Electrodynamics Simulator
Group #4

Senior Design Final Report

Due 4/28/06

Name	Degree	Role	Contact
Zipporah Fleming	EE	EM fields research Mathematician Output verification	zffleming@fsu.edu
Franklin Pulido	CpE	Visualizations engineering Software engineering	fjp03@fsu.edu
Rene Sanchez	CpE	Team manager Window form designer Output verification coding	rfs4591@fsu.edu
Chris Taylor	EE	Initial state physics coding Next state physics coding Volume rendering coding	cat02e@fsu.edu
Michael P. Frank	PhD	Faculty advisor	mfrank@fsu.edu

Executive Summary

The electrodynamics simulator was designed to be a visualization tool for arbitrary time-varying electromagnetic fields in the time domain in free space. The finite-element, finite-difference discrete simulation of Maxwell's equations had to be reversible. It had to be able to visualize the electromagnetic field lines and the intensities of the field in the space. It was required to work in real time. It had to be extensible so that programmers would be able to simulate any kinds of fields using the code. Users had to be able to speed up, slow down, and pause the simulation. They should be able to fly through the space and also rotate around it.

Our solution implemented Maxwell's equations in a reversible manner, acting on a 3d matrix representing the simulation space. The matrix elements are all integers, allowing fast updates of large matrices and a simple reversible rule for updating forward or backward in time. In free space, the update rule did not have to take into account the permeability and permittivity of each position in the space. There is a slowdown factor in the physics engine that allows the user to adjust the speed of the simulation. Free-motion and trackball cameras were designed to view the scene. We offer two types of visualizations, based on a fast OpenGL render engine.

All of these features are accessible through an intuitive wxWidgets-based user interface, and all features were coded by us in C++ so that we have complete control over our source code licensing.

Our original project budget analysis indicated that we were not going to be spending much money to meet our design goals. This turned out to be accurate, but we have prepared a complete list of all budgetary items. Furthermore, in chapter 9 "Product Cost and Commercialization" we have run budgetary analysis on how much the product would cost to market as a professional software package.

Our original project deliverables schedule was predictably inaccurate. However through this project we learned a lot about project management and the lessons we learned are reported throughout this document. The Statement of Work section will cover iterative steps taken during the design, based on constraints. The Schedule and Work Breakdown section covers the timeline we were working within. The System-level Design and Detail-level Design documents completely describe the design. The design constraints we encountered are described in chapter 7. To be sure we delivered a working simulator, its functionality was verified. Our test cases and results are presented in the Design Verification chapter.

Table of Contents

Chapter 1 Introduction

Trying to mentally visualize an electromagnetic field is a difficult task. For students, the best visualization aids available are often two-dimensional textbook pictures and the occasional MATLAB program to generate a few slices of a much larger 3D picture. Our project, the Electrodynamics Simulator, not only allows for a fully-interactive 3D visualization of a static field, but also dynamic fields. The user is able to view not only the field strength but also the field direction at each point in a 3D space. They are able to view the space from any angle and adjust properties of the simulation to instantly see the results. It has potential to be an excellent teaching tool for electrodynamics systems, similar to Mefisto, but with the potential to be more useful for research than Mefisto, as it is extensible.

The software is designed to be easy to use but also powerful. It contains many of the niceties of good user interface design, such as a greeter window and a toolbar for common functions and color-coded objects in the visualization (blue for E-field, red for B-field). It is powerful because it has the potential to simulate any arbitrary free-space electromagnetic field configuration using a discrete reversible leap-frog updating rule. However for this project, the only initial field state configuration is for a rectangular waveguide. It allows the user to simulate not only waves propagating down a waveguide at the critical frequency for the waveguide, but also waves that would not propagate at all.
The problem statement, from Dr. Frank our faculty advisor:

> "The electrodynamics simulation and 3D visualization engine is a well-designed (ie. extensible), discrete finite-element time-domain simulator for the Maxwell's equations in free space (bounded or periodic), together with an animated 3D visualization interface for rendering the fields, either as E & B fields, or in terms of vector and scalar (A & V) potentials. The simulation should be dynamically rendered, and the user should be able to speed up, slow down, pause, reverse, fly through, and rotate the simulated system. This tool would be very useful for educational and pedagogical purposes, and would also be a starting point for more advanced multi-domain physical simulators to be used in ongoing device physics research. Knowledge of electromagnetic field theory, finite-element modeling, finite-difference numerical techniques for solving differential equations, computational geometry, graphics programming, 3D rendering, user-interface design, and physics engine design are all necessary to complete this project."

For us, this means that we will be writing software to be run on PCs that will simulate the electromagnetic fields by discretizing the problem. The result of the simulation will be presented in 3D, and it is desired to be very interactive. So, the presentation of the fields as they change over time is one half of the project, and the other half is the simulation it is based on. All components need to be written well, so that this simulator may be used as a stepping stone for future work, perhaps by another Senior Design project team. This program will only support a few modes, just for testing its functionality, but it should be extensible enough so other people can easily design their own simulations by changing the source code.

Chapter 2 Statement of Work

2.1 Problem Statement

From Dr. Frank's description shown in the introduction, it's clear that our program MUST:

- Electrodynamics simulation
- E-field and B-field
- Real-time 3D visualization engine
- Extensible
- Discrete, finite-element, time-domain
- Maxwell's equations, free space, bounded or periodic
- User time controls: speed up, slow down, pause, reverse
- User space controls: fly through, rotate
- Should be useful as an educational tool

2.2 Design Process

Task 1: Gather all necessary information relating electromagnetic field theory

- **Subtask 1:** Gather information about field theory from University libraries

  Zipporah found that some college textbooks (in our References section) contained the equations we needed to create the initial state generator for a wave propagating in a waveguide.

  Initially we had researched plane waves, but gave up on those because they only have 2D solutions and we wanted to fill the whole 3D space as an example application for our simulator.

- **Subtask 2:** Gather information from Internet research

  We found some nice examples online for 3D electrodynamics simulators, but the textbooks were more valuable for explaining what a waveguide was and how to simulate it.

- **Subtask 3:** Summarize all information collected

  Armed with the equations from the textbook, we implemented C++ code that generated a state of the wave in the waveguide at time t = 0 for the initial state generator.

Task 2: Research current visualization technologies available in the market

- **Subtask 1:** Gather information about 3D visualization from university libraries

  Before looking into this, we did Internet research and found what we needed.

- **Subtask 2:** Gather information from Internet research

  We had decided to use OpenGL, and Franklin knew of a website (nehe.gamedev.net) that explained how to create many different effects with OpenGL. The rendering code we ended up with was entirely original, but that site was excellent for learning the library functions needed to create the rendering code.

  We found some animated 3D visualizations of electrostatic systems online.

- **Subtask 3:** Summarize all information collected

  Dr. Frank had specified that the camera should rotate and fly, so we came up with two camera styles, the Free camera (flies) and the Trackball camera (rotates).

  We decided to have a separate visualization for intensity using a technique called Volume Rendering that some of us had heard of before. By searching around, a good reference (astronomy.swin.edu.au) was found on how to do it.

