
Appendix B ActiveX control Visual Basic Code .............................................................. 57
Appendix C ActiveX control eMbedded Visual C++ Code .............................................. 65
LIST OF FIGURES

Figure 1.1: Pocket PC based device  6
Figure 1.2: Palm based device  6
Figure 3.1: VRML browser for Pocket PC  14
Figure 3.2: Home page for Statics web portal  15
Figure 3.3: E-book for Electrical and Statics topics  16
Figure 3.4: Example problems  17
Figure 3.5: Quiz Section  17
Figure 4.1: Login page for Pocket PC  19
Figure 4.2: Login page for Desktop/Laptop  20
Figure 4.3: Learning System Structure  22
Figure 4.4: Objectives of a selected module  23
Figure 4.5: Content explained vocally  23
Figure 4.6: Images, animation and testing by quiz makes training interactive  24
Figure 5.1: Tablet PC  28
Figure 5.2: Lecture generation process  30
Figure 5.3: Block diagram of the online lecture generation process  31
Figure 5.4: ActiveX control drawing board  32
Figure 5.5: Freehand drawing of forces acting on a block  34
Figure 5.6: Final Lecture shown as a movie  35
Figure 6.1: General Quiz format  38
Figure 6.2: General Quiz format – Pocket PC device  39
Figure 6.3: Drawing board pops upon the main window  40
Figure 6.4: Giving force magnitude and angle 41

Figure 6.5: Force magnitude and angle 41
LIST OF TABLES

Table 1: Available Pocket PC device specifications 7
ABSTRACT

E-learning should continue to encompass more than one new available technology for creating more interactive education environments. One such technology is the wireless Personal Digital Assistant (PDAs), which can be used for e-learning. With the usage of wireless Personal Digital Assistants in classrooms the learner can engage in higher level thinking through interactivity and feedback.

This thesis researches how the PDAs can be used effectively for education and training, and proposes the development of training and education aiding Pocket PC applications. In order to create a collaborative learning system, sharing of information between learners is an important aspect. Also accessing and sending information irrespective of the time and place form a vital part of an online learning system. PDA with its wireless features, small size, makes it an ideal tool for accessing, sending and sharing information. In the process of creating a collaborative learning system environment a PDA specific web portal was developed so that students can access vital course related information either in a classroom or from any other remote place. A web portal including e-book, examples, problems, and sample examinations for the Fundamentals of Engineering Examination was developed for Pocket PC. This portal allows students to access the learning material in class or at any remote place at any time with ease.

PDAs use pen-based input and hence are best suited for drawing applications. A drawing tool was developed for PDA that allows the user to draw a free body diagram. Online classes generally use multiple-choice questions for testing the knowledge gained by a student through e-learning. In engineering courses, a diagram helps to elaborate the
problem before actually performing the calculations. To make the testing process more real-time and compatible with all PCs, the drawing tool was developed for both PDAs and laptops/desktops. PDAs will use pen-based input for drawing, and laptops/desktops will use the mouse. This drawing tool helps the students elaborate more the concepts they understand in the form of a diagram. During a test the student submits the multiple-choice answer along with the diagram. The drawing tool is actually an ActiveX control that will be installed on the user’s system through the web. Once installed ActiveX control can be called-up through the course page wherever necessary. The multiple-choice answer, along with the diagram information, is passed to the server and stored in a database. A Solution recognition program is used to compare a student’s diagram with the actual solution. The diagram is recognized based on the given solution for that particular question. The comparison results are used for grading purposes. Grading of the whole test, which includes the multiple-choice questions and free body diagram, is done the moment student submits the solution.

Wireless PDAs depicting the system on the training field create an ideal environment for the learning process. Hence training applications which can be accessed wirelessly on the field was developed for Pocket PC. The Tinker Air Force Base Online Training System currently hosted by the University of Oklahoma (http://tinker.ou.edu) implements web-based online training support for Tinker Air Force Base Personnel. This system provides training about various environmental regulations, and also training for the correct maintenance procedures for cabin pressurization of the KC-135 series aircraft. Some of the modules of the environmental regulations were developed for Pocket PC for use in training and guiding workers at the training site.
CHAPTER 1 INTRODUCTION

E-learning is an online-based knowledge repository, which provides various modes for learning, including education, training, and collaborative learning, apart from acting as a medium for communication and information sharing. The system through which dissemination of information is done consists of a computer, a network and Web-based technologies. The network can be the Internet (World Wide Web), or a company/institution intranet/extranet. The end user also needs a tool through which the content is displayed. It can be a desktop computer, a laptop or a wireless personal digital assistant (PDA). This thesis discusses the role of PDAs in education and training and developing related applications.

Whether e-learning or classroom based learning only the engaged learners learn more. E-learning has an advantage of making the class more interactive by the usage of multimedia [1] and mobile learning technologies. E-learning has features like streaming audio and video, animations explaining the topic, simulations creating the virtual environment of the topic properties, links to the related information on the web and e-book covering all the details of the topic. With the help of these features, e-learning has made it possible for learners to choose what they want to learn, how they want to learn, and when they want to learn. Also e-learning has the advantage of real-time expertise, hands-on experience and experimentation. With the availability of advanced media streaming, higher bandwidth and mobile learning technologies, in the future e-learning will have interesting implications on how people are educated [2].

With its high degree of accessibility and flexibility e-learning can cater to individual learning styles, of how they want to learn and when they want to. E-learning
thus has the potential of making a significant difference in educating/training an individual or improving organizational performance. With a web-enabled environment, individuals can access courses, individual topics and performance support resources at any time and from anywhere, while in the office, at home or while traveling [3].

1.1 E-learning Usage

The e-learning market covers the academic, corporate and consumer fields. Broadly the fields fall under two categories called education and training which is generalized and defined as follows.

- **Education**: Uses rich multimedia and interactive content in delivering the content to the end user by the usage of Internet tools. Makes possible to have interclass collaboration.
- **Training**: E-learning provides the means of providing company specific training to the workforces.

The availability of broadband technology has increased the online learning options by making use of videoconferencing, advanced animation techniques, and virtual scientific laboratories. In addition to its cost-effectiveness, its main advantages are its flexibility, its convenience to users, its wide reach, its easy accessibility, its consistency and its repeatability.

1.2 Role of Multimedia in E-learning

Delivering content to the end user and thus imparting the skills and knowledge is the ultimate goal of e-learning. The presentation of content needs to be compelling to the audience it targets, offering the learner a resource that is seen as appealing, valuable and productive to his/her goals. This is made possible with the usage of multimedia e-learning
programs. Audio, video, animations and simulations play a vital role in the learning process [4]. The inclusion of audio and video makes learning much more effective [5]. Effective use of audio and video in online training courses requires streaming technologies. Streaming technologies allow us to send audio, video and other forms of multimedia files across standard computer lines (in small packets) while at the same time being able to hear or see the files while they are downloading. In other words, we see or hear what is available the minute it arrives, without having to wait for the entire file to download before we start.

1.3 Adapting technology and process

Educators use different tools to streamline their work. E-learning should continue to encompass more than one new technology that is available in creating a more interactive education environment.

Encompassing technology into education has led e-learning through CDROM, Internet/World Wide Web, multimedia and mobile technologies. Using technology in implementing a learning system improves quality of learning apart from improving the access to education and training. Mobile and wireless technologies are inspiring educators and learners with their flexibility and adaptability to different education and training environments [5,6,7]. Various Mobile and Wireless technologies existing in the market can be classified as PDAs (Personal Digital Assistant), laptop, mobile communication products etc.

1.3.1 PDA

PDA, a term coined in 1992 means Personal Digital Assistant, which was basically used to store contact information. With implemented features like wireless
connectivity, IR beaming, and synchronization gone are those days where PDAs were used to act as mere organizers. PDA is a convenient device that can be used in developing e-learning applications and in training purposes. As they can receive and send information at any place and at any time, they have the ability of creating a collaborative learning environment. The type of applications it supports differ from accessing websites to running PDA specific applications such as Shockwave applications, Visual Basic applications, C++ applications, ActiveX controls etc.

1.3.1.1 History of PDAs:

The first truly functional PDA was developed by Apple Newton in 1993. However, Newton sales were lower than expected and was soon cancelled. Palm Computing founded what we now refer to generically as the “Palm”, in 1992. In 1995, US Robotics acquired Palm Computing, and the Palm Pilot 1000 and 5000 were released in March 1996. After the release of these units, the market for the PDA grew dramatically each year, and 15.5 million PDAs are expected to be shipped in 2002 [8]. Currently the major platforms for handheld devices are Pocket PC, Palm OS, Symbian OS, Blackberry and Linux.

Currently Palm, Handspring, Sony, TRGPro and IBM manufacture the most popular Palm OS devices. All of these companies together currently control more than 86 percent of the handhelds sold [9].

Microsoft entered the PDA market by having the first version of Windows CE installed on the Sharp Electronics Mobilon in 1997. Windows CE has gone through several versions, with the latest version termed "Pocket PC". Casio, Compaq, and Hewlett Packard manufacture the most popular Pocket PC devices. With the latest release
of the Pocket PC OS, sales have increased. Pocket PCs now represent over 10 percent of all handheld devices sold [9], and offer greater multimedia capabilities than Palm OS units.

1.3.1.2 Disadvantages with PDAs:

The screens on current PDAs are small and their memory is limited so they are not intended to replace a laptop or desktop computer. PDAs with color screens are not cost effective to use.

1.3.2 Types Of Devices Available:

Currently, there are two major types of PDAs in the market. They are the Microsoft’s Pocket PC and Palm OS handhelds. The main difference between the Palm and the Pocket PC is multimedia. The term multimedia describes a number of diverse technologies that allow visual and audio media to be taken and combined in new ways for the purpose of communicating. Palm is simple for organizing appointments, addresses, and to-do lists. The Pocket PC has almost all features of a notebook computer.
Pocket PC is a new windows operating system created specifically to power a handheld computer. The system architecture features software modules that can be assembled in a variety of ways. By using only essential components, software size can be kept low to ensure performance is high. Developers are familiar with windows tools, which can be used in developing the required applications. eMbedded Visual C++ software for Pocket PC was used in developing the drawing tool for online quiz.

Macromedia flash player plug-in is available for Pocket PC, which enables multimedia content to be viewed on Pocket PC. Pocket PC's Windows Media Player can play MP3 files and video files. Another important feature is the availability of VRML browser with which 3D visuals can be viewed. E-learning becomes more effective with multimedia usage in describing courses. Support for some of these multimedia features in Pocket PC has helped in choosing this particular device as an e-learning tool. Multimedia
feature is used in developing the training system. Another main advantage of Pocket PC is that it supports multitasking. The discussed features are responsible in choosing Pocket PC as tool for the e-learning process.

### 1.3.2.2 Some available Pocket PC device specifications

The table below shows the various Pocket PC devices available in market along with their specifications.

<table>
<thead>
<tr>
<th>Model Name</th>
<th>Display</th>
<th>Memory</th>
<th>CPU</th>
<th>List Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compaq iPAQ H3955</td>
<td>240x320</td>
<td>64 MB SDRAM</td>
<td>400 MHz Intel X-Scale processor</td>
<td>$499.94</td>
</tr>
<tr>
<td>Toshiba e740</td>
<td>240 x 320</td>
<td>64 MB RAM and 32 MB ROM</td>
<td>400 MHz Xscale processor</td>
<td>$599.94</td>
</tr>
<tr>
<td>Casio Cassiopeia EM-125</td>
<td>240 x 320, 65,536 color or monotone</td>
<td>32MB RAM, 16MB ROM</td>
<td>150 MHz VR4122</td>
<td>$299</td>
</tr>
<tr>
<td>Casio Cassiopeia EM-500</td>
<td>240 x 320, 65,536 color</td>
<td>16 MB RAM</td>
<td>150 MHz VR4122</td>
<td>$399</td>
</tr>
<tr>
<td>IPAQ H3955</td>
<td>240 x 320, 4,086 colors</td>
<td>64 MB SDRAM</td>
<td>400 MHz Intel X-Scale processor</td>
<td>$499.94</td>
</tr>
<tr>
<td>IPAQ H3635</td>
<td>240 x 320</td>
<td>32 MB RAM, 16 MB ROM</td>
<td>206 MHz Intel StrongARM SA-1110 32-bit RISC</td>
<td>$549.99</td>
</tr>
<tr>
<td>Jornado 525</td>
<td>240 x 320, 256 colors</td>
<td>16 MB RAM, 16 MB ROM</td>
<td>133 MHz 32-bit Hitachi processor</td>
<td>$299.99</td>
</tr>
</tbody>
</table>

*Table 1. Available Pocket PC device specifications.*

Source: [10,11,12,13,14]

### 1.4 Objective Of the Research

E-learning can be more effective with the introduction of PDAs. Though PDAs can’t be used for all the online learning features currently implemented for desktops/laptops, certain device applications are suggested keeping in mind the processing and the wireless transmission speeds. This thesis discusses the usage of PDAs in education and training, and proposes the development of training and education aiding Pocket PC applications.
The objective consists of the following features

- Build a PDA specific portal consisting of e-book, examples, animations, and quizzes. The portal provides access to all the course material through web from any remote site. Chapter 3 explains this portal.

- Develop an online training system for PDAs. Whole module is developed in Flash. The importance of this module is that on-field training can be given at the site where actually the process is taking place. Chapter 4 explains this training system.

- Build an online lecture generation system. This tool can be used to record/create lectures and can be made available online as Flash movies. This tool is accessed through Internet Explorer and has the ability to record/create lectures from a remote site and stores them on the server. This serves as an important feature of online learning. Classroom lectures are recorded and made available online dynamically. Chapter 5 explains this system.

