APPLICATION OF WEB-BASED
INTERACTIVE AND MULTIMEDIA
TECHNOLOGY IN AN INTRODUCTORY
ENGINEERING COURSE

A Thesis Presented to

The Faculty of the

Fritz J. and Dolores H. Russ
College of Engineering and Technology

Ohio University

In Partial Fulfillment
Of the Requirement for the Degree
Master of Science

by

Liang Zhang

March, 2001
ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Bhavin V. Mehta, for his advice and support during my work.

I would also like to thank Dr. Eric P. Steinberg in the Department of Civil Engineering, another faculty member involved in this project, for his notes, suggestion, and comments. Thanks to Dr. Gregory G. Kremer from Mechanical Engineering for his suggestions and comments.

Finally, I offer my greatest thanks to my wife, without whose support and encouragement I would have never made it this far.
TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION .......................................................... 1
  1.1 Statics ............................................................................... 2
  1.2 Web-Based Learning Education ........................................... 3
  1.3 Thesis Organization ............................................................ 5

CHAPTER 2 LITERATURE SURVEY .................................................. 6

CHAPTER 3 STATICS ON THE WEB ............................................... 13
  3.1 HTML .............................................................................. 14
  3.2 Animated GIF File ............................................................ 17
  3.3 Problem Solving Examples .................................................. 19
  3.4 Three-Dimensional Models Using VRML ............................. 23

CHAPTER 4 INTERACTIVE AND DYNAMIC EXAMPLES ON WEB ...... 27
  4.1 2-D Interactive Vector Program ......................................... 28
  4.2 3-D Interactive Rotation Java Model .................................... 32
    4.2.1 VecDouble Class ....................................................... 33
      4.2.1.1 Adding and Subtracting Vectors ......................... 33
      4.2.1.2 Vector Dot Product ........................................... 33
      4.2.1.3 Vector Cross Product ...................................... 34
      4.2.1.4 Length of a Vector ............................................ 34
      4.2.1.5 Normalizing a Vector ........................................ 34
      4.2.1.6 Multiplication of a Vector by a Scalar ................. 34
# Table of Contents (continued)

4.2.1.7 The Surface_Normal Method .................................. 35

4.2.2 Matrix Class ........................................................... 35
  4.2.2.1 Multiplying a Matrix by a Matrix ......................... 36
  4.2.2.2 Multiplying a Vector by a Matrix ......................... 36
  4.2.2.3 Translating a Vector ........................................... 36
  4.2.2.4 Scaling a Vector ................................................. 37
  4.2.2.5 Rotating a Vector .............................................. 37
  4.2.2.6 The transform Matrix ........................................... 37

4.2.3 3D Interactive Rotation Java Model ............................. 38
  4.2.3.1 The SolidStuff Class ........................................... 38
  4.2.3.2 The DisplayFacet Method ..................................... 39
  4.2.3.3 Creating a Rectangular Solid .................................. 41
  4.2.3.4 Initializing a Solid ............................................. 43
  4.2.3.5 Algorithm of Rotation About an Arbitrary Axis ........ 44
  4.2.3.6 Work10( ) Method .............................................. 44
  4.2.3.7 Step 1: Translating the solid ............................... 45
  4.2.3.8 Step 2: Rotate about the X axis ............................ 47
  4.2.3.9 Step 3: Rotate the solid about Y axis ..................... 48
  4.2.3.10 Step 4: Rotate the solid about Z axis for angle θ .... 49
  4.2.3.11 Step 5: Inverse of Step 3 .................................... 50
  4.2.3.12 Step 6: Inverse of Step 2 .................................... 51
Table of Contents (continued)

4.2.3.13 Step 7: Inverse of Step 1 .......................... 52

CHAPTER 5 CONCLUSIONS AND FUTURE WORKS ..................... 54

5.1 Conclusions .......................................................... 54

5.2 Future Work .......................................................... 66

BIBLIOGRAPHY ................................................................ 68
LIST OF FIGURES

Figure 3.1 Page 2 of Chapter 9 ................................................................. 15
Figure 3.2 GIF animation for Page 5 of Chapter 1 ................................. 18
Figure 3.3 Problem solving example in Chapter 2 ................................... 21
Figure 3.4 Step 2 for the same example ............................................... 22
Figure 3.5 Results of Step 2 and hint of Step 3 for that example ................. 23
Figure 3.6 Regular 2-D view to a 3-D model on Page 12 of Chapter 3 ............ 25
Figure 3.7 View the same model in last figure as a 3-D model by Cosmo-player .... 26
Figure 4.1 Addition of vectors ................................................................. 29
Figure 4.2 2-D vector program shows wrong results when click “Your answer” .... 30
Figure 4.3 Click “Correct answer” to get the correct results graphically .......... 31
Figure 5.1 The front page of the Web site ................................................ 54
Figure 5.2 A page with voice file in Chapter 2 .......................................... 55
Figure 5.3 Page 6 of Chapter 8 ................................................................. 56
Figure 5.4 An example for problem saving in Chapter 2 ............................. 57
Figure 5.5 The page linked to GIF animation .......................................... 58
Figure 5.6 2-D interactive vector Java program ....................................... 59
Figure 5.7 2-D interactive vector Java program ....................................... 60
Figure 5.8 3-D interactive rotation solid Java program (initial solid) ............... 61
Figure 5.9 3-D interactive model by Java ............................................... 62
List of Figures (continued)

Figure 5.10 3-D interactive rotation model by Java ........................................ 63
Figure 5.11 3-D interactive rotation Java model .............................................. 64
Figure 5.12 3-D interactive rotation Java model .............................................. 65
CHAPTER 1
INTRODUCTION

Statics, an introductory engineering course, was developed as a web-based course at Ohio University. The web-based course contains animated GIF, Java-based animations, 3-D models using Virtual Reality Modeling Language (VRML), and Java-based interactive tutorials and examples to demonstrate the principles of mechanics in a more dynamic and interactive form. The primary emphasis was on using the site as a supplement to the regular class lectures. The web-based course is geared towards interactive examples and analysis models where the student can actually solve a myriad of problems on a specific topic until students have a better understanding of concepts.

The web site also includes the lecture notes for review. This research concentrates on issues and applications related to HTML, animated Gif file, the Java-based applications developed for interactive problem solving, and two & three-dimensional Java-based interactive models.

1.1 Statics

Statics is a core engineering course taken by undergraduate students in engineering at most of the universities in the United States. Since Statics is one of the first engineering courses in the curriculum, students often have difficulty with rigorous problem solving and with assessing whether or not their calculated results are physically meaningful. [1]
Problem solving is important both as a useful skill and as a means for teaching important concepts in Statics. Often for learning to take place in the context of problem solving, the student must not be given the answer but must struggle and work through the solution step-by-step, receiving guidance only at the proper time. Therefore, the standard method of assigning and grading homework may not maximize all students learning because it does not necessarily force the student to correctly solve every important problem and it does not provide immediate feedback. Student learning may be increased by eliminating partial credit and requiring students to continually rework and resubmit their homework until it is correct, in order to receive credit for the work. This is different to the traditional method of marking the problem wrong and showing the error or the correct solution procedure. The forced solution method may be better for promoting student learning than the traditional homework/grading method but it is very inefficient due to the amount of grading/guidance time necessary and the time lapse between the student’s initial attempt at solving the problem and the feedback from the instructor. Often the student’s calculated results are not physically realistic, but he/she is not aware of that until the homework problems are reviewed and returned. Also, since all students normally solve the same homework problems, there is always the possibility that some students will copy the work of other students or receive significant assistance and will miss out on the learning experience provided by solving the problem. With web-based problem solving examples, the instructor might improve his/her teaching significantly in above aspects. [1]

Most engineering applications are truly three-dimensional. The concepts of Statics are very much dependent on the students visualizing the physics of the problem
and understanding and interpreting the behavior of the solution (e.g., vectors, free-body diagrams, motion etc.). It is not easy for the instructor to construct an actual model of the problem in a classroom and demonstrate the physical phenomena of that model on a 2D chalkboard. Using the tools available on the World Wide Web (like VRML and Java) one can perform interactive model building and demonstrate the concepts of Statics associated with that model in three dimensions. The students will be able to interactively change model parameters (e.g., force or dimensions) on the model and graphically view the effects of those changes. This will definitely increase the student understanding and learning of objects in the three-dimensional world. This is obviously much better visually than the instructor who illustrates the object in class using the chalkboard.

1.2 Web-Based Learning Education

The World Wide Web is a hypermedia system that uses universally accepted protocols over non-proprietary networks that encourage the sharing of information. The Web allows anyone with a browser to transfer files from thousands of possible sources to his/her computer in a nonlinear fashion. Accessing information has never before been possible on this scale, making the educational potential of the Web enormous.