  In addition to representing intensity we wanted to represent field direction, but the space had too much resolution to draw an arrow at each cell. One of the animated 3D visualizations of electrostatic systems that we saw was based on test charges that would move in the field. We did not use this approach, but realized that we could represent the field lines by making points move along the lines, and having a lot of randomly-placed points scattered about the space. Since the magnetic field lines curve back on themselves, the points should whirl around in loops. The color of the points could be used to indicate the field strength the point is experiencing.

Task 3: Build physics engine

- **Subtask 1:** Create the algorithms for the fine-grained dynamical evolution of arbitrary electromagnetic field configurations.

  These turned out to be very simple and we only updated this once to include a slowdown factor so that the simulation isn't over in a blink of the eye.

- **Subtask 2:** Create the algorithms for the different regions of varying permeability and permittivity.

  We decided to save this as an optional task because it would affect many parts of the project. It ended up not being included.

- **Subtask 3:** Create the algorithms for calculating the magnetic field and the electric field.

  These equations were easy to create, based on the equations from the textbooks.

- **Subtask 4:** Create all the functions for the different calculations and algorithms.

  Initially we were going to have an InitialStateGenerator class and a NextStateGenerator class that would be used by the Driver program, but we ended up integrating both of them into a single PhysicsEngine class that takes fewer source files, fewer lines of code, less time to execute, and is easier to understand.

  The final design is a PhysicsEngine class with members for the different calculations and algorithms involved.

- **Subtask 5:** Update the functions so they also work backwards.

  This turned out to be so simple we did it from the start.

- **Subtask 6:** Make/update the functions so that they update and refresh themselves.

  This ended up being part of the implementation of the PhysicsEngine class, instead of the Driver program.

- **Subtask 7:** Compile all the functions and build the physics engine.

  Ditto.

- **Subtask 8:** Test the individual modules and the engine.

  We saved testing until near the end of the project, and the results are in the Design Verification section of this document (Chapter 8).

Task 4: Build the rendering engine

- **Subtask 1:** Create cameras to view the scene.

  We initially were going to make a camera that would be able to fly through the scene much like a space simulator game. After that was done, the controls seemed clunky for just quickly looking at the visualization from different angles. Chris set out to research how to do a "trackball" camera that rotates around the origin when the mouse is dragged. It turned out to be complicated, but fortunately a website (nehe.gamedev.net) that we had already visited had a tutorial on how to accomplish that effect.

- **Subtask 2:** Create a volume rendering visualization.

  Based on the Internet reference mentioned earlier, implementing it took longer than understanding it. It went through a few revisions because at first we were not sure whether we had to render triangles from back to front or in any order. In OpenGL, objects rendered first are overwritten by objects rendered later, so the first version of the volume rendering visualization only worked half the time. This was quickly corrected, and the only other major change was made when we decided to support rectangular prism regions instead of just cubes.

- **Subtask 3:** Create a point sprite-based visualization.

  At first we were unsure of how many sprites to render, how large they should be, whether they should scale with the size of the canvas, what color they should be, how fast they should move, etc. After the visualization started working, we adjusted the parameters until things looked right, over the course of the rest of the project, on and off.

- **Subtask 4:** Create axes and a grid lattice.

  The grid lattice show visually where the cells of the matrix are in the virtual space, as well as indicates whether boundaries are periodic or bounded by color.

  At first the axes were planned to be based on solid objects (cylinders and cones), but because this would have necessitated rendering objects in the order of their relative position to the camera (to render from back to front) it was easier to not do that and instead use vector art for the axes.

Task 5: Build the visual interface

- **Subtask 1:** Create the different windows for entering the information

  This seemed to be difficult until we found that there are free tools to design the windows visually. Rene used one called wxDesigner.

- **Subtask 2:** Create different windows for displaying information.

  Ditto.

- **Subtask 3:** Create a window for the project settings

  This went through a rewrite every time the initial state generator gained a new parameter but took only about an hour each time.

- **Subtask 4:** Create the camera view

  The selection between cameras was handled by the Visualization Canvas, designed to integrate OpenGL with the windowing system.

- **Subtask 5:** Design the controls for the camera

  The camera controls are a part of the Visualization Canvas, because it receives the key press events. The events are translated into event codes which are independent of which key was pressed. This way, two keys can activate the same event and key bindings can be reassigned later if that feature ends up being written.

Task 6: File system

- **Subtask 1:** Design the file format according to the requirements specifications

The requirements specification was very specific about how the format worked, so this did not go through revisions.

- **Subtask 2:** Design the interface to handle the input files
- **Subtask 3:** Design the interface to handle the output files

This was actually very easy, as wxWidgets had a function for displaying standard File Open and File Save dialog boxes.

Task 7: Integrate file system and visual interface

- **Subtask 1:** Integrate the file system to the visual interface

The callbacks from the windowing system call functions in the Driver program, which call the file system functions.

- **Subtask 2:** Test the visual interface and file system

This was easy; we just tried saving and loading files.

Task 8: Integrate the visual interface, the rendering and the physics engine

- **Subtask 1:** Compile and link all the code together

The VisualizationCanvas class implements a panel in the Windowing System that wraps the visualizations in a form that the Windowing System understands. The physics engine communicates with the rendering system through the matrix. All information needed to render is stored in the matrix, so the components do not need to communicate directly.

- **Subtask 2:** Create a probe that will be able to display the information of a specific point on the plane.

This was considered an optional feature and was not implemented.

- **Subtask 3:** Assign the different hot keys for calling their respective function.

The Requirements Specification gave some good defaults to use. Through testing those were revised to be easier to learn and use.

Task 9: Functionality test

- **Subtask 1:** Test the different components and the whole program.

The testing plan is discussed in chapter 8. It did not require revision for this part.

- **Subtask 2:** Compare the functionality and accuracy of our program to others such as Mefisto.

The initial state values and visualization were tested with Mefisto and MATLAB, and these results are in chapter 8. The testing plan was adequate, though we also wrote

a small test case program to test the initial state generator separately from the program.

Task 10: Wrapping up

- **Subtask 1:** Create a program installer

  The program ended up being a single executable, so this optional task never got done.

- **Subtask 2:** Set all the preset and colors

  We initially had all the field visualizations in green. However, rendering the E-field in blue and the B-field in red made it more clear which was being viewed.

- **Subtask 3:** Make the program more attractive to the user, if we can add buttons to the toolbars and/or menu bars

  We built a toolbar into the main window with pretty icons. The icons were revised later to look more professional and easier to understand. The background colors of the windows were set to white instead of grey so the text is more visible.

Task 11: Management

- **Subtask 1:** Check that everyone is meeting their deadlines

  Our initial estimates for the time things took was off by a week in some cases. The changes in deadlines are discussed in chapter 4.

- **Subtask 2:** Arrange the meetings

  Meetings were every Tuesday at 4:30 PM during the Spring semester in a dedicated meeting room.

- **Subtask 3:** Keep control of the budget

  We did not have many expenses, so people volunteered to pay for food and things like the posterboard for the Design Faire.

- **Subtask 4:** Revise all the documents before submitting

  We prepared working documents as the project progressed, which were revised as the parts they covered changed. No document went through more than 5 revisions.