- Implement a drawing tool for both laptops/desktops and PDAs that is to be used with e-courses quiz. The drawing tool is used to draw free body diagrams while answering statics quiz questions. These diagrams and multiple-choice answers to the quiz are graded on the server. This feature has been implemented for both desktops/laptops and Pocket PC. Currently, online quiz uses only multiple-question format to test learner’s knowledge. This drawing tool helps in giving partial credit to the free body diagram. Chapter 6 explains this drawing tool in detail.
CHAPTER 2 LITERATURE SURVEY

2.1 Online learning for education and training purposes

Online learning has the potential to change the way people learn. As it has no specific time schedule, the learner has the option of learning on demand. Online learning is in the process of becoming truly parallel to the traditional classroom-based learning. This involves creating more engaging multimedia learning scenarios and adapting new upcoming technologies into the learning process [15]. The most sophisticated online learning tries to bridge the gap between the instructor and students and thus increases the interaction between them.

Education and training is poised to become one of the largest sectors in the world economy. Merrill Lynch estimated global expenditures on education and training at over US$2 trillion. About one third of this spending is in North America, half in Europe and the other developed market economies, and 15 percent in the developing world [16]. Corporate and campus agendas have started to recognize e-learning as having the power to really transform the performance, knowledge and skills. The International Data Corporation estimates that the corporate spend on e-learning alone will increase from $1bn in 1999 to over $11bn in 2003 [17].

2.2 Changing trends in e-learning

E-learning continues to find new ways to reduce learning time and thus increase the learner’s retention. The goal is to deliver the most engaging key course material concepts in the most effective and efficient manner. This involves utilizing the upcoming technologies in creating the learning environment and testing for its reliability. One such area where there has been a large increase in research activity is the personal
digital assistant (PDA) usage as a learning tool. This is in part due to its small size and information availability anywhere and at any time.

### 2.2.1 PDAs role in e-learning

PDAs can serve as an important tool in education and training. PDAs come with various important features like wireless connectivity, handwriting recognition, synchronization, IR beaming, ability to develop customized applications, etc. These features can help improve collaborative-based learning and design [18].

Physicians were quick adopters of PDA devices, since the units provided the ability to quickly record notes, reference drug information, and track multiple patients [19]. Dr Steven Lakoff used PDAs in clinical settings with the Constellation Project [20]. This project was designed to explore the uses and benefits of mobile computing within a medical residency setting. The release of the Palm Pilot 1000 increased physician use of PDA devices, and software was quickly devised that allowed for rudimentary medical calculations, storage of large amounts of reference information, and patient tracking. Currently a large number of health-related PDA Web sites are available that offer free information and software, while others provide subscription-based information, including complete electronic books.

### 2.2.2 Features available in PDAs

PDAs that used to be mere organizer are coming with various features making it an important tool in education and training. Many PDAs use a pen like stylus to enter information. PDAs typically have 2 MB to 32 MB of built-in memory. We may also be able to add memory or removable storage to a PDA, or connect it to an external monitor, a network, or a modem. Printers and external keyboards can be added to PDAs
as well. Monochrome or gray-scale displays 2 to 16 grays; 256 to 64,000 colors. PDAs multimedia capabilities allow it to play a full-motion video with sound. It is an important aspect for any educational or training purpose. Synchronzing with PC through Serial, USB and IR is possible. Wireless connection through ethernet, wireless LAN, IrDA and cell phone is possible.

2.3 Wireless devices in classroom

PDAs help students gain better access to student services and information. The device can be used by the students in accessing and sharing course related information, participate in classroom discussions, etc. North Star Middle School [21], Eau Claire, Wisconsin, has given Palm handhelds to some selected students in grade 6-8. They were used for beaming notes, downloading assignments and tracking tasks. It resulted in effective collection, organization, reviewing, and sharing of information.

Specific course-related software could be prepared using the PDA related design environment and tools which would be an interactive teaching environment that takes the user input in creating a design or computing the result. Dr. Terry E. Shoup [22,23] discusses the utility of palm-sized computers for augmenting the design of machine elements. He developed a program that handles the manipulation of a three-dimensional state of stress and stress concentration factors. Course specific application development for PDAs plays an important role in creating collaborative learning environment as these programs facilitate a computational process that would not be possible with a traditional hand-held calculator.
2.4 PDAs usage in industrial training

In industry, PDAs are used to increase the efficiency and productivity of the employees and also for better service to the customers. These devices come in handy especially during on field training or acquiring information [24]. Albert Florens Storm (Eye institute) doctors [25] use a PDA specific program that allows them to enter information into the palm device. The software’s features, which include automatic time and date stamping, pull-down menus of surgical procedures and a virtual numeric keyboard, make it possible to enter a report in less than ten seconds. The data is then synchronized to a central database, where administrators can query it any way they choose. The resulting database is a valuable tool for analyzing doctors’ performance and treatment patterns.

Gartner Dataquest [26] projects that 15.5 million PDAs will be shipped in 2002, an 18 percent increase from 2001 shipments of 13 million units. E-learning will account for almost half of the projected $16.9 billion business skills training market by 2004, while growth in CD-ROM, videotape and satellite training will slow considerably, according to Cushing Anderson [27], program manager of learning services research for market analyst IDC in Framingham, Mass. E-learning is going to see numerous education/training programs developed for PDAs that would aid in coming up with an interactive learning environment.
CHAPTER 3  PDA BASED EDUCATION

PDAs come with various features such as wireless connectivity, handwriting recognition, IR beaming and synchronization that can be used in creating a collaboration-based learning and design system. With the usage of these features E-learning and training can be more effective as the learner can be engaged in higher level thinking through interactivity and feedback [28]. Engineering subjects require solving problems with the help of diagrams. PDAs are best suited as a drawing tool as they have pen-based input. This thesis discusses the usage of PDAs in education and training, and proposes the development of training and education aiding Pocket PC applications.

3.1 Target Device

The research objective is to develop an easy-to-use PDA based web portal for engineering education and training. Before actually continuing with the development of the PDA-based web portal, selection of target device was crucial as it is important not only to come up with an interactive information system but also the device should be able to support the features which we are looking to implement on the device [29].

Multimedia tools developed for Pocket PC [30], such as Macromedia Flash Player [31] and VRML browsers aid in viewing 2D and 3D visuals respectively. Animation developed using Flash can have sound embedded into flash movies. Currently Flash 4.0 and Flash 5.0 versions are supported on Pocket PC. This is one of the limitations as the latest version available for desktop/laptop, that is, Flash 6.0 is not available for Pocket PC and hence cannot use all the features available in Flash. Pocket Cortona is a 3D browser for viewing VRML scenes on Pocket PC and Handheld PCs. This is an important feature for creating an effective training environment as it supports development of data
visualization, location-based navigation, field maintenance, and field sales applications [32,33]. Considering the multimedia features available, Pocket PC has been chosen for implementing the proposed project.

![VRML browser for Pocket PC.](image)

**Figure 3.1:** VRML browser for Pocket PC.

### 3.2 PDA based E-book

Information from wireless PDAs can be accessed from anywhere within the wireless service region. Also their lightweight and less start-up time makes accessing and sending information much easier. Hence a web portal separately built for PDAs serves a great purpose as it gives the learners more flexibility in learning [34,35]. Fundamentals of Engineering Examination, conducted by National Council of Examiners for Engineering and Surveying (NCEES), is used to determine the examinee’s understanding of basic science, engineering science, and discipline-specific subjects. A web portal including e-book, examples, problems, and sample test for Fundamentals of Engineering examination in engineering is developed. The topics in electrical engineering and statics section are covered. Each topic would contain brief descriptions about the topic with examples, simulations and problems.
A PDA based web portal for statics and electrical engineering section is developed. One such site is shown in Figure 3.2. The portal discusses each topic in detail with theory, figures, animations, examples, and problems. To minimize the file size, animations and figures were done in Flash, which uses vector-based graphics. In most cases, vector-based animations are 10% the file size as pixel-based animations, such as AVI, MPEG, or QuickTime files. However, one drawback of vector-based animations is that video cannot be used. The animations usually consist of both graphic motion and narration explaining a certain topic.

**Figure 3.2:** Home page for Statics web portal listing Topics.

Each topic is divided into three sections. One section deals with the theory part, explaining in detail the concepts of that topic. Figure 3.3. shows the theory part covered
in “3 Phase Circuits” and “Statics: Vectors and Forces” section. This first section contains images and simulations developed in Flash apart from theory.

**Figure 3.3:** E-book for Electrical and Statics topics.

The second section discusses some examples on the selected topic. These are mostly problems containing equations and images. Figure 3.4 is one such example.

The third section is the quiz section in which the user can test his/her knowledge gained by solving the problems. Figure 3.5 is one such example. The problems are developed using Flash, as they are made interactive. That is, they take user input and show the solution if the user chooses the right answer; choosing a wrong answer will give the option of going back and selecting a different answer.
Example 2

An unbalanced Δ-connected load with an effective line voltage is shown in Fig. 3. Calculate the line currents.

Figure 3.4: Example problems.

Problem 3

Problem 4

Figure 3.5: Quiz Section.
CHAPTER 4  DESIGN OF PDA BASED TRAINING SYSTEM

Wireless PDA devices can be used for training purposes, since the units provide the ability to quickly record notes, reference information, and track multiple events. With the aid of wireless PDAs the employees can be given on-field training. The main advantage of giving on-field training is that the employee can observe the practical performance of the equipment/product while demonstrating the features of the equipment/product theoretically [36]. The wireless feature helps in providing information irrespective of the field site and also helps in monitoring the employee training progress. Usage of multimedia technologies in delivering the training information makes the training more effective. Animations, sound, and graphics on the PDAs depicting the system on the training field create an ideal environment for the learning process.

This chapter presents the various technical aspects involved in designing the web-based training system for PDAs. The different components that constitute a training system are the database, front-end, learning environment, and server. In this chapter, the design of a database, front-end, general framework of the lessons structure, and the various user-tracking features are explained.

4.1 Navigation and Layout

The Tinker Air Force Base Online Training system developed is currently hosted at the University of Oklahoma and can be accessed at http://tinker.ou.edu. Tinker Air Force Base Online Training is a web-based online training support for Tinker Air Force Base personal. It provides training on various environmental regulations and also trains on the correct maintenance procedures for cabin pressurization of the KC-135 series aircraft. The system trains and guides employers to meet the standard requirements.
Throughout the website, the trainee is taught what each regulation is and how to meet each requirement. Two common plug-ins, Flash and Director Shockwave players, are used for delivering the learning material.

To demonstrate the usage and advantages of wireless PDA devices for training purposes, one lesson in the Aerospace NESHAP module was taken and all its objectives were implemented for Pocket PC devices. Since the Director Shockwave plug-in is not available for Pocket PC, all the lesson objectives are developed using Flash. Flash plug-in for Pocket PC is used in devices for delivering learning material. The Flash lesson objective uses images, graphics, and animations in explaining the content. Also the developed Flash movies have file size in the range of 5 KB to 1 MB. Due to their small size it is easier to transmit Flash files wirelessly.

![Internet Explorer 8:35a](http://eml.ou.edu/ppc/Tinker/tink.png)

**Figure 4.1:** Login page for Pocket PC.
4.2 User Interface

The homepage detects the type of browser the request is coming from; that is, whether it is from a desktop/laptop or a Pocket PC device. Depending on the browser type, the homepage redirects to the respective training material prepared for the specific device. The first page that appears when the training system is accessed is shown in Figure 4.1 and Figure 4.2 for Pocket PC and desktop/laptop respectively. The user should have a valid login ID and password to access the training material. The login ID is also used in tracking the user training process.

Figure 4.2: Login page for desktop/laptop.
4.3 Learning System Structure

A modular and hierarchical structure is incorporated in designing the learning system. The learning material is classified into modules at each level. The structure of the learning system is shown in Figure 4.3. The structure is designed in close relationship to that of an academic curriculum. The top level in the system structure is broadly divided into categories called modules. This stage is similar to the different streams and majors in the academic curriculum that a student can take. The modules consist of individual lessons that provide information on “module” related topics. The lessons are further divided into objectives that provide a fundamental learning experience for the users with explicit material focusing on specific topics. These objectives, which are the building blocks for the entire learning system, are comparable to the different chapters in a textbook.

The last level in the learning system is the content organization and presentation in each objective. The contents in an objective are displayed by segregating them into frames. This level in the system is similar to the pages of each chapter in the textbook. As each chapter may vary in the number of pages, the objectives are also designed to vary from 1–16 frames, depending on the volume of contents. Though each frame displays specific content, they are made self-contained in a single environment for an objective. This is done to maintain continuity in learning and minimize waiting time to load each frame. This frame-based structure is designed to display small chunks of material and help the user grasp concepts gradually before proceeding to the next one. The frame-based system combines sound, images, and quizzes with the material in delivering the information to the end user.
4.4 Design of Pocket PC based Training System

As stated in the previous section, the Director Shockwave plug-in is not available for Pocket PC devices, so all lesson content is developed using Flash. The training objectives of a particular module are displayed in a plain HTML page as shown in Figure 4.4. The user chooses an objective to start with and will have to go through all the frames step by step before taking the quiz. All frames, except the quiz, contain text material delivered with a voice explaining the contents with or without images and animations.
4.5 Tracking user progress

Once the user logs in, the server can track the objectives taken or completed by user. A database resides on the server, which stores all the training information of the user. Each objective quiz score is stored for that particular user in the database. The information from the Flash frames to the database is passed with the help of ASP files. Upon submitting the quiz, an ASP file is called which collects the answers and stores them into the database before redirecting to the next objective. The quiz format is shown in Figure 4.6.
Figure 4.6: Images, animation and testing by quiz makes training more interactive.

Flash files have the advantage of showing animations and embedding sound that makes the training environment more effective. Also the vector based drawings helps in reducing the file size compared to other available image formats. In most cases, vector-based animations are 10% the file size as pixel-based animations, such as AVI, MPEG, or QuickTime files.