With the Web, an instructor may do more teaching in less time. Facilities might not be needed, but a Web site is required. It is not necessary to have a concentrated group of students in one particular geographic area, all of whom have the time and resources to attend class at a set time. If the instructor is comfortable with fifteen virtual students, he/she only needs to find fifteen people on the planet that need the information and have access to the Web. The instructor can train across time zones. This is a vitally
important concept--asynchronous training across all time zones. Students in Sydney can hold a synchronous chat with a student in Atlanta or can post messages and e-mail to each other. Students from every countries or even continents can share their thoughts or ideas in the same course Web site at the same--or different--times.

The instructor can train a diverse student base. With students from all over, he/she can provide the platform for the students and "let them go." Over and over, the most valuable part of students Web studying experience are getting information and perceptions from other people in other geographic areas and other countries--effectively widening students own personal worlds.

The Web not only is a supplement to real course teaching, but also allows instructors to teach students who would otherwise not have the opportunity to learn.

As mentioned before, a course Web site can significantly help students learn much about a course; it can even help students to understand some concepts they might not understand without a Web site, such as three-dimensional models. The Web-based course also helps students in problem solving. It is also a supplement to the lectures of the real class.

In addition, many potential students cannot attend a traditional classroom, eliminating the opportunity to learn and improve their lives. A single instructor can literally reach the globe from a single source, without regard for geographic or time limitations. Ferrying a particular subject matter expert or industry leader to remote locations around the world is prohibitively expensive, and few experts want to spend all of their time in hotels and lecture halls. The Web does provide the opportunity for all interested students to be involved in studying.
Several instructors can teach simultaneously from the same Web site. The thought of putting five instructors in one classroom at the same time might be outrageous, but perfectly allowable on the Web. Different instructors can log in to the Web site and provide input to students from their own perspectives. This allows the student to benefit from multiple ideas and opinions rather than just one. When you combine this concept with the possibility of having instructors from many countries and many cultures contribute to a single Web-based course, the possibilities are limitless.

1.3 Thesis Organization

Chapter 2 provides a brief review of literature and other research in the field of Web-based learning education. The literature is extremely limited because actually the online course is a really new technology for the whole education environment. It just came true several years ago. The fundamental introduction of the "Statics" course is discussed in Chapter 3. Some basic concepts are introduced here, such as HTML, animated Gif files, and interactive problem solving examples. Chapter 4 contains two Java programs that are used in the Web site and also are the main focus of this research. One is a 2D resultant vectors program and the other is a 3D interactive solid program. The algorithms and developing processes are discussed here. Also most of the functions of the programs are explained step by step. Chapter 5 includes a discussion of results, conclusions and recommendations for future work. The appendices contain the step by step instructions to use the two Java based problem solvers (tutorials), one for 2D resultant vector generation and the other for rotation of a 3D object about an arbitrary axis.
CHAPTER 2
LITERATURE REVIEW

As the Internet and the World Wide Web saturate our life and culture, people are, for the fourth time this century, in the middle of a major campaign attempting to translate technological advances into solutions of university problems. After radio, film, television, and the computer, the Web has become another powerful technology being used to transform teaching and learning. With its increasing capacity for multimedia, multimode communication and information presentation, easy access to an ever-growing body of information, and new ways of data representation, the web presents educators with exciting opportunities to enhance teaching and learning by setting up Web-based courses.

However, the history of educational technology has shown that technological potentials do not easily transfer into direct educational benefits. Two issues have been identified that determine the degree to which the potential of a given technology is realized: if and how the technology is used. Neither one of these issues was considered seriously in early attempts to introduce technology into universities. It was widely believed that instructors would naturally embrace technology and readily integrate it into their teaching because one of the most appealing features of the new computer technologies is the idea that it opens up many new opportunities to make teaching and learning more exciting. It was also believed that using technology would automatically improve teaching and learning. As a result of the gradual realization that technology alone neither leads to automatic use nor more effective teaching and learning, an
increasing number of efforts have been directed to promoting more widespread, effective uses of technology.

During nearly 30 years of teaching mathematics at the University of California, Santa Cruz, Dr. Edward M. Landesman [2] experienced many instructional situations—some that worked and some that did not. In the late 1960’s, while teaching a large calculus class, he made arrangements with the media services group on campus to videotape a lecture. When he viewed the video tapes several days later, he saw himself teaching in a way that somehow he never had imagined. There were aspects that caused him to question the whole delivery process. He saw himself as very active and highly engaged in the instruction, trying very hard to assure that the students were comprehending what he was saying. At the same time, he saw the students sitting there in a fairly passive way, some taking notes, some listening intently, others nodding, and some not paying attention at all. In many ways he really didn’t know whether or not learning was actually taking place.

So, in an accidental way, his viewing of the tape had a profound influence that would change his way of thinking about teaching and learning for the next thirty years. He began to experiment with video and made a series of over 100 black-and-white videotapes, each about 15 to 20 minutes long, covering the main topics of calculus. They were not at all interactive and they made little use of graphics aside from line drawings. It was far from the ideal in achieving an active learning environment with a first rate technology by today’s standards. These tapes led to other tapes for the teaching of precalculus, using color, some graphics, and some animation. A further endeavor was an
interactive videotape project for the teaching of prealgebra, and later, an interactive laser
disc project for the learning of precalculus.

As the technology was becoming more sophisticated, Dr. Landesman, too, was
learning how to use it to maximize the teaching and learning effort. Six years ago he
began working with a team of mathematicians, instructional designers, and engineers to
produce computer-based, interactive multimedia materials for the teaching and learning
of mathematics. The technology is now at the stage where they can satisfy all of the
major characteristics that allow students to be active learners who are engaged in their
learning with rich media-driven applications, sophisticated graphical representations,
sequential learner-controlled learning screens, multiple representations, and real time
feedback. Such learning materials now play an important role in many colleges and
universities to help students learn various courses material. From the results of his
research, not only are students learning mathematics in courses taught with visual
technologies, but also longitudinal studies conducted by some institutions are showing
that students who have been exposed to these interactive learning materials perform
better in subsequent courses. [2]

In the last 25 years, course designers have used computer-mediated instruction to
explore the various ways that a programmed computer system could be used to provide
adaptive, interactive instruction that would accommodate individual learner needs.
Computer-mediated instruction can be as simple as a drill and practice tutorial or as
complicated as a multimedia extravaganza with many levels of conditional branching,
inquiry, questioning, and other adaptations to individual learner needs.
After 1990, more and more instructors set up an online course by using the Internet. The reason that they set up an online course is apparently because in a browser-only format that the largest possible audience has ready access. [3] The day of the computer-based virtual classroom is growing rapidly nearer. Technology has enabled instructors to reach beyond the boundaries of the classroom. [4] Web pages are easy to create, and have far more capability than the disk-based educational software of the past. Connections between computers, via the Internet, allow the teacher to have unparalleled capability for delivering instruction to, interacting with, and evaluating the off-campus student. [4] The Internet-connected computer allows students to access most class information at a time of their choosing. Moreover, they can access the class information using their home computer. This mode of delivering instruction promises to create new educational opportunities for a large number of people. [4] Online courses are no longer confined to experimental ventures of innovative faculty members. They have become an integral part of curricula at most institutions. As entire degrees become available online, there is an increasing need for more “mainstream” faculty to teach online. [5] Academic and industrial course can be enhanced with links to resources on the Web, or the courses can be delivered virtually—completely via the Web. [6]

However, most Web sites normally have many limitations. Generally speaking, most of the designers of an online course only think about the contents of the Web site but discard the very important aspect of an online course; a good online course should be an interesting, abstractive and interactive instead of static. Also the online course must be a course supplement, if it fails to achieve this, it will likely fail. [3]
For most students, learning requires something more than just exposure to the material—and that something is often hard to identify, harder to quantify, and harder still to control. An online course can approach, and even surpass, the learning model of a conventional classroom. [7] However, most of the online courses have been the development of Web sites the feature educational resources: lessons, activities, articles and just general information on leading edge topics. [8]

When one searches in the Web, one finds many online courses that are only static. Most of them have a similar structure and style: notes are the prominent contents and they also have some assignments and examples. Basically they have introductions in the beginning; sometimes they have syllabus. They have many links (that are notes) to the syllabus or directly go to the notes of the course. The users are guided step-by-step to learn the course until the course is finished. Of course, users can learn a lot from this kind of Web site. The big disadvantage of this sort of online course is that it is not dynamic and interactive. Users can’t enhance their knowledge by interacting with the online course that is a very important characteristic of the real class. Many institutions are converting lecture notes or other paper-based materials to HTML for the World-Wide Web, but, with little support provided for the student, the gains are minimal. Simply converting the publication medium is inadequate for supported distance education, which aims to engage the student in a community of learning. [9] As students do in real class, in an online course students can also do the following things--they should be informed that they are wrong and try again when they try to solve some example problems, they should visually view some results after they input some values for understanding some specific concepts (for example, three-dimensional model), and
they should have the ability to prove some concepts by giving some operations on the online course. This format is similar to that is available in a traditional course setting.

For example, the department of Civil Engineering in the University of Toledo has a Web site about engineering course. But this site is more like a static one. It has a few Java applets, but they have more notes and introduction information.