2.4 Role of Members in Competing Tasks

Zipporah Fleming was in charge of researching the different kinds of waves, how are they supposed to look, calculate all the points in the matrix, verify that the calculations made by our program match these calculations and create the algorithms needed in order to calculate the E & B fields for those waves.

Chris Taylor was in charge of merging all the different components together, making sure that every part worked properly, he also designed and researched the math involved in the creation of the 3d engine. He was also in charge of making the axes, the volume rendering and fusing the interface system with the rendering using a visualization canvas.

Franklin Pulido was in charge of researching and designing the camera, lattice, point sprites and rendering system that were used in the project. He also designed how the file system worked.

Rene Sanchez was the project manager. He was in charge of coordinating the meetings, assigning tasks, evaluating the different options, checking the progress of the project, building the users interface, helping build the physics engine by testing the initial state generator by hand and through separate test case programs. He also integrated part of the users interface with the file system.

Table1 Task Assignments

Task	Task Description	Members
1	Research EM Fields	Zipporah
2	Research Visualizations	Franklin, Chris
3	Physics Engine	Zipporah, Rene, Chris
4	Rendering Engine	Franklin, Chris
5	User Interface	Rene, Chris
6	File System	Franklin
7	Integrate FS and UI	Rene
8	Integrate Physics and Rendering	Chris
9	Verification	Rene, Zipporah
10	Wrapping up	All
11	Management	Rene

Chapter 3

Project Budget & Justifications

3.1 Personnel Costs

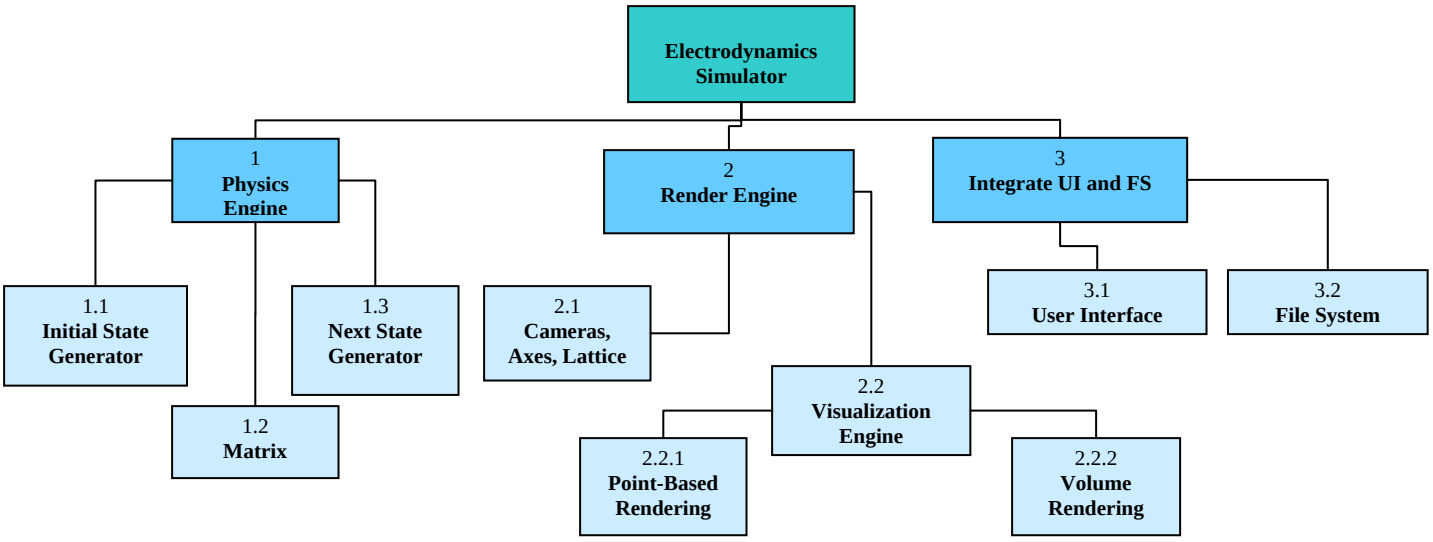
Name	Hourly Salary	Hours	Cost	Justification
Rene Sanchez	$50.00	160	$8000.00	Team manager Window form designer Output verification coding
Chris Taylor	$50.00	160	$8000.00	Initial state physics coding Next state physics coding Volume rendering coding
Dr. Michael Frank	$100.00	64	$6400.00	Consulting
Franklin Pulido Jr	$50.00	160	$8000.00	Visualizations engineering Software engineering
Zipporah Fleming	$50.00	160	$8000.00	EM fields research Mathematician Output verification
Total Personnel Cost			$38400.00	

3.2 Expenses

Item	Cost	Justification
Posterboard	$15.00	We needed a posterboard panel at the design faire
Projector rental	$10.00	Having a projector on the presentation day enabled people to see our project in action, without having to squint at a small screen
Food at meetings	~$120.00	By eating together we were able to work longer
wxDesigner license	$29.00	Rene used wxDesigner to create the look and feel of the windows of the project, quickly and easily
Total Expenses	$184.00	

