Technology Enabled Active Learning: System Architecture & Implementation

by

Anup V Mantena

Bachelor of Technology, Civil and Environmental Engineering, Indian Institute of Technology, Madras 1999

Submitted to the Department of Civil and Environmental Engineering
in partial fulfillment of the requirements for the degree of

Master of Science

at the

Massachusetts Institute of Technology

September 2001

© Massachusetts Institute of Technology 2001. All rights reserved.
Technology Enabled Active Learning: System Architecture & Implementation

by

Anup V Mantena

Submitted to the Department of Civil and Environmental Engineering on August 17, 2001, in partial fulfillment of the requirements for the degree of Master of Science

Abstract

This thesis describes the design and implementation of a set of software tools as part of the Technology Enabled Active Learning (TEAL) / Studio project at MIT. These tools include a simulation package for creating electromagnetic simulations and a framework for collaboration software based on asynchronous messaging. We also describe Interactive Question Service (IQS). IQS is a tool built using the collaboration framework to gauge the student performance in realtime. The software tools described here should contribute to a better learning experience for the students.

Thesis Supervisor: John W. Belcher
Title: Professor

Thesis Reader: Steven R. Lerman
Title: Professor
Acknowledgments

I would like to thank Prof John Belcher for giving me an oppurtunity to work on the TEAL/Studio project and for his constant guidance and support that made this thesis possible. I would also like to express my gratitude to Prof Steven R Lerman for his invaluable advice and guidance he offered me over the last two years.

I would like to take this oppurtunity to thank Andrew Mckinney and Philip Bailey for the tremendous amount of help, advice and patience that they gave me over the entire duration. Thanks to all the RA’s and staff of CECI for making CECI such a wonderful place to work.

My thanks to my family for all the love, affection, encouragement and wonderful moments they shared with me over the years.

And last but not the least I would like to thank Eswar, Bharath, Sugata, Sahoo and all other friends for making my stay at MIT a memorable experience.
## Contents

1 Introduction ........................................ 9
  1.1 Motivation ....................................... 9
  1.2 New Trends In Teaching Physics .................. 10
    1.2.1 Desktop Experiments ......................... 10
    1.2.2 Conceptual Questions ......................... 11
    1.2.3 Peer Instruction ............................... 11
    1.2.4 Studio Physics ................................ 11
  1.3 Case for Simulations, Desktop Experiments and Collaboration as Effective Teaching tools .... 12
  1.4 TEAL ............................................... 13
    1.4.1 Introduction .................................. 13
    1.4.2 Strategy ....................................... 14
    1.4.3 Road Map ....................................... 14
  1.5 Areas Of Research ................................ 14
  1.6 Thesis Roadmap .................................... 15

2 Technology Overview ................................ 16
  2.1 Java ............................................... 16
  2.2 XML ................................................ 17
  2.3 Databases ......................................... 18
  2.4 Database Connectivity .............................. 19
    2.4.1 RDA (Remote Data Access) .................... 19
    2.4.2 ODBC/JDBC .................................... 19
2.5 JMS (Java Messaging Service) .................................. 20

3 Physical Simulations Architecture .................................. 22
  3.1 Introduction .................................................. 22
  3.2 Design Objectives ............................................ 22
  3.3 Architecture .................................................. 23
    3.3.1 Salient Features ......................................... 23
    3.3.2 Important Components/Classes and Interfaces ............. 24
  3.4 Simulation Tasks ............................................. 29
    3.4.1 EMObejects Creation ..................................... 29
    3.4.2 Rendering Of Objects ................................... 30
    3.4.3 Drawing Graphs .......................................... 30
    3.4.4 Simulation Physics ....................................... 31
  3.5 Simulation Setup and Lifecycle ................................ 32
    3.5.1 Simulation Setup from the Configuration File .......... 32
    3.5.2 Lifecycle ................................................. 34

4 Collaboration Software Architecture ................................ 36
  4.1 Introduction .................................................. 36
  4.2 Architecture .................................................. 37
    4.2.1 Choice of Messaging System ............................... 37
    4.2.2 Protocol Driven .......................................... 39
    4.2.3 Client Server ............................................ 39
    4.2.4 Collablets ................................................ 41
  4.3 Implementation ............................................... 43
    4.3.1 Third Party Software products used ..................... 43
    4.3.2 Security System .......................................... 43
    4.3.3 Collablets part of the framework ......................... 44
    4.3.4 Collablet Manager Interface(CMI) ......................... 46
5 Interactive Question Service (IQS) 47
5.1 Introduction ........................................ 47
5.2 Advantages of using IQS .......................... 48
5.3 IQS Requirements ................................. 48
5.4 Implementation ................................. 49
  5.4.1 Question Lifecycle .......................... 50
  5.4.2 Communication Protocol .................. 51
  5.4.3 Triggers ........................................ 51
  5.4.4 WWW Interface to the system ............. 53

6 Conclusion 54
  6.1 Computer Simulations ........................ 54
  6.2 Collaboration Framework ..................... 54
  6.3 Scope for Future Work ....................... 55
    6.3.1 Computer Simulations .................. 55
    6.3.2 Collaborative applications ............. 55

A Sample Simulation 57
  A.1 Configuration File .......................... 57
  A.2 Screenshot ..................................... 60

B Messaging server setup 62
List of Figures

3-1 Creation of EMobjects .................................................. 29
3-2 Setup and Initialization Of the Simulation ............................. 33
3-3 Simulation Lifecycle ....................................................... 34

4-1 Publish/Subscribe Messaging ............................................. 38
4-2 Point-To-Point Messaging ................................................ 39
4-3 Collaboration Software Architecture .................................. 40
4-4 Collabtlet Lifecycle ........................................................ 42
4-5 Login Process ............................................................. 45
4-6 Classroom Collabtlet Communications overview .................. 46

5-1 Interactive Question Service Architecture ............................ 49
5-2 Data Flow Through IQS .................................................... 50
5-3 Communication Protocol Used .......................................... 52

A-1 Simulation Screenshot ...................................................... 60
List of Tables

B.1 Topics used by the framework. .............................................. 62
B.2 Queues used by the framework. .............................................. 63
B.3 Groups in the framework. ...................................................... 63
B.4 User accounts used by the framework. ................................. 64
Chapter 1

Introduction

The work presented in this thesis is part of Technology Enabled Active Learning (TEAL)/Studio project at MIT. TEAL/Studio project aims to reform the way physics is being taught in undergraduate classes. The focus of this thesis will be on the different software tools that were designed and developed to be used as part of the TEAL/Studio project.

1.1 Motivation

Physics is an important element in the education of engineers, scientists as well as practitioners of other physical and biomedical sciences. For this reason it has been an integral part of any technical education and it is not possible to overemphasize its importance. Research conducted into the efficiency of different teaching techniques show that the traditional methods of teaching physics are inadequate. For example, the traditional method of lecturing and then handing out problem sets for the student works very well when all the students in the class are highly motivated. In practice this is hardly ever the case. A majority of the students don’t even think of the problem set until it is due the next day. The kind of questions asked in the problem sets is very important for them to be effective. If the problems are quantitative in nature students can get by without spending much time on them by plugging the values into equations.
The test clearly showed that what I thought they were understanding they did not understand. Professor Eric Mazur of Harvard University

Students usually learn physics by associating what they learn in class with real life experiences. This works very well for fundamental physics like Newtonian physics where the concepts taught in class can be directly correlated with everyday experiences. It gets complicated for advanced topics like Electromagnetism where the students find it difficult to comprehend and visualize what is being taught in the class. Physics in general is a difficult subject to master when compared to other subjects and each student learns at a different pace. Often some students fall back in the class and gradually lose interest in the class. So it becomes very important to constantly monitor the performance of students in the class and make the learning process interactive and a memorable experience for the student.