Northern Arizona University has a Web-based travel and Tourism course. This web site is designed for delivery via the World Wide Web to service the need for knowledge about the management of tourism and travel industries. The design focuses on the development of higher level learning skills of analysis, synthesis and evaluation. The model provides a format for integrating interaction with content and interpersonal interaction. Students interact with the content through the exploration of World Wide Web sites selected to enhance or reinforce a concept presented in the text. Interpersonal interaction takes place as the instructor communicates ideas to the students via a lecture, e-mail and a list serve. The students communicate with the instructor via e-mail and the list serve. But this type of interaction is not common in a traditional course. Students appear freer to discuss questions and concerns with the instructor via e-mail than they are in the classroom. [10]

The research for this thesis was conducted as follows. Statics is an introductory engineering course that almost all of the engineering undergraduate students must take. This course is developed as a Web-based course. The course contains animated GIF, Java-based animations, 3-D models using Virtual Reality Modeling Language (VRML), interactive tutorials, and examples to demonstrate the principles of mechanics in a more dynamic and interactive form.
• By animation GIF, students can view an animation GIF file instead of static one.

• By VRML, students can view a lot of 3-D models by zoom, pan, and rotate so that they can visually understand what a 3-D model looks like.

• By numerous examples, students can learn to solve problems step-by-step and also get help and hints from the site. They can also check their results step-by-step.

• By 2-D vector program, students can fully understand the concept of the summation of vectors.

• By Java based problem solvers, students can solve problems interactively, with the problem solvers guiding them through the steps of the problem and providing hints or feedback whenever necessary.
CHAPTER 3
STATICS ON THE WEB

This chapter describes some basic features of the Statics Web site. Basically, when people design a web site, they usually use HTML (Hypertext Markup Language) as the fundamental language. However, HTML is so simple that it can’t do as much work as they want. For instance, they can’t make a dynamic and interactive Web site with only HTML. Therefore, some other scripting and high level programming language is used to implement this interactive Web site. GIF animation and three-dimensional models and examples are developed in this project by the Java programming language.

The steps of the design sequence for this research are:

- Set up a front page for this Web site.
- Implement syllabus and accomplish the important notes for most of the chapters.
- Make GIF animations.
- Generate many three-dimensional VRML models by Solid Edge, which can be viewed by a VRML plug-in on the Web.
- Provide some examples for problem solving step-by-step.
- Incorporate voice files and video files on Web site.
- Provide two-dimensional interactive Java program.
- Provide three-dimensional interactive Java applet programs.

This chapter describes the above steps except the interactive Java programs, which will be discussed in Chapter 4.
3.1 HTML

It's common knowledge that all documents on the WWW should be written in so-called HTM--Hyper Text Markup Language. Much less is known about what to count as HTML. To make things really obfuscated, it should be noted that HTML is a markup language. Markup means that it will mark different elements of your document, but how this document will be seen by Web wandering individuals depends on which Web browsers are running on a multitude of various operating systems.

As is known, HTML is used as the basic language for all the pages of this Web site. Each page is set up in the following manners:

1. Use “Paint” package that is in “accessories” in most PC in the public computer lab in Stocker Center to create a Bitmap file. Put the important information for a specific note page in this Bitmap file except the title and the similar link margin for each page at the end of every page.

2. Use Microsoft photo editor or other photo shop packages to open the Bitmap file, change the background of the Bitmap file as “transparent” and save it as a GIF file so that it can be easily put in a HTML file.

3. Use ftp to transfer the file from a PC to bobcat.ent.ohiou.edu (login name for this Web site account is “statics”), and of course, transfer the desired file to the specific directory (all the GIF files are in the same directory so that it is easy to handle them).

4. Create an HTML file for that page where the GIF file is the part of the HTML file.
The reason that "Paint" is used to create Bitmap file is as follows. In almost all the pages in this Web site, not only the text contents, but also a lot of equations, graphics, and symbols are needed. With "Paint" (See Figure 3.1), all these necessary contents can be put easily in the same Bitmap file, and it is also very convenient to modify the file if necessary.

However, the Bitmap file can’t be put into HTML file. Therefore, the Bitmap file needs to be edited as a GIF file so that it can be used in a HTML file. To describe it thoroughly, the following example HTML file for Page 2 of Chapter 9 follows.

```
<html>
<head>
<title>notes</title>
```

Figure 3.1 : Page 2 of Chapter 9

```
R = \int dF = \int k_y \, dA = 0
M_x = \int y \, dF = \int k y^2 \, dA = k \int y^2 \, dA

Moments of inertia:
I_x = \int y^2 \, dA \quad I_y = \int x^2 \, dA

I = \text{Moment of inertia about X axis}
I = \text{Moment of inertia about Y axis}
y = \text{Distance of } dA \text{ from X axis}
x = \text{Distance of } dA \text{ from Y axis}
```
Chapter Nine

9.2 Moment of inertia of an area

Page

[1] // links to other pages in this chapter

[2]

[3].

[11] // total pages are 11
The above HTML file is the style for most pages in this Web site. Some other pages' HTML are a bit different because they might have some other links to voice file, VRML file, video file, or Java programs.

3.2 Animated Gif File

As one surfs the Web, an individual will come across many little animated pictures that add life and interest to the Web pages. Animation is performed by a series of frames displayed in sequence with each picture nearly the same but with small differences in position. This makes objects appear to move the same way as in movies. The animations simply use a whole series of graphics files, often in GIF format. A GIF animation file can contain all of the frames. If necessary, some time delay is added between frames.
In Statics course, sometimes students are expected to be able to understand some particular concepts, which are easier to understand with animation. For instance, in “the principle of Transmissibility”: the conditions of equilibrium or motion of a body remain unchanged if a force on the body is replaced by a force of the same magnitude and direction along the line of action of the original force. In this case, this important concept is shown to students on the Web. (See Figure 3.2) It must be easier to understand if this concept is shown by animation than just to put the text on the Web.
Some graphics can be drawn to show a force that is in a different place, and then use a GIF animation package to produce the animation.

The theory of a GIF animation package is simple: just draw some graphics to show the different places of a force and save them as GIF files. By using the Microsoft GIF Animation package, link several GIF files and adjust the time between every individual GIF file frame and save them as a whole GIF file. The final GIF file is an animation file. There are several animation files of this kind and the “Principle of Transmissibility” is one of them.

3.3 Problem Solving Examples

In engineering courses, problem solving is so important both as a useful skill and as a means for teaching the important concepts that we must provide strong support for students to grasp these concepts. But for true learning to take place (in the context of problem solving), the students must not be given the answer but must struggle and work through the solution step-by-step, receiving guidance only at the proper times. Therefore, the standard method of assigning and grading homework does not maximize student learning because it does not force the students to correctly solve every important problem and it does not provide immediate feedback. Eliminating partial credit and requiring students to continually rework and resubmit their homework until it is correct might increase students learning. [1]

As opposed to the traditional method of marking the problem incorrect and showing the error or the correct solution procedure, an instructor using the forced solution method would identify the error and provide guidance but would require the
students to continually rework the problem until it is correctly solved. This forced solution method is better for promoting student learning than the traditional homework/grading method but it is very inefficient due to the amount of grading/guidance time necessary and the time lapse between the students’ initial attempt at solving the problem and the feedback from the instructor.

Occasionally a student’s calculated results are not physically realistic. However, they are not aware of that until their homework problems are reviewed and returned. Also, since all students normally solve the same homework problems, there is always the possibility that some students will copy the work of others or be provided significant assistance and will miss out on the learning experience provided by solving the problem.

Therefore, in Statics Web site, some problem solving examples are presented in this way:

1. First of all, the question and the graphics for this question are provided. Just below the question, there is a link to the final result. Students can check with the results if they think that they already have correct results. However, some hints are also provided for those students who can’t get the final results by themselves or perhaps have no idea of where to start.

2. For hints, there are normally several steps. From the first step of the hint, students should know how to start. They can check their result for the first step with the correct results of that step. Example 1 of Chapter is shown in Figure 3.3.
3. When students click the button “Click here to check your answer”, they get the next page that not only provides the result of the first step, but also provides the hint of step two. They should check the results of the step one first. If they have the same results as the correct results, then they may go on to read the hint of step two and continue to solve the problem. Otherwise, students should go back to step one to check what’s wrong (See Figure 3.4).
Step 2:
Find components of the forces for the crane and the forces for the tug. To check your answer, click here!

Click here to check your answer!

Back to the front!

Figure 3.4: Step 2 for hint of this example

4. When students click “Click here to check to check your answer!”, they get the next page. Similar to the previous page, they are first required to check the results of step 2 with the correct results of that step. If their results are wrong, they should go back and try to figure out the reason. Otherwise, they may read the hint of step 3 and then try to follow step 3, and then they can check their results by going to the next page (See Figure 3.5).
5. Finally students can check their results with the correct result for step 3 (last step in this question). If their results are wrong, they need to go back again to check the reason. Otherwise, they have finished this problem successfully.