Chapter 4

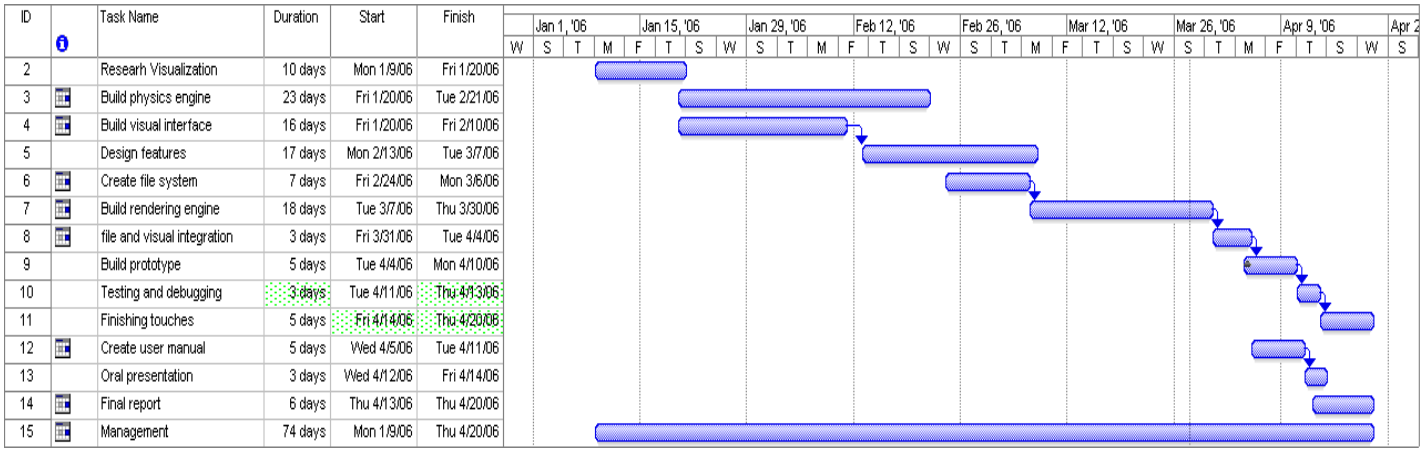
Schedule & Work Breakdown



ID	Task	Description	Effort	Skills	Dependency	Who
	Electrodynamics Simulator	Prototyping. Testing.	2d	Coding	All	All
1	**Physics Engine**	**Integer the Initial State Generator and the Next State Generator. Offer a clean interface to other modules for updating the Matrix based on real seconds passing.**	**18d**	**Coding, Fields**	**1.1 1.2 1.3**	**Chris Zipporah Rene**
1.1	Initial State Generator	Research the equations needed. Scale the physical units to fit within the matrix data.	10d	Coding, Fields	1.2	Zipporah Rene
1.2	Matrix	Create a reusable Matrix object.	2d	Coding	None	Chris
1.3	Next State Generator	Discretize Maxwell's Equations in a reversible way. Update quickly with few calculations in each step.	10d	Coding, Fields	1.3	Chris
2	**Render Engine**	**Integrate the Cameras, Axes, Grid lattice and visualizations into the VisualizationCanvas.**	**10d**	**Coding**	**2.1 2.2**	**Chris Franklin**

ID	Task	Description	Effort	Skills	Dependency	Who
2.1	Camera, Axes, Lattice	Research and design two camera styles: one for ease of use and one for utility. (Ended up being Trackball / Free camera styles) Design an axes overlay. Design a grid lattice to go around the active visualization.	8d	OpenGL	None	Chris Franklin
2.2	Visualization Engine	Integrate the two visualization styles so they can be switched between in real time and errors can be handled.	2d	OpenGL	2.2.1 2.2.2	Chris Franklin
2.2.1	Point-Based Rendering	Research and design a particle system that demonstrates the field lines in the 3D space.	16d	OpenGL	1.2	Franklin
2.2.2	Volume Rendering	Research and design a volume rendering system that displays field intensity in the 3D space.	16d	OpenGL	1.2	Chris
3.1	**Integrate UI and FS**	**Be able to go from disk to settings to disk with the UI.**	**7d**	**Coding**	**3.1.1 3.1.2**	**Rene Chris**
3.1.1	User Interface	Create all the windows specified in the Requirements Specification. Update the Project Settings window based on the Initial State Generator.	10d	wxWidgets	None	Rene Zipporah
3.1.2	File System	Offer classes to abstract file access to binary files and XML-based text files.	5d	Coding	None	Franklin Chris

Comparing our proposed Schedule with the breakdown, we found out that it actually changed a lot; also the team members that were assigned to each part had some minor changes. This were due because some resource allocation was needed and some unexpected parts had to be build. Also we didn't consider that some parts would require more time that it actually took and thus making the plans to change.



ID	Task Name	Duration	Start	Finish
2	Research Visualization	10 days	Mon 1/9/06	Fri 1/20/06
3	Build physics engine	23 days	Fri 1/20/06	Tue 2/21/06
4	Build visual interface	16 days	Fri 1/20/06	Fri 2/10/06
5	Design features	17 days	Mon 2/13/06	Tue 3/7/06
6	Create file system	7 days	Fri 2/24/06	Mon 3/6/06
7	Build rendering engine	18 days	Tue 3/7/06	Thu 3/30/06
8	file and visual integration	3 days	Fri 3/31/06	Tue 4/4/06
9	Build prototype	5 days	Tue 4/4/06	Mon 4/10/06
10	Testing and debugging	3 days	Tue 4/11/06	Thu 4/13/06
11	Finishing touches	5 days	Fri 4/14/06	Thu 4/20/06
12	Create user manual	5 days	Wed 4/5/06	Tue 4/11/06
13	Oral presentation	3 days	Wed 4/12/06	Fri 4/14/06
14	Final report	6 days	Thu 4/13/06	Thu 4/20/06
15	Management	74 days	Mon 1/9/06	Thu 4/20/06

Chapter 5 System-level Design

See attached document “System Level Design”. This document is created as an extended view of the interconnections and functionality of the main components of the Electrodynamics Simulator.

Chapter 6 Detailed-level Design

See attached document “Detail-level Design: How We Implemented Our Master Plan”. It covers how the all the major components were implemented.

Chapter 7 Design Constraints

7.1 Purpose

This section of the document will furnish the reader with all of the different design choices that were taken regarding code standards and physical limitations which impacted the different components and subsystems within the program. We will also explain some of the program's hardware and software needs, and the different supported field configurations.

7.2 Impacting Factors for the different system components

Listed below is a brief description of the crucial constraints of the main components of the simulation.

Display Devices- Because this project needs to be run on PCs, and the simulation would need to be outputted to the screen, and so a graphics card is needed to correctly run our tool.

Physics System- The physics engine is the source of all the mathematical modeling that takes place throughout the entire system. Because we have to adhere to real standards whole integers were used in the matrix system.

Rendering System- This system is the basis of all the graphical modeling which takes place within the program. And so, high demanding graphics are drawn on the screen which may not display properly in some computers that do not have a graphics card. And so, we had the limitation of rendering graphics that could be displayed in most computers.

7.3 Project Constraints

These are the realities of the project and, some of the limitations that were acknowledged ahead of time. And so, we had to design our project keeping these crucial factors in mind. The following is a list of constraints ordered according to their level of priority:

- Time constraints
- Economic constraints (matching costs with resources)
- Safety and environmental impact constraints
- Meeting engineering specifications and standards
- Catering to our primary and secondary markets
- Preferring to fulfill "needs" before "wants"

7.4 Time Constraints

Since the project needed to be completed by the end of the Spring 2006 semester a considerable amount of work needed to take place to successfully implement this ambitious project. Meaning that we worked on this project from Jan. 9, 2006 to April 20, 2006. That affords 124 days of effort including weekends, and only 60 weekdays. Meaning that we had about 240 man hours to spend on the whole project since everyone in average worked one hour every weekday.

7.5 Economic Constraints

In theory our project had no significant economic constraints. The simulation software that we used for comparison testing was MATLAB, and the College has a site license for it already. The graphics and miscellaneous libraries that we used are either free as in beer, or free as in speech. The College has a license for Visual Studio, which we will use. For presenting, Franklin Pulido borrowed for us a large projector, and the materials for poster board and paper are negligible. In real terms, we have produced a product worth money, for no more cost than our time in its construction.

We could consider our time worth money based on how much our part-time jobs are paying. As a result a complete budget was constructed this information is later exposed in Chapter nine.

7.6 Safety and Environmental Impact Constraints

The safety concerns are important for our project. It may be used some day to test if a microwave phase antenna array produces too much power at certain angles, and it might generate incorrect results. For this reason our testing was stringent, and in addition the user interface was built fool-proof to avoid user error. As a result, strict error checking was implemented and incremental testing was performed

7.7 Meeting Engineering Specifications and Standards

Based on the stated needs from nearly all our interviews, the product would be useful for teaching field theory and examining the effects of various field configurations over time. If the product is inaccurate, then it would not be suitable for these important uses. The IEEE standards for engineering conduct dictate we must do our best to prevent this from happening.