1.2 New Trends In Teaching Physics

Realizing the drawbacks and limitations involved in the traditional methods of teaching physics many universities and organizations are actively involved in physics education research (http://www.physics.umd.edu/perg/homepages.htm). As part of their efforts many new innovative methods of instructions have been tested with varying degree of success. Some of the successful methods are presented here.

1.2.1 Desktop Experiments

These are experiments involving simple devices that can be used by students in a regular classroom as well as at home. These experiments usually do not require very elaborate setups and are not very complex in nature, so they can be very easily integrated into the lecture. Typically these experiments are used to introduce new concepts to the students or to demonstrate a physical phenomenon.

---

1McKay Professor of Applied Physics Eric Mazur's reaction on the results of a diagnostic test he gave his students to assess their understanding of Newtonian mechanics. Mazur had until then depended chiefly on the traditional lecture format.
1.2.2 Conceptual Questions

Instructors pose conceptual questions to the students during the course of the class. They are different from the ordinary questions because they intrigue the student and force them to think deeply about the material presented in the class. This often leaves a lasting impression on the student. Even if the students don’t answer the question correctly the process they go through in attempting to solve the question has a huge impact on the students understanding of the subject. Since these questions are mentally absorbing they also serve as a good tool to keep the students interested in the class.

1.2.3 Peer Instruction

There is usually a high level of interaction and activity found in small groups. The idea behind peer instruction is to organize students into small groups and then engage them in useful activities, which could include doing experiments, solving problems etc. The purpose is to engage these groups of students in an intellectually stimulating activity so that they could learn from the high level of interaction and discussion that usually accompany these activities.

1.2.4 Studio Physics

The studio approach to teaching is a major shift in the way physics is taught from the traditional way, which is dominated by lecture. The studio classrooms are set up to facilitate group work and increase the level of student activity. The essential ingredients in a studio class other than lectures are desktop experiments, problem solving, active group discussions, computer simulations of physical phenomenon and question-answer sessions. The instructors supervise and coordinate the activities of the students in the class and help them with any problems they may have.

Prototypes \(^2\) of studio classrooms have been deployed in some universities with

\(^2\)Full-fledged studio classrooms have been implemented in North Carolina State University and Rensselaer Polytechnic Institute and other universities in the United States.
very impressive results. These prototypes differ from each other based on the em-
phasis they place on the different ingredients of a studio class. Nonetheless all these
implementations have one feature in common: they actively engage the students in
the class.

1.3 Case for Simulations, Desktop Experiments and
Collaboration as Effective Teaching tools

Physics is best learnt when the student is an active participant in the class rather than
being a passive participant. Any tool that can help the students visualize and absorb
the physical principles that are being taught in the class is of immense value. If these
tools are interactive and actively engage the students then they are very effective in
getting them accustomed to the physical principles that are being taught in the class.
Desktop experiments, Computer Simulations, Peer Instruction and Collaboration are
a few such tools, which can be incorporated into a classroom to make the learning
process an intellectually rich and rewarding experience.

Desktop experiments involve simple experiments which the students can perform
during the class. Often these experiments are conducted before the introduction of
a theoretical concept, to drive home a point made in the class, to verify the theory
taught in class. Since they are very tightly integrated with the theory taught in
the class they have a tremendous impact on the student. This solves the problem
that exists with traditional laboratory sessions. There often is a delay between the
introduction of a concept to the students and when they actually do the corresponding
experiment in the traditional laboratory, which results in the students doing the
experiments without understanding the physics behind it. The desktop experiments
can be accompanied by thought provoking conceptual questions, which force the
students to think about the under lying concepts and in the process develop insight
into and an interest for the subject.

Desktop experiments are the ideal solution to provide the students with practical
hands on experience, but they have many practical limitations. There may be many logistical issues associated with conducting desktop experiments in classroom with large number of students. Desktop experiments are best suited for simple experiments and often it is not possible to do complex experiments in the classroom. In some cases it may not be economically feasible to conduct the experiments in class. With the latest advances in computer hardware and graphics it has become possible to create high quality simulations. These computer simulations can be used very effectively as a substitute for desktop experiments to simulate a wide variety of physical phenomenon ranging from very the simple to the very complex. Since computer simulations can be made very interactive and are highly configurable they enhance the user experience. These computer simulations can be easily integrated with other course material. Computer simulations also offer the added advantage of multiple uses without any added cost.

Intuition plays an important role in learning physics, and each student perceives the material differently. Physics is a subject where there can be more than more than one way of doing things or analyzing the phenomenon. Since students are the same level, they learn much faster when they interact among themselves. It is necessary to create an environment in the classroom which is conducive for student-student and student-instructor collaboration, so that effective transfer of knowledge takes place.

1.4 TEAL

1.4.1 Introduction

The Teal/Studio Project at MIT is based on the Studio Physics approach to physics education and has been designed to help students develop much better intuition about, and conceptual models of, physical phenomena. This approach is centered on an "active learning" approach—that is, a highly collaborative, hands-on environment, with extensive use of educational technology.
1.4.2 Strategy

The basic plan is to merge lecture, recitations, and hands-on laboratory experience into a technologically and collaboratively rich experience for incoming freshmen. Students will gather in groups of nine, with twelve or so such groups in a common area, for five hours per week. The students will be exposed to a mixture of instruction, laboratory work with desktop experiments, and collaborative work in smaller groups of three, in a computer rich environment (one networked laptop per three students, with data acquisition links between laptop and experiments). The desktop experiments and computer-aided analysis of experimental data will give the students direct experience with the basic phenomena. Formal and informal instruction, aided by media-rich interactive software for simulation and visualization, will then aid students in their conceptualization of this experience.

1.4.3 Road Map

The first prototype implementation of this approach will be in Fall 2001. Based on the results of the first use in Fall 2001, the course will be revised in Spring and Summer of 2002 and offered again in the Fall of 2002, again to off-term 8.02 with about 130 students. This second trial will be used to test revised materials and procedures, and to train faculty who will participate in the on-term implementation in the following term. In the Spring of 2003 TEAL/Studio will be instituted in the on-term version of 8.02, with 700 students, about seven full time Physics faculty, seven teaching assistants, and seven undergraduate aids.

1.5 Areas Of Research

Software tools are needed to teach physics as envisioned by the TEAL/Studio project. The aim of the research is to develop a set of extensible software tools, which can be effectively used in a studio class. The focus has been on:

3Course Offered by MIT Physics Department. Course title: Electricity & Magnetism I
- A toolset for developing Computer Simulations. The toolset has to be very extensible and fairly comprehensive because the goal is to make the generation of new simulations very simple. The simulations generated have to be very interactive and extremely configurable.

- Framework for developing Collaborative Applications (Collablets): Using this framework it should be possible to simulate all the communications that take place in a classroom. It should be a scalable, robust and secure system that can be used even by geographically dispersed individuals.

