Virtual 3D Worlds for virtual or remote Laboratories with regards to learning/educational undertakings.

Fabio Ricardo dos Santos
Graz University of Technology, Austria

ABSTRACT

The transportation of a campus classroom and/or laboratory into a three dimensional virtual representation has changed remote learning, especially in engineering education.

This thesis is concerned with shared virtual environments for collaborative learning and working, mainly between students and tutors/professors in a virtual control experiment setting.

The underlying foundation for this idea is the iLab Project, which is a scalable architecture for sharing online experiments used by students and institutions. This research project is conducted and developed by the Center for Educational Computing Initiatives (CECI) at the Massachusetts Institute of Technology (MIT). The main goal of the iLab Project is to provide a broad set of experimentation resources for students at MIT and elsewhere. The iLab initiative has grown tremendously over the past two years, being adopted by numerous partner universities around the globe.

3D Virtual Worlds are computer-based simulated environments intended for their users to inhabit and interact via avatars with other avatars and the virtual environment. These avatars are typically three-dimensional graphical representations. Such modeled worlds may appear similar to the real world and represent a powerful new media for instruction and education. Many higher education institutions are advancing their online teaching methods by offering educational applications in 3D Virtual Worlds because they offer attractive features and possible applications for education. The integration of possible functional iLabs by prototyping an iLabs application in such a 3D environment (such as Second Life or Open Wonderland) seems to be a very innovative tool for distance learning. Open Wonderland is a Java open source toolkit and is completely extensible for new features in the existing 3D World, like a prototype for iLab application. The platform is ideally suited to the education sector because it provides a higher level of security than is possible on other virtual platforms.

The combination of the advantages both of 3D Virtual Worlds and the iLabs Project will be realize an appreciation for work with labs for university use and is the primary goal for this master thesis. The Project will open the doors of education to new possibilities for learning and collaboration in a globally connected world, and education itself will be fundamentally altered. This method of immersive learning in 3D virtual worlds environments does not replace traditional methods of evaluation, but instead offers additional ones, particularly for distance learning.

The research project goals is organized as follows:

- Research how iLab Application prototype and TEALSim Simulations should be integrated into a virtual 3D World.
- iLab prototype should be ported from Open Wonderland Version 0.4 to the Version 0.5, which was in the beginning of the project as Developer Release available, to extend its functionality in 3D context.
- Program interactive 3D Interfaces in Wonderland Release 0.5
- Create an entire environment for the simulation in Project Wonderland using 3D Graphics Software Tools.
- Test the outcome/result with students at MIT.
- Write the theoretical part of master thesis, including stand of the art (STAR) problems and results.

Our first collaborative virtual environment, a proof of concept, provides full functionality of one physics experiment, though there are still some performance issues to be resolved.

The next step for integrating TEALsim and iLabs in Open Wonderland is porting our system from Wonderland's version 0.4 to 0.5. Our goal is a system redesign in order to support adding flexibility to multiple physics simulations. The performance improvements in Wonderland 0.5 will allow a large number of avatars in our future scenario, where they will be able to run even more physics experiments, through a new 3D user interface.

The expected outcome of research in this topic is that the 3D context should make learning and working with online laboratories easier and more attractive. The possibility of immersive learning should be emphasized to users of iLab and scale up to large numbers of users worldwide.

1. INTRODUCTION

In the past few years, especially due to the increasing availability of faster and robust hardware and software, 3D environments have become common technology. Virtual laboratories, scientific visualizations, and collaborative work approaches are just some of the successful fields of 3D graphics applications.

One essential term is Virtual Environment (VE), which is a computer generated spatial environment, where the stimulation of diverse human senses gives the user a feeling of being immersed. Immersion refers to how deep the user is emotionally involved within a specific virtual environment [1].

Nowadays, a great challenge in educational technology research is building technology that not only supports the learning process, but also connects students and educators in a way that enables effortless cooperation, even when both parties are geographically spread.

Our first collaborative virtual environment provides full functionality of one physics experiment. This environment results from the integration of internet-accessible physics experiments (iLabs) combined with the TEALsim 3D simulation toolkit in Open Wonderland [2]. Students and educators within this environment are represented as avatars that can remotely control experimentation equipment, visualize physics phenomena generated by the experiment, and then discuss results. This environment was developed based on the Technology Enabled Active Learning (TEAL) classroom idea to support social interactions and encourage students' active learning and interest in an environment that fosters conceptual change [3].