Best software engineering practices dictate that we write clear, maintainable, easy-to-debug code that may be verified with test cases. Furthermore because our project may be inherited by future classes, we have documented everything at all steps, using things such as engineering notebooks and central version management tools.

Chapter 8 Design Verification

This document provides evidence through various testing scenarios that the design output for the various system components meets or exceeds the requirements as specified in the requirement specification document. Within this section is a compilation of evidence that reveals how the various system components work as a stand alone program, in addition to providing feedback on their accuracy, as well as their ability to work together as a unit. This information is provided via the use of plots, error charts, in addition to graphical schematics generated to provide theoretical data for rendered graphic generation.

8.1 Physics Engine

TM Mode Field Equations

$$E_{zs} = E_0 \sin\left(\frac{m\pi x}{a}\right)\sin\left(\frac{n\pi y}{b}\right)e^{-j\beta z} \quad (1)$$

$$E_{ys} = \frac{-j\beta}{\beta_u^2-\beta^2}\left(\frac{n\pi}{b}\right)E_0 \sin\left(\frac{m\pi x}{a}\right)\cos\left(\frac{n\pi y}{b}\right)e^{-j\beta z} \quad (2)$$

$$H_{xs} = \frac{j\omega\epsilon}{\beta_u^2-\beta^2}\left(\frac{n\pi}{b}\right)E_0 \sin\left(\frac{m\pi x}{a}\right)\cos\left(\frac{n\pi y}{b}\right)e^{-j\beta z} \quad (3)$$

$$E_{xs} = \frac{-j\beta}{\beta_u^2-\beta^2}\left(\frac{m\pi}{a}\right)E_0 \cos\left(\frac{m\pi x}{a}\right)\sin\left(\frac{n\pi y}{b}\right)e^{-j\beta z} \quad (4)$$

$$H_{ys} = \frac{-j\omega\epsilon}{\beta_u^2-\beta^2}\left(\frac{m\pi}{a}\right)E_0 \cos\left(\frac{m\pi x}{a}\right)\sin\left(\frac{n\pi y}{b}\right)e^{-j\beta z} \quad (5)$$

Hz=0 (6)

TE Mode Equations

$$H_{zs} = H_0 \cos\left(\frac{m\pi x}{a}\right)\cos\left(\frac{n\pi y}{b}\right)e^{-j\beta z} \quad (7)$$

$$E_{ys} = \frac{-j\omega\mu}{\beta_u^2-\beta^2}\frac{m\pi}{a}H_0 \sin\left(\frac{m\pi x}{a}\right)\cos\left(\frac{n\pi y}{b}\right)e^{-j\beta z} \quad (8)$$

$$H_{xs} = \frac{-j\beta}{\beta_u^2-\beta^2}\frac{m\pi}{a}H_0 \sin\left(\frac{m\pi x}{a}\right)\cos\left(\frac{n\pi y}{b}\right)e^{-j\beta z} \quad (9)$$

$$E_{xs} = \frac{j\omega\mu}{\beta_u^2 - \beta^2}\frac{n\pi}{b}H_0 \cos\left(\frac{m\pi x}{a}\right)\sin\left(\frac{n\pi y}{b}\right)e^{-j\beta z} \quad (10)$$

$$H_{ys} = \frac{j\beta}{\beta_u^2 - \beta^2}\frac{n\pi}{b}H_0 \cos\left(\frac{m\pi x}{a}\right)\sin\left(\frac{n\pi y}{b}\right)e^{-j\beta z} \quad (11)$$

Ez=0 (12)

$$f_{c_{mn}} = \frac{1}{2\sqrt{\mu\epsilon}}\sqrt{\left(\frac{m}{a}\right)^2 + \left(\frac{n}{b}\right)^2} \quad (13)$$

Note: We replaced Ho A/m with the Bo Wb/m^2

$\beta = \sqrt{\beta u^2 - \beta x^2 - \beta y^2}$ (14)

β_X = mπ/a (15)

β_Y = nπ/b (16)

$\beta u = \omega\sqrt{\mu\varepsilon}$ (17)

f- frequency (Hz)
a- longest side of rectangle correlates to x axis
b- shorter side of the rectangle correlates to y axis
m- number of half wave variations in the x direction
y-number of half wave variation in the y direction
w- angular velocity (rad/s)
β - phase Constant
Fc- Cut-off frequency dependent on m and n

8.1.1 Physics Engine Verification Tests

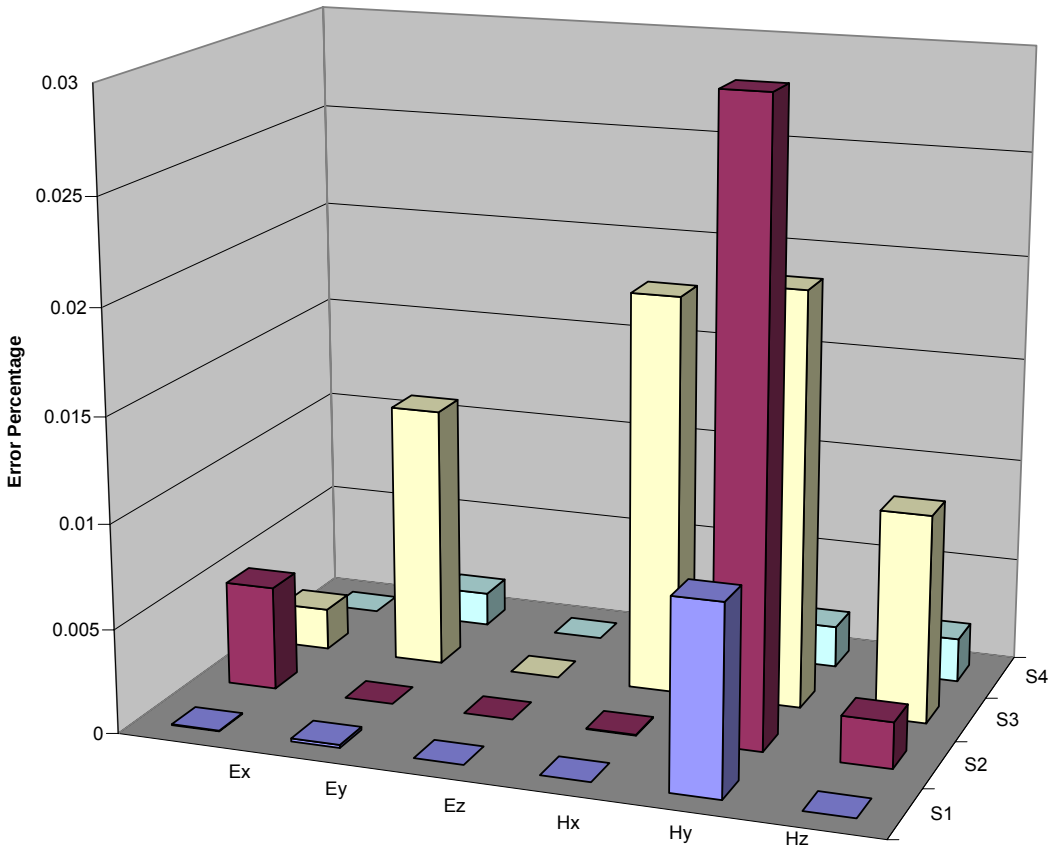
The Physic's Engine is the module within the program that calculates the various components within the rectangular waveguide, it is also in charge of determining various parameters that define the shape and cut off frequency of the waveguide. Therefore testing of this unit was imperative to the accuracy of the graphics rendering. This unit was tested by comparing results generated via the physics engine initial state generator, and was then compared to the theoretical results that were computed by hand using identical forms of input. Below is a chart that reveals the comparison of the experimental data versus the theoretical results.