4.6 Implementation

Tinker can use these PDA based modules in training Air Force Base personnel on the field where they need to observe the work operations of a certain equipment. For example consider the NESHAP testing requirements module explaining proper solvent waste addition and removal procedure. It would be more helpful had the trainee had the opportunity to see how the solvent addition is done and sludge is stored in sealed
containers. Having this module available through wireless PDA, gives the trainee the option of actually going to the specific proper solvent addition site and taking training lessons over there. Currently the PDA based module training is still in testing phase at Tinker on shop floor learning.
CHAPTER 5  CREATING ONLINE LECTURES

Lectures are a key aspect of any instructional process. Formal lectures can be an efficient and effective way of delivering information and ideas to students. Making lectures available online is the feature that would make e-learning a full fledge online learning tool. Though classroom lectures are currently delivered over the web using video streaming technologies, there are many disadvantages associated with this process. One disadvantage is the time taken in making and editing the lectures. Another disadvantage is the movie size, which makes it unreasonable to keep it online unless the whole lecture is broken into parts. To overcome these obstacles, a Flash movie creation of the formal lecture is suggested, which supports vector-based input data and sound file in MP3 format. The Flash movie generation is a dynamic process; that is, the moment lecture delivery is finished it will be made available online and does not require any manual editing.

5.1 Online Lecture Formats

An online lecture is actually a classroom lecture that is converted to online media based format. It is a popular method of delivering information and ideas in e-learning systems. Some of the available online lecture formats are discussed below.

5.1.1 Text-Based Online Content

A text-based content involves breaking a topic into sections and presenting each idea with clear headings and graphics. It is a navigable document with hyperlinks that take the student to further information under the clearly marked headings. This is also commonly called an e-book, which organizes all topics in chapter form as in a textbook.
Text-based format should not act as a substitute for formal classroom lectures as it can be a dull process for the student reading the material and also lacks interactivity.

5.1.2 Multimedia-Based Lecture

Multimedia-based lecture is an effective instructional program that integrates video, animation, sound, text, and graphics in explaining a topic. An advantage of Multimedia-based lecture is that the file size can be made smaller by the usage of vector graphics and also the lecture can be made interactive. The disadvantage is the nature of making animated lectures for all topics is time consuming.

5.1.3 PowerPoint-Based Lecture

The method of delivering lectures using MS PowerPoint is the most commonly used process of delivering classroom information. A PowerPoint presentation is easy to upload onto the web site and make available online. The file size is relatively small compared to video that makes it much easier download the lecture. If audio is embedded into the PowerPoint lectures explaining the topic, then the process comes closer to the formal classroom lecture except that it would have slides like a presentation and it would not involve any of the details of the lecture presentation explained on the classroom board.

5.1.4 Full-Motion Video Lecture

Another method of making online lectures involves recording the formal lecture given in a classroom. The recorded lecture is then modified and placed on the web. This is the best form of making online lectures as compared to the previously mentioned lecture formats. Still, it has the disadvantage of large file size. Therefore it is not practical to put a whole lecture session online.
As seen from these discussed online lecture formats, full-motion video lecture is the best available form of online lecture; still, its large file size has limitations. One reason behind the large file size is the redundancy of recording unnecessary frames. What is required is the lecturer input on the chalkboard with synchronizing sound. One solution for the presenter is to use a PC tablet instead of chalkboard and pen for writing. Tablet-pc is used to capture the lecturer input and sound and then the data is converted into a movie on the server.

5.2 Wireless Chalkboard

Tablet PC is a portable PC that uses multi-modal input [37]. We can input with keyboard, pen or voice. It has an electromagnetic pen, touch screen and software that recognizes letters written on a screen. Figure 5.1 shows a tablet pc. Because of its multi-modal input, wireless feature and its compatibility with Windows XP, tablet pc would be ideal to implement the online lecture generation system. It acts as a wireless chalkboard that takes the user inputs and transfers the data wirelessly to the server.

Figure 5.1: Tablet PC [38].
Screen resolution for Tablet PC is 800 x 600 and 120 dpi (Dots Per Inch). Pocket PC has a resolution of 240 x 320 and 106 dpi. Pixel count for Tablet PC is 6.25 times more than Pocket PC. The more dots, the better is the resolution. Writing a single character on Tablet PC uses more dpi than Pocket PC, which makes Tablet PC display more crisp and clear.

The lecture creation program consists of freehand writing of text and drawings on a digitizing board and speaking into a sound microphone. The freehand writing of text and drawings is done with a mouse or a pen-based input. This process is shown in Figure 5.2. To permit the recording of both the graphics and sound input, an ActiveX control client program was developed. This program runs on the local user computer and will be called up by the browser. This pen-based input of text and drawings simulates the traditional chalkboard. The text and drawings details are drawn and stored in a vector-based format to keep the file size small. In most cases vector-based drawings are 10% the file size as pixel-based graphics. This permits the use of 56 kbps modem connections. The speakers voice is synchronized with the pen-based input and both are embedded into a single movie after the completion of the lecture. Therefore, as the presenter is giving a normal lecture, the process is being recorded and a movie is generated. However it should be noted that the lecture could not be viewed on the web at the same time while it is being created.
Figure 5.2: Lecture Generation Process.

5.3 Online Lecture Generation System

Figure 5.3 shows the block diagram of the online lecture generation process. The system can be divided into two parts, the client side and the server side. Client side is the user PC where the lecture recording takes place. And server side is the web server where the movie generation takes place from the data collected on the client side.
The client side consists of two major components. One is the registered ActiveX control drawing tool and the other is the wave to mp3 conversion Component Object Model component. The Component Object Model (COM) is a software architecture that allows applications to be built from binary software components.
5.3.1.1 ActiveX Drawing Tool

ActiveX control is the user interface that implements a virtual chalkboard. The board consists of a series of drawing tools, which can be used to write or draw vector-based drawings on the screen using the pen-based input or a mouse. The tools allow users to draw rectangles, lines, arrows, or circles. Using the freehand tool, one can perform freehand drawing. Users can use it to write on the screen just like a chalkboard. As the drawings are vector-based images the x and y pixel data co-ordinates of the user input are stored. This pixel data is used in developing the movie on the server. The drawing tool also records sound during the lecture provided a microphone is connected to the client system and the speaker speaking into it. The sound recording can be turned on or off during the presentation.

Figure 5.4: ActiveX control drawing board.
With a graphic tablet attached to the computer, freehand drawing can simulate writing on a chalkboard. For sound recording purposes, a microphone attached to the computer can be used.

When the drawing board is accessed through the web, the ActiveX control is automatically installed on the client system in either of the two cases; if it has not been installed earlier or if the ActiveX control is a newer version than the one existing on the user’s computer. Accessing the drawing board with the installed/updated ActiveX control displays an interface as shown in Figure 5.4. First the lecture name (the title of the lecture) is entered in the text box with label Lecture Name. The lecture recorded will be saved with this name. Then a drawing tool is selected and sound recording is started if needed. The lecturer can then start delivering the lecture in normal fashion using the drawing board on the PC rather than the chalkboard. Figure 5.5 illustrates the freehand drawing of a block and forces acting on it. ASP Flash Turbine generates dynamic Flash movies from Active Server Page scripts. A Flash file template exists on the server that reads in the text file details having the draw commands written in ASP Flash Turbine format. The x and y co-ordinates of the drawings and writings on the board are stored in a text file according to the ASP Flash Turbine syntax specifications of each drawing type. For example if the drawing is a straight line, then the starting point co-ordinates and the end point co-ordinates are stored with the Line command. Similar procedures are followed for other drawings. The recorded sound is stored as a temporary wave file on the client system. When the presentation is finished the sound recording is stopped and the submit button is clicked.
Once the submit button is clicked, the server initiates the COM component on the client side to convert the wave file to mp3. After the conversion, an ASP file uploads the mp3 file along with text file to the server. On the server side, the ASP Flash Turbine integrates the sound file and text file into the flash template. The generator template file consists of ASP Flash Turbine commands to load the text file data and integrate the mp3 sound file. This template is called from an ASP file that passes on the file information to be considered for integration. The ASP file also generates the Flash file (.swf) from the generator template and places it on the server with the lecture name. The Flash file can be made available online the moment it has been generated. The whole process of generating flash format of the lecture takes only a few seconds.

A snapshot of the final lecture generated is shown in Figure 5.6. The Flash movie consists of play, pause, and rewind buttons. The rewind button takes the movie to the first
frame. A 15 seconds lecture developed using this system would have a flash file size is approximately 100 KB whereas a 15 seconds formal lecture recorded in a classroom would be approximately 5MB after compressing the movie to around 40% quality.

Figure 5.6: Final Lecture shown as a movie.
The present thesis deals with the development of a drawing tool for PDAs and laptops/desktops, which will be used by the students while taking online quiz. The developed drawing tool can be used for free hand drawing on PDAs or laptops/desktops. PDAs use pen-based input for drawing whereas laptops/desktops use a mouse. As the pen-based system is best suited for free hand drawing, use of PDAs is recommended. This drawing tool helps the students in elaborating the concepts they have understood in the form of a diagram. Currently, most online classes use a multiple-choice format for testing purposes. In engineering courses it helps to elaborate the problem in the form of a diagram before actually performing the calculations. Correct diagram also can be a source of partial credit. Implementation of this drawing tool can help students in explaining the problem with a free body diagram. During a test the student submits the multiple-choice answer along with the diagram. A solution recognition program is used to compare the student’s diagram with the actual solution. The diagram is recognized based on the given solution for that particular question. The comparison results are used for grading purposes. Grading of the whole test, which includes the multiple-choice questions and free body diagram, is done dynamically the moment the student submits the solution.

The advantage of using this drawing tool is that the learner can elaborate more in answering the questions. With the help of drawings and equations the learners can express their answers more clearly. So in the end if a student gets a wrong answer due to a calculation mistake or by usage of the wrong sign, the learner would not be getting zero marks but eventually be graded partially according to the procedure followed. For
example, say a student gets the value of force, $F = 250\text{N}$, and suppose that the answer should be $F = -250\text{N}$ due to the opposite direction of force. In multiple-choice questions, if the students chooses $F = 250\text{N}$ then the mark given will be zero. But with the drawing details, answers would be graded according to the procedure followed and not just on the final answer. Thus, the student solution is partially graded in doing so.

The drawing tool is basically a drawing board that enables free hand drawing to be done on a PDA’s or laptop’s/desktop’s screen. PDAs will use pen-based input for drawing whereas laptops/desktops will use a mouse. The drawing board is actually an ActiveX control, which can be installed on the user’s system through the Web. Once installed, the course pages will call the ActiveX control wherever it is necessary. When the quiz questions are displayed, the user has the option of drawing a free body diagram. To draw, the user needs to open the drawing board provided on the quiz page and start drawing. The pen-based input will be converted into vector format, which is stored in a text file. The text file details, along with multiple-choice answers, will be sent to the server where it will be stored in the flash file format. The multiple-choice answer, along with the diagram information is stored in a database. A solution recognition program is used to compare the student’s diagram with the actual solution. The solution recognition program is a COM component that takes the input data text file and compares it to the actual solution existing on the server. The diagram is recognized based on the given solution for that particular question. The comparison results are used for grading purposes.
6.1 Quiz Structure

The quiz format with ActiveX drawing board embedded is shown in Figure 6.1 and Figure 6.2 for both desktop and Pocket PC respectively. It can be divided into three columns. The first column is comprised of a figure depicting the question. The second column is the question with multiple-choice answers. And the third column has a rectangle-shaped box in which the free body diagram shows up in the form of a Flash movie. And the button with caption “Draw FBD” pops up the drawing board upon clicking on it.

Figure 6.1: General Quiz format – desktop/laptop.
First, the remote user has to login so the system can track the person taking the quiz and store their results in a database. Then the relevant quiz is opened and solved for respective solutions. After solving, the correct answer is chosen from the given multiple choices and if necessary a free body diagram is drawn. To draw the free body diagram for a particular question the “DrawForces” button for that particular question is clicked. Upon clicking a new window pops up as shown in Figure 6.3, which contains the drawing tools required to express the problem in free body diagram format.
Figure 6.3: Drawing board pops upon the main window.

The drawing board contains all the basic tools for drawing a free body diagram. It has a force component in the shape of an arrow, which helps the user in actually specifying angle and value of the force there by creating the force at the given angle proportional to the magnitude mentioned. Figure 6.4 and Figure 6.5 explain the above-mentioned features.
Figure 6.4: Giving force magnitude and angle.

Figure 6.5: Force magnitude and angle.
After the free body diagram has been drawn, the user clicks on the “submit” button. The vector-based drawing is then stored and the data is transferred to the server. The data is decrypted and passed as draw commands to the Flash template and a Flash movie is generated utilizing ASP Flash turbine. The Flash movie is the same as what the user has drawn in a vector based format on the ActiveX control board. This generated Flash movie is stored under the user’s folder for that particular quiz. The force values for a particular question are compared with the answers and graded accordingly. Also the chosen multiple-choice answer is also considered while grading and giving marks to the whole question. The chosen multiple-choice answer and final grade is stored in the database while the Flash movie and drawing details are stored under the user folder under specific quiz. The drawing details are converted to so the two-dimensional vector data can be stored in a single file. The Flash file is small and is easy to read, offering a good solution for publishing and storing user information.

Engineering Media Lab (EML) has been successfully developing and using two online courses, Statics-ENGR 2113 and Dynamics–AME 2523 for the past four years. These two online courses have a knowledge repository that contains course-material, lecture movies, homework problems and test problems. The knowledge repository is kept intact by a web-based database. This whole system can be used in implementing the proposed Desktop/PDA based drawing tool.
CHAPTER 7  DESIGN TOOLS

7.1 Windows CE and Pocket PC

All the PDA applications developed for this thesis are developed and tested for Pocket PC platform. Pocket PC include a set of system and application components from the Windows CE OS. All the applications developed for PDAs can be accessed wirelessly through web. The Pocket Internet Explorer, a Windows CE OS application component, makes this possible.