### 1.6 Thesis Roadmap

The various technologies used in developing the software are introduced in Chapter 2. The architecture developed for creating computer simulations and the various steps involved in their creation and execution is described in Chapter 3. A brief overview of the framework that was developed to create collaborative software applications for use in a studio format class is presented in Chapter 4. Chapter 5 introduces Interactive Question Service, a collaborative application built using the framework presented in Chapter 4. Chapter 6 summarizes the work done, conclusions made therein and gives an outline for the future work.
Chapter 2

Technology Overview

A brief introduction to the various technologies used in developing the software is presented here.

2.1 Java

Java has been used extensively to program the software developed. Java (introduced by Sun Microsystems) was the first programming language that wasn't tied to any particular operating system or microprocessor. Applications written in Java will run anywhere, eliminating one of the biggest headaches for computer users: incompatibility between operating systems and versions of operating systems. Java started in 1990 when a team of Sun researchers developed technology for the convergence of digitally controlled consumer devices and computers.

Java is directly derived from C++ and was designed with many goals in mind. Its designers wanted a new language which was familiar, simple, object-oriented, platform independent, high-performance, threaded, robust and secure. To this end a number of more complicated features of C++ such as pointers, multiple inheritance and operator overloading were omitted from Java. Java is a language of the 1990's and quickly gained fame as the language of the Internet. It was recognized that a language for the Internet would present immense security worries. Users would not want to download routines to their local machines if these routines had the potential to wreck their local
working environments. The creators of java built into the language mechanisms, which prevent remotely loaded routines from taking control of the machine they run on.

2.2 XML

Extensible Markup Language (XML for short) is a new language designed to make information self-describing. This simple sounding change in how computers communicate and exchange data has the potential to extend the Internet beyond information delivery to many other kinds of human activity. Since XML the specification was completed in the early 1998 by the World Wide Web Consortium (usually called W3C), the standard has spread like wild fire through science and into industries ranging from manufacturing to medicine. This enthusiastic response is fueled by a hope that XML will solve one of the Webs biggest problems: although every kind of information is available online it can be extremely difficult to find the information one needs. The problem arises from the nature of Webs main language, HTML (shorthand for Hyper Text Markup Language). Although HTML is the most successful electronic Publishing language ever invented, it is superficial. In essence it describes how a Web browser should arrange text, images and widgets on a page. HTML's concern with appearance makes it a relatively simple language to learn, but this simplicity also has its costs. XML makes it possible, despite the use of incompatible computer systems to create a data format that all can read and write. Unlike most computer data formats, XML markup also makes sense to humans, because it contains nothing more than ordinary text.

The unifying power of XML comes from a few well-chosen rules. One is that tags almost always come in pairs. Like parenthesis, they surround the text to which they apply. And like quotation marks, tag pairs can be nested inside one another to multiple levels. The nesting rule automatically forces certain simplicity on every XML document, which takes the structure of a tree. Each element in the document represents a parent, child or sibling from other element; relationships are unambigu-
ous. Trees cannot represent every kind of information, but they can represent most kinds, which we need computers to understand. Trees are extremely convenient to manipulate using computers.

2.3 Databases

E. F. Codd, a scientist with IBM, developed the Relational Model in the late 1960’s. The relational model introduced the idea of data independence and mathematical set concepts as the foundation for database architecture. Up to that point, database applications directly accessed the data files to manipulate them. In addition, the data was stored in records consisting of fields of individual data items.

The relational model makes the Database Management System (DBMS) itself responsible for accessing and managing the data. The user database application asks the DBMS for the data or passes the data along to the DBMS for storage, and the DBMS then accesses its own files to do the appropriate processing. The details of how the data is stored on the server’s disk are hidden from the user’s application.

The relational model also introduced the concepts of data tables, where the data is presented to the user as a series of columns in one or more rows. The columns and rows are respectively equivalent to fields and records used by the other databases. This concept gave the DBMS more flexibility in sorting and presenting the data to the user, while hiding the actual detail of how the data is stored and manipulated. Thus, the relational model both increases database flexibility and provides greater data security. The relational model remained a paper concept until the late 1970’s, when minicomputers and mainframes began to have enough processing power for experimental relational databases to be developed. IBM performed the initial experiments, and eventually brought its own Relational DBMS to the market. At the same time, IBM released the Structured Query Language (SQL), which eventually became the standard programming language for relational databases. Oracle Corporation was actually the first to release a commercial Relational DBMS that used SQL. Other companies such as Ingres, Informix and Sybase soon followed with their
own implementations of a commercial Relational DBMS.

2.4 Database Connectivity

Databases are usually run on dedicated hardware with fast hard disks and large amounts of memory for faster data access. So standards have been developed to access the data in databases over the network.

2.4.1 RDA (Remote Data Access)

RDA was the first standard to define protocols and services for accessing databases over network connections (actually, over Open Systems Interconnect (OSI\(^1\))). A consortium called SQL Access Group (SAG) was the first to implement the RDA standard over OSI protocol stack and then the now popular TCP/IP stack.

2.4.2 ODBC/JDBC

ODBC stands for Open Database Connectivity. It defines a vendor-independent API (Application Programming Interface) as defined by the Microsoft Corporation, for accessing data in relational and non-relational databases using Structured Query Language (SQL) as a standard for accessing data. This allows an application developer to develop, compile and ship an application without targeting a specific DBMS. Users can then add modules called database drivers that link the application to their choice of database management systems.

Data access from the Java is done with the Java Database connectivity standard, JDBC. JDBC is an object-oriented application-programming interface (API) based on the RDA standard, with the interface defined by Javasoft as part of the core Java distribution. The JDBC API defines Java Classes to represent connections to the database, SQL statements, database metadata and other database objects that enable

\(^{1}\)Created by the International Organization for Standardization (ISO) to develop standards for data networking
a java program to interact with a database and process results. As with ODBC, most
database vendors must provide JDBC drivers that implement the specific functions
native to their respective databases.

2.5 JMS (Java Messaging Service)

Messaging is a method of communication between software components or applica-
tions. A messaging system is a peer-peer facility: A messaging client can send
messages to and receive messages from another messaging client. Each client con-
ects to a messaging agent that provides facilities for creating, sending and receiving
messages. Messaging enables distributed communication that is loosely coupled. A
component sends a message to a destination and the recipient can retrieve it from the
destination. However the sender and receiver do not have to be available at the same
time in order to communicate. In fact, the sender need not know anything about the
receiver, nor does the receiver need to know anything about the sender. They only
need to know what message format and destination to use. In this respect messaging
differs from tightly coupled technologies such a Remote Method Invocation (RMI),
which require an application to know a remote application’s methods.

The Java Message Service is a Java API that allows applications to create, send,
receive and read messages. Designed by Sun and several partner companies, the JMS
API defines a common set of interfaces and associated semantics that allow programs
written in the Java programming language to communicate with other messaging
implementations.

The JMS API enables communication that is not only loosely coupled but also:

- Asynchronous: A JMS Provider can deliver messages to a client as they arrive;
a client does not have to request messages in order to receive them.

- Reliable: The JMS API can ensure that a message is delivered once and only
  once. Lower levels of reliability are available that can afford to miss or receive
duplicate messages. The JMS provider also stores the messages for later delivery
to client when he comes online.
Chapter 3

Physical Simulations Architecture

3.1 Introduction

The TEAL/Studio class will require a large number of computer simulations to be used as part the class. Having a framework on which all these simulations are based offers the following advantages.

- Creation of new simulations is well defined and much simpler.
- Maintainence of existing simulations becomes easy.
- Any improvements made to the framework are automatically reflected in all the simulations.