***Error Equation*:** **E=**$(y - x)/y$

***y**- number value corresponding to the physics engine result calculation*
***x**- number value corresponding to the theoretical result*

Table 1. Physics Engine Error Calculations for the Theoretical Results Vs. Experimental Results for fields components of the waveguide				
Waveguide Component	Series 1 Error	Series 2 Error	Series 3 Error	Series 4 Error
Ex	0.000039	0.005	0.002	0
Ey	0.000144	0	0.0126	0.001624
Ez	0	0	0	0
Hx	0	0.0000444	0.0192	0.0006
Hy	0.00904	0.03	0.02	0.002
Hz	0	0.002213549	0.01	0.0021

Chart 1: Physic's Engine Initial State Expected Results vs. Theoretical Results



S1- denotes series 1 testing parameters as specified in table below
S2- denotes series 2 testing parameters as specified in table below
S3- denotes series 3 testing parameters as specified in table below
S4- denotes series 4 testing parameters as specified in table below

Taken at time t=0

Table 2. Parameter Values Used for Each Testing Series				
VARIABLES	S1 TM	S2 TE	S3 TM	S4 TE
f	100e6 Hz	100e6 Hz	1e9 Hz	1e9 Hz
a	5 m	5 m	5 m	6
b	2.5 m	2.5 m	6 m	10
m	1	1	1	1
n	1	1	1	1
Bo	N/A	1 Wb/m^2		.001 Wb/m^2
Eo	1 V/M	N/A	1 V/M	N/A

8.1.2 Explanation of Results

The error calculation results reveal a maximum error of 3 percent when comparing the theoretical result values to that of the physics engine. This error occurs in series two of the testing cycle, and correlates to the Hy component of the magnetic field while in TE mode. Through reviewing equation (11) used in generation of this particular result, the error is likely to have occurred via concerns with significant figures used in the various computational techniques, one being the computer generated the solution, the second being the solution generated through the use of Texas Instruments, TI-89 Titanium calculator. This can be revealed through analysis of the constants being used in the calculation of this particular variable, for instance the variable B_U is used several times in the computation for this particular component of the Magnetic field, within just this one variable several constants are used; pi ~3.14159625), uo~4pi x 10^7 H/m, and the constant εo~8.854x 10^-12 F/m. Within the constant $B/(B_U\text{^}2-B\text{^}2)$ pi is used several times as well, this can be revealed through analysis of the equations 14-17. As more constants were calculated are used in parallel with high frequency the error reveals a trend of increasing accordingly. The remaining error values were within a reasonable range in comparison to the theoretical results. In efforts to improve the error percentages in future calculations, more significant digits will be implemented within this system module.

The series containing the largest overall error is series three, this results from the use of an extremely large frequency being used, in parallel to the m, n, a, b, and amplitude values being used for this particular testing cycle. These factors only enhance the difference values that were previously carried as a result of significant figures. Despite this fact, the overall result of error for this particular series was only an average of approximately .01063 which is about 1.06 percent. This is considerably small, it appears to be within a reasonable error threshold.

8.2 Rendering System

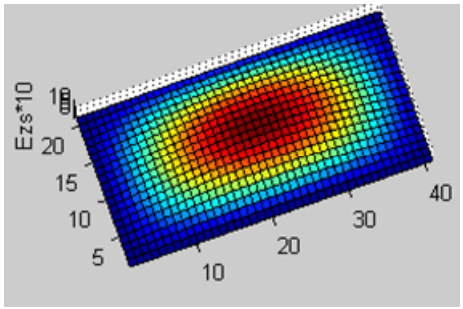
The rendering system is the program module that generates the graphics displayed to the user. These graphics are based on the information that is transferred from the physics engine. Therefore after verifying the physics engine generates the correct output to a certain degree; we then test the visual effects that are displayed to the user. The way in which this test is carried out is through the use and implementation of various test statistics in MATLAB. We create equivalent parameter scenarios in both our program environment as well as the of the MATLAB program, this then allows us to determine that the results are similar as far as the total number of half wave variations on either the y or x axis.

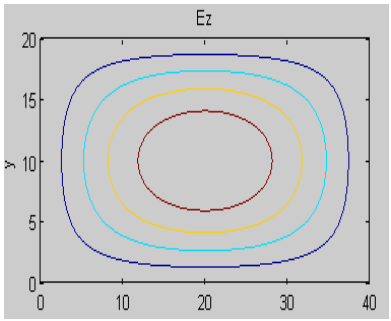
8.2.1 Rendering System Verification Tests

Testing presents schematic generated in MATLAB and compares this data to the rendered graphics in the Electrodynamics Simulator. This comparison enables verification of the rendering engine, in addition to providing evidence that this modules works properly in parallel with the physics engine.

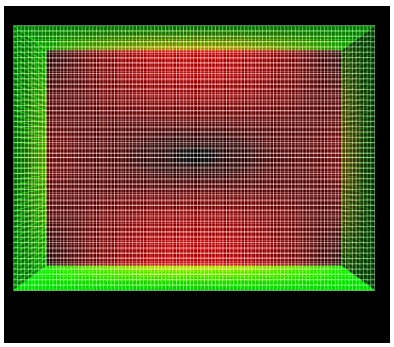
----------Testing Series 1----------

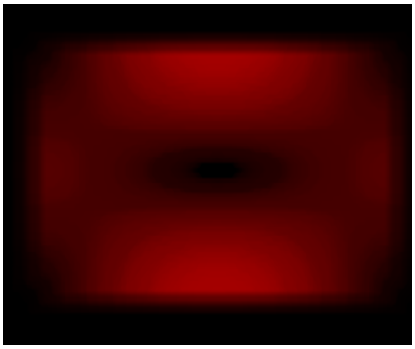
% Initialize variables
m=1;n=1;
a=40;b=20;