This study explores the process of conversion of the TEALsim simulation package to the rendering engine jMonkeyEngine (JME), which is the graphics engine used in Open Wonderland version 0.5. Additionally, we have built a Wonderland cell class that can dynamically load TEAL simulations into Wonderland by the end of the research project. This will provide a collaborative environment similar to the current 0.4 version, but using the new graphics engine.

Furthermore, we implemented an automatic generation of the simulation's controls in the Wonderland environment either as buttons, sliders or other control elements within the 3D space or as elements in the Heads Up Display. These controls will provide the standard Java event handling model.

Therefore, avatars have the ability to interact directly with TEALsim elements, including moving elements and activating sensors. Changes in the environment would be updated in real time. Finally, this report outlines future directions and challenges to come, based on the latest research.

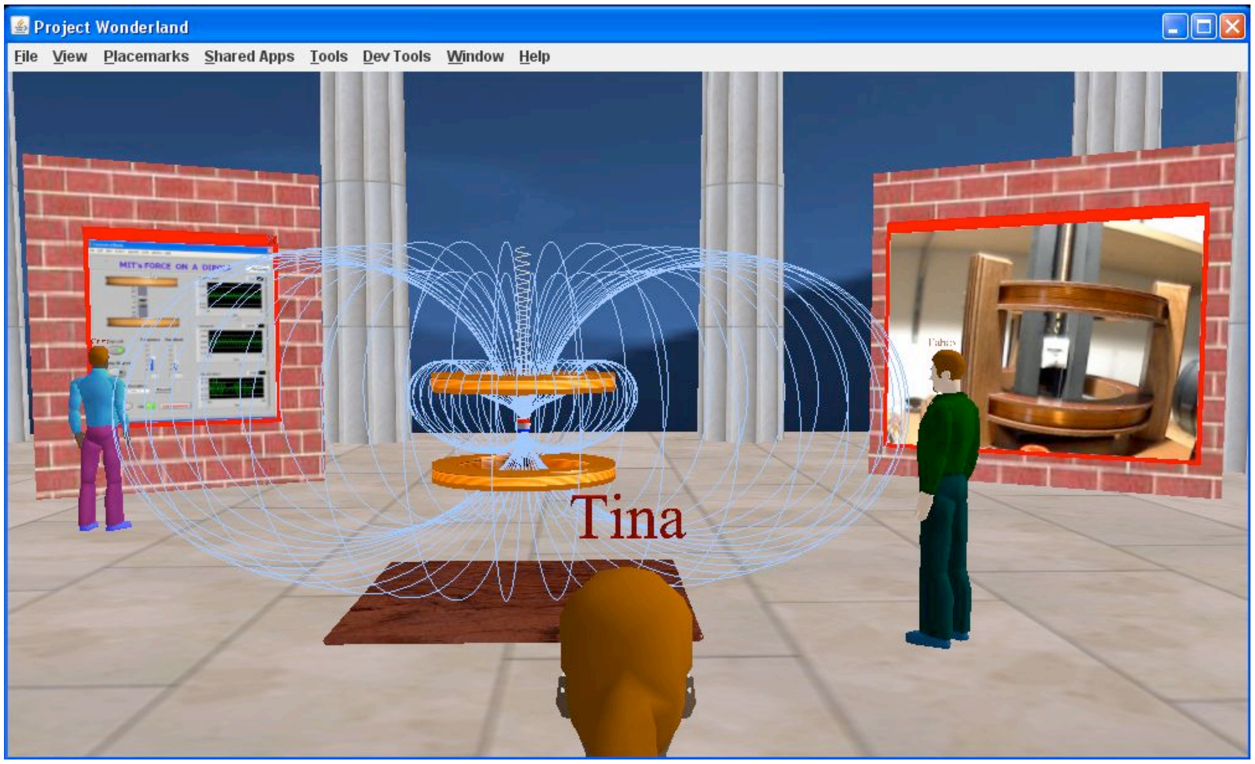
Project Wonderland
File View Placemarks Shared Apps Tools Dev Tools Window Help
MIT's FORCE ON A DIPOLE
Tina

FIGURE 1: CLOSE-UP VIEW OF THE 'FORCE ON A DIPOLE' EXPERIMENT IN WONDERLAND 0.4

2. BACKGROUND TECHNOLOGIES

2.1 Terminology

During the course of this report we will mention, cite and use various terms related to technologies, such as programming languages or rendering engines, and also often refer to other similar scientific projects. For better reader comprehension, we decided to dedicate some words to clarify some important terms, so our readers can easily follow the technical terminology used during the development of this project.