Microsoft Windows CE is an open, scalable, 32-bit operating system (OS). Windows CE offers the application developer the familiar environment of the Microsoft Win32 application programming interface (API), ActiveX controls, message queuing (MSMQ) and Component Object Model (COM) interfaces, Active Template Library (ATL), and the Microsoft Foundation Classes (MFC) Library. ActiveSync provides easy connectivity between the desktop and the embedded device, whether by serial connection, infrared port, or network cable. There is support for multimedia, communications (TCP/IP, SNMP, TAPI, and more), and security.

7.2 Embedded Visual C++

The drawing tool developed for Pocket PC, which is an ActiveX control, uses the eMbedded Visual C++ 3.0 environment. The ActiveX control is registered on the Pocket PC device and called through Pocket Internet explorer. Currently ActiveX controls for Pocket PC device cannot be developed using eMbedded Visual Basic environment.
Microsoft Visual C++ provides an integrated development environment that allows the programmer to write, build, and debug 32 bit C/C++ programs. Microsoft eMbedded Visual C++ 3.0 enables the development of Windows CE-based applications using an integrated development environment (IDE) similar to that used in developing desktop Visual C++ applications. It also includes a variety of tools that help us to develop new software uniquely appropriate for Windows CE platforms and devices.

7.3 Microsoft Windows Platform SDK for Pocket PC

With Pocket PC SDK we can develop applications for Pocket PC mobile devices using either Microsoft eMbedded Visual C++ 3.0 or Microsoft eMbedded Visual Basic 3.0. The SDK comes with a Pocket PC x86 Emulator. Pocket PC x86 Emulator simulates the Pocket PC device environment on our local workstation. Using the Pocket PC x86 Emulator, applications can be debugged on the workstation where the development work goes on. This is much faster compared to debugging over serial or USB connections. The PDA based e-book and drawing tool are widely tested using the Pocket PC emulator.

The Pocket PC SDK, in conjunction with the Microsoft eMbedded Visual Tools 3.0, delivers a complete development environment for creating applications for Pocket PC devices based on Microsoft Windows CE.

7.4 Microsoft Visual Basic

The drawing tool for the e-courses, which is an ActiveX control for desktops/laptops, is developed using Visual Basic 6.0. Visual Basic language has been chosen because it is the fastest way to build and test an application’s feasibility.

Visual provides the user with a complete set of tools to simplify rapid application development. The graphical user interface creation is simplified with a set of prebuilt
objects, which can be added directly instead of writing numerous lines of code in
describing the appearance and location of the interface elements. The Visual Basic
Scripting Edition (VBScript), a scripting language, is a subset of the Visual Basic
language.

7.5 ActiveX Control

This technology is used in developing a drawing tool for both laptops/desktops
and Pocket PC. ActiveX controls for Pocket PC cannot be developed using eMbedded
Visual Basic. As Visual Basic language is the fastest way to build and test an application,
two different languages have to be used in developing the drawing tool application. For
laptops/desktops Visual Basic language has been used and for Pocket PC eMbedded
Visual C++ has been used.

ActiveX controls can run in a wide variety of containers—Visual Basic, Visual
C++, Microsoft Access, and, Microsoft Internet Explorer 3.0. They can be very useful for
creating powerful web sites for serving Windows based clients.

7.6 COM

Grading of the e-courses quiz, which includes the multiple-choice questions and
free body diagram is done by the COM component that is residing on the server. Also the
COM component installed on the client side does the conversion of the sound file from
wave to mp3 format.

The Component Object Model (COM) is a way for software components to
communicate with each other. It is a binary and network standard that allows any two
components to communicate regardless of what machine they are running on (as long as
the machines are connected), what operating systems the machines are running (as long
as it supports COM), and what language the components are written in. The Component Object Model (COM) is a software architecture that allows applications to be built from binary software components.

**7.7 Active Server Pages**

The free body diagram drawing tool is kept intact by the ASP pages and the user-entered information is passed on to the server using the ASP technology. Also the ASP page initiates the COM component that grades the user’s answers. The drawing board for creating online lectures is displayed using the ASP page and also initiates the COM component on the client side that converts the wave sound file to mp3 format.

Active Server Pages (ASP) provides a framework for using existing scripting languages, such as JavaScript, VBScript, or PerlScript and reusable ActiveX server components, and integrating them into the HTML to create dynamic and powerful Web-based business solutions. Active Server Pages enables server-side scripting for IIS with native support for both VBScript and Jscript. The server compiles the ASP code on the fly and the resulting output is standard HTML. By using ASP, Web pages can be dynamic, full of ever-changing content, and browser independent.

**7.8 ASP Flash Turbine**

The online lecture recordings that contain the writings on drawing board and sound recorded are combined together and made it into a movie by ASP Flash Turbine. Also the free body diagram drawn by the user while taking online quiz is converted into Flash movie by ASP Flash turbine.

ASP Flash Turbine generates dynamic Flash movies from Active Server Page scripts. With ASP Flash Turbine, Flash movies can be generated with content obtained
from a variety of sources like text files, image files, MP3, ODBC compliant databases, etc. Tables, Lists and Charts can also be generated from the accessed data. Its Draw Script feature enables the user to draw directly into Flash movies.

ASP Flash Turbine works with Internet Information Server (and PWS) 4.0 and 5.0 in creating dynamic Flash movies from ASP scripts. We can use all the scripting languages supported by ASP, like VBScript, JavaScript and PerlScript to access any data source and generate dynamic content for web site application. ASP Flash Turbine is implemented as a COM DLL, offering the best possible performance on IIS web servers. Turbine allows us to separate presentation from content – as we will no longer need to update Flash movies by hand to reflect changes on information.

7.9 Microsoft Access

Access database is used to store the student login information, quiz and grading details. Microsoft Access is a relational database management system (DBMS, a program that facilitates the storage and retrieval of structured information on a computer’s hard drive). Microsoft Access provides users with one of the simplest and most powerful database solution for small-scale projects.

7.10 Macromedia Flash

Flash is used in developing animated movies for PDA based e-book. The training module developed for Pocket PC is developed completely in Flash. Also the free body diagram drawn by the user while taking online quiz is converted into Flash.

Flash is used for creating graphics and animation, which can be played on web sites. The movies are primarily vector graphics, but they can also contain imported bitmap graphics and sounds. Flash movies can incorporate interactivity to permit input
from viewers, and can create nonlinear movies that can interact with other Web applications. Web designers use Flash to create navigation controls, animated logos, long-form animations with synchronized sound, and even complete, sensory-rich Web sites. Flash movies are compact, vector graphics, so they download rapidly and scale to the viewer's screen size. The Flash plug-in resides on the local computer, where it plays back movies in browsers or as stand-alone applications. The Flash plug-in is the device used to display the movies created in the Flash authoring application.

7.11 Macromedia Dreamweaver

Macromedia Dreamweaver is a HTML editor for designing and managing Web sites and pages. Dreamweaver includes many coding tools and features: an HTML, CSS, JavaScript reference, a JavaScript Debugger, and code editors that allow the user to edit JavaScript, XML, and other text documents directly in Dreamweaver.
CHAPTER 8 CONCLUSIONS

PDAs (personal digital assistants) represent a new phase in educational technology. They are powerful, portable computing devices, which provide great educational opportunities. The latest advance in PDAs is the Pocket PC. Its wireless capability, multimedia support, and typical windows environment for creating third party applications makes it the most powerful device available. This thesis involves designing and developing Pocket PC applications for education and training purposes.

PDA friendly web sites were created for education and training purposes. An e-book covering topics in statics, and some topics in electrical engineering have been developed. The web portal includes e-book, examples, and sample problems. The portal provides access to reference material useful in a classroom. And away from the classroom too at any remote site the learner can access the material and train himself/herself. Within a classroom the learners can mutually exchange information between their respective PDAs and participate in a collaborative study. PDA’s small size and less weight make them easy to carry and travel with to any place.

One module of the Tinker AFB Online Training System currently hosted by the University of Oklahoma (http://tinker.ou.edu) was developed for Pocket PC. With the help of PDAs the employer can be trained on the field site while actually observing the practical performance. This type of onsite training is not possible for all kind of field sites by using laptop. And also laptops are not convenient compared to that of PDAs.
An ActiveX drawing tool was developed, which will be called-up inside the Internet Explorer. The drawing tool is used to draw free body diagrams while answering statics quiz questions. These diagrams and multiple-choice answers to the quiz are graded on the server. This feature has been implemented for both desktops/laptops and Pocket PC. Currently, online quiz uses only multiple-question format to test learner’s knowledge. This drawing tool helps in giving partial credit to the free body diagram.

Apart from this, an online lecture generation tool was developed. This tool can be used to record/create lectures and can be made available online as Flash movies. This tool is accessed through Internet Explorer and has the ability to record/create lectures from a remote site and stores them on the server. This serves as an important feature of online learning. Classroom lectures are recorded and made available online dynamically.

8.1 Accomplishments

- Pocket PC web portal for statics and electrical engineering sections of Fundamentals of engineering exam was developed.
- One module of Tinker AFB Online Training System was developed for Pocket PC.
- Drawing tool was developed for Pocket PC and laptop/desktop and was implemented for the multiple-question quiz format.
- Online lecture generation was developed.

8.2 Recommendations for Future Research

More content on the web will have to be considered for use on a PDA, and it is important that whoever converts old content and designs new content has a grasp on how to deliver the content in a concise manner and keep in mind the wireless speed.
availability. The e-courses web site hosted by EML lab (http://emlou.edu) can be made portable with PDAs. Also, all the modules of the Tinker AFB Online Training System can be redesigned and developed for Pocket PC. The online lecture generation tool will be most effective if implemented for Pocket PC, as it has pen-based input. Further research can be done to implement the whole system on Pocket PC.

Microsoft Visual Studio .NET and the .NET framework provides full set of development tools to develop applications and XML web services. An important feature of .NET is the common language run time that gives the freedom of choosing the programming language most suited. Web applications developed for this thesis requires developing ActiveX controls, COM components and integrating them inside scripting languages. As .NET comes with a single integrated development environment all the required applications can be developed and tested using a single interface. The Web Services feature of .NET makes component methods available over HTTP (Hypertext Transfer Protocol). Client applications that reside on Pocket PC can be developed using .NET and Web Services feature can be used for data communication.
REFERENCES


       bin/tais/pc/pc_prodList.jsp?comm=CS&plin=Toshiba%20Pocket%20PC


       e-learning Magazine, p 18.


       Retrieved July 10, 2002, from
       http://www.informationweek.com/story/IWK20010726S0024


       Graders “doin’ science” – Collecting Data and looking for patterns in a Virtual Field.

       Visualization Tool for the Pocket PC. 2nd IEEE International Symposium on
       Bioinformatics and bioengineering (BIBE’01), Bethesda, Maryland.


APPENDICES

Appendix A  ASP Script for generating dynamic Flash movies

Appendix B  ActiveX control Visual Basic Code

Appendix C  ActiveX control eMbedded Visual C++ Code
Appendix A  ASP Script for generating dynamic Flash movies

<%@ Language=VBScript %>
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
</HEAD>
<BODY>

<%'
create the Turbine object:
Set Turbine=Server.CreateObject("Turbine.ASP")
' Give .swt template file to use:
Turbine.Template = "lecture.swt"
Turbine.Variable("lecName") = "lecture.txt" 'Textfile to be used inside flash template
Turbine.Variable("lecSound") = "lecture.mp3" 'MP3 file that need to be embedded

' now generate the movie to the web browser
Turbine.GenerateToFile "result.swf"
Turbine.GenerateFlash
Response.Redirect "result.htm" 'Display the generated Flash Movie
%>

</BODY>
</HTML>
Option Explicit

Private Sub cmdArrow_Click()
    figName = "arrow" 'arrow selected
End Sub

Private Sub cmdCircle_Click()
    figName = "circle" 'circle selected
End Sub

Private Sub cmdFreeHand_Click()
    figName = "freehand" 'freehand mode selected
End Sub

Private Sub cmdLine_Click()
    figName = "line" 'line tool selected
End Sub

Private Sub cmdRect_Click()
    figName = "rectangle" 'rectangle selected
End Sub

Private Sub cmdSubmit_Click() 'force values are ready to be submitted
    Dim h, fs, a, subChoice
    subChoice = figName
    fileAns.Close
End Sub

Private Sub cmdClear_Click() 'clear the drawing board
    UserControl.Cls
    Erase getXCord
    Erase getYCord
    Dim fileAnsDel, fileAnsTempDel, fileTempDel, fso
    Set fso = CreateObject("Scripting.FileSystemObject")
    fileAns.Close
    fileAnsTemp.Close
    Set fileAnsDel = fso.GetFile("C:\\stdAnswer.txt")
    Set fileAnsTempDel = fso.GetFile("C:\\tempFile.txt")
    Set fileTempDel = fso.GetFile("C:\\cpTemp.txt")
    fileAnsDel.Delete
    fileAnsTempDel.Delete
    fileTempDel.Delete
    Set fileAns = fileObj.CreateTextFile("C:\\stdAnswer.txt", True)
    Set fileAnsTemp = fileObjTemp.CreateTextFile("C:\\tempFile.txt", True)
End Sub