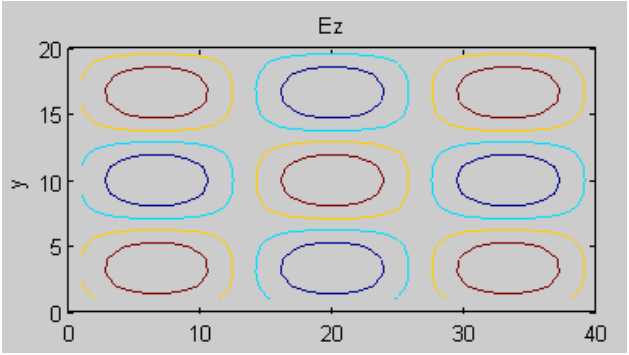


Electrodynamic Simulator Generated Graphics
m=1
n=1
a/b= .5

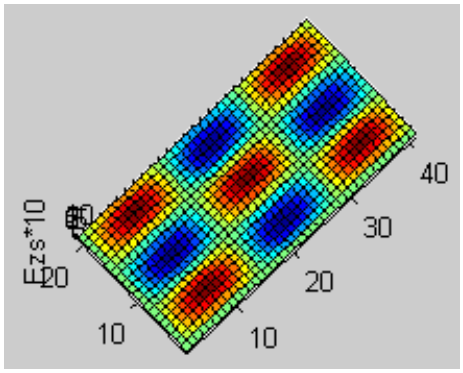




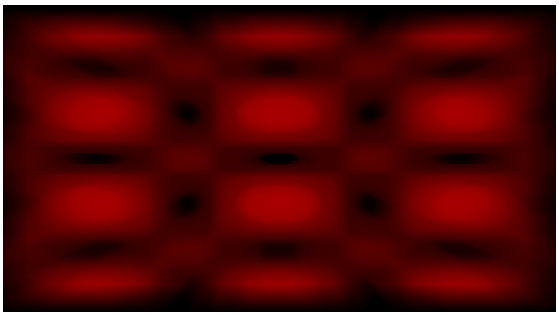
----------Testing Series 2----------

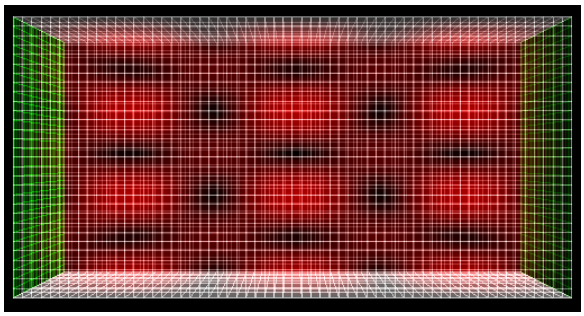


Matlab Generation Results
% Initialize variables
m=3;n=3;
a=40
b=20



Electrodynamic Simulator Generated Graphics
m=3
n=5
a/b= .5

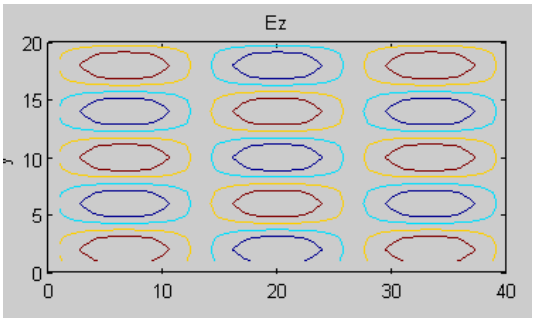


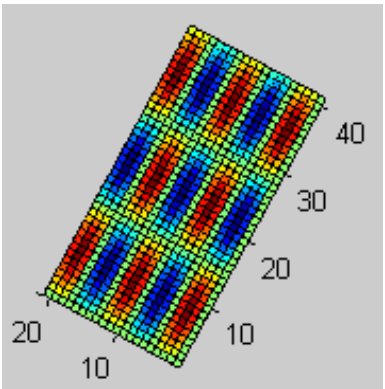


----------Testing Series 3----------

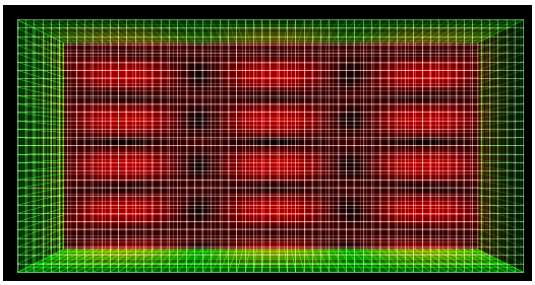
Matlab Graphics Generation

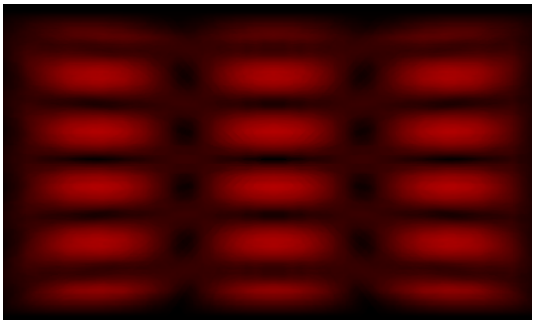
```
%   Initialize variables
m=3;n=5;
a=40;b=20;
```





Electrodynamics Simulator Generated Graphics
m=3
n=5
a/b=.5

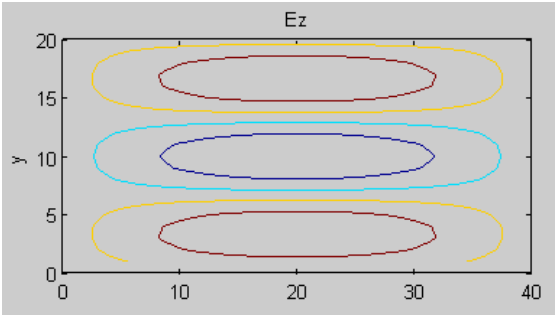


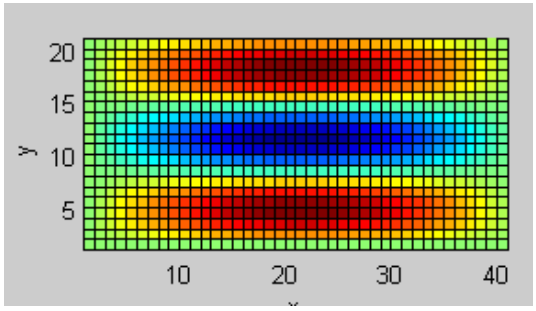


----------Testing Series 4----------

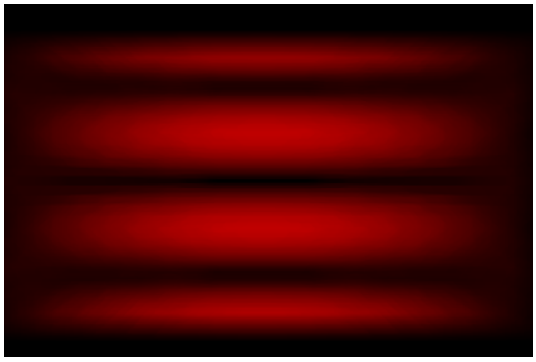
Matlab Graphic Generation

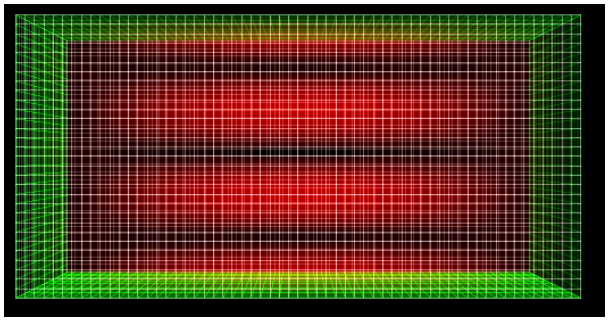
```
%  Initialize variables
m=1;n=3;
a=40;b=20;
```





Electrodynamics Simulator Generated Graphics
m=1
n=3
a/b=.5





8.2.2 Explanation of Rendering System Test Results

The testing carried out for rendering was conducted in several ways. The first form of testing was carried out via user observation and verification. After carefully studying the behavior of the rectangular wave guides, in addition to any variations that occur as a result of manipulation of the m and n values corresponding to half-wave variations on either the x or y axis, each member of the group conducted a test comparing the hypothesized

visualization from prior experience with waveguides, to the graphics rendered via the rendering engine module in the electrodynamics simulation engine. In each conduction of the individual testing neither of the teams members reported errors in the visualizations that were rendered.