In computer graphics, a scene is constructed using a scene graph and data structure, which represents all objects to be shown in a graphical virtual environment. A scene graph is built from nodes in a graph, where a node normally can have only one parent. One single operation applied on one node can propagate to all its children. For example, it is easier to move or transform objects grouped into a compound object (rather than) as all single objects separately.

Nevertheless, it is difficult to codify how a scene graph should be, since programmers very often change and adjust its principles to fit their applications.

Beginning with the lowest level, APIs such OpenGL, Direct3D and QuickDraw3D are responsible for the primitives, which allow for the creation and manipulation of 3D objects.

OpenGL is a hardware independent library designed for efficient processing of 3D applications, although it can also handle 2D scene descriptions (z=0). This API has recently become the most widely used and supported in the industry [12].

In 3D computer graphics we can construct a virtual environment using Java3D, which is a graphic library developed for the Java language. We can use high-level constructs to create and manipulate geometries. Additionally, we can produce three dimensional web displays with Java3D. Substantial virtual worlds can be efficiently defined and rendered with this API. Java3D is an interface on top of OpengGL or Direct3D and it’s based on the concept of scene-graph. It will become clear later in this session how the comprehension of Java3D supports the development of this project.

Java 3D acts on a gap between VRML, that focus on describing 3D content, and OpenGL, which is an API based on C, used in rendering geometries as triangles, lines, etc.

The process of generating an 3D image from a model is carried by the rendering engines. Due to lack of graphics engine written in Java, the open source jMonkey Engine (jME) was built [7].

jME is an API dedicated for high performance scene graph based graphics; in other words, a 3D framework for Java developers. One of its greatest advantages is that it is a cross-platform software, having 3D games that can run on Windows, Mac OS or Linux. The feature list of jME is extensive and includes embedded integration of Java Applet and SWT (Standard Widget

Toolkit). JME Desktop System is responsible for the rendering of Swing components in jME scenes.

In the Wonderland, jME help us to specify 3D objects in a scene. This concerns an object's size, appearance, and their relation to each other. jME can perform not only computer graphics operations such as lighting and texture, but also complex techniques like particle systems.

Nodes in jME can be either Leaf or Internal nodes. The first, also called geometry, contains data to be processed and core game elements, and has no children. The last can be also referred to as "nodes" and are for management purposes. Each node contains important information about itself such as transforms, bounding volumes, render states and controllers.

Figure 2 exemplifies state of the art shader support.