The simulations generated using this framework should be very easy to use so that students spend most of their time understanding the physics involved rather than figuring out how to use the simulation.

3.2 Design Objectives

The framework has been designed to meet the following requirements.

- Effort involved in creating new simulations should be minimal.
• The architecture should be modular, which makes it possible to develop reusable software components, which can then be reused in multiple simulations.

• The different simulation tasks should be very clearly separated from each other. As an example, the physics governing the simulated objects should be clearly separated from other aspects of the simulation.

• Provide a consistent and extensible method of incorporating the physics into the simulation.

• Provide a consistent and flexible API to manage objects that have to be rendered on the screen.

• Provide an easy and convenient API for drawing graphs representing the variation of different parameters of the simulation.

• Should be able to dynamically alter the state of the different Electromagnetic objects in the simulation and even add and remove Electromagnetic objects dynamically.

• Support Collaboration among different students and the instructor.

3.3 Architecture

3.3.1 Salient Features

Modular and Extensible

The whole simulation process is abstracted into a set of abstract classes and interfaces. These classes and interface are further grouped into modules depending on their functionality. Different modules communicate with each other through well-defined interfaces. If the existing components do not provide the required functionality, new implementations can be written and plugged in, as long as they confirm to the interface used to communicate with other components.
Intelligent Components

The idea behind a modular and extensible architecture is the ability to develop new simulations by putting together software components that can talk to each other and perform the required task. For this to work effectively the individual software components should have some intelligence built into them. By doing this, the entire program logic is decentralized. A simple example of this is the different objects that have to be rendered on the simulation panel. All these objects know how to render themselves on the screen. So they can be plugged into any simulation and they are automatically rendered on the screen.

Event Driven Mechanism

One of the important features of the simulations is their dynamic nature, which lets the users add and remove objects to the simulation dynamically. This poses certain difficulties regarding the communications between different Simulated Objects because communication channels don’t exist between the newly added objects. Adopting an event driven mechanism solves this problem. All the communications between the simulated objects takes place through events.

3.3.2 Important Components/Classes and Interfaces

The following are the main Components in any simulation.

EMobject

All the physical and electromagnetic properties of objects like Point Charge, Electric Dipole, etc are abstracted into a base class called EMobject. The different electromagnetic objects are represented by special implementations of this base class.

EMobjectHandler

EMobjectHandler is a class used to model the physical and the electromagnetic behavior of an electromagnetic object. Different EMobjectHandlers are developed for
each electromagnetic object modeling the required behavior. In its simplest form
the EMobjectHandler for a Point Charge models the dynamics of a Point Charge by
calculating the Force acting on the Point Charge using the equation

\[ \vec{F} = q(\vec{E} + \vec{v} \times \vec{B}) \]  \hspace{1cm} (3.1)

In short EMobjectHandler encapsulates the Electro Magnetic behavior of an EMob-
ject. There could be various reasons for developing different EMobjectHandlers for
the same EMobject. Example: If only one degree of freedom is active for a simulation,
we want to use a handler which takes this into account instead of using a generic three
Degrees of Freedom Handler, for performance reasons. Different simulations may be
emphasizing different physical principles, making it necessary to associate a different
behavior to the EMobject. This is achieved by using a different EMobjectHandler for
the EMobject in different simulations.

**ForceModel**

This is an interface which encapsulates the necessary properties of physical forces
like Friction, Gravity etc. By creating new implementations of this interface different
forces can be simulated.

**ImpulseModel**

All the necessary properties of any object which exert an impulse on the EMObjects
in the simulation are encapsulated in this interface. A simple example is the imagi-
ary boundary of the simulation area. Whenever an EMObject hits the boundary it
gets reflected back so that the object never leaves the simulation area. This is very
easily modeled as an impulse being exerted on the EMObject whenever it reaches
the boundary. Another place where this could be used is when there are collisions
between EMObjects.
Simulation Model

This is the brain of the simulation. All the EMobjects that are added to the simulation register themselves with the Simulation Model. And for every simulation step the simulation model updates the different EMobjects accordingly.

Transform

The coordinate system in which the EMobject properties like position and velocity are measured and the coordinate system used for rendering, is different. The conversion between the two different coordinate systems is handled by the Transform. Sometimes all the objects that are being rendered on the screen may have to be scaled up or down to emphasize certain physical behavior of the EMobjects. Transform presents a convenient way of handling this as well.

Drawable

This is an interface to be implemented by all objects that have to be rendered on the screen. So whenever a new object that has to be rendered on the screen is added to the simulation, if it implements this interface it is registered with the simulation to be rendered at the end of every simulation step.

FieldLines

Electromagnetic field lines present an excellent way to depict the variation of electric and magnetic fields as the simulation progresses. Every EMobject for which the field lines have to be drawn has a FieldLines object associated with it. Points relative to the EMobject can be specified through which the field lines will be drawn. Since the field lines have to be rendered as part of the simulation, FieldLines implements the Drawable interface.
Simulation Applet

This is the class that actually starts the simulations and acts as a container for the simulation. This class features a thread which is used to control the progress of the simulation. This thread is used to trigger each subsequent step of the simulation.

SimulationPanel

This is the panel where all the simulation related rendering takes place. This panel maintains a list of all the objects that need to be rendered onto the screen. So at the end of every simulation step all these are objects redrawn on the screen. This object also is in charge of generating and propagating events at the end of every simulation cycle. Objects interested in these events should register themselves with the SimulationPanel.

GraphPanel

This class acts as a panel for the different graphs to be drawn. This is also the panel where the actual graphs are displayed. Different graphs that have to be displayed register themselves with the GraphPanel. After every simulation cycle all the graphs are updated, keeping them up to date with the simulation.

Graph

This class encapsulates the different properties of a graph. It provides the GraphPanel with points through which the graph has to be drawn.

SidePanel

Different Simulations may have different GUI requirements. For example some simulations may want to display the constantly varying parameter values for the different EMOBjects. Some simulations may want to plot and display different graphs relevant to the simulation. Some simulations may want to add some controls (example: Buttons to add and remove EMOBjects). The SidePanel acts as a container for all the
simulation specific GUI components.

Configuration File

The configuration file is an XML file which contains information regarding the simulation setup and initialization. At the start of a simulation this file is parsed and the simulation environment is setup and initialized with default objects, as specified in the configuration file.

Factory Classes

Factory classes are used to parse the XML configuration file and then create simulation objects according to the properties read from the configuration file. Every simulation object whose properties can be set in the XML file has a Factory class associated with it. Some of the important factory classes used are listed here.

- **SimulationFactory**: This class is responsible for reading the simulation’s configuration file and then setting up the simulation accordingly. It is also responsible for initializing the simulation with parameters read from the configuration file. It uses various other helper classes to assist in the process.

- **EMObjectFactory**: This class is used by the SimulationFactory to create EMobjects that will be added to the simulation. Some of the other tasks that are performed by this class are initializing the created EMobjects with default values read from the configuration file, setting the EMobjectHandler for the EMObject and then setting the FieldLines for the EMObject created. New Custom EMObjectFactories can be easily developed to have custom creation of EMobjects.

- **FieldLinesFactory** This class is used by the EMObjectFactory to set the field lines that have to be drawn for the EMObjects created by the EMObjectFactory.
3.4 Simulation Tasks

The various processes involved in a simulation like creating EMobjects, rendering different objects on the screen etc and how all these different processes fit together is explained below.