Private Sub cmdOK_Click() 'Get the force Values and store them inside the text file as answers
    fileAns.writeline txtForceName & " " & txtMag & " " & txtAngle
    Line (getXCord(0), getYCord(0))-(getXCord(xCord), getYCord(yCord)), RGB(255, 255, 255) 'draw the given force
    Dim tempX, tempY
    Dim fMag, trXCord, trY cord, length
    Dim pi As Double
    pi = 4 * Atn(1)
    fAngle = Clnt(txtAngle)
    fMag = Clnt(txtMag) * 10
    Call storeData("arrow")
End If
txtForceName.Text = ""
txtMag.Text = ""
txtAngle = ""
figName = ""
lblForceName.Visible = False
lblMag.Visible = False
lblAngle.Visible = False
txtForceName.Visible = False
txtMag.Visible = False
txtAngle.Visible = False
cmdOK.Visible = False
Call refresh
If fMod = True Then
    Line (arrTemp(1), arrTemp(2))-(arrTemp(3), arrTemp(4)), RGB(255, 255, 255)
End If
End Sub
Private Sub UserControl_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
mouDown = 1
xCordStart = X
yCordStart = Y
xCord = 0
yCord = 0
getXCord(xCord) = xCordStart
getYCord(yCord) = yCordStart
End Sub
Private Sub UserControl_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
Dim choice As String
Dim radius As Integer
choice = figName
If mouDown = 1 And figName <> "rotate" Then
    xCord = xCord + 1
    yCord = yCord + 1
    getXCord(xCord) = xCordStart
    getYCord(yCord) = yCordStart
Select Case choice
    Case "circle"
        If xCord = 1 Then
            Call findRadius(xCordStart, yCordStart, getXCord(xCord), getYCord(yCord))
            radius = radVal
            Circle (xCordStart, yCordStart), radius, RGB(0, 0, 0)
        Else
            Call findRadius(xCordStart, yCordStart, getXCord(xCord - 1), getYCord(yCord - 1))
            radius = radVal
            Circle (xCordStart, yCordStart), radius, RGB(255, 255, 255)
        End If
End Case
    Case "rectangle"
        If xCord = 1 Then
End Case
End Sub
Private Sub UserControl_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If figName = "arrow" Then
        Call drawArrow
    End If

    If figName = "line" Then
        Call drawLine
    End If

    If figName <> "arrow" Then
        Call storeData(figName)
    End If

    mouDown = 1
End Sub

Function findRadius(xCordStart, yCordStart, xCordNext, yCordNext)
End Function

Function findAngle(xCordStart, yCordStart, xCordLast, yCordLast)
Dim slope
Dim pi
If xCordLast = xCordStart Then
    theta = 90
Else
    pi = 4 * Atn(1)
    slope = (yCordLast - yCordStart) / (xCordLast - xCordStart)
    theta = Atn(slope)
End If
End Function

Private Sub UserControl_Show()
Dim fileDrawDetails, fileDrawDetailsObj
If Dir("C:\tempFile.txt") = "" Then
    lblCount = 1
    rectTheta = 0
    fMod = False
    Set fileObj = CreateObject("Scripting.FileSystemObject")
    Set fileAns = fileObj.CreateTextFile("C:\stdAnswer.txt", True)
    Set fileObjTemp = CreateObject("Scripting.FileSystemObject")
    Set fileAnsTemp = fileObjTemp.CreateTextFile("C:\tempFile.txt", True)
    Set cpReqObj = CreateObject("Scripting.FileSystemObject")
    Set cpReq = cpReqObj.CreateTextFile("C:\cpTemp.txt", True)
    'Dim fileDrawDetails, fileDrawDetailsObj
    Set fileDrawDetailsObj = CreateObject("Scripting.FileSystemObject")
    Set fileDrawDetails = fileDrawDetailsObj.CreateTextFile("C:\stdFBD.txt", True)
Else
    Set fileObj = CreateObject("Scripting.FileSystemObject")
    Set fileAns = fileObj.CreateTextFile("C:\stdAnswer.txt", True)
    Set fileObjTemp = CreateObject("Scripting.FileSystemObject")
    Set fileAnsTemp = fileObjTemp.CreateTextFile("C:\tempFile.txt", True)
    Dim fileTempData
    Open "C:\cpTemp.txt" For Input As #1
    While Not EOF(1)
        Line Input #1, fileTempData
        fileAnsTemp.writeline fileTempData
    Wend
    Close #1
End If
End Sub

Function drawArrow()
Dim iX1, iX2, iX3, iX4, iX5, iY1, iY2, iY3, iY4, iY5 As Integer
Dim fTheta, fAlpha, fLength, fLenHyp As Double
iX1 = getXCord(0)
iY1 = getYCord(0)
iX2 = getXCord(xCord)
iY2 = getYCord(yCord)
fLength = Sqr((iX2 - iX1) * (iX2 - iX1) + (iY2 - iY1) * (iY2 - iY1))
fTheta = Atn((iY2 - iY1) / (iX2 - iX1))
fAlpha = Atn(3 / fLength)
fLenHyp = Sqr((fLength * fLength) + 9)
iX3 = fLenHyp * Cos(fAlpha + fTheta) + iX1
iY3 = fLenHyp * Sin(fAlpha + fTheta) + iY1
iX4 = 2 * iX2 - iX3
iY4 = 2 * iY2 - iY3
iX5 = (iX2 * (fLength + 4) - (4 * iX1)) / fLength
iY5 = (iY2 * (fLength + 4) - (4 * iY1)) / fLength

Line (iX2, iY2)-(iX3, iY3)
Line (iX2, iY2)-(iX4, iY4)
Line (iX5, iY5)-(iX3, iY3)
Line (iX5, iY5)-(iX4, iY4)

lblForceName.Visible = True
lblMag.Visible = True
lblAngle.Visible = True
txtForceName.Visible = True
txtMag.Visible = True
txtAngle.Visible = True
cmdOK.Visible = True
End Function
Function drawLine()
Call findAngle(xCordStart, yCordStart, getXCord(xCord), getYCord(yCord))
txtInfo.Text = theta
End Function
Function storeData(figDetails)
Open "C:\cpTemp.txt" For Append As #1
Open "C:\stdFBD.txt" For Append As #2
Select Case figDetails
Case "circle"
  fileAnsTemp.writeline "circle"
  fileAnsTemp.writeline radVal
  fileAnsTemp.writeline getXCord(0)
  fileAnsTemp.writeline getYCord(0)
  Print #1, "circle"
  Print #1, radVal
  Print #1, getXCord(0)
  Print #1, getYCord(0)
  Print #2, "LineStyle(4, #FF0000)"
  Print #2, "Circle(" & getXCord(0) / 25 & "," & getYCord(0) / 25 & ",", & radVal / 25 & ",", & "1,1)"
End Function
Case "rectangle"
    fileAnsTemp.writeline "rectangle"
    fileAnsTemp.writeline getXCord(0)
    fileAnsTemp.writeline getYCord(0)
    fileAnsTemp.writeline getXCord(xCord)
    fileAnsTemp.writeline getYCord(xCord)
    fileAnsTemp.writeline rectTheta

    Print #1, "rectangle"
    Print #1, getXCord(0)
    Print #1, getYCord(0)
    Print #1, getXCord(xCord)
    Print #1, getYCord(xCord)
    Print #1, rectTheta

    Print #2, "LineStyle(4, #FF0000)"
    Print #2, "Rectangle(" & getXCord(0) / 25 & "," & getXCord(0) / 25 & "," & getXCord(xCord) / 25 & "," & getXCord(xCord) / 25 & ")"

Case "arrow"
    fileAnsTemp.writeline "arrow"
    fileAnsTemp.writeline getXCord(0)
    fileAnsTemp.writeline getYCord(0)
    fileAnsTemp.writeline getXCord(xCord)
    fileAnsTemp.writeline getYCord(yCord)
    fileAnsTemp.writeline txtForceName.Text
    fileAnsTemp.writeline arrTheta+

    Print #1, "arrow"
    Print #1, getXCord(0)
    Print #1, getYCord(0)
    Print #1, getXCord(xCord)
    Print #1, getYCord(yCord)
    Print #1, txtForceName.Text

    Print #2, "LineStyle(4, #FF0000)"
    Print #2, "Line(" & getXCord(0) / 25 & "," & getXCord(0) / 25 & "," & getXCord(xCord) / 25 & "," & getXCord(xCord) / 25 & ")"

Case "line"
    fileAnsTemp.writeline "line"
    fileAnsTemp.writeline getXCord(0)
    fileAnsTemp.writeline getYCord(0)
    fileAnsTemp.writeline getXCord(xCord)
    fileAnsTemp.writeline getYCord(xCord)
    fileAnsTemp.writeline lineTheta

    Print #1, "line"
    Print #1, getXCord(0)
    Print #1, getYCord(0)
    Print #1, getXCord(xCord)
    Print #1, getYCord(xCord)
    Print #1, lineTheta

    Print #2, "LineStyle(4, #FF0000)"
Print #2, "Line(\& getXCord(0) / 25 & ", & getYCord(0) / 25 & ", & getXCord(xCord) / 25 & ", & getYCord(xCord) / 25 & ")"

Case "freehand"
  Print #2, "Line(\& getXCord(0) & ", & getYCord(0) & ", & getXCord(xCord) & ", & getYCord(xCord) & ")"
  fileAnsTemp.writeline "No figure"
End Select
Close #1
Close #2
End Function

Function refresh()
  Dim file_data
  Dim arrFileData(10)
  Open "C:\tempFile.txt" For Input As #1
  While Not EOF(1)
    Line Input #1, file_data
    Select Case file_data
      Case "rectangle"
        Line Input #1, arrFileData(1)
        Line Input #1, arrFileData(2)
        Line Input #1, arrFileData(3)
        Line Input #1, arrFileData(4)
        Line Input #1, arrFileData(5)
        Line (arrFileData(1), arrFileData(2))-(arrFileData(3), arrFileData(2)), RGB(0, 0, 0)
        Line (arrFileData(1), arrFileData(2))-(arrFileData(1), arrFileData(4)), RGB(0, 0, 0)
        Line (arrFileData(1), arrFileData(4))-(arrFileData(3), arrFileData(4)), RGB(0, 0, 0)
        Line (arrFileData(3), arrFileData(2))-(arrFileData(3), arrFileData(4)), RGB(0, 0, 0)
      Case "arrow"
        Line Input #1, arrFileData(1)
        Line Input #1, arrFileData(2)
        Line Input #1, arrFileData(3)
        Line Input #1, arrFileData(4)
        Line Input #1, arrFileData(5)
        Line (arrFileData(1), arrFileData(2))-(arrFileData(3), arrFileData(4)), RGB(0, 0, 0)
      Case "circle"
        Line Input #1, arrFileData(1)
        Line Input #1, arrFileData(2)
        Line Input #1, arrFileData(3)
        Circle (arrFileData(2), arrFileData(3)), arrFileData(1), RGB(0, 0, 0)
      Case "line"
        Line Input #1, arrFileData(1)
        Line Input #1, arrFileData(2)
        Line Input #1, arrFileData(3)
        Line Input #1, arrFileData(4)
        Line Input #1, arrFileData(5)
        Line (arrFileData(1), arrFileData(2))-(arrFileData(3), arrFileData(4)), RGB(0, 0, 0)
    End Select
  Wend
Close #1
End Function

Select Case currentLabel
Case lblForce1
lblForce1.Visible = True
Case lblForce2
    lblForce2.Visible = True
Case lblForce3
    lblForce3.Visible = True
Case lblForce4
    lblForce4.Visible = True
Case lblForce5
    lblForce5.Visible = True
Case lblForce6
    lblForce6.Visible = True
End Select

End Function

' Module declarations
Option Explicit
Public xCordStart, yCordStart
Public xCordEnd, yCordEnd
Public xCord, yCord, xCordRot, yCordRot
Public getXCord(1000) As Integer
Public getYCord(1000) As Integer
Public getXCordRot(1000) As Integer
Public getYCordRot(1000) As Integer
Public getXCord1() Public getXCord1()
Public radVal As Double
Public figName As String
Public mouDown As Integer
Public theta As Double
Public rectTheta As Double
Public arrTheta As Double
Public lineTheta As Double
Public toSelect As Boolean
Public fileAns, fileObj, fileObjTemp, fileAnsTemp, cpReqObj, cpReq
Public b, h, d
Public lblCount
Public fMod As Boolean
Public currentLabel
Public arrTemp(10)
 Appendix C  ActiveX control eMbedded Visual C++ Code

'ActiveX control Pocket PC drawing tool code.
// UsrInt.cpp : implementation file

#include "stdafx.h"
#include "resource.h"
#include "UsrInt.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// CUsrInt dialog

CUsrInt::CUsrInt(CWnd* pParent /*=NULL*/)
: CDialog(CUsrInt::IDD, pParent)
{
    MakeAllFlagsFalse();
    FigCounter=0;
    FigFHandCounter=0;
}

BOOL CUsrInt::OnInitDialog()
{
    CDialog::OnInitDialog();
}

void CUsrInt::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //AFX_DATA_MAP
    DDX_Control(pDX, IDC_LINE, m_Line);
    DDX_Control(pDX, IDC_CIRCLE, m_Circle);
    //AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CUsrInt, CDialog)
    //AFX_MSG_MAP
    ON_BN_CLICKED(IDC_LINE, OnLine)
    ON_BN_CLICKED(IDC_RECT, OnRect)
    ON_BN_CLICKED(IDC_CIRCLE, OnCircle)
    ON_BN_CLICKED(IDC_FHAND, OnFhand)
    ON_BN_CLICKED(IDC_SELECT, OnSelect)
    ON_BN_CLICKED(IDC_SUBMIT, OnSubmit)
    ON_BN_CLICKED(IDC_CLEAR, OnClear)
    ON_WM_LBUTTONDOWN()
    ON_WM_MOUSEMOVE()
END_MESSAGE_MAP
ON_WM_LBUTTONDOWN()
ON_BN_CLICKED(IDC_FORCE, OnForce)
ON_BN_CLICKED(IDC_ROTATE, OnRotate)
ON_BN_CLICKED(IDC_LINE1, OnLine1)
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CUsrInt message handlers

void CUsrInt::OnLine()
{
    // Line selected
    MakeAllFlagsFalse();
    bDrawLine=true;
}

void CUsrInt::OnRect()
{
    // Rectangle selected
    MakeAllFlagsFalse();
    bDrawRect=true;
}

void CUsrInt::OnCircle()
{
    // Circle selected
    MakeAllFlagsFalse();
    bDrawCircle=true;
}

void CUsrInt::OnFhand()
{
    // Freehand mode
    MakeAllFlagsFalse();
    bDrawFreeHand=true;
}

void CUsrInt::OnSelect()
{
    // Selecting drawn figure
    MakeAllFlagsFalse();
    bDrawSelect=true;
}

void CUsrInt::MakeAllFlagsFalse()
{
    bDrawLine=false;
    bDrawArrow=false;
    bDrawRect=false;
    bDrawCircle=false;
    bDrawFreeHand=false;
    bDrawSelect=false;
    bDrawCompress=false;
bDrawRotate=false;
}

void CUsrInt::OnLButtonDown(UINT nFlags, CPoint point)
{
    // when pen is on the display
    CordIndex=0;
    XCordData[CordIndex]=point.x;
    YCordData[CordIndex]=point.y;

    if((nFlags && MK_LBUTTON) && bDrawSelect){
        aSelectedCord[0]=point.x;
        aSelectedCord[1]=point.y;
        SelectFigure();
        ChkForCompression(XCordData[CordIndex],YCordData[CordIndex]);
    }

    if((nFlags && MK_LBUTTON) && bDrawRotate){
        aSelectedCord[0]=point.x;
        aSelectedCord[1]=point.y;
        ChkForRotation(XCordData[CordIndex],YCordData[CordIndex]);
    }