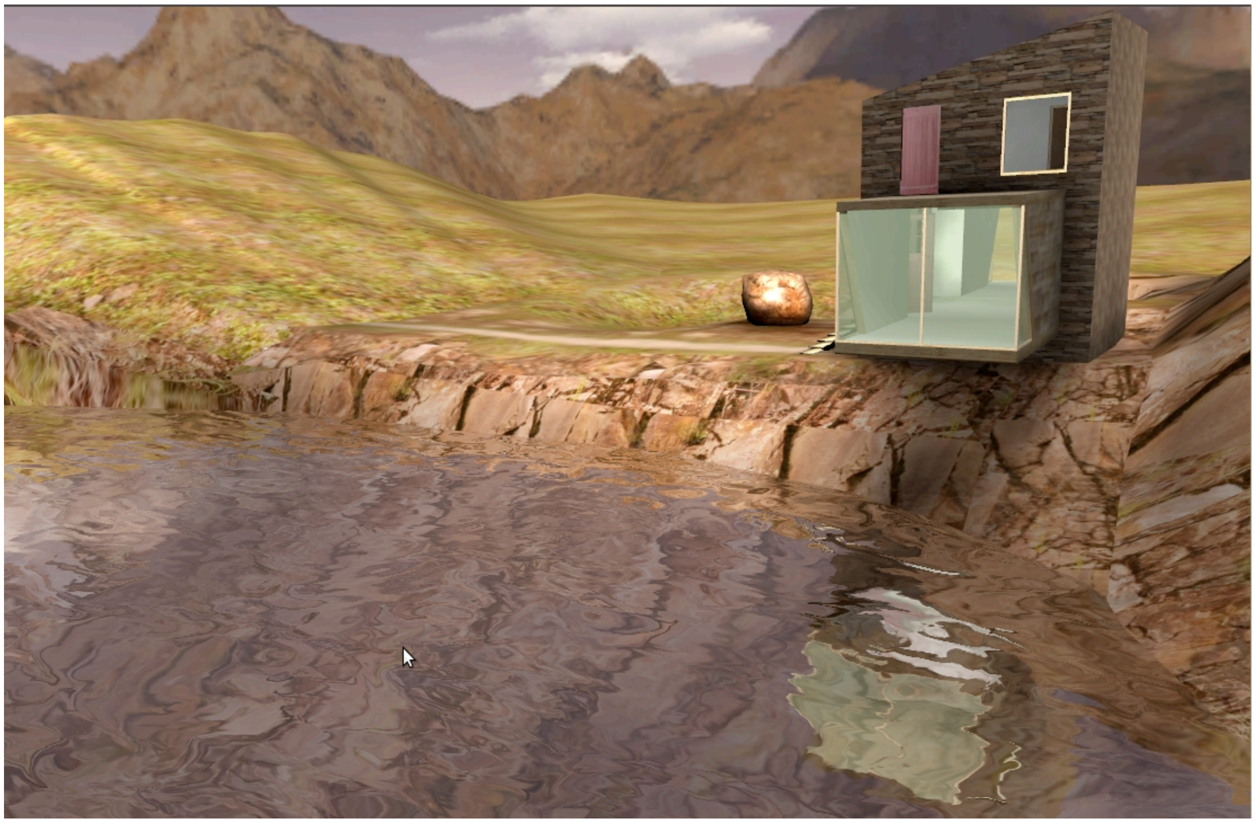

FIGURE 2: ART SHADER SUPPORT IN JME

Above the jMonkey Engine scene graph, we have MTGame as a Multi-Threaded Game engine. His design was driven by growing number of multi-threaded client systems. Thus, a game engine, which could benefit greatly from this, could process even more complex worlds. Its main purpose is 3D graphics content processing and rendering. MTGame extends the capabilities of jME, offering a fully parallel processing model and a fully featured rendering system, which supports all major rendering techniques. In addition to that, three plug-able systems are included: picking, collision, and a physics and input system for some input devices. Essentially, MTGame is a library for scene graph management and concurrency management.

Also, what is particular to MTGame is its component based design model, which makes the systems easy to extend by developers. New components can be added to the systems without affecting other system components.

Animations and some other changes that happen in wonderland scene graph over time are managed by MTGame. Developers can use MTGame to animate various 3D Object attributes like position, color and lighting.

Here, a list of all the features of MTGame that extend jME capabilities:

- A fully parallel processing model.
- A fully featured rendering system with support for all major rendering techniques.
- A pluggable Picking/Collision system that supports both queries and collision resolution.
- A pluggable Physics system that allows for dynamic simulation of real world physics.
- A pluggable input system for processing of Mouse and Keyboard events as well as other
- devices. [6]

Remote laboratories are real equipment laboratories that can be operated and controlled remotely through an experiment interface [13]. Compared to a conventional laboratory, remote labs make it easier a lot of processes, including scheduling of equipment. Less lab space and staff are required. Extremely expensive experiments can be efficiently shared among students and staff living in geographically remote locations.

2.2 Related Projects

iLabs

An iLab stands for an internet-accessible experiment that can be reachable remotely at any time over the internet. Compared to conventional laboratories, iLabs are available 24-7 and are easily shared. Since they provide reliable access to unique remote resources, they are less expensive and complex than conventional labs. Users can access these online laboratories from around the world through a single standard administrative interface. In engineering education, iLabs enrich the scope of experiments students have available to them in the course of their academic careers [4].

In order to illustrate the iLabs project we have Figure 3 showing the architectural overview of the three-tiered iLab system. Clients and servers communicate using Webservices/Communication between clients and servers is powered by Webservices. Shared generic services such as user authentication, credential management, registration, and authentication are handled by the Service Broker. It is not necessary for all this administrative tasks to be implemented for each different Weblab. The user can submit experiments and manipulate parameters through a virtual interface. The Client is implemented here in this example as a Java Applet running on the student’s browser. The third tier is the Server, where is actual experiment is running, so the test hardware and the device under test. Once the experiment is finished, the Server provides the Service Broker with results.