3.4 Three-Dimensional Models Using VRML

Most of the engineering applications related to Statics are truly three-dimensional. The concepts of Statics are very much dependent on the students visualizing the physics of the problem and understanding and interpreting the behavior of the solution (e.g., vectors, free-body diagrams, motion etc.). It is not easy for the
instructor to construct an actual model of 3D problem in a classroom and demonstrate the physical phenomena (dynamics) of that model on a 2D chalkboard. Using the tools available on the World Wide Web (like VRML), one can perform interactive model building and demonstrate the concepts of Statics associated with that model in three dimensions.

VRML (Virtual Reality Modeling Language) is a platform-independent file format for sharing three-dimensional worlds on the Web. VRML worlds can be interactive and animated and can include embedded hyperlinks to other Web documents. With VRML, we can put three-dimension VRML models on web. By VRML model, students can dynamically zoom, rotate, and pan the model to visualize in all three dimensions. Now it is easier for students to understand how a three-dimension model works as it does in real life.

There are many packages that can be used to create a VRML file. Solid Edge is used in this project. Solid Edge software is a solid modeling system for ease of use and rapid implementation. Solid Edge, with STREAM technology, provides performance for mechanical assembly design and drafting, combining production-proven solid modeling and unparalleled ease-of-use. Written for Windows NT and Windows-98 operating systems, Solid Edge is a CAD solution for design engineers moving from 2-D drafting to 3-D solids-based design. By using Solid Edge, a 3D graphics can be easily drawn and then saved as .wrl file, which can be viewed as VRML file. Figure 3.6 and Figure 3.7 show a VRML file on the Statics Web site.
Addition of couples:

\[ M = rR \]

\[ M = F_1r \]

\[ M_2 = rF_2 \]

\[ \bar{M} = \bar{r} \times \bar{R} = \bar{r} \times (\bar{F}_1 + \bar{F}_2) = \bar{r} \times \bar{F}_1 + \bar{r} \times \bar{F}_2 \]

\[ \bar{M} = \bar{M}_1 + \bar{M}_2 \]

Figure 3.6: Regular 2-D view to a 3-D model on Page 12 of Chapter 3
Figure 3.7: View of the same model in the Figure 3.6 as a 3-D model by Cosmo-player
CHAPTER 4
INTERACTIVE AND DYNAMIC EXAMPLES ON WEB

Nowadays, more and more Web sites are dynamic and interactive. The advantage of a dynamic and interactive Web site is that users are able to interact with the Web site so that they can get a more direct experience from the Web site, especially for an engineering course Web site. For instance, if students can interactively change model parameters (e.g., force or dimensions) on the model and graphically view the effects of those changes, they can get more intuitive experience. It is also beneficial for their full understanding of some important concepts and handling kinds of problems related to those concepts.

On the Statics Web site, there are several interactive and dynamic examples with which students can view the result graphically after changing some parameters. These examples are created by the Java programming language. The Java programming language platform provides a portable, interpreted, high-performance, simple, object-oriented programming language and supports a run-time environment. The advantages of using the Java programming language are:

- Java programming language is object oriented, yet it's still very simple.
- Java development cycle is much faster because Java technology is interpreted.

The compile-link-load-test-crash-debug cycle is obsolete--now you just compile and run.
• Java applications are portable across multiple platforms. Applications are written once and never need to be imported--they will run without modification on multiple operating systems and hardware architectures.

• Java applications are robust because the Java runtime environment manages memory.

• Java interactive graphical applications have high performance because multiple concurrent threads of activity in the application are supported by the multithreading built into the Java programming language and runtime platform.

• Java applications are adaptable to changing environments because one can dynamically download code modules from anywhere on the network.

• Java end users can trust that their applications are secure, even though they're downloading code from all over the Internet; the Java runtime environment has built-in protection against viruses and tampering. [17]

4.1 2-D Interactive Vector Program

Vectors are an important concept in an engineering course. Vectors are defined as mathematical expressions possessing magnitude and direction, which add according to the parallelogram law. For example, two forces acting at a right angle to each other, one of 4 lb and the other of 3 lb, add up to a force of 5 lb, not to a force of 7 lb. Forces are not the only expressions which follow the parallelogram law of addition. As you know, displacements, velocities, accelerations, and moment are other examples of physical quantities possessing magnitude and direction and which are added according to the parallelogram law. Vectors may represent all these quantities mathematically.
Ordinary numbers or scalars represent those physical quantities, which do not have direction, such as volume, mass, or energy.

Some operations that can be done on vectors are:

- Addition of vectors
- Subtracting vectors
- Vector dot product
- Vector cross product

On this Web site, the first one--adding vectors by a Java program--is graphically solved. By definition, vectors add according to the parallelogram law. Thus, attaching the two vectors to the same point A and constructing a parallelogram, using P and two sides of the parallelogram obtain the sum of two vectors P and Q. The diagonal that passes through A represents the sum of the vectors P and Q, and this sum is denoted by $P + Q$. The fact that the sign $+$ is used to denote both vector and scalar addition should not cause any confusion if vector and scalar quantities are always carefully distinguished. Thus, we should note that the magnitude of the vectors $P + Q$ is not, in general, equal to the sum $P + Q$ of the magnitudes of the vectors P and Q (See Figure 4.1).

![Figure 4.1: Addition of vectors](image-url)
In the interactive and dynamic Java program, students are given this question (See Figure 4.2).

![Image of a 2-D vector program showing wrong result when click "Your answer".](image-url)

**Figure 4.2**: 2-D vector program shows wrong result when click "Your answer"

The question is: given two vectors--the first one from point A to point B and the second one from point A to point C. Determine the addition of these two vectors. Assume the first point of the resultant vector is still point A, give the coordinate of the second point and then check your result with the correct answer. Point A is fixed and point B and C are generated randomly.

Notice that in order to generate a generic example, a random number generating algorithm is used to generate the coordinates for point B and C. That means every time
that the user opens this applet, they will get a different value for B and C for this question. After students calculate the resultant vector and input the coordinate of the second point of the resultant vector, they may click the "**Your answer**" button to check their result graphically. If their result is wrong, they will get an unrealistic graphic, which informs them that their calculation is wrong. If they click the "**Correct answer**" button, they will view graphics that show them the addition of the two vectors. Also students can get some hints by clicking the "**Hint**" button or resetting the coordinates by clicking "**Reset x,y**" button (See Figure 4.3).

![Improved version for vector - Netscape](image)

**Figure 4.3**: Click “Correct answer” to get the correct result graphically
4.2 3-D Interactive Rotation Java Model

The technique used to create a three-dimensional model on a two-dimensional screen was to define each solid as a collection of facets, each of which was a quadrilateral. If the surface of the solid was curved, we tried to use many small quadrilaterals so that the fact approximated the curve by a collection of flat surfaces was not noticeable. The solid was defined to fit in a box that was + or - 1 unit in the X direction, + or – 2 units in the Y direction, and + or – 3 units in the Z direction and centered at the origin.

Vectors and matrices were used to describe and manipulate a rectangular solid object. Before this step, however, one needed to learn how to define and perform mathematical operations with vectors and matrices in Java. Just to clear up any misconceptions, Java has a Vector class, but it defines variable content arrays and has nothing to do with mathematical vectors. If one has worked with C++, it is known that not only can Vector and Matrix classes be defined, but also various mathematical operators (such as + and -) can be overridden so that they perform the corresponding operations on vectors or matrices. This is a very elegant way of handling these mathematical operations. Unfortunately, Java does not have this capability. This makes handling vectors and matrices in Java a little more difficult than when using C++, but a little less complicated than using ordinary C.

Creating and moving solid objects and converting them into a two-dimensional picture required the creation and manipulation of three-dimensional vectors. Some of these manipulations required processing a four-by-four matrix. The following sections describe the procedures involved.
4.2.1 VecDouble class

The *VecDouble* class [see the appendix for the source code] defined three-dimensional vectors (using double floating point numbers) and defined the methods used for their processing. You'll observe that the *VecDouble* class has three double type variables—\(x\), \(y\), and \(z\). There were two constructors for a *VecDouble*. If no parameters were passed when a new *VecDouble* was created, the variables \(x\), \(y\), and \(z\) were all set to 0.0. If three double type numbers were passed to the new *VecDouble*, then \(x\), \(y\), and \(z\) take on these values.

4.2.1.1 Adding and Subtracting Vectors

To add two vectors, the \(x\), \(y\), and \(z\) components of the two vectors first were added. This was shown in the method `addvee` in the above listing. To subtract one vector from another, each element from the corresponding element of the other vector was subtracted. This is shown in the `subvee` method in the listing.

4.2.1.2 Vector Dot Product

There are two forms of vector multiplication. The first of these is the vector dot product. It is a double floating point number obtained by multiplying together each pair of corresponding vector elements and then summing the results. This is done in the method `dotvee` shown in the above listing.
4.2.1.3 Vector Cross Product

The other form of vector multiplication is the vector cross product. The result of this multiplication is a new vector whose elements are composed of combinations of multiplication of the original vector elements. You can see how this is done by looking at the cross method in the above listing.