    CDialog::OnLButtonDown(nFlags, point);
}

void CUsrInt::OnMouseMove(UINT nFlags, CPoint point)
{
    // as pen moves draw the respective figure
    CordIndex=CordIndex+1;
    XCordData[CordIndex]=point.x;
    YCordData[CordIndex]=point.y;

    CClientDC* pDC=new CClientDC(this);
    pDC->SelectStockObject(NULL_BRUSH);

    if((nFlags && MK_LBUTTON) && bDrawLine){
        pDC->SetROP2(R2_NOT);
        pDC->MoveTo(XCordData[0],YCordData[0]);
        pDC->LineTo(XCordData[CordIndex-1],YCordData[CordIndex-1]);

        pDC->SelectStockObject(BLACK_PEN);
        pDC->MoveTo(XCordData[0],YCordData[0]);
        pDC->LineTo(XCordData[CordIndex],YCordData[CordIndex]);
    }

    if((nFlags && MK_LBUTTON) && bDrawRect){
        pDC->SetROP2(R2_NOT);
        pDC->Rectangle(XCordData[0],YCordData[0],XCordData[CordIndex-1],YCordData[CordIndex-1]);

        pDC->SelectStockObject(BLACK_PEN);
    }
void CUsrInt::SelectFigure()
{
    int iSelFigName;
    int iMaxXCord, iMaxYCord, iMinXCord, iMinYCord;

    if((nFlags && MK_LBUTTON) && bDrawCircle){
        pDC->SetROP2(R2_NOT);
        pDC->Ellipse(XCordData[0], YCordData[0], XCordData[CordIndex-1], YCordData[CordIndex-1]);

        pDC->SelectStockObject(BLACK_PEN);
        pDC->Ellipse(XCordData[0], YCordData[0], XCordData[CordIndex], YCordData[CordIndex]);
    }

    if((nFlags && MK_LBUTTON) && bDrawFreeHand){
        pDC->SelectStockObject(BLACK_PEN);
        pDC->MoveTo(XCordData[CordIndex-1], YCordData[CordIndex-1]);
        pDC->LineTo(XCordData[CordIndex], YCordData[CordIndex]);

        CordFHandStore[FigFHandCounter][0]=4;
        CordFHandStore[FigFHandCounter][1]=XCordData[0];
        CordFHandStore[FigFHandCounter][2]=YCordData[0];
        CordFHandStore[FigFHandCounter][3]=XCordData[CordIndex];
        CordFHandStore[FigFHandCounter][4]=YCordData[CordIndex];
        FigFHandCounter=FigFHandCounter+1;
    }

    if((nFlags && MK_LBUTTON) && bDrawArrow){
        pDC->SetROP2(R2_NOT);
        pDC->MoveTo(XCordData[0], YCordData[0]);
        pDC->LineTo(XCordData[CordIndex-1], YCordData[CordIndex-1]);

        pDC->SelectStockObject(BLACK_PEN);
        pDC->MoveTo(XCordData[0], YCordData[0]);
        pDC->LineTo(XCordData[CordIndex], YCordData[CordIndex]);
    }

    if((nFlags && MK_LBUTTON) && bDrawSelect && bDrawCompress){
        OnCompressFig(iSelFigCounter, XCordData[CordIndex], YCordData[CordIndex]);
    }

    if((nFlags && MK_LBUTTON) && bDrawRotate && bDrawCompress){
        OnRotateFig(iSelFigCounter, XCordData[CordIndex], YCordData[CordIndex]);
    }

    OnRefreshForm();
    CDIalog::OnMouseMove(nFlags, point);
}

void CUsrInt::SelectFigure()
double fRadius,fPtRadius;
double dSlope1,dSlope2;
double dSlopeDiff;

CClientDC* pDC=new CClientDC(this);
pDC->SelectStockObject(NULL_BRUSH);

for(int figIndex=0;figIndex<=FigCounter;figIndex++){
  iSelFigName=CordStoreData[figIndex][0];
  switch(iSelFigName)
  {
    case 0: //Line
      if((CordStoreData[figIndex][1]-CordStoreData[figIndex][3]) &&
         (CordStoreData[figIndex][1]-aSelectedCord[0]))
        {
          dSlope1=(CordStoreData[figIndex][2]-
                   CordStoreData[figIndex][4])/(CordStoreData[figIndex][1]-
                   CordStoreData[figIndex][3]);
          dSlope2=(CordStoreData[figIndex][2]-
                   aSelectedCord[1])/(CordStoreData[figIndex][1]-
                   aSelectedCord[0]);
          dSlopeDiff=dSlope1-dSlope2;

          if(!dSlopeDiff)
          {
            pDC->SelectStockObject(BLACK_PEN);
            pDC->Rectangle(CordStoreData[figIndex][1]-
                            4,CordStoreData[figIndex][2]-4,
                            CordStoreData[figIndex][1]+4,CordStoreData[figIndex][2]+4);
            pDC->Rectangle(CordStoreData[figIndex][3]-
                            4,CordStoreData[figIndex][4]-4,
                            CordStoreData[figIndex][3]+4,CordStoreData[figIndex][4]+4);
            iSelFigCounter=figIndex;
            GetDlgItem(IDC_ROTATE)->EnableWindow(TRUE);
          }
          break;
        }
    case 1: //Rectangle
      if(CordStoreData[figIndex][3]>CordStoreData[figIndex][1])
        {
          iMaxXCord=CordStoreData[figIndex][3];
          iMinXCord=CordStoreData[figIndex][1];
        }
      if(CordStoreData[figIndex][1]>CordStoreData[figIndex][3])
        {
          iMaxXCord=CordStoreData[figIndex][1];
          iMinXCord=CordStoreData[figIndex][3];
        }
      if(CordStoreData[figIndex][2]>CordStoreData[figIndex][4])
        {
          iMaxYCord=CordStoreData[figIndex][2];
          iMinYCord=CordStoreData[figIndex][4];
        }
      if(CordStoreData[figIndex][4]>CordStoreData[figIndex][2])
        {
          iMaxYCord=CordStoreData[figIndex][4];
          iMinYCord=CordStoreData[figIndex][2];
        }
      break;
  }
}
if(aSelectedCord[0]>=iMinXCord && aSelectedCord[0]<=iMaxXCord && aSelectedCord[1]>=iMinYCord && aSelectedCord[1]<=iMaxYCord)
{
    pDC->SelectStockObject(BLACK_PEN);
    pDC->Rectangle(CordStoreData[figIndex][1]-4,CordStoreData[figIndex][2]-4,CordStoreData[figIndex][1]+4,CordStoreData[figIndex][2]+4);
    pDC->Rectangle(CordStoreData[figIndex][3]-4,CordStoreData[figIndex][4]-4,CordStoreData[figIndex][3]+4,CordStoreData[figIndex][4]+4);
    pDC->Rectangle((((CordStoreData[figIndex][1]+CordStoreData[figIndex][3])/2)-4),(((CordStoreData[figIndex][2]+CordStoreData[figIndex][4])/2)-4),(((CordStoreData[figIndex][1]+CordStoreData[figIndex][3])/2)+4),(((CordStoreData[figIndex][2]+CordStoreData[figIndex][4])/2)+4));
    pDC->Rectangle((((CordStoreData[figIndex][1]+CordStoreData[figIndex][3])/2)-4),((CordStoreData[figIndex][2]-4),(((CordStoreData[figIndex][1]+CordStoreData[figIndex][3])/2)+4),((CordStoreData[figIndex][2]+4));
    pDC->Rectangle(CordStoreData[figIndex][1]-4,(((CordStoreData[figIndex][2]+CordStoreData[figIndex][4])/2)-4),CordStoreData[figIndex][1]+4,(((CordStoreData[figIndex][2]+CordStoreData[figIndex][4])/2)+4));
    iSelFigCounter=figIndex;
    GetDlgItem(IDC_ROTATE)->EnableWindow(TRUE);
}
break;

case 2: //Ellipse

fRadius=abs((CordStoreData[figIndex][2]-

CordStoreData[figIndex][4])/2);

fPtRadius=sqrt(((CordStoreData[figIndex][1]+CordStoreData[figIndex][3])/2)-
aSelectedCord[0])*(CordStoreData[figIndex][1]+CordStoreData[figIndex][3])/2- 
aSelectedCord[0])+(CordStoreData[figIndex][1]+CordStoreData[figIndex][3])/2- 
aSelectedCord[1])*(CordStoreData[figIndex][2]+CordStoreData[figIndex][4])/2- 
aSelectedCord[1]));

if(fRadius>=fPtRadius)
{
    pDC->SelectStockObject(BLACK_PEN);
pDC->Rectangle(((CordStoreData[figIndex][1]+CordStoreData[figIndex][3])/2)-4),((CordStoreData[figIndex][2]-4),(((CordStoreData[figIndex][1]+CordStoreData[figIndex][3])/2)+4),(CordStoreData[figIndex][2]+4));
pDC->Rectangle(((CordStoreData[figIndex][1]+CordStoreData[figIndex][3])/2)-4),((CordStoreData[figIndex][4]-4),(((CordStoreData[figIndex][1]+CordStoreData[figIndex][3])/2)+4),(CordStoreData[figIndex][4]+4));
pDC->Rectangle(CordStoreData[figIndex][1]-4,(((CordStoreData[figIndex][2]+CordStoreData[figIndex][4])/2)-4),CordStoreData[figIndex][1]+4,(((CordStoreData[figIndex][2]+CordStoreData[figIndex][4])/2)+4));
pDC->Rectangle(CordStoreData[figIndex][3]-4,(((CordStoreData[figIndex][2]+CordStoreData[figIndex][4])/2)-4),CordStoreData[figIndex][3]+4,(((CordStoreData[figIndex][2]+CordStoreData[figIndex][4])/2)+4));

iSelFigCounter=figIndex;
GetDlgItem(IDC_ROTATE)->EnableWindow(TRUE);

break;
case 3: //Arrow
if((CordStoreData[figIndex][1]-CordStoreData[figIndex][3]) && (CordStoreData[figIndex][1]-aSelectedCord[0]))
{
    dSlope1=(CordStoreData[figIndex][2]-CordStoreData[figIndex][4])/(CordStoreData[figIndex][2]-CordStoreData[figIndex][4]);
    dSlope2=(CordStoreData[figIndex][2]-aSelectedCord[1])/(CordStoreData[figIndex][2]-aSelectedCord[0]);
    dSlopeDiff=dSlope1-dSlope2;

    if(!dSlopeDiff)
    {
        pDC->SelectStockObject(BLACK_PEN);
pDC->Rectangle(CordStoreData[figIndex][1]-2,CordStoreData[figIndex][2]-2,CordStoreData[figIndex][1]+2,CordStoreData[figIndex][2]+2);
pDC->Rectangle(CordStoreData[figIndex][3]-2,CordStoreData[figIndex][4]-2,CordStoreData[figIndex][3]+2,CordStoreData[figIndex][4]+2);
iSelFigCounter=figIndex;
GetDlgItem(IDC_ROTATE)->EnableWindow(TRUE);
    }
}

break;
default:
bDrawSelect=false;
break;
BOOL CUsrInt::PreCreateWindow(CREATESTRUCT& cs)
{
    // TODO: Add your specialized code here and/or call the base class
    cs.lpszClass=AfxRegisterWndClass(CS_DBLCLKS,AfxGetApp()->LoadStandardCursor(IDC_CROSS),(HBRUSH)(COLOR_WINDOW+1),AfxGetApp()->LoadIcon(IDC_CROSS));

    return CDialog::PreCreateWindow(cs);
}

void CUsrInt::OnLButtonUp(UINT nFlags, CPoint point)
{
    if(bDrawLine)
    {
        CordStoreData[FigCounter][0]=0;
        CordStoreData[FigCounter][1]=XCordData[0];
        CordStoreData[FigCounter][2]=YCordData[0];
        CordStoreData[FigCounter][3]=XCordData[CordIndex];
        CordStoreData[FigCounter][4]=YCordData[CordIndex];
        FigCounter=FigCounter+1;
    }

    if(bDrawRect)
    {
        CordStoreData[FigCounter][0]=1;
        CordStoreData[FigCounter][1]=XCordData[0];
        CordStoreData[FigCounter][2]=YCordData[0];
        CordStoreData[FigCounter][3]=XCordData[CordIndex];
        CordStoreData[FigCounter][4]=YCordData[CordIndex];
        FigCounter=FigCounter+1;
    }