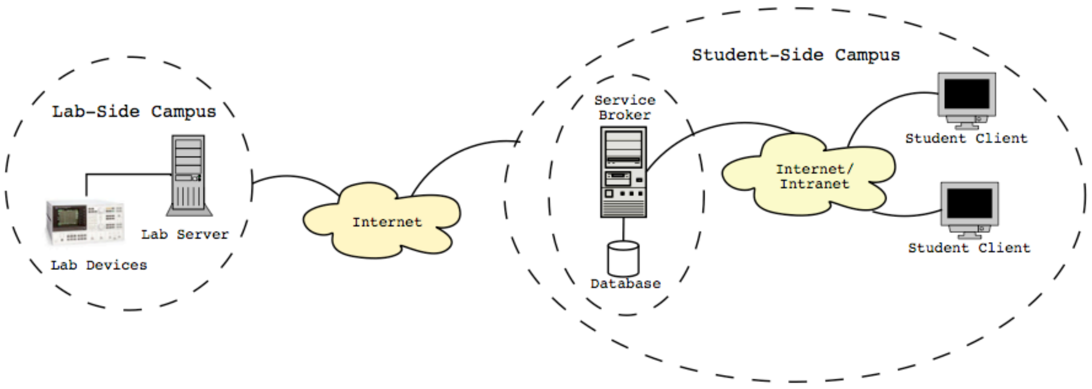

FIGURE 3: 3 TIER SYSTEM

iLabs considers three different categories of online experiments. When the entire course of the experiment is defined before the experiment itself begins, we call them Batched Experiments. For example, in order to characterize different semiconductor devices, we prepare a test protocol before the execution. Sensor experiments enable users to analyze real data stream without influencing the event under observation.

The last category, interactive experiments, refers to those experiments where the user can both monitor and interact with an experiment during its execution. Students can change input values and the system reports the consequent results dynamically. In this project we will explore the last category of experiments further. [14]

TEALSim

TEALSim, the TEAL simulation system, is designed as a framework for authoring, presenting and controlling simulations in a variety of domains. It was developed by the Technology Enabled Active Learning (TEAL) Project at Massachusetts Institute of Technology (MIT) [3]. With TEALSim and its API, developers can produce full featured interactive simulations, without having to touch the inner system. The system's flexible framework enables more experienced programmers to extend system functionality in order to meet their own requirements. Among others, the objectives of this project are to increase students' conceptual and analytical understanding of the nature and dynamics of electromagnetic fields and phenomena, and also to foster students' visualization skills.

Figure 4 illustrates an example of how TEALSim is useful in electromagnetism, by helping students visualize and process phenomena. During the Force on a Dipole experiment, TEALsim enables students to see the invisible magnetic field lines, which, naturally, are not visible in real settings. This visualization behaves according to changes in simulation input values made by students, giving them a better understanding of electromagnetic fields. Such visualizations allow students to make abstract ideas concrete. [15]

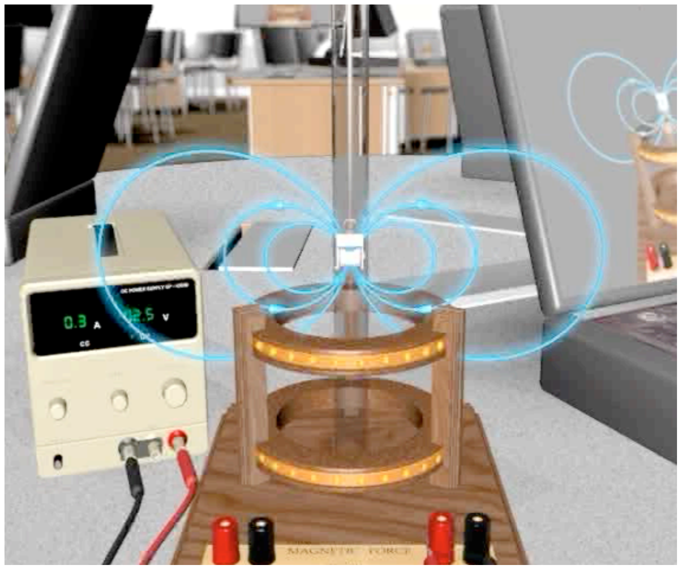

FIGURE 4: THE TEACHSPIN™ APPARATUS FROM THE MAGNETOSTATICS SESSION.

Open Wonderland

Open Wonderland is a toolkit for building 3D virtual worlds, which has been developed by Sun Microsystems Laboratories [5]. Previously, Open Wonderland toolkit was known as Project Wonderland, an open source project developed and funded by Sun Microsystems Laboratories. Since the beginning of 2010, Open Wonderland is governed by the Open Wonderland Foundation, which is a non-profit corporation, committed to maintaining and enhancing Wonderland as a virtual world platform.

Based on 100% open source Java technology, Wonderland is extensible so developers and graphic artists can extend its functionalities to create new collaborative 3D virtual worlds. The focus is on business and education collaboration. Wonderland also supports a high level of communication via highly immersive audio, and enables desktop application sharing, among other features.

Here is a brief overview of Wonderland's architecture. Wonderland's client is designed as a browser for wonderland world, where each world has different content and behaviors. Therefore, clients connect to a Server to download contents. The Server side hosts a number of different applications managed through the web. The figure 5 shows the Wonderland software stack for the 0.5 Version.

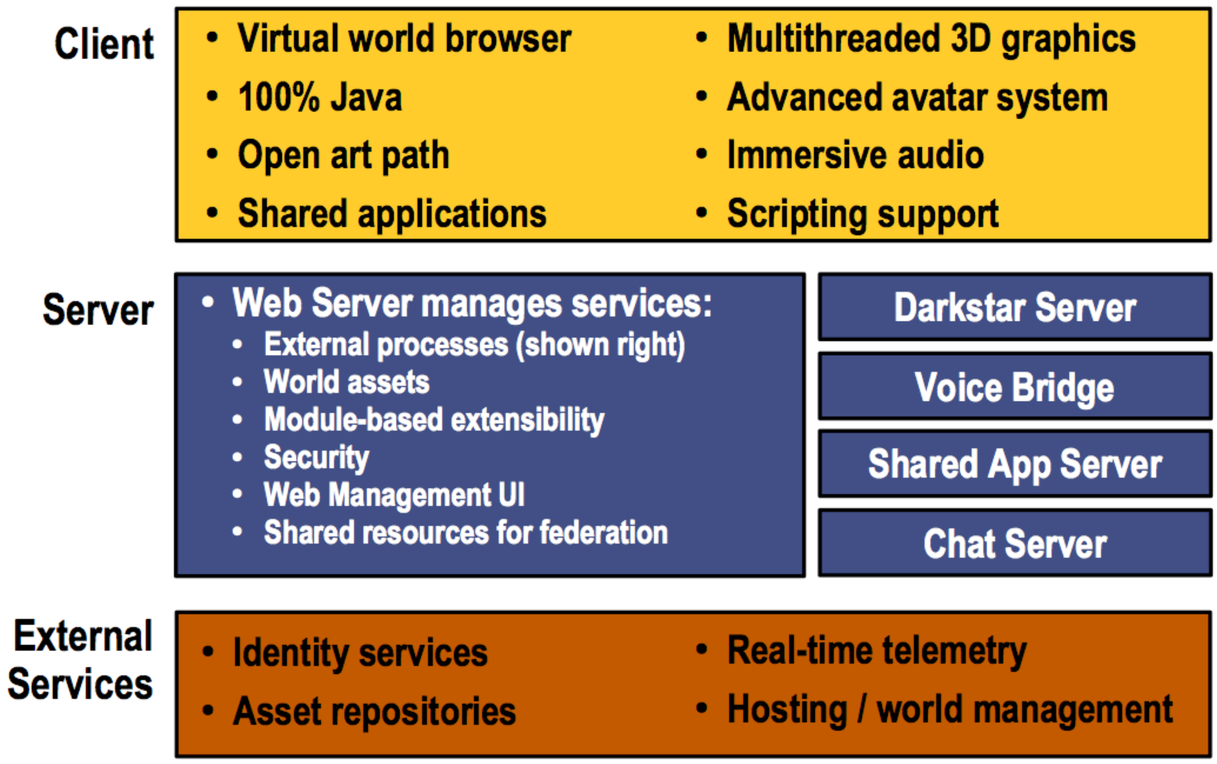

FIGURE 5: WONDERLAND SOFTWARE ARCHITECTURE, SUN MICROSYSTEMS.

Wonderland is organized as a modular system. This can be seen as a mechanism to facilitate packaging and sharing of wonderland extensions. Each module can contain artwork, code, scripts, assembled worlds and other resources.

In the new redesigned version 0.5 of Wonderland, JMonkey Engine is one of the supported rendering engines. In the Wonderland project, jME helps us to specify 3D objects in a scene [5]. This concerns an object's size, appearance and how they are related to each other.

Besides the increase in performance, another important feature of version 0.5 is the possibility of embedded Swing components for user interface development in the world. By creating a swing-based interface, we can interact directly with the experiment instead of interacting with a LabVIEW application through a VNC viewer.

Regarding the assembly of virtual worlds, Wonderland has an open art path. It supports the direct import of .kmz models designed with the Google tool Google SkechtUP. Additionally, it can import models generated by other industry standard tools such as Photoshop, GIMP, Maya and Blender. The majority of the time, we export our models as Collada Files.

Content can be added to the world using the web browser and later edited and resized using in-world tools.

A world is divided into Cells, which are organized similar to scene graph tree structures. So, a node in a wonderland scene graph corresponds to a Cell, where we hold data about a visible object in the scene. Examples for a visible object could be a web browser or a 3D interface element.

3. Collaborative Virtual Learning Environment

3.1 Overview

In CLVE students are represented as avatars, through them users communicate with each other, cooperate with other students or educators to solve common physics problems and experience physics experiments. This Virtual Environment extends the real studio physics classroom in many ways in order to support students understanding of physics concepts [15].

Students are able to interact with a remote experiment through a 2D LabVIEW front panel, for example changing experiment input parameters such as amplitude or frequency. This panel has been presented via the graphical sharing desktop system Virtual Networking Computer (VNC) until now since the LabVIEW application is running remotely in the computer wired to the experiment.

Synchronously, a 3D visualization of the simulation is generated from the input and output values in the experiment equipment. For instance, in the case of the Force on a Dipole Experiment, shown in Figure 1, students are able to compare the generated magnetic field lines with the real physical experiment, which is streamed in real time through a video camera. The network camera is provided by the iLab ServiceBroker.

Due to the lack of support of swing components in the 0.4 Version of Wonderland, a button was developed to start the virtual representation of the experiment and synchronize with the real hardware.

Furthermore, a whiteboard was added to the environment to use for post experiment discussion results. Textchat functionality is also provided. These tools give students the ability to interact with each other during the experiment and discuss results of their work, as it is in a real face-to-face collaboration.

3.2 Challenges to Address after first Prototype

In this section, we will cover several different ways of dealing with issues concerning our prototype in wonderland 0.4 version. Some of them concern the overall software architecture and its performance and scalability, while others refers to the pedagogical values of a such a tool for graduate and undergraduate level courses.

The fifth version of wonderland clients relies on modified jME classes to, among other things, specify 3D objects in a scene--specifically, an object's size, appearance, and how they are related to each other. Therefore, we have to integrate jMonkeyEngine (jME) as an optional rendering engine for the TEALSim project in Wonderland. We thoroughly explore the process of conversion of the TEALsim simulation package to the rendering engine jMonkeyEngine (JME). Our goal is to have a Wonderland cell class that can dynamically load TEAL simulations into Wonderland by the end of the research project.

In the previous virtual environment version, a button was implemented in order to start and stop the virtual simulation of the real experiment synchronized with the real hardware. The main reason for this process was the lack of support of swing components in the 0.4 Version of Wonderland. In the newest version, we are able to wire the Wonderland swing provided API to the TEAL simulation objects on our simulations.

During the development we were not quite sure if we should implement either 2D HUD (Heads-up Display) or 3D interfaces in Wonderland cells. While a 2D version of a swing element can boost performance and remove clutter from the plain view display, 3D user interface elements closely resemble real objects, providing a highly immersive environment. We have been carrying out some test and short evaluations to discover advantages of different types of interfaces in such learning scenarios. Apart from the fact that students enjoyed dealing with 3D elements, we believe that in the cognitive domain, such objects can improve memory spatial use, since students can easily recognize and memorize these objects. [15].

Similar to other international researchers on remote labs and educators, we are also strongly interested in the pedagogical values for graduate and undergraduate level courses.