4.2.1.4 Length of a Vector

The method lengthvec returns the length of a vector. This is simply the square root of the sum of the squares of the vector elements.

4.2.1.5 Normalizing a Vector

A vector is normalized when it is converted to a unit vector having the same direction as the original vector. The method normvec does this. First the method calls the lengthvec method to find the vector's length. The only pathological case where the vector cannot be normalized is that the original vector length is zero. The method first checks for this case; if it is encountered, the vector is simply returned unchanged. Otherwise the length then divides each element of the vector.

4.2.1.6 Multiplication of a Vector by a Scalar

A vector is multiplied by a scalar (an ordinary double floating point number) by multiplying each element of the vector by the scalar. The method scalemult in the above listing performs this operation.
4.2.1.7 The Surface Normal Method

The surface normal method, in the above listing passed four vectors that define the four corners of a quadrilateral. The method returns a vector of unit length that is at right angles to the planar surface defined by the four vectors passed to it. The method began by computing a vector dir1 that was in the direction from the first passed vector to the second. A second vector dir2 was then computed having a direction from the first passed vector to the fourth passed vector. The cross product of these two vectors was then taken to obtain a vector perpendicular to the plane defined by the three passed vectors. The length of this perpendicular vector was then taken. The method then performed the same process using the third, fourth, and second passed vectors. Next, if the length of either perpendicular vector was zero, the other perpendicular vector was normalized and returned as the surface normal vector. If both perpendicular vectors were non-zero, the returned surface normal vector was the average of the two perpendicular vectors.

4.2.2 The Matrix Class

A $4 \times 4$ element matrix was used to multiply a vector in order to perform scaling, rotation, translation, or a combination of all three. The class Matrix [see the appendix for the source code] defined a Matrix object as consisting of a $4 \times 4$ array of double floating point numbers and set up the methods to be used with this class. A Matrix object could be initialized in either of two ways. The first method passed no parameters to the object; all of the matrix elements were then set to zero. The second type of constructor passed 16 double floating-point numbers to the object. These became the elements of the matrix in the order $a(0,0)$, $a(0,1)$, $a(0,2)$, $a(0,3)$, $a(1,0)$, $a(1,1)$, and so forth.
4.2.2.1 Multiplying a Matrix by a Matrix

Multiplication of matrices was done so that those matrices could be used for the scaling, translation, and rotation of vectors into a single combined matrix that required only a single set of operations to transform a vector rather than several sets of operations. The programming to implement this is shown in `mult_mat` method in the above listing.

4.2.2.2 Multiplying a Object by a Matrix

When a three-dimensional object was multiplied by a 4 x 4 matrix, the result was a new three-dimensional object. Depending upon the element values of the matrix, this multiplication could have represented a translation, rotation, or scaling of the old vector or a combination of all three, depending upon the values of the matrix elements. The actual multiplication process is quite similar to that used in obtaining the cross product of two vectors. The programming is shown in the method `MatCrossVec` in the listing above.

4.2.2.3 Translating a Vector

The method `translate` in the above listing performed the necessary operations to create the translation matrix. The method passed a vector that contained the translation in the x, y, and z directions. The function then generated the appropriate translation matrix. When the matrix operated upon a vector, the length and direction of the vector were unchanged, but its position was changed as specified by the three parameters.
4.2.2.4 Scaling a Vector

The method scale in the above listing accomplished creating a scaling matrix. This method passed a vector that consisted of the three scaling parameters in the x, y, and z directions, respectively. It then generated the appropriate scaling matrix. When the matrix operated upon a vector, the position of the vector was unchanged, but its length component in each of the three axis directions was modified by the corresponding scaling parameter.

4.2.2.5 Rotating a Vector

The method began by setting up a new matrix called \( a \), which contained all zero elements. Next, it set all of the matrix elements that are to be 1 or -1. It then computed the sine and cosine of the rotation angle and placed them in the proper matrix elements (with the proper sign) for the rotation axis selected. The effect of this matrix on a vector was that the vector was rotated by the specified angle about the selected axis, but retained the same length and the same orientation with respect to the plane defined by the two axes not selected.

4.2.2.6 The Transform Matrix

The transform method in the above listing combined scaling, rotation about each of the three axes, and translation into one overall transformation matrix by multiplying the individual matrices together. It passed three vectors--\( tr \), \( sc \), and \( ro \). The \( tr \) vector contained the amount of translation for the x, y, and z directions, respectively. The vector \( sc \) contained the scaling factors in the x, y, and z directions, respectively. The vector \( ro \) contained the rotation angles about the x, y, and z axes, respectively.
4.2.3 3-D Interactive Rotation Java Model

Using the mathematical tools (VecDouble and Matrix class) developed in the preceding steps, we can construct a solid that might make up a three-dimensional picture and project it onto a two-dimensional surface to produce a display.

4.2.3.1 The SolidStuff Class

Basically, different solids will be drawn. It turns out that there are a number of parameters and vectors that are common to all solid object classes, as well as the DisplayFacet method, which is used by every solid object whenever it wants to paint a facet on the display. Rather than repeat these in every solid object class, they are included in a class called SolidStuff. Then each solid object class is defined to extend SolidStuff, thereby causing it to inherit all of this class's characteristics.

SolidStuff begins by defining angle, which is the angle of the viewer vector around the z axis, and tilt, which is the angle between the observer vector and the (x,y) plane. Next, four trigonometric parameters based on these angles are defined. ViewDist is set to be the distance from the screen to the viewer. Next a vector Viewer is created to represent the direction of the viewer from the center of the viewing screen. Then the vector light, which is the direction of the light source from the viewing screen, is created. Several integers and double floating point parameters are defined in our solid object computations. The object color of class Color is set up and initialized to represent the color of the object to be painted. Four vectors--facet3D1, facet3D2, facet3D3, and facet3D4--are set up to represent each of the four corners of the facet to be painted on the screen. A few temporary vectors are set up and then the vectors tr, which represents the translation of the solid object from
its default position at the center of the screen, \( sc \), which represents the scaling of the solid object in each axis direction, and \( ro \), which represents any rotation of the object about one of the three axes. Finally \( trans \), which will contain the final matrix that is used to perform the transformation of each specified point, is defined.

### 4.2.3.2 The DisplayFacet Method

Suppose that the four vectors—\( facet3D1 \), \( facet3D2 \), \( facet3D3 \), and \( facet3D4 \)—that make up the corners of a three-dimensional facet that we want to display on the screen have already been defined. These vectors are passed to the method \( DisplayFacet \). The method begins by setting the values for four parameters that control the overall color characteristics of every object in the scene. The parameter \( ambient \) specifies the fraction of maximum color that is to be used by any portion of an object that is not illuminated by the light source. If you set this to 0.0, such portions will be black; if you set to 1.0, these portions will be in the brightest shade of the selected color. The parameter \( diffuse \) specifies the percentage of maximum brightness that is to be controlled by the mathematics of light source and viewer direction. The parameter \( specular \) specifies the fraction of maximum color that is to be applied to highlights on curved surfaces. The parameter \( glossiness \) is a power that controls the size of the highlight. The method then defines a few more parameters and vectors that are needed in the calculations and then sets up an object \( poly \) of class \( Polygon \). Finally, the method instantiates the matrix \( Rotator \), which is used to rotate all points to compensate for the angular changes specified at the beginning of \( SolidStuff \).

The first step in displaying a facet is to compute the normal vector to the facet's surface, \( surf\_norm \). Then the \( dotvec \) method is used to find the cosine of the angle between
the Viewer vector and the surface normal vector (beta). If the cosine of this angle is negative, the viewer cannot see this particular facet, so the method returns without doing anything else. Otherwise, the method calculates the cosine of theta, the angle between the light vector and the surface normal vector. If this is negative, no light from the light source is reaching the facet, so intensity is set to the ambient light level. Otherwise, the cosine of alpha, the angle between the reflected light from the facet and the Viewer vector, is computed. Then intensity is set to the sum of the ambient light, the diffuse light factor multiplied by the cosine of theta, and the specular light factor multiplied by the cosine of alpha raised to the glossiness power. This last term is the Phong shading factor which creates highlights that are seen when illuminating curved surfaces. The Math class methods min and max are used to assure that intensity remains within the boundaries of 0.0 to 1.0.

Now the proper intensity of the facet color is known, but the actual position of each corner of the facet on the two-dimensional screen needs to be established. Start with a three-dimensional point in the world coordinates used to develop a scene and then convert it to a two-dimensional projection onto the viewing screen. The viewer is looking at the origin of the world coordinate system; at some point between the viewer and the origin is the viewing screen, positioned at right angles to the view-to-origin vector. First multiply the facet vector by the matrix Rotator to correct for the rotation around the z axis specified by angle and the angle with the z axis specified by tilt. Then add in the pers vector, which defines the perspective from which the scene is viewed. Use the addPoint method to first convert the three-dimensional to a two-dimensional projection, then add offsets to establish the coordinate system at the center of the screen, and then save this new point as part of the definition of the polygon Poly. Do this for each of the four corners of the polygon and then
for the first corner again to make sure the polygon is closed. The \texttt{setColor} method is then used to set the color for the facet. This color is the color specified as the maximum color for the facet with each component multiplied by \textit{intensity}. Finally, call the \texttt{fillPolygon} method to paint the facet on the screen.