    if(bDrawCircle)
    {
        CordStoreData[FigCounter][0]=2;
        CordStoreData[FigCounter][1]=XCordData[0];
        CordStoreData[FigCounter][2]=YCordData[0];
        CordStoreData[FigCounter][3]=XCordData[CordIndex];
        CordStoreData[FigCounter][4]=YCordData[CordIndex];
        FigCounter=FigCounter+1;
    }

    if(bDrawArrow)
    {
        DrawTriangle(1);
        CordStoreData[FigCounter][0]=3;
        CordStoreData[FigCounter][1]=XCordData[0];
        CordStoreData[FigCounter][2]=YCordData[0];
        CordStoreData[FigCounter][3]=XCordData[CordIndex];
        CordStoreData[FigCounter][4]=YCordData[CordIndex];
        FigCounter=FigCounter+1;
    }
}
if(bDrawCompress)
{
    bDrawCompress=false;
}
CDialog::OnLButtonUp(nFlags, point);

void CUsrInt::OnRefreshForm()
{
    int figName;
    CClientDC* pDC=new CClientDC(this);
    pDC->SelectStockObject(NULL_BRUSH);
    for(int figIndex=0;figIndex<=FigCounter;figIndex++){
        figName=CordStoreData[figIndex][0];
        switch(figName)
        {
            case 0:
                pDC->SelectStockObject(BLACK_PEN);
                pDC->MoveTo(CordStoreData[figIndex][1],CordStoreData[figIndex][2]);
                pDC->LineTo(CordStoreData[figIndex][3],CordStoreData[figIndex][4]);
                break;
            case 1:
                pDC->SelectStockObject(BLACK_PEN);
                pDC->Rectangle(CordStoreData[figIndex][1], CordStoreData[figIndex][2], CordStoreData[figIndex][3], CordStoreData[figIndex][4]);
                break;
            case 2:
                pDC->SelectStockObject(BLACK_PEN);
                pDC->Ellipse(CordStoreData[figIndex][1], CordStoreData[figIndex][2], CordStoreData[figIndex][3], CordStoreData[figIndex][4]);
                break;
            case 3:
                pDC->SelectStockObject(BLACK_PEN);
                pDC->MoveTo(CordStoreData[figIndex][1], CordStoreData[figIndex][2]);
                pDC->LineTo(CordStoreData[figIndex][3], CordStoreData[figIndex][4]);
                break;
            default:
                break;
        }
    }
}
void CUsrInt::OnOK()
CDialog::OnOK();

void CUsrInt::OnCancel()
{
    CDialog::OnCancel();
}

void CUsrInt::OnForce()
{
    MakeAllFlagsFalse();
    bDrawArrow=true;
}

void CUsrInt::OnCompressFig(int iFigCnt, int iNewXCord, int iNewYCord)
{
    int iCompFigName;
    CClientDC* pDC = new CClientDC(this);
    pDC->SelectStockObject(NULL_BRUSH);
    iCompFigName = CordStoreData[iFigCnt][0];
    switch(iCompFigName)
    {
        case 0: //Line
            pDC->SetROP2(R2_NOT);
            //pDC->SelectStockObject(GRAY_BRUSH);
            pDC->MoveTo(CordStoreData[iFigCnt][1], CordStoreData[iFigCnt][2]);
            pDC->LineTo(CordStoreData[iFigCnt][3], CordStoreData[iFigCnt][4]);
            CordStoreData[iFigCnt][3] = iNewXCord;
            CordStoreData[iFigCnt][4] = iNewYCord;
            pDC->SelectStockObject(BLACK_PEN);
            pDC->MoveTo(CordStoreData[iFigCnt][1], CordStoreData[iFigCnt][2]);
            pDC->LineTo(CordStoreData[iFigCnt][3], CordStoreData[iFigCnt][4]);
            break;
        case 1: //Rectangle
            //pDC->SetBkColor(RGB(100,192,192));
            pDC->SetROP2(R2_NOT);
            //pDC->SelectStockObject(GRAY_BRUSH);
            pDC->Rectangle(CordStoreData[iFigCnt][1], CordStoreData[iFigCnt][2], CordStoreData[iFigCnt][3], CordStoreData[iFigCnt][4]);
            CordStoreData[iFigCnt][2], CordStoreData[iFigCnt][3], CordStoreData[iFigCnt][4]);
CordStoreData[iFigCnt][3]=iNewXCord;
CordStoreData[iFigCnt][4]=iNewYCord;

//Redraw the new rectangle
pDC->SelectStockObject(BLACK_PEN);
pDC->Rectangle(CordStoreData[iFigCnt][1],
,CordStoreData[iFigCnt][2],CordStoreData[iFigCnt][3],CordStoreData[iFigCnt][4]);
break;
case 2: //Ellipse
   pDC->SetROP2(R2_NOT);
pDC->Ellipse(CordStoreData[iFigCnt][1],
,CordStoreData[iFigCnt][2],CordStoreData[iFigCnt][3],CordStoreData[iFigCnt][4]);
   CordStoreData[iFigCnt][3]=iNewXCord;
   CordStoreData[iFigCnt][4]=iNewYCord;
   pDC->SelectStockObject(BLACK_PEN);
pDC->Ellipse(CordStoreData[iFigCnt][1],
,CordStoreData[iFigCnt][2],CordStoreData[iFigCnt][3],CordStoreData[iFigCnt][4]);
break;
case 3: //Arrow
   pDC->SetROP2(R2_NOT);
pDC->MoveTo(CordStoreData[iFigCnt][1],
,CordStoreData[iFigCnt][2],CordStoreData[iFigCnt][3],CordStoreData[iFigCnt][4]);
   CordStoreData[iFigCnt][3]=iNewXCord;
   CordStoreData[iFigCnt][4]=iNewYCord;
   pDC->SelectStockObject(BLACK_PEN);
pDC->MoveTo(CordStoreData[iFigCnt][1],
,CordStoreData[iFigCnt][2],CordStoreData[iFigCnt][3],CordStoreData[iFigCnt][4]);
break;
default:
   break;
}

void CUsrInt::ChkForCompression(int iXCCord, int iYCCord)
{
   int iCompFigName;
   int iMaxXCord,iMaxYCord,iMinXCord,iMinYCord;
   CClientDC* pDC=new CClientDC(this);
pDC->SelectStockObject(NULL_BRUSH);

iCompFigName=CordStoreData[iSelFigCounter][0];
switch(iCompFigName)
{
    case 0: //Line
        if((((iXCCord>=(CordStoreData[iSelFigCounter][1]-4))
          && (iXCCord<=(CordStoreData[iSelFigCounter][1]+4))) 
          ||
          ((iXCCord>=(CordStoreData[iSelFigCounter][3]-4))
            &&
            (iXCCord<=(CordStoreData[iSelFigCounter][3]+4))))
          &&
          (((iYCCord>=(CordStoreData[iSelFigCounter][2]-4))
            &&
            (iYCCord<=(CordStoreData[iSelFigCounter][2]+4)))
            ||
            ((iYCCord>=(CordStoreData[iSelFigCounter][4]-4))
              &&
              (iYCCord<=(CordStoreData[iSelFigCounter][4]+4))))
        {
            bDrawCompress=true;
            pDC->SelectStockObject(R2_NOT);
            pDC->Rectangle(CordStoreData[iSelFigCounter][1]-4,
                          CordStoreData[iSelFigCounter][2]-4,
                          CordStoreData[iSelFigCounter][1]+4,
                          CordStoreData[iSelFigCounter][2]+4);
            pDC->Rectangle(CordStoreData[iSelFigCounter][3]-4,
                          CordStoreData[iSelFigCounter][4]-4,
                          CordStoreData[iSelFigCounter][3]+4,
                          CordStoreData[iSelFigCounter][4]+4);
        }
        break;
    case 1: //Rectangle

        if(CordStoreData[iSelFigCounter][3]>CordStoreData[iSelFigCounter][1])
        {
            iMaxXCord=CordStoreData[iSelFigCounter][3];
            iMinXCord=CordStoreData[iSelFigCounter][1];
        }

        if(CordStoreData[iSelFigCounter][1]>CordStoreData[iSelFigCounter][3])
        {
            iMaxXCord=CordStoreData[iSelFigCounter][1];
            iMinXCord=CordStoreData[iSelFigCounter][3];
        }

        if(CordStoreData[iSelFigCounter][2]>CordStoreData[iSelFigCounter][4])
        {
            iMaxYCord=CordStoreData[iSelFigCounter][2];
            iMinYCord=CordStoreData[iSelFigCounter][4];
        }

        if(CordStoreData[iSelFigCounter][4]>CordStoreData[iSelFigCounter][2])
        {
            iMaxYCord=CordStoreData[iSelFigCounter][4];
            iMinYCord=CordStoreData[iSelFigCounter][2];
        }
}
if(((iXCCord>=iMaxXCord-4) && (iXCCord<=iMaxXCord+4)) ||
((iXCCord>=iMinXCord-4) && (iXCCord<=iMinXCord+4)) ||
((iXCCord>=(((iMaxXCord+iMinXCord)/2)-4)) && (iXCCord<=(((iMaxXCord+iMinXCord)/2)+4))))
&&
((iYCCord>=iMaxYCord-4) && (iYCCord<=iMaxYCord+4)) || ((iYCCord>=iMinYCord-4) && (iYCCord<=iMinYCord+4)) ||
((iYCCord>=(((iMaxYCord+iMinYCord)/2)-4)) && (iYCCord<=(((iMaxYCord+iMinYCord)/2)+4))))
{
    bDrawCompress=true;
    pDC->SelectStockObject(R2_NOT);
    pDC->Rectangle(CordStoreData[iSelFigCounter][1]-4,
                   CordStoreData[iSelFigCounter][2]-4,
                   CordStoreData[iSelFigCounter][1]+4,
                   CordStoreData[iSelFigCounter][2]+4);
    pDC->Rectangle(CordStoreData[iSelFigCounter][3]-4,
                   CordStoreData[iSelFigCounter][4]-4,
                   CordStoreData[iSelFigCounter][3]+4,
                   CordStoreData[iSelFigCounter][4]+4);
    pDC->Rectangle((((CordStoreData[iSelFigCounter][1]+CordStoreData[iSelFigCounter][3])/2)-4),
                   CordStoreData[iSelFigCounter][2]-4,
                   (((CordStoreData[iSelFigCounter][1]+CordStoreData[iSelFigCounter][3])/2)+4),
                   CordStoreData[iSelFigCounter][2]+4);
    pDC->Rectangle(CordStoreData[iSelFigCounter][1]-4,
                   (((CordStoreData[iSelFigCounter][2]+CordStoreData[iSelFigCounter][4])/2)-4),
                   CordStoreData[iSelFigCounter][1]+4,
                   (((CordStoreData[iSelFigCounter][2]+CordStoreData[iSelFigCounter][4])/2)+4));
    break;
}
case 2: // Ellipse

if(CordStoreData[iSelFigCounter][3]>CordStoreData[iSelFigCounter][1])
{
    iMaxXCord=CordStoreData[iSelFigCounter][3];
    iMinXCord=CordStoreData[iSelFigCounter][1];
}
if(CordStoreData[iSelFigCounter][1]>CordStoreData[iSelFigCounter][3])
    {
        iMaxXCord=CordStoreData[iSelFigCounter][1];
        iMinXCord=CordStoreData[iSelFigCounter][3];
    }

if(CordStoreData[iSelFigCounter][2]>CordStoreData[iSelFigCounter][4])
    {
        iMaxYCord=CordStoreData[iSelFigCounter][2];
        iMinYCord=CordStoreData[iSelFigCounter][4];
    }

if(CordStoreData[iSelFigCounter][4]>CordStoreData[iSelFigCounter][2])
    {
        iMaxYCord=CordStoreData[iSelFigCounter][4];
        iMinYCord=CordStoreData[iSelFigCounter][2];
    }

if((((iXCCord>=iMaxXCord-4) &&
    ((iXCCord>=(iMaxXCord+iMinXCord)/2)-4)) || ((iXCCord<=(iMaxXCord+iMinXCord)/2)+4)) ||
    ((iYCCord>=(iMaxYCord+iMinYCord)/2)-4)) && ((iYCCord<=(iMaxYCord+iMinYCord)/2)+4))
    {
        bDrawCompress=true;
        pDC->SelectStockObject(R2_NOT);
        pDC->Rectangle((((CordStoreData[iSelFigCounter][1]+CordStoreData[iSelFigCounter][3])/2)-4),
                        (((CordStoreData[iSelFigCounter][2]+CordStoreData[iSelFigCounter][4])/2)-4),
                        ((((CordStoreData[iSelFigCounter][1]+CordStoreData[iSelFigCounter][3])/2)+4),
                        (((CordStoreData[iSelFigCounter][2]+CordStoreData[iSelFigCounter][4])/2)+4));
    }
}

break;
}
case 3: //Arrow
    if(((iXCCord>=(CordStoreData[iSelFigCounter][1]-2))
        & (iXCCord<=(CordStoreData[iSelFigCounter][1]+2)) ||
        (iXCCord>=(CordStoreData[iSelFigCounter][3]-2)) & (iXCCord<=(CordStoreData[iSelFigCounter][3]+2))))
    {
        pDC->Rectangle(CordStoreData[iSelFigCounter][1]-4, CordStoreData[iSelFigCounter][1]+4,
                        CordStoreData[iSelFigCounter][1]-4, CordStoreData[iSelFigCounter][1]+4);
        pDC->Rectangle(CordStoreData[iSelFigCounter][2]-4, CordStoreData[iSelFigCounter][2]+4,
                        CordStoreData[iSelFigCounter][2]-4, CordStoreData[iSelFigCounter][2]+4);
        pDC->Rectangle(CordStoreData[iSelFigCounter][3]-4, CordStoreData[iSelFigCounter][3]+4,
                        CordStoreData[iSelFigCounter][3]-4, CordStoreData[iSelFigCounter][3]+4);
        pDC->Rectangle(CordStoreData[iSelFigCounter][4]-4, CordStoreData[iSelFigCounter][4]+4,
                        CordStoreData[iSelFigCounter][4]-4, CordStoreData[iSelFigCounter][4]+4);
    }