3.4.1 EMObjects Creation

The creation of the new EMObjects is handled by the EMObjectFactories. These factories are created during the setting up of the simulation. After these factories are
created they are registered with the SimulationModel for later use in the simulation. Using EMObjectFactories makes it possible to set simulation specific properties to the EMObjects being created. The Emobject creation process has been briefly illustrated in figure 3-1.

There are two different situations in which EMObjects are created in the simulation.

1. During the initialization process of the simulation where EMObjects are created with properties as specified in the configuration file.

2. When the user decides to add an EMObject during the simulation. In this case, the EMObjectFactory creates an EMObject with default properties. Then the EMObject is presented to the user for further change in the properties, like mass, velocity and other physical properties before it is added to the simulation.

3.4.2 Rendering Of Objects

SimulationPanel is the region where all the simulation related rendering takes place. And all objects that have to be rendered on the SimulationPanel implement the Drawable interface. All objects that implement this interface know how to render themselves on the screen when given access to a Transform and a Graphics objects. Some of the objects that implement this interface are EMObject and FieldLines. The SimulationPanel maintains a list of objects that have to be rendered (those that implement the interface Drawable). So whenever a new object which implements this interface is added to the simulation, it is registered with the SimulationPanel and the SimulationPanel automatically renders it on the screen at the end of every simulation step. To stop the object from being rendered on the screen it has to be unregistered from the SimulationPanel.

3.4.3 Drawing Graphs

GraphPanel is the region where all the graphs for the simulation are drawn. So every simulation which wants to use graphs to depict the variation of different simulation
parameters as they change should have the GraphPanel as one of its components. Individual Graphs can then register themselves with the GraphPanel. At the end of every simulation step the GraphPanel requests all the registered Graphs to plot their latest values.

3.4.4 Simulation Physics

One of the important goals of the architecture is to separate the physics of the simulation from every other aspect of the simulation. This is exactly what the SimulationModel achieves. All the EMobjects that have to be simulated are registered with the SimulationModel. Then for every simulation step, the SimulationModel computes the next state of all the registered EMobjects upon instructions from the SimulationPanel. The simulation essentially is solving differential equations involving the properties of EMobjects. Depending on the complexity of the equations and the interdependence of the properties of different EMobjects it may or may not be possible to dynamically add and remover EMobjects to and from the simulation. As an Example, consider the interaction of point charges. Their behavior is purely a function of the total electric field acting on it due to all the EMobjects. The point charge doesn't need any knowledge from the other EMobjects except their Electric Field contribution. Hence in this simulation it is possible to add and remove point charges dynamically. In contrast, consider the interaction of two electric dipoles. To solve the equations of motion we have chosen to use the conservation of linear and angular momentum, which very closely couples the two electric dipole together. The moment a new electric dipole is added to the simulation the equations to be solved using this approach change drastically, making it impossible to add and remove EMobjects dynamically unless certain approximations are made. So there are two types of SimulationModels one which allows us to add and remove EMobjects dynamically; and one which doesn't let us change the number of EMobjects in the simulation.
Fixed EMObjects

The SimulationModel is specifically crafted for a particular complex simulation and as such each simulation has a separate SimulationModel when this approach is used.

Variable number of EMObjects

The important feature in this approach is the use of EMobjectHanlders, which know how to control the behavior of the EMobject associated with it. They make extensive use of ForceModel, ImpulseModel and CollectionOfEMobjects to model different behaviors. By using different EMobjectHandlers for different simulations it is possible to use the same SimulationModel for all the simulations without any changes. Most simulations fall in this category.

3.5 Simulation Setup and Lifecycle

3.5.1 Simulation Setup from the Configuration File

The first step in any simulation is setting it up according to the configuration file (see Figure 3-2). First the SimulationPanel class for the simulation is read from the configuration file and then an instance of it is created and initialized. Some of the properties that can be set are Size of the panel, Color of the background and so on. The properties of the panel could also be the SimulationModel to use and the SidePanel to use for the simulation. One property of the SimulationPanel is whether it uses a predetermined SimulationModel and SidePanel or whether they are read in from the configuration file. In either case after the initialization of the SimulationPanel, the simulation has a SimulationModel and a SidePanel.

Next the EMObject properties for the simulation are read from the configuration file. During this process the different EMObjects that are allowed in the simulation are read in and appropriate EMObjectFactories are created for them. These EMObjectFactories are then registered with the SimulationModel. With this the simulation setup is complete.
The next step is to initialize the simulation based on the information from the configuration file. Some of the common tasks that are performed during the simulation initialization are to create an appropriate transform for the simulation, and then add some objects that have to be rendered on the SimulationPanel (like axes representing the coordinate system in which the simulation takes place). Then the EMObjects that have to be added to the simulation are processed. A reference to the appropriate EMObjectFactory is got from the SimulationModel and is used to create the appropriate EMObject with the appropriate properties and EMObjectHandler set.
for the EMobject (depending on the SimulationModel used). These EMobjects are then added to the Simulation. After this the graphs node is processed to register the appropriate graphs with a GraphPanel, if the simulation supports graphs. This completes the initialization of the simulation.

### 3.5.2 Lifecycle

The Simulation Applet is the starting point of any simulation. The first step in any simulation is to setup and initialize the simulation according to the properties read from the configuration file. To aid in this process the SimulationApplet makes use of simulation factory that parses the configuration file (an XML file) for the simulation and then reads the simulation and initializes it.

Now the simulation is ready to start. The Thread in the simulation applet is started which periodically advances the simulation through time steps. The simulation can be controlled by the different controls added to the SidePanel. As the simulation progresses, all the different listeners for events or updates are notified as

![Simulation Lifecycle Diagram](image)
and when appropriate events occur. The simulation now proceeds until the user stops it or until a certain number of simulation steps have been completed.
4.1 Introduction

A TEAL/Studio class involves high level of interaction between the students and the instructors. These interactions, while actively engaging the students in the class, also generate valuable information which can be used to track the progress of the students, check the effectiveness of teaching techniques being used in the class, and fine tune them. Obtaining this information and analyzing it could be a very demanding task without proper tools to aid in the process. The availability of networked computers for all the students in a studio class make it possible to use collaborative software applications to aid in the learning process, as well as collecting this useful information.

These collaborative applications should be easy to use and run on a variety of computer platforms. The information should be exchanged securely, and the solution has to be scalable to support large number of students. A framework has been created for developing collaborative applications meeting the above requirements.
4.2 Architecture

4.2.1 Choice of Messaging System

Collaboration among different users invariably involves sending data across the network. How these communications between the client and server are handled is the most important design issue. From the network point of view this can be done either through synchronous or asynchronous messaging.

Synchronous messaging over the network is possible using technologies like CORBA\(^1\) and JavaRMI\(^2\). These technologies attempt to mimic the behavior of a system that runs in one process. When a remote procedure is invoked the caller is blocked until the procedure completes and returns control to the caller. This synchronized model allows the developer to view the entire system as it runs in one process. But at the same time this synchronized nature of RPC tightly couples the client (software making the call) to the server (the software servicing the call). This client cannot proceed until the server responds. Though this tightly coupled nature makes it very easy to develop distributed applications it creates highly interdependent systems where a failure in any of the subsystems can have an immediate and debilitating impact on the entire system. This makes it unsuitable for the development of scalable and fault tolerant solutions.

Problems involving the availability of subsystems is not an issue with Asynchronous messaging. The different software components communicate with each other using one-way messages that require no immediate response. In other words there is no blocking involved. Once a message is sent, the messaging client can move on to other tasks; it doesn’t have to wait for a response. In this paradigm all the subsystems are decoupled from each other so a failure in one doesn’t impede the operation of others.

---

\(^{1}\) Common Object Request Broker Architecture (CORBA) is an open distributed object computing infrastructure standardized by the Object Management Group (OMG).

\(^{2}\) Java RMI is an Object Oriented analogue to RPC (Remote Procedure Call), which is used to invoke methods and functions in remote procedural-based applications. Java RMI framework is a solution for a distributed system implemented entirely in Java.
Partial network failure is a fact of life, during which the system can miss many messages. In recognition of this, messaging systems provide guaranteed delivery, which ensures that intended consumers will receive a message even if partial failure occurs. Guaranteed delivery uses a store and forward mechanism, which means that the underlying message server will write the incoming messages to a persistent store if the intended consumers are not available. When the receiving system becomes online at a later time, the store-forward mechanism will deliver all the messages that the consumer missed while it was offline. Providing this kind of fault tolerance in synchronous communication models is impossible. After considering all the above facts, Asynchronous messaging was selected to handle the communications between different components. The following two messaging paradigms have been extensively used in developing the applications.

Publish/Subscribe

![Publish/Subscribe Messaging](image)

A publish/subscribe (pub/sub) messaging system (see Figure 4-1) is an event driven model where information consumers and producers participate in the transmission of messages using topics. Topics are electronic channels to which the producers of messages publish their messages. These messages are then sent to all the
subscribers for that topic. A topic is analogous to a news group or a list server: when a message is sent to a list server or newsgroup, it is delivered to all the subscribers.

**Point-to-Point**

![Point-To-Point Messaging Diagram](image)

Figure 4-2: Point-To-Point Messaging

In point to point messaging (see Figure 4-2), messages from different sources are routed to an individual consumer. This is achieved by using queues to which messaging applications send messages. The clients then receive these messages from the queue.

**4.2.2 Protocol Driven**

Because of the asynchronous nature in which messages are sent and received by the components, it is necessary to have a communication protocol which can be used to process the messages. In general every service offered by the framework has a communication protocol which all the components adhere to when interacting with this service.

**4.2.3 Client Server**
Collaboration Software Architecture

Clients Connect to the appropriate Topics and Queues

Messing Server

Collables connect to appropriate Topics and Queues.

Collablers

Figure 4-3: Collaboration Software Architecture
The client/server software architecture is a versatile and modular paradigm that is intended to improve usability, flexibility, interoperability and scalability of software components. It is an extension of modular programming. The fundamental assumption in modular programming is that the separation of a large piece of software into its constituent parts ("modules") creates the possibility for easier development and better maintainability. Client/server computing takes this a step farther by recognizing that those modules need not all be executed within the same memory space. With this architecture, the calling module becomes the "client" (that which requests a service), and the called module becomes the "server" (that which provides the service). The logical extension of this is to have clients and servers running on the appropriate hardware and software platforms for their specific functions. For example, database management system servers running on platforms specially designed and configured to perform queries, or file servers running on platforms with special elements for managing files.

Because of the above reasons and the nature of the application, which consists of a number of clients distributed across multiple machines, Client Server architecture is the obvious choice. The Collaboration server, which offers the collaboration services is the server component of the application. The clients (students/instructors) access these services through the messaging layer that exists between them (see Figure 4-3).

4.2.4 Collablets

A collablet is defined as a component of the collaboration server, which offers a particular collaboration service. Enforcing a common scheme on all collaboration applications/services provides a simple and consistent mechanism which allows new collaborative applications to be developed and deployed. It also makes the management of these services very easy for the collaboration server.
Overview

A collablet in its most general form is a Java class which is run inside a collablet container. Each collablet has the same lifecycle (see Figure 4-4):

- Collablet loaded into the Collablet Container (Collaboration Server) and initialized.
- The Collablet is started.
- Zero or more request are served by the collablet.
- The collablet can be stopped and restarted.
- Collablet stopped and unloaded from the container.

Typically a collablet has its own set of messaging resources (topics and queues) through which all the collablet related communications take place. And every collablet also has a queue dedicated for its management. Using this queue it is possible
to start and stop the service and also to do service specific configuration. Having separate messaging resources for each collablet makes it possible to impose finer access controls over the system resources. The basic framework has five collablets Collablet Manager, Authentication Collablet, Classroom Collablet, Poll Collablet and Help Collablet. The services offered by these collablets are explained in greater detail later.

4.3 Implementation

4.3.1 Third Party Software products used

SwiftMQ Router

This is a Java Messaging Service compliant high performance messaging system, which supports both Publish/Subscribe and Point-To-Point messaging. This product has been used to provide the messaging infrastructure. The messaging server resources are accessed using the JMS API.

Oracle8i

Oracle 8i from Oracle Corporation is a market leader in high performance relational database systems. This DBMS has been used to store the application data. The application interacts with the DBMS using JDBC.

4.3.2 Security System

Different users perform various tasks which require access to multiple system resources. It is very important to prevent unauthorized access to these resources. Therefore it is necessary to enforce a security system which controls the user's access to system resources. Since the user has to invariably go through the messaging server to access any of the system resources, and since each system resource has its own dedicated communication channels, controlling access to these communication channels suffices.
Groups are fundamental to the JMS security system. They serve as a security profile for the different users. Different groups representing different security profiles are created and given access to the different topics and queues of the messaging server. JMS users belonging to these groups are created and they are mapped to the appropriate Collaboration application users. When the application users log into the messaging server it is possible to restrict access to system resources very efficiently.

4.3.3 Collablets part of the framework

Collablet Manager

Collablet Manager is a collablet, which is used to manage other collablets. The following are the tasks handled by the collablet manager.

- Initialize the container in which the collablets will run.
- Load various collablets into the container and initialize them.
- Start and Stop Collablets that have loaded into the container.
- Unload collablets from the container after they are no longer needed.

Authentication Collablet

As the name suggests the main purpose of the authentication Collablet is to authenticate new users logging into the system. Whenever a new request comes in it verifies the users credentials and sends back the information, which the client uses to log on to the messaging system. The login process is briefly illustrated in the above flow chart.

Classroom Collablet

The classroom collablet is responsible for creating a virtual classroom in which the users can communicate with each other. By virtue of the messaging model used it becomes very easy to implement this. The collablet performs the role of a modulator.
and stores any important information generated during the session. Though it is possible to allow student-student communication, this feature has been turned off for practical considerations.

Help Collablet

During the course of the class students may have some doubts regarding the material presented in the class or they may need some help to with the class work. In such cases students can use the help collablet to send a help request to all the instructors in the class. One or more than one instructor can respond to a help request sent by the student. This is usually followed by exchange of information between the student and instructor/s till the issue is resolved. All the communications that take place in
this process are stored in the database. This information can later be analyzed to find out the topics with which students are having problems.

Poll Collablet

It may be necessary to monitor the activity of students participating in the class. This collalbet can be used to constantly poll the students for their activity information. The collected information is then sent to the instructors.

4.3.4 Collablet Manager Interface (CMI)

CMI is a command line tool for remote management of the collablets in the collaboration server. Using this tool it is possible to remotely load and unload, start and stop collablets.
Chapter 5

Interactive Question Service (IQS)

5.1 Introduction

One of the many challenges any instructor faces in handling a class is maintaining the student’s level of interest. In some cases though the students start off enthusiastically, they gradually lose interest and by the end of the class they are totally lost. In other cases the students gradually lose interest in the course itself. It is absolutely essential to prevent this from happening.

Presenting the students with questions to answer as part of the lecture goes a long way in keeping the students interested in the class. This technique is much more effective when the type of questions asked in the class are mentally stimulating. If the student responses to these questions are recorded and analyzed, it is possible to identify students falling behind in the class long before it is too late. However it is practically impossible to collect this kind of information in a conventional classroom.

In a studio class the situation is radically different from that of a conventional classroom. Because the students use networked computer as part of the regular class activity, this could be a very good medium through which questions could be sent to the students. Interactive Question Service is a comprehensive tool for use in a studio class room, which lets the instructors pose questions to the students, store the subsequent student responses to these questions and provide the instructor with real time analysis of this data. This data can be used to track the progress of the
students.

5.2 Advantages of using IQS

1. It is possible to consistently track the performance of all the students during the class for the entire duration of the course.

2. Accurate statistics on the performance of the class are instantaneously available to the instructor. The instructor can then use this information to make the teaching process more effective.

3. The collected information when combined with the assignment and test scores can reveal valuable insights.

4. In a traditional classroom the questions are answered by the students vocally (they spell out the answer) or physically (raise hands to indicate their answer). Some students may be too shy to do this or they may be hesitant to participate for the fear of choosing the wrong answer before fellow classmates. IQS offers the students a way to answer the questions anonymously, thus encouraging everyone in the class to participate.

5.3 IQS Requirements

1. The application should support a large number of users including both students and instructors at the same time.

2. The instructor should be able to track the responses of different users and should be able to set triggers. (Example: The instructor should be notified if a student gets a large number of questions wrong).

3. All the activity and results should be logged so that the data can be analyzed after the class.

4. Web Interface to the data store for convenient access to the data and analysis.
5.4 Implementation

The Interactive Question Service has been implemented as a collablet (IQSCollablet) using the framework presented in the previous chapter. This collablet uses the Publish/Subscribe messaging model to handle all the communications with the students and the instructors. In particular two topics are used to handle the communications.

![Interactive Question Service Architecture](image)

Figure 5-1: Interactive Question Service Architecture
- t_iqscollablet.student: This topic is dedicated for communications between the IQSCollablet and the student. Both the IQSCollablet and the student can publish and subscribe to this topic.

- t_iqscollablet.instructor: This topic is dedicated for communications between the IQSCollablet and the instructor. Both the IQSCollablet and the student can publish and subscribe to this topic.

5.4.1 Question Lifecycle

Figure 5-2: Data Flow Through IQS
Instructors connected to IQS create new questions. These questions are then sent to students connected to IQS. After the students answer the question their answers are sent to the server, which processes the information and then updates the students and instructors with the latest statistics regarding the question. The students are allowed to change their answers as long as the instructor does not close the question. The instructor in the mean time can send some hints to the students for answering the question. The instructor then closes the question and sends the answer to the students. IQS executes the triggers associated with the question and then sends the results back to the instructors.

5.4.2 Communication Protocol

Like any other collablet IQSCollablet, students and instructors to send and receive messages use a communication protocol. This protocol has been briefly described in Fig 5.3.

5.4.3 Triggers

Triggers that can be set on the questions are very powerful tools. Two triggers have been developed to be used with the IQS.

The first trigger analyzes the performance of the whole class at the close of each question and if 50% (configurable) of the class get the answer wrong, the trigger sends a message to the instructor. Using this information the instructor can take appropriate action, for example, more time on the topic so that students get familiar with it.

The other trigger is based on the performance of each individual student in the class. At the end of each question, the past performance of each student is studied. If the performance of the student doesn't meet the expectations of the instructor, a message is sent to the instructor informing him about the student. This helps the instructor to identify the students that may need help.
<table>
<thead>
<tr>
<th>Instructor</th>
<th>Question Collabnet (Instructor Stub)</th>
<th>Question Collabnet (Student Stub)</th>
<th>Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUESTION_INST_NEW</td>
<td>This message is sent when the instructor sends a new question.</td>
<td>QUESTION_SVR_INST_NEW</td>
<td>The server after checking that there is no active question, informs all the instructors about the new question.</td>
</tr>
<tr>
<td>QUESTION_INST_HINT</td>
<td>The message is sent when the instructor sends a hint to the current question.</td>
<td>QUESTION_SVR_INST_STATUS</td>
<td>The updated status for the questions are sent to the students.</td>
</tr>
<tr>
<td>QUESTION_INST_HINT</td>
<td>This message is sent when the instructor sends a hint to the current question.</td>
<td>QUESTION_SVR_STD_HINT</td>
<td>This message is sent when there is a hint for the question.</td>
</tr>
<tr>
<td>QUESTION_INST_CLOSE</td>
<td>This message is sent when the instructor closes a question.</td>
<td>QUESTION_SVR_STD_CLOSE</td>
<td>The students get this message when the current question is closed. After this message is received the students will not be able to change their answers.</td>
</tr>
<tr>
<td>QUESTION_INST_ANSWER</td>
<td>This message is sent when the instructor intends to send the answer of the question to the students.</td>
<td>QUESTION_SVR_STD_ANSWER</td>
<td>The students are sent this message when the current question has been answered.</td>
</tr>
</tbody>
</table>

Figure 5-3: Communication Protocol Used
5.4.4 WWW Interface to the system

IQS generates a lot of data over a period of its usage. So a web interface has been developed to conveniently access this data.
Chapter 6

Conclusion

6.1 Computer Simulations

The simulation package developed is extremely modular in nature. It uses a library of well defined simulation objects to perform the different simulation tasks. New objects can be added to the library as the need arises. New simulations are created by specifying the simulation objects to use along with their properties in a configuration file. This makes the process of creating new simulations and maintaining them extremely easy. The package is not yet complete because it lacks a consistent mechanism for handling the GUI elements and laying them out on the screen. This feature is being worked on by other members of the TEAL project and should be complete very soon.

6.2 Collaboration Framework

The framework developed is a prototype. Though it has all the required functionality, it has to be thoroughly tested and proper error checking has to be built into it before it can be used in the studio classroom. Some of the areas that can be improved are:

- The collablet container in which the collablets run should allow inter-collablet communication.

- Encryption schemes are necessary to provide better security for the data that
6.3 Scope for Future Work

6.3.1 Computer Simulations

Though the current simulations architecture is very generic and can handle any kind of physical simulation it still can be improved. Some of the possible improvements are.

- All the simulations are currently rendered in 2D. The architecture should be extended to handle 3D rendering.

- A simulation editor (GUI) could be developed for creating new simulations using simulation objects from the library.

6.3.2 Collaborative applications

Some of the interesting areas to explore are.

Improved IQS (Interactive Question Service)

The current implementation of IQS's analysis service is based on how the students respond to a single question. Though this by itself is of immense value, it can be extended to analyze students responses to a series of questions. IQS could also be made more interactive by making the questions students asked a function of the answer to the previous question. For example, when a student answers a question incorrectly, IQS will present to the student a new question. This new question would give the students a chance to correct their previous mistakes.

Collaborative Simulations

Students may need some help in using the computer simulations or they may want to share the results that they have obtained with fellow students or the instructors. So
a collaboration module could be built into the simulations which lets the students and instructors share their simulation desktops and also the observations they make.
Appendix A

Sample Simulation

In this appendix we present the configuration file and a screenshot for a simple simulation that models the behaviour of a magnetic dipole falling through a conducting ring.

A.1 Configuration File

```xml
<?xml version="1.0"?>
<simulation>
<title>Test Simulation</title>
<gui-settings>
<panel class="teal.simulation.control.EMWorldPanel">
<size x="500" y="700"/>
<background color="black"/>
</panel>
</gui-settings>
<emobject-settings>
<emobject type="PointCharge" allowed="true">
<handler>teal.simulation.emworld.PointChargeHandler2D</handler>
<fieldlines draw="true"/>
```
<symmetry x="false" y="false"/>
<step size="5"/>
<point radius="10" angle="1.57"/>
<point radius="10" angle="-1.57"/>
<point radius="20" angle="1.57"/>
<point radius="20" angle="-1.57"/>
</fieldlines>
</emobject>

<emobject type ="MagneticDipole" allowed="true">
<handler>teal.simulation.emworld.MagneticDipoleHandler1D</handler>
<fieldlines draw="true">
<symmetry x="false" y="false"/>
<step size="5"/>
<point radius="75" angle="1.57"/>
<point radius="75" angle="-1.57"/>
</fieldlines>
</emobject>

<emobject type ="RingOfCurrent" allowed="true">
<handler>teal.simulation.emworld.RingOfCurrentHandler1D</handler>
<fieldlines draw="true">
<symmetry x="true" y="true"/>
<step size="5"/>
</fieldlines>
</emobject>

</emobject-settings>
<initialize>
<emobjects>

<RingOfCurrent mass="1" current="0" stationary="true" radius="100"
id="RingOfCurrent1" resistance="10" inductance="100">
<position x="0" y="0" z="0"/>
<velocity x="0" y="0" z="0"/>
</RingOfCurrent>

<MagneticDipole mass="1" stationary="false" mi="1" id="MagneticDipole1">
<position x="0" y="200" z="0"/>
<velocity x="0" y="-0.5" z="0"/>
<dipolemoment x="0" y="10" z="0"/>
</MagneticDipole>

</emobjects>

<drawable>
<object class="teal.geometry.Axis"/>
</drawable>

<transform class="teal.math.TransformXY"/>
<graphs>
<graph class="teal.gui.graph.Current">
<properties target="RingOfCurrent1"/>
</graph>
</graphs>
</initialize>
A.2 Screenshot

Figure A-1: Simulation Screenshot

The important components visible in the screenshot (see Figure A-1) are:

- SimulationPanel: The panel in which the falling magnet and the ring are rendered.

- GraphPanel: The panel where the graph of current in the ring with respect to time is drawn.
- ControlPanel: The panel with the Add, Edit and Remove buttons which can be used to add EMobjects to the simulation, edit properties of EMobjects in the simulation and remove EMobjects from the simulation respectively.
Appendix B

Messaging server setup

In this appendix we present the different topics, queues, groups and user accounts created for use by the collaboration framework.

Table B.1: Topics used by the framework.

<table>
<thead>
<tr>
<th>Topics</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_classcollablet_service</td>
<td>Used by the PollCollablet to monitor the student activity</td>
</tr>
<tr>
<td>t_classroom</td>
<td>Used by ClassRoomCollablet to handle communications with the students and instructors.</td>
</tr>
<tr>
<td>t_help</td>
<td>HelpCollablet uses this topic to provide help functionality to the students.</td>
</tr>
<tr>
<td>t_iqscollablet_instructor_service</td>
<td>Used by IQSCollablet to handle communications with the instructor</td>
</tr>
<tr>
<td>t_iqscollablet_student_service</td>
<td>Used by IQSCollablet to handle communications with the student</td>
</tr>
</tbody>
</table>
Table B.2: Queues used by the framework.

<table>
<thead>
<tr>
<th>Queues</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>q_mgmt_manager</td>
<td>Management queue for CollabletManager.</td>
</tr>
<tr>
<td>q_mgmt_authentication</td>
<td>Management queue for AuthenticationCollablet.</td>
</tr>
<tr>
<td>q_mgmt_iqs</td>
<td>Management queue for IQSCollablet.</td>
</tr>
<tr>
<td>q_mgmt_class</td>
<td>Management queue for ClassCollablet</td>
</tr>
<tr>
<td>q_service.auth</td>
<td>Used by authentication collablet to provide authentication service</td>
</tr>
<tr>
<td>q_mgmt_poll</td>
<td>Management queue for PollCollablet</td>
</tr>
<tr>
<td>q_mgmt_help</td>
<td>Management queue for HelpCollablet</td>
</tr>
</tbody>
</table>

Table B.3: Groups in the framework.

<table>
<thead>
<tr>
<th>Groups</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>g_instructors</td>
<td>All the instructors belong to this group.</td>
</tr>
<tr>
<td>g_students</td>
<td>All the students belong to this group.</td>
</tr>
<tr>
<td>g_administrators</td>
<td>All the administrators belong to this group.</td>
</tr>
<tr>
<td>g_authcollablet_service</td>
<td>AuthenticationCollablet belongs to this group.</td>
</tr>
<tr>
<td>g_iqscollablet_students</td>
<td>IQSCollablet student stub uses this group.</td>
</tr>
<tr>
<td>g_iqscollablet_instructors</td>
<td>IQSCollablet instructor stub uses this group.</td>
</tr>
<tr>
<td>g_collabletmanagement</td>
<td>Collablets connect to their management queues as members of this group.</td>
</tr>
<tr>
<td>g_public</td>
<td>Users connect to the AuthenticationCollablet as a member of this group.</td>
</tr>
<tr>
<td>g_poll</td>
<td>PollCollablet connects to the messaging service as a member of this group.</td>
</tr>
<tr>
<td>g_help</td>
<td>HelpCollablet connects to the messaging service as a member of this group.</td>
</tr>
</tbody>
</table>
### Table B.4: User accounts used by the framework.

<table>
<thead>
<tr>
<th>User Name</th>
<th>Group</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>u_instructor</td>
<td>g_instructors</td>
<td>Instructors are mapped to this user account</td>
</tr>
<tr>
<td>u_student</td>
<td>g_students</td>
<td>Students are mapped to this user account</td>
</tr>
<tr>
<td>u_administrator</td>
<td>g_administrators</td>
<td>Administrators are mapped to this user account</td>
</tr>
<tr>
<td>u_public</td>
<td>g_public</td>
<td>Students and Instructors access the messaging system as this user during login</td>
</tr>
<tr>
<td>u_poll</td>
<td>g_poll</td>
<td>Poll collablet uses this account to access the messaging system</td>
</tr>
<tr>
<td>u_help</td>
<td>g_help</td>
<td>Help collablet uses this account to access the messaging system</td>
</tr>
<tr>
<td>u_auth_collablet_service</td>
<td>g_authcollablet_service</td>
<td>AuthenticationCollablet uses this account to access the JMS system.</td>
</tr>
<tr>
<td>u_iqs_collablet_student</td>
<td>g_iqscollablet_students</td>
<td>Used by IQS collablet to communicate with students</td>
</tr>
<tr>
<td>u_iqs_collablet_instructor</td>
<td>g_iqscollablet_instructors</td>
<td>Used by IQS collablet to communicate with instructors</td>
</tr>
<tr>
<td>u_collab_mgmt</td>
<td>g_collablet_management</td>
<td>Used by collablets to connect to management queue</td>
</tr>
</tbody>
</table>
Bibliography