\textbf{4.2.3.3 Creating a Rectangular Solid}

Refer to the source codes for \texttt{arbrotate.java} from the appendices. The \texttt{arbrotate} class (which extends \texttt{Applet}) is used to generate a rectangular solid and then rotate this solid for some angle by an arbitrary axis, which is defined by the two points whose coordinates are entered by users. Of course, users also enter the rotation angle.

In \textit{init()} method, initialize the class and set 10 buttons and twenty four text fields.

In the south part of layout, set two buttons and eight text fields. The first button is “Initial”; by clicking, users can get the original rectangular solid to show up in the screen that is defined in the \textit{init()} method. The first three text fields are the coordinates for the first point to define the axis. The next three text fields are the coordinates for the second point to define the axis. The next text field is the angle that users want to rotate the solid around the axis. The value of these seven text fields will be used for all the rest steps. The next text field is the number of steps that users estimate. The last component is a button by which users can check if the number of steps that they enter is correct.

In the east part of the layout, there are sixteen text fields to which users can enter the values of transform the matrix for each step. Under these text fields there are eight buttons. Users don’t have to use all these eight buttons. They have to be left there because the buttons have to be initialized even though users don’t know how many steps
they should use. In other words, after users check their step numbers, there is no way to set the exactly number of buttons at that time; they must be set in constructor (before users know how many they should use). Therefore, users only need to use some of them.

In action() method, the program performs differently by adjusting which buttons the mouse is on. However, after getting the scalar value from the text field and changing the value to double type, it will always call paint method to draw the result graphically.

In getmatrixA() method, the whole matrix value is always obtained from users input. First of all, getText() method is called for each text field to get a string (getText() is a method of TextField class). Then the string is changed to a Double class object by calling new Double(string) (constructor). Finally, the Double object is changed to a double value by calling doubleValue() (doubleValue() is a method of Double class).

In equal() method, one wants to judge if the value that users enter is equal to the correct value. Because the two values both are double type, they can hardly be exactly the same. This simple method is deployed to say they are equivalent if the relative error between the two values is less than 1 percent.

In setmatrix() method, all the sixteen values of the four-by-four matrix are initialized to zero by calling setText() method (setText() is a method of TextField class).

In paint() method, depending on the case (which button is clicked), one of the nine different methods—work0, work1, work2, work3, work4, work5, work6, work7, work10—will be called.
4.2.3.4 Initializing a Solid

If the button "Initial solid" is pushed, \texttt{work0()} method, which just draws a rectangular cube initially without any other interactive operation, is called. First of all, \texttt{setmatrix()} is called to set all the values of the matrix to zero. Then the X, Y, Z axis are drawn. The translation, scale and rotation vectors were set so that the object can be set in the middle of the screen and is big enough for users to observe. In this case, a transition vector is set as (300, 150, 0) while no rotation vector is set. Scaling vector is set as (20, 20, 20). Finally, for each one of the six facets in the solid, draw the vectors for the four points of each facet, transform each point to a visible, middle set point, and then call \texttt{DispalyFacet()} method to show this facet of the solid. Then use a loop to show all the facets of the cube.

In order to provide some random initial values for the user-defined axis, a random number generating algorithm is deployed to create some random number to fill the text fields for the coordinates and the rotation angle.

\begin{verbatim}
Random r = new Random(); // create an object r for Random class
vx1 = Math.abs(r.nextInt())%11 / 5.0; // keep the x coordinate for point 1 in [0,2.0]
va = Math.abs(r.nextInt())%91 ; // keep the rotation angle in [0, 90]
x1field.setText(Double.toString(vx1)); // write the value of vx1 as a string to x1field
anglefield.setText(Integer.toString(va)); // write the value of va as a string to
// anglefield
\end{verbatim}

Of course the user can enter whatever values he wants for the coordinates and rotation angles. This random number generating algorithm just provides another choice.
4.2.3.5 The Algorithm of Rotation About an Arbitrary Axis in Space

Before presenting about the details about each step, the basic algorithm of rotation about an arbitrary axis in space is provided. The axis is passing through a point \((x_1, y_1, z_1)\) and has direction cosines as \((c_x, c_y, c_z)\) while

\[
\begin{align*}
    c_x &= \frac{(x_2 - x_1)}{\text{length of two points}}, \\
    c_y &= \frac{(y_2 - y_1)}{\text{length of two points}}, \\
    c_z &= \frac{(z_2 - z_1)}{\text{length of two points}}.
\end{align*}
\]

Hence, rotating the object about this axis by some angle \(\theta\) has the following seven steps:

1. Translate the object so that point \((x_1, y_1, z_1)\) is at the origin.
2. Rotate about X axis to make the arbitrary axis coincident to the plain that is composed by X and Z axis.
3. Rotate about Y axis to make the arbitrary axis coincident to Z axis.
4. Rotate about Z axis by \(\theta\) degree.
5. Inverse of step 3.
6. Inverse of step 2.
7. Inverse of step 1.

4.2.3.6 WorkIO() Method

If a value is typed into the text field “Step number” and then the button “Check step number” is pushed, WorkIO() method will be called. In this method, the X, Y, Z axes are drawn and then the program judges if the number that users enter is equal to seven. If yes, users can start from the first step. Otherwise, they have to try again.
4.2.3.7 Step1: Translating the solid so that point \((x_1, y_1, z_1)\) is at origin.

The user needs to input the coordinates of the two points that decide the arbitrary axis first. Then input the angle, which is around the axis. After doing that, he/she needs to enter the translation matrix value into the matrix text fields.

If all the necessary values are typed into the necessary text fields and then the button “Step 1” is pushed, the program calls \(Work1()\) method, which shows a solid that has been translated the magnitude. Thus, the point \((x_1, y_1, z_1)\) is at the origin.

Refer to the \(Work1()\) method, \(getMatrixA()\) is called to get the values that users enter from the text fields. Then the X, Y, Z axes are drawn, and the program calculates some necessary globe variables that it needs in the following steps. \(length\) is the length between two points. \(cx, cy\) and \(cz\) are direction cosines for the arbitrary axis. \(x_1, y_1, z_1, x_2, y_2, z_2\) are coordinates of two points. So

\[
\begin{align*}
    cx &= \frac{(x_2 - x_1)}{length}; \\
    cy &= \frac{(y_2 - y_1)}{length}; \\
    cz &= \frac{(z_2 - z_1)}{length}; \\
    d * d &= cy * cy + cz * cz; \\
    xangle &= Math.acos(\frac{cz}{d}) / 0.017453292; // from radian to degree \\
    yangle &= Math.acos(\frac{d}{0.017453292});
\end{align*}
\]

\(xangle\) is the angle that we will rotate the solid about X axis in Step 2 and \(yangle\) is the angle that the solid will be rotated about Y axis in Step 3. Notice that \(cz / d\) is used in the above formulas, so one more judgment is added before \(xangle\) line: if \(d\) is equal to zero, then

\[
xangle = 0;
\]
angle = 90;

Three objects are used for VecDouble class: tr, sc, ro, and one object for Matrix class: tran. tran is set by calling transform() method of Matrix class.

\[ tr = \text{new VecDouble}(x_1, y_1, z_1); \]

\[ sc = \text{new VecDouble}(1.0, 1.0, 1.0); \]

\[ ro = \text{new VecDouble}(0,0,0); \]

\[ tran = tran.\text{transform}(tr, sc, ro); \]

tran1 is the matrix that is obtained from the matrix text fields and Cube.trans is a matrix by calling method transform() and using Cube.tr, Cube.sc, Cube.ro as arguments.

\[ Cube.trans = Cube\text{.trans}\text{.transform}(Cube.tr, Cube.sc, Cube.ro); \]

\[ tran1 = \text{new Matrix} (av11,av12,av13,av14,av21,av22,av23,av24, \]

\[ av31,av32,av33,av34,av41,av42,av43,av44); \]

If the matrix that users enter is not equal to the correct matrix, the wrong translation solid is drawn to users. For each facet of the solid, there are four points. Transform the coordinates of each point by using the wrong matrix users enter and calling MatCrossVec() method of Matrix class first; next transform them to the center of the screen visibly and then call DisplayFacet() method to show the facet. All the six facets are drawn; then the solid is done. Also there is a count variable: flag. If flag is less than 4, some hints are provided to users to let them try again. If they try more than 3 times with the wrong results, the correct translation matrix will be shown directly to them and they are allowed to use this matrix.

If the matrix that users enter is equal to the correct matrix, the correct translation solid is drawn. Transform the coordinates of each point by using the correct matrix users
enter and transform them to the center visibly, as discussed earlier. Then call 
DisplayFacet() method to show the facet. The other procedures are the same as those 
used when the matrix users enter is wrong. After that, draw some strings on the screen 
and set flag to zero so that at the beginning of every step flag is zero because it is a globe 
variable. Finally, call dispose() to completely eliminate the object that invokes it.

4.2.3.8 Step 2: Rotate about the X axis.

After the user finishes the first step successfully, he/she should enter the rotation 
matrix about X axis into the text fields and then click the button “Step 2” to call 
Work2() method to check the result. The matrix is obtained from the text fields. It is set 
to a new matrix tran2, and three axes are drawn. The translation matrix is calculated as 
done in Step 1 and then multiplied with tran2 by calling mult_mat() and gain the matrix 
tran1 that includes everything needed in the first two steps.

\[
\text{tran2} = \text{new Matrix}(av11, av12, av13, av14, av21, av22, av23, av24, 
\quad av31, av32, av33, av34, av41, av42, av43, av44) ;
\]

\[
\text{tr} = \text{new VecDouble}(x1, y1, z1) ;
\]

\[
\text{sc} = \text{new VecDouble}(1.0, 1.0, 1.0) ;
\]

\[
\text{ro} = \text{new VecDouble}(0, 0, 0) ;
\]

\[
\text{tran} = \text{tran}.\text{transform} (\text{tr}, \text{sc}, \text{ro}) ; \quad // \text{correct transform matrix after step 1}
\]

\[
\text{tran1} = \text{tran1}.\text{mult_mat} (\text{tran}, \text{tran2}) ; \quad // \text{wrong transform matrix after multiplying}
\]

\[
\quad \text{// the wrong matrix that users enter}
\]

If the matrix that users enter is not equal to the correct matrix, the wrong 
translation solid is drawn. When transforming the coordinates of each point, the program
uses the wrong matrix that users enter and call \textit{MatCrossVec()} method of \textit{Matrix} class, then call \textit{DisplayFacet()} method to show the facet. A loop is then used to show all the six facets as was done in the last step.

If the matrix that users enter is equal to the correct matrix, the correct translation solid is drawn for users. The coordinates of each point are transformed by using the correct transform matrix \textit{tran} and then the solid is drawn as done in the last step. The count still used \textit{flag} to help users. Before \textit{dispose()} method is called, \textit{setmatrix()} is called to set all the default values of the matrix to zero.

\begin{verbatim}
tr = new VecDouble(x1, y1, z1);
sc = new VecDouble(1.0, 1.0, 1.0);
ro = new VecDouble(xangle, 0, 0);
tran = tran.transformnew(tr, sc, ro); // correct transform matrix after 2 steps
\end{verbatim}

4.2.3.9 Step 3: Rotate solid about Y axis so that the arbitrary axis coincident to Z

After the user finishes the second step successfully, they enter the rotation matrix about Y axis into the text fields and then click the button “Step 3” to call \textit{Work3()} method to check the results. It is similar to \textit{Work2()}, except that when transforming the solid, if the matrix that the user enters is wrong, matrix \textit{tran1} is obtained by using the wrong matrix that users enter:

\begin{verbatim}
tr = new VecDouble(x1, y1, z1);
sc = new VecDouble(1.0, 1.0, 1.0);
ro = new VecDouble(xangle, 0, 0);
tran = tran.transformnew(tr, sc, ro); // correct transform matrix after 2 steps
\end{verbatim}
$tran_2 = new \text{Matrix}(av11, av12, av13, av14, av21, av22, av23, av24,$

$\quad av31, av32, av33, av34, av41, av42, av43, av44);$  

$tran_1 = tran_1.\text{mult\_mat}(tran, tran_2); \quad // \text{wrong transform matrix after multiplying}$

$\quad // \text{the wrong matrix users enter}$

Otherwise, the correct matrix $tran$ is used to transform the solid:

$ro = new \text{VecDouble}(xangle, yangle, 0.0);$  

$tran = tran.\text{transformnew}(tr, sc, ro); \quad // \text{correct transform matrix after 3 steps}$

The either the wrong transformed solid or the correct transformed solid is drawn, some strings are drawn, and do some other similar jobs as we did before.

\[4.2.3.10 \text{ Step 4 : Rotate the solid about Z axis for angle } \theta.\]

After the user finishes the third step successfully, they should enter the rotation matrix about Z axis into the text fields and then click the button "Step 4" to call \text{Work4()} method to check the results. It is similar to \text{Work2()}, except that when transform the solid, if the matrix that the user enters is wrong, matrix $tran_1$ is obtained by using the wrong matrix that users enter:

$tr = new \text{VecDouble}(x1, y1, z1);$  

$sc = new \text{VecDouble}(1.0, 1.0, 1.0);$  

$ro = new \text{VecDouble}(xangle, yangle, 0.0);$  

$tran = tran.\text{transformnew}(tr, sc, ro); \quad // \text{correct transform matrix after 3 steps}$

$tran_2 = new \text{Matrix}(av11, av12, av13, av14, av21, av22, av23, av24,$

$\quad av31, av32, av33, av34, av41, av42, av43, av44);$  

$tran_1 = tran_1.\text{mult\_mat}(tran, tran_2); \quad // \text{wrong transform matrix after multiplying}$
 Otherwise, the correct matrix is used in `tran` to transform the solid:

```java
ro = new VecDouble(xangle, yangle, angle);
sc = new VecDouble(1.0, 1.0, 1.0);
tran = tran.transformnew(tr, sc, ro); // correct transform matrix after 4 steps
```

Then either the wrong transformed solid or the correct transformed solid is drawn, some strings are drawn, and do some other similar jobs as it was done before.

### 4.2.3.11 Step 5: Inverse of step 3 (rotate back about Y axis).

After the user finishes the fourth step successfully, they should enter the back rotation matrix about Y axis into the text fields and then click the button “Step 5” to call `Work5()` method to check the results. It is similar to `Work3()`, except that when transforming the solid, if the matrix that the user entered was wrong, matrix `tran1` is obtained by using the wrong matrix that users enter:

```java
tr = new VecDouble(x1, y1, z1);
sc = new VecDouble(1.0, 1.0, 1.0);
ro = new VecDouble(xangle, yangle, angle);
tran = tran.transformnew(tr, sc, ro); // correct transform matrix after four steps
tran2 = new Matrix(av11, av12, av13, av14, av21, av22, av23, av24, av31, av32, av33, av34, av41, av42, av43, av44);
tran1 = tran1.mult_mat(tran, tran2); // we get wrong transform matrix by multiply
```

Otherwise, the correct matrix `tran` is used to transform the solid.
temp1 = temp1.rotate(2, (-yangle)); // rotation matrix about Y for –yangle degree
ro = new VecDouble(xangle,yangle,angle);
sc = new VecDouble(1.0,1.0,1.0);
tran = tran.transformnew(tr, sc, ro); // correct transform matrix after four steps
tran = tran.mult_mat(tran,temp1); // correct transform matrix after five steps

Then either the wrong transformed solid or the correct transformed solid is
drawn, some strings are drawn, and some other similar procedures are done.

4.2.3.12 Step 6 : Inverse of Step 2 ( rotate back about X axis)

After the user finishes the fifth step successfully, they should enter the back
rotation matrix about X axis into the text fields and then click the button “ Step 6 ” to
call Work6() method to check the results. It is similar to Work20(), except that when
transforming the solid, if the matrix that the user enter is wrong, matrix tran1 is obtained
by using the wrong matrix that users enter:

```
temp1 = temp1.rotate(2, (-yangle)); // temp1 is a rotation matrix about Y axis for
// degree –yangle.
ro = new VecDouble(xangle,yangle,angle);
sc = new VecDouble(1.0,1.0,1.0);
tran = tran.transformnew(tr, sc, ro); // correct transform matrix after 4 steps
tran = tran.mult_mat(tran,temp1); // correct transform matrix after 5 steps
tran2 = new Matrix(av11,av12,av13,av14,av21,av22,av23,av24,
av31,av32,av33,av34,av41,av42,av43,av44);
tran1 = tran1.mult_mat(tran,tran2); // wrong transform matrix after multiplying
```
// the wrong matrix that users enter.

Otherwise, the correct matrix tran is used to transform the solid.

\[
\text{temp1} = \text{temp1.rotate}(2, (-yangle)); \quad \text{// rotation matrix about Y for } -yangle \text{ degree}
\]

\[
\text{temp2} = \text{temp2.rotate}(1, (-xangle)); \quad \text{// rotation matrix about X for } -xangle \text{ degree}
\]

\[
\text{ro} = \text{new VecDouble(xangle,yangle,angle)};
\]

\[
\text{sc} = \text{new VecDouble(1.0,1.0,1.0)};
\]

\[
\text{tran} = \text{tran.transformnew(tr, sc, ro); } \quad \text{// correct transform matrix after 4 steps}
\]

\[
\text{tran} = \text{tran.mult_mat(tran,temp1);} \quad \text{// correct transform matrix after 5 steps}
\]

\[
\text{tran} = \text{tran.mult_mat(tran,temp2);} \quad \text{// correct transform matrix after 6 steps}
\]

Then either the wrong transformed solid or the correct transformed solid is drawn, some strings are drawn, and some other similar procedures was done.

4.2.3.13 Step 7: Inverse of Step 1 (translate back the solid)

After the user finishes the sixth step successfully, they should enter the back rotation matrix about X axis into the text fields and then click the button "Step 7" to call WorkZ() method to check the results. It is similar to the Work2(), except that when transforming the solid, if the matrix that the user enter is wrong, matrix tranl is obtained by using the wrong matrix that users enter:

\[
\text{temp1} = \text{temp1.rotate}(2, (-yangle)); \quad \text{// rotation matrix about Y for } -yangle \text{ degree}
\]

\[
\text{temp2} = \text{temp2.rotate}(1, (-xangle)); \quad \text{// rotation matrix about X for } -xangle \text{ degree}
\]

\[
\text{ro} = \text{new VecDouble(xangle,yangle,angle)};
\]

\[
\text{sc} = \text{new VecDouble(1.0,1.0,1.0)};
\]

\[
\text{tran} = \text{tran.transformnew(tr, sc, ro); } \quad \text{// correct transform matrix after 4 steps}
\]
\[
\text{tran} = \text{tran.mulf_mat(tran, temp1)}; \quad \text{// correct transform matrix after 5 steps}
\]
\[
\text{tran} = \text{tran.mulf_mat(tran, temp2)}; \quad \text{// correct transform matrix after 6 steps}
\]
\[
\text{tran2} = \text{new Matrix(av11, av12, av13, av14, av21, av22, av23, av24,}
\]
\[
av31, av32, av33, av34, av41, av42, av43, av44);
\]
\[
\text{tran1} = \text{tran1.mulf_mat(tran, tran2)}; \quad \text{// wrong transform matrix after multiplying}
\]
\[
\quad \text{// the wrong matrix users enter}
\]

Otherwise, the correct matrix \( \text{tran} \) is used to transform the solid.

\[
\text{temp1} = \text{temp1.rotate(2, (-yangle))}; \quad \text{// rotation matrix about Y for \(-\)yangle degree}
\]
\[
\text{temp2} = \text{temp2.rotate(1, (-xangle))}; \quad \text{// rotation matrix about X for \(-\)xangle degree}
\]
\[
\text{ro} = \text{new VecDouble(xangle, yangle, angle)};
\]
\[
\text{temp1} = \text{temp1.rotate(2, (-yangle))}; \quad \text{// rotation matrix about Y for \(-\)yangle degree}
\]
\[
\text{temp2} = \text{temp2.rotate(1, (-xangle))}; \quad \text{// rotation matrix about X for \(-\)xangle degree}
\]
\[
\text{trback} = \text{new VecDouble((-x1), (-y1), (-z1))};
\]
\[
\text{temp3} = \text{temp3.translate(trback)}; \quad \text{// translation back matrix}
\]
\[
\text{ro} = \text{new VecDouble(xangle, yangle, angle)};
\]
\[
\text{sc} = \text{new VecDouble(1.0, 1.0, 1.0)};
\]
\[
\text{tran} = \text{tran.transformnew(tr, sc, ro)}; \quad \text{// correct transform matrix after 4 steps.}
\]
\[
\text{tran} = \text{tran.mulf_mat(tran, temp1)}; \quad \text{// correct transform matrix after 5 steps.}
\]
\[
\text{tran} = \text{tran.mulf_mat(tran, temp2)}; \quad \text{// correct transform matrix after 6 steps.}
\]
\[
\text{tran} = \text{tran.mulf_mat(tran, temp3)}; \quad \text{// correct transform matrix after 7 steps.}
\]

Then either the wrong transformed solid or the correct transformed solid is drawn, some strings are drawn, and some other similar procedures are done. This concludes the 3D interactive arbitrary rotation Java program.
CHAPTER 5
CONCLUSION AND FUTURE WORK

5.1 Conclusion:

The goal of this work was to develop a Web site in assisting instructors to teach students more efficiently and to assist students to understand the course material better. The Web site was developed for the course, which includes the notes for 10 chapters, GIF animations, Java 2-D and Java 3-D interactive programs, 3-D VRML models, many examples in solving the problems, voice and video files, and a class syllabus. The following graphics are some copies from the web site. See Figure 5.1 for the front page.

![Figure 5.1: The front page of the Web site](image-url)
The basic page for most of the pages in this Web site is set up by HTML (See Figure 5.2). At the bottom right of this page, there is a graph, which links to the instructor's voice file. There are also a link to a VRML file at the bottom of this page, and links to other pages to this chapter.

Figure 5.2: A page with voice file in chapter 2.
The next figure is an example of a regular page for Chapter 8. In this page (See Figure 5.3), a square-threaded screw is drawn by Paint software and is saved as a BMP file. By using Microsoft Photo Editor the BMP file is saved as GIF file, which can be used directly in HTML code. As mentioned before, with Paint not only sentences, but also graphics can be drawn very easily in the same figure.
For problem solving in Chapter 2, users can check their answers with the “Right Result” link or they can receive hints and solve the problem step-by-step (See Figure 5.4).

Figure 5.4 : An example for problem solving in Chapter 2
The next figure (See Figure 5.5) is Page 5 of Chapter 1. In this page, a link to a GIF animation will show the principle of transmissibility when the button at the bottom is clicked.

Figure 5.5 : The page linked to GIF animation
Next figure is the Java applet for 2-D vector operation (See Figure 5.6). When the coordinate values are entered wrong, the program will show the wrong vector after the button “Your answer” is clicked. The user can try again.

![2-D interactive vector Java program.](image)

Figure 5.6 : 2-D interactive vector Java program.
When the button “Correct answer” is clicked (See Figure 5.7), the correct result is displayed graphically. If the results are not the same as this result, users should try again.

![Figure 5.7: 2-D interactive vector Java program](image-url)
The next figure (See Figure 5.8) shows the initial solid for a 3-D interactive rotation model after the button “Initial” is clicked. The user must enter a value into “Step numbers” text field and click the button “Check Step numbers” to check if the total steps estimate are correct. When the correct step number is obtained, the user can proceed. The coordinates of two points, which decide the rotation axis must be entered (or the user can use the default values that are generated randomly). Also one must enter the rotation angle about that axis (or use the default angle that is generated randomly).

Figure 5.8: 3-D interactive rotation model by Java (initial solid)
The first step is to enter the translation matrix into the right-hand side matrix text fields (See Figure 5.9). If any value of the matrix is wrong, the program will draw the incorrect translated solid and indicate that it is wrong when the button “Step 1” is clicked. The user will need to try again in order to start to work on step 2.

Figure 5.9 : 3-D interactive model by Java
If the user has tried more than three times and still can’t achieve the correct result, the program will give the correct matrix for each step (See Figure 5.10).

![3-D interactive rotation model by Java](image)

*Figure 5.10: 3-D interactive rotation model by Java*
When the matrix is entered correct, the correctly translated solid will be drawn. Then the user can enter the rotation matrix about X axis, and click the button “Step 2” to check the results (See Figure 5.11).

Figure 5.11 : 3-D interactive rotation Java model
Steps 2 through 7 are similar to step 1 except that the matrix values and the transforming graphic results are different for each step. The next figure is the last one after the user finishes all the seven steps successfully for this Java interactive model (See Figure 5.12).

Figure 5.12 : 3-D interactive rotation Java model
5.2 Future work

This Web site is still a work in progress, with more technology features to be added as the program develops. Each step taken thus far has brought new insights into the complex process for making the transition from traditional classroom instruction to the Web environment for course delivery.

The World Wide Web provides valuable new avenues for education. A glimpse of the opportunities it affords is beginning to emerge. Taking advantage of these opportunities requires considerable effort on the part of those who participate. Using the Web is time-consuming and labor-intensive, if productive outcomes are to be derived. Students and faculty may find valuable resources and increased opportunities in communication through the Web, but at the expense of continuous effort and time consumption. Establishing an interactive and dynamic Web site course can help overcome time consumption difficulty, while providing students with quick and convenient ways to find useful information. Web pages create a much more interactive learning environment thereby increasing the effectiveness of learning.

A well-designed Web site course provides the balance of real and virtual classrooms and class sessions. This ideally makes the class a more continuous environment rather than an environment, which is done in one or two hours and then set aside for the remainder of the week.

As noted, this is a good Web site course for both instructors and students to use for the course “Statics”. Nevertheless, more work can be done in the future. For example, more interactive examples need to be implemented by Java for students to use the site more frequently and conveniently in the future.
More graphical examples for problem solving should be added to the site. Normally, examples of problem solving are of much more benefit to students in completely understanding concepts and lectures than just reading notes or books.
BIBLIOGRAPHY