break;
void CUsrInt::OnRotate()
{
    // Rotating selected figure
    bDrawRotate=true;
    if(bDrawSelect){
        SelectForRotation();
        bDrawSelect=false;
    }
}

void CUsrInt::OnRotateFig(int iFigCnt, int iNewXCord, int iNewYCord)
{
    int iCompFigName;
    int iTranXCord[4], iTranYCord[4];
    float fRotAngle;
    CClientDC* pDC=new CClientDC(this);
    pDC->SelectStockObject(NULL_BRUSH);
    iCompFigName=CordStoreData[iFigCnt][0];
    switch(iCompFigName)
    {
        case 0: //Line
            if(!iNewXCord){
                fRotAngle=90;
            } else {
                fRotAngle=atan2((iNewYCord-CordStoreData[iFigCnt][2]),(iNewXCord-CordStoreData[iFigCnt][1]));
                if(fRotAngle<0){
                    fRotAngle=-fRotAngle/100;
                } else {
                    fRotAngle=fRotAngle/100;
                }
            }
            break;
    default:
        break;
    }
}
pDC->SetROP2(R2_NOT);
pDC->MoveTo(CordStoreData[iFigCnt][1], CordStoreData[iFigCnt][2]);
pDC->LineTo(CordStoreData[iFigCnt][3], CordStoreData[iFigCnt][4]);
pDC->SelectStockObject(BLACK_PEN);

iTranXCord[1] = (CordStoreData[iFigCnt][1] * cos(fRotAngle)) - (CordStoreData[iFigCnt][2] * sin(fRotAngle));
iTranYCord[1] = (CordStoreData[iFigCnt][1] * sin(fRotAngle)) + (CordStoreData[iFigCnt][2] * cos(fRotAngle));
PDC->MoveTo(iTranXCord[1], iTranYCord[1]);
PDC->MoveTo(CordStoreData[iFigCnt][1], CordStoreData[iFigCnt][2]);

iTranXCord[2] = abs((CordStoreData[iFigCnt][3] * cos(fRotAngle)) - (CordStoreData[iFigCnt][4] * sin(fRotAngle)));
iTranYCord[2] = abs((CordStoreData[iFigCnt][3] * sin(fRotAngle)) + (CordStoreData[iFigCnt][4] * cos(fRotAngle)));
PDC->LineTo(iTranXCord[2], iTranYCord[2]);
CordStoreData[iFigCnt][3] = iTranXCord[2];
CordStoreData[iFigCnt][4] = iTranYCord[2];

break;

case 1: //Rectangle
pDC->SetROP2(R2_NOT);
pDC->Rectangle(CordStoreData[iFigCnt][1], CordStoreData[iFigCnt][2], CordStoreData[iFigCnt][3], CordStoreData[iFigCnt][4]);
CordStoreData[iFigCnt][3] = iNewXCord;
CordStoreData[iFigCnt][4] = iNewY Cord;

//Redraw the new rectangle
pDC->SelectStockObject(BLACK_PEN);
pDC->Rectangle(CordStoreData[iFigCnt][1], CordStoreData[iFigCnt][2], CordStoreData[iFigCnt][3], CordStoreData[iFigCnt][4]);
break;

case 2: //Ellipse
pDC->SetROP2(R2_NOT);
pDC->Ellipse(CordStoreData[iFigCnt][1], CordStoreData[iFigCnt][2], CordStoreData[iFigCnt][3], CordStoreData[iFigCnt][4]);
CordStoreData[iFigCnt][3]=iNewXCord;
CordStoreData[iFigCnt][4]=iNewYCord;

pDC->SelectStockObject(BLACK_PEN);
pDC->Ellipse(CordStoreData[iFigCnt][1],CordStoreData[iFigCnt][2],CordStoreData[iFigCnt][3],CordStoreData[iFigCnt][4]);

break;

case 3: //Arrow
pDC->SetROP2(R2_NOT);
pDC->MoveTo(CordStoreData[iFigCnt][1],CordStoreData[iFigCnt][2]);
pDC->LineTo(CordStoreData[iFigCnt][3],CordStoreData[iFigCnt][4]);

CordStoreData[iFigCnt][3]=iNewXCord;
CordStoreData[iFigCnt][4]=iNewYCord;

pDC->SelectStockObject(BLACK_PEN);
pDC->MoveTo(CordStoreData[iFigCnt][1],CordStoreData[iFigCnt][2]);
pDC->LineTo(CordStoreData[iFigCnt][3],CordStoreData[iFigCnt][4]);
break;

default:
break;
}
}

void CUsrInt::ChkForRotation(int iXCCord, int iYCCord)
{
int iCompFigName;
int iMaxXCord,iMaxYCor,iMinXCord,iMinYCor;

CClientDC* pDC=new CClientDC(this);
pDC->SelectStockObject(NULL_BRUSH);

iCompFigName=CordStoreData[iSelFigCounter][0];
switch(iCompFigName)
{
 case 0: //Line
  if(((iXCCord<=(CordStoreData[iSelFigCounter][1]-4)) &&
    (iXCCord>=(CordStoreData[iSelFigCounter][1]+4))) ||
    (iXCCord>=(CordStoreData[iSelFigCounter][3]-4)) &&
    (iXCCord<=(CordStoreData[iSelFigCounter][3]+4))) &&
    (((iYCCord>=(CordStoreData[iSelFigCounter][2]-4)) &&
    (iYCCord>=(CordStoreData[iSelFigCounter][2]+4))) ||
    (iYCCord>=(CordStoreData[iSelFigCounter][4]-4)) &&
    (iYCCord<=(CordStoreData[iSelFigCounter][4]+4))))
    
    bDrawCompress=true;

    break;
}

case 1: //Rectangle
if(CordStoreData[iSelFigCounter][3]>CordStoreData[iSelFigCounter][1])
{
    iMaxXCord=CordStoreData[iSelFigCounter][3];
    iMinXCord=CordStoreData[iSelFigCounter][1];
}

if(CordStoreData[iSelFigCounter][1]>CordStoreData[iSelFigCounter][3])
{
    iMaxXCord=CordStoreData[iSelFigCounter][1];
    iMinXCord=CordStoreData[iSelFigCounter][3];
}

if(CordStoreData[iSelFigCounter][2]>CordStoreData[iSelFigCounter][4])
{
    iMaxYCord=CordStoreData[iSelFigCounter][2];
    iMinYCord=CordStoreData[iSelFigCounter][4];
}

if(CordStoreData[iSelFigCounter][4]>CordStoreData[iSelFigCounter][2])
{
    iMaxYCord=CordStoreData[iSelFigCounter][4];
    iMinYCord=CordStoreData[iSelFigCounter][2];
}

if((((iXCCord>=iMaxXCord-4) && (iXCCord<=iMaxXCord+4)) ||
     ((iXCCord>=iMinXCord-4) && (iXCCord<=iMinXCord+4)) ||
     (((iXCCord>=(iMaxXCord+iMinXCord)/2)-4) && ((iXCCord<=(iMaxXCord+iMinXCord)/2)+4))) &&
     (((iYCCord>=iMaxYCord-4) && (iYCCord<=iMaxYCord+4)) || ((iYCCord>==(iMaxYCord+iMinYCord)/2)-4) &&
     ((iYCCord>=(iMaxYCord+iMinYCord)/2)+4))))
{
    bDrawCompress=true;
}

break;

case 2: //Ellipse

if(CordStoreData[iSelFigCounter][3]>CordStoreData[iSelFigCounter][1])
{
    iMaxXCord=CordStoreData[iSelFigCounter][3];
    iMinXCord=CordStoreData[iSelFigCounter][1];
}

if(CordStoreData[iSelFigCounter][1]>CordStoreData[iSelFigCounter][3])
{
iMaxXCord=CordStoreData[iSelFigCounter][1];
iMinXCord=CordStoreData[iSelFigCounter][3];

if(CordStoreData[iSelFigCounter][2]>CordStoreData[iSelFigCounter][4])
{
    iMaxYCord=CordStoreData[iSelFigCounter][2];
iMinYCord=CordStoreData[iSelFigCounter][4];
}

if(CordStoreData[iSelFigCounter][4]>CordStoreData[iSelFigCounter][2])
{
    iMaxYCord=CordStoreData[iSelFigCounter][4];
iMinYCord=CordStoreData[iSelFigCounter][2];
}

if(((iXCCord>=(iMaxXCord+iMinXCord)/2)-2) &&
    ((iYCCord>=(iMaxYCord+iMinYCord)/2)-2) &&
    ((iXCCord<=(iMaxXCord+iMinXCord)/2)+2) &&
    ((iYCCord<=(iMaxYCord+iMinYCord)/2)+2))
{
    bDrawCompress=true;
}

break;

case 3: //Arrow
    if(((iXCCord>=(CordStoreData[iSelFigCounter][1]-2)) ||
        (iXCCord<=(CordStoreData[iSelFigCounter][1]+2))) &&
        ((iXCCord>=(CordStoreData[iSelFigCounter][3]-2)) &&
        (iXCCord<=(CordStoreData[iSelFigCounter][3]+2))) &&
        ((iYCCord>=(CordStoreData[iSelFigCounter][2]-2)) &&
        (iYCCord<=(CordStoreData[iSelFigCounter][2]+2))) ||
        ((iYCCord>=(CordStoreData[iSelFigCounter][4]-2)) &&
        (iYCCord<=(CordStoreData[iSelFigCounter][4]+2))))
    {
        bDrawCompress=true;
    }
    break;

default:
    break;
}

void CUsrInt::SelectForRotation()
{
    int iCompFigName;
    CClientDC* pDC=new CClientDC(this);
pDC->SelectStockObject(NULL_BRUSH);
    //CPen myPen1(PS_SOLID, 1, RGB(100,50,100));
pDC->SelectStockObject(BLACK_PEN);
iCompFigName=CordStoreData[iSelFigCounter][0];
switch(iCompFigName)
{
    case 0: //Line
        pDC->Ellipse(CordStoreData[iSelFigCounter][1]-4,
                     CordStoreData[iSelFigCounter][2]-4,
                     CordStoreData[iSelFigCounter][1]+4,
                     CordStoreData[iSelFigCounter][2]+4);
        pDC->Ellipse(CordStoreData[iSelFigCounter][3]-4,
                     CordStoreData[iSelFigCounter][4]-4,
                     CordStoreData[iSelFigCounter][3]+4,
                     CordStoreData[iSelFigCounter][4]+4);
        break;
    case 1: //Rectangle
        pDC->Ellipse(CordStoreData[iSelFigCounter][1]-4,
                     CordStoreData[iSelFigCounter][2]-4,
                     CordStoreData[iSelFigCounter][1]+4,
                     CordStoreData[iSelFigCounter][2]+4);
        pDC->Ellipse(CordStoreData[iSelFigCounter][3]-4,
                     CordStoreData[iSelFigCounter][4]-4,
                     CordStoreData[iSelFigCounter][3]+4,
                     CordStoreData[iSelFigCounter][4]+4);
        pDC->Ellipse(CordStoreData[iSelFigCounter][1]-4,
                     CordStoreData[iSelFigCounter][4]-4,
                     CordStoreData[iSelFigCounter][1]+4,
                     CordStoreData[iSelFigCounter][4]+4);
        pDC->Ellipse(CordStoreData[iSelFigCounter][3]-4,
                     CordStoreData[iSelFigCounter][2]-4,
                     CordStoreData[iSelFigCounter][3]+4,
                     CordStoreData[iSelFigCounter][2]+4);
        break;
    case 2: //Ellipse
        pDC->Ellipse(((CordStoreData[iSelFigCounter][1]+CordStoreData[iSelFigCounter][3])/2)-4,
                      ((CordStoreData[iSelFigCounter][2])+CordStoreData[iSelFigCounter][4])/2-4,
                      (((CordStoreData[iSelFigCounter][1]+CordStoreData[iSelFigCounter][3])/2)+4),
                      ((CordStoreData[iSelFigCounter][2])+CordStoreData[iSelFigCounter][4])/2+4);
        pDC->Ellipse(((CordStoreData[iSelFigCounter][1]+CordStoreData[iSelFigCounter][3])/2)-4,
                      ((CordStoreData[iSelFigCounter][4])-CordStoreData[iSelFigCounter][2])/2-4,
                      (((CordStoreData[iSelFigCounter][1]+CordStoreData[iSelFigCounter][3])/2)+4),
                      ((CordStoreData[iSelFigCounter][4])-CordStoreData[iSelFigCounter][2])/2+4);
        pDC->Ellipse(((CordStoreData[iSelFigCounter][2])+CordStoreData[iSelFigCounter][4])/2-4,
                      ((CordStoreData[iSelFigCounter][1]+CordStoreData[iSelFigCounter][3])/2)-4,
                      (((CordStoreData[iSelFigCounter][2])+CordStoreData[iSelFigCounter][4])/2)+4),
                      ((CordStoreData[iSelFigCounter][1]+CordStoreData[iSelFigCounter][3])/2)+4);
        pDC->Ellipse(((CordStoreData[iSelFigCounter][2])+CordStoreData[iSelFigCounter][4])/2-4,
                      ((CordStoreData[iSelFigCounter][3])-CordStoreData[iSelFigCounter][1])/2-4,
                      (((CordStoreData[iSelFigCounter][2])+CordStoreData[iSelFigCounter][4])/2)+4),
                      ((CordStoreData[iSelFigCounter][3])-CordStoreData[iSelFigCounter][1])/2+4);
        break;
    case 3: //Arrow
        break;
    default:
        break;
}