3.3 Redesigning the Collaborative Virtual Learning Environment (CLVE)

The redesign consists principally in the development of a new collaborative virtual learning environment, taking into consideration previous evaluations of our proof of concept and some of the new features of wonderland presented in the fifth version. For the new collaborative environment, new objects were created using content creation tools available such as 3ds max or SketchUp. Moreover, we implemented an automatic generation of buttons, sliders, and other control elements as 3D objects in world and/or as elements in the HUD. It is generally worth pointing out the several issues related to the modification of TEALSim architecture, the process of incorporating jME in TEALsim, and the controversy over 3D versus 2D user interfaces.

3.4 Developing new models and importing them into the world

There are two different ways to import art work in Wonderland: per drag and drop or thorough the import-model Menu item. While the first one targets end users, the last one is adapted for developers. Our models are developed using 3ds Max and further exported as collada files. The XML schema Collada makes easy the interexchange of 3D digital assets.

3.5 Adapting the TEALsim Architecture

TEALsim is separated in three main parts according the Model-View-Controller design pattern: Model represented by the simulation, user interface corresponds the control and View constitute of viewer and renderer. Interfaces, such as TElement, provides the basic functionality of the element, leaving further customization to its implementation [8].

A typical simulation scenario will include a simulation engine, a viewer and UI element and objects being simulated. TSimulation brings all this components together and is responsible for the entire logic in a particular simulation. SimPlayer loads a TSimulation object so that it can be presented to the user. This class is an implementation of TFramework.

The process of creating simulations will not change considerably through the integration of TEAL with Wonderland. The integration will provide modules, extensions to Open Wonderland, which implement TEAL framework. Once installed, such modules will provide simulation engines that run within the Wonderland environment and manage the communication between server and client, the last being where the render engine will be executed.

The Viewer, through the rendering engine, is responsible for displaying 3D simulation elements on the screen. It also controls the interaction with rendered objects. We have been working on redesigning TEALSim code to support later rebinding, using java3D or jME engines; (also changes the world engine)

For several reasons, the viewer is tightly coupled to the SimEngine. The simulation engine has to inform the Viewer as soon as a simulation object has changed. In other words, the viewer should report back to the simulation every time a user manipulates a visual object in the viewer.

Our original TEALsim design was a monolithic process, whereas now we are moving towards a client-server model. The Client is concerned with the rendering aspect, while the control is handled together by clients and servers.

3.6 Incorporating jME in TEALsim

Basically, three TEAL packages should be redesigned: scene graph, render, and geometries.
As Java3D API, jME is also a scene-graph based rendering engine. A scene graph organizes the data in a tree structure, normally spatially. In the case of the jME, scene graph nodes can be called either Nodes or Geometries, depending if we are considering an internal node or a leaf node.

Node's relevant information, like transforms have to be considered during the integration, because Java3D uses different utilities classes. Java3D fundamental types are slightly different than jME ones. These are Matrix,Vector and Quaternion.

Vector3f, Vector2f, Quaternions, Matrices are in jME float because LWJGL only supports float. That is, our double precision values handle the application side and are converted to float at the scene graph level.

Besides other minor changes, we start to substitute Java3D native geometries through jME fundamental shapes: box, cone, cylinder, sphere, tourus.

A factory method is used to facilitate the rendering engine migration. This method just instantiates an object. They are useful here because we can leave the concrete implementation of the conversion to the factory method. In the simulation we didn't know whether to create Java3D or jME instances. Instead we can use "render" and leave the instantiation of implementation to a factory method.

3.7 MTGame

Although jME is a robust scene graph based rendering engine, it does not provide everything necessary for rendering 3D real time simulation in Wonderland.

MT Game is a full performance Multi-Threaded Game engine, which fulfills this need and also takes advantage of new multicore client systems. In the API hierarchy, MT Game sits on top of the jMonkey engine scene graph and the Wonderland 3D client sits on the top of MT Game.

It extends the JME features by providing a fully parallel processing model and its rendering system supports all major rendering techniques. Besides that, it offers plugable systems including picking and collision resolution and an input system for processing not just mouse and keyboard events but also other devices. In addition, we use MT Game collada model loader to import our models into Wonderland.

MT Game is designed following a component model; this enables us to dynamically add new features to new models with almost no programmer intervention. All data will get into the system through a component. In this model our base object is called the Entity object, which is simply a container object for components that are responsible for the object visual and behavior. These components are created and managed by Managers. The Worldmanager object provides the access to all four Managers in the system: Render Manger, Input Manager, CollisionManager, and Physics

3.8 User Interface