Although, the individualized testing went extremely well, in order to better verify the results generated, testing was also carried out using a MATLAB program provided in Wentworths, “Fundamentals of Electromagnetic”, this program generates a rectangular waveguide plot in the Z direction. In order to properly compare the results generated via the use of this program to those generated through incorporation of the code use within the rendering system, the view was always taken from the Z axis, although it is completely functional from all perspectives. Through analysis of the results which can be viewed in section 2.2 of this document, there was never an instance were a discrepancy occurs in the graphics generated in the rendering module. The MATLAB program provides ample verification that this module is completely functional in addition to revealing the driver and the physics engine are also working correctly in a parallel to this system component. The accuracy of the results revealed there was no error in all test trials that were carried out.

8.3 Windowing System

The windowing system was the first module created within the system as a whole. This module allows interaction to occur between the user and the internal actions that take place within the driver program, the physics engine, in addition to the graphics that are rendered thereafter. Testing of this system was carried out through the creation of message box pop up menus. The way this particular testing was carried out involved the user selecting certain options from the available menu i.e “Stop”, “Forward” or “Backwards”. Based on the option selected by the user a pop up box comes up, emulating the action to be carried out in the event that “Stop” is selected a box displaying the phrase pops up to verify the function is working. If a value was entered within the testing phase being processed, the pop up also displayed the value that was entered by the user. Through the implementation of this technique, all menu items and actions were verified as being completely functional.

In addition to this testing technique, each individual ran the program several times to verify that all functions produced the proper protocols. Through this method identification of bugs were easily identified and corrected accordingly. This particular method of testing was conducted at each progressive stage of the programming process in order to eliminate major functionality problems when accessing the final prototype. Upon the final testing all parts were verified as being functional.

Chapter 9

Product Cost & Commercialization

This document reveals the manufacturing costs associated with the production, packaging and labor hours required to complete full design implementation of the electrodynamic's simulation engine as specified by the customer. Based on these estimates a reasonable consumer price has been tabulated for commercial use and manufacturing. These estimates are based on average cost analysis for software, advertising, and labor. Included are all necessary transactions aiding in the process of transforming this product into an item available for commerce.

9.1 Product and Manufacturing Costs

Below is a brief description of estimated costs incorporated to manufacture this product for commercial use. In order to determine overall costs for the project the following areas were accessed: Labor, Manual and documentation, software, advertising, and distribution.

Budget Item	Cost Per Unit/Hour ($)	Units/Hours Required	Total Cost (x1000 $)
Manufacturing Costs			
Labor	100	240	60
Manufacturing costs			
Manual & documentation	1	10000	10
Software Distribution (CD)	.15	8000	1.2
Packaging	1	8000	8
Software Distribution (internet)*	3000	1	3
Support for users	20	1560	31.2
Advertising/Promotion			
Journal advertising spots	400	10	4.0
Internet banner ads and a website	80	4	0.32
		Subtotal	117.72

*this cost is associated with the cost of web hosting and bandwidth. This is an alternative to packaging.

9.2 Consumers

This product was created in order to promote a better understanding of the characteristics and behavior of electromagnetic fields. It was implemented with the student use in mind. The target market is geared toward academia. Teachers, researchers and students should find this product extremely useful.
It should serve as learning and as well as a researching tool. In order to better promote this tool, a product demo will be available free online for student use.

Chapter 10 Conclusion

The electromagnetic simulator is a basic program, which is meant to grow. This tool is designed for the use of students, teachers and as well as researchers as a stepping stone in order to create visualizations in 3 dimensions of electromagnetic fields. In order to achieve this we had to spend a lot of time inside the design room because not many of us, if some at all had much experience in the areas required in order building this tool. Fortunately we counted with the guidance of an excellent advisor who would guide us for every step of the making of this tool.

In order to design something we first need to know what we are designing. So our first step was to go ahead and hear what was this project about, create a list of needs for it and based on those needs then develop a requirement specification. After building the requirements specifications we had an idea of what areas we had to start researching in order to do a solid program. After compiling all this information we sat down and did the top level design which consists primarily of 3 things: a Physics engine, a Rendering engine and an interface between the user and the program. After evaluating each member of the team strengths we went ahead and decided to divide these main parts into many smaller parts and thus the Detailed Level Design document was created. Having this done now we went ahead and started building the tool.

Since we didn't have to build hardware, our product costs were near zero so most of our budget is a virtual budget. Still the practice for it was good; hopefully we got a good approximation from it.

As you can see in our finished product we have a working tool that meet most, if not all the requirements that we thought were possible at the beginning of the semester. The only things that where left out where some of the optional features because we couldn't finish on time. But as you can see anyone can keep adding features to our tool. The basics are there you just need to improve it. The way the program is made is in modules, an experienced programmer can go ahead and go into our physics engine and add other initial states wave forms such as spherical & cylindrical waveforms, dialectics waves and some other objects inside the simulation area with different properties such as reflection and absorption. Who know even one day this tool will be able to simulate very complex products such as micro-circuits or antennas.

Chapter 11 References

[1] Ian Sommerville, Software Engineering, 6th ed., Essex, England: Pearson, 2001, chapters 1-5, 22, 23.

[2] Karl T. Ulrich & Steven D. Eppinger, Product Design and development, 2nd ed., Burr Ridge, IL: McGraw-Hill, 2000, chapters 1-4,8,9,14.

[3] Pozar, David. Microwave Engineering. 3rd ed. Danvers,MA: Wiley and Sons, 2005. 113-126.

[4] Ulaby, Fawwaz T. Fundamentals of Applied Electromagnetics. Media ed. Prentice Hall, 2004. 294-301.

[5] Wentworth, Stuart M. Fundamentals of Electromagnetics with Engineering Applications. 1st ed. Hoboken,NJ: Wiley and Sons, 2005. 339-346.

[6] MEFISTO-3 D Pro. Faustus Scientific Corporation, 2001.

[7] Paul Bourke Hardware accelerated volume rendering, November 2003 http://astronomy.swin.edu.au/~pbourke/opengl/glvol/

[8] Jeff Molofee OpenGL tutorials, NeonHelium Productions!, http://nehe.gamedev.net/

Chapter 12 Appendices

Attached should be:

- The System-level Design document
- The Detail-level Design document
- The meeting minutes
- And the four progress status reports from the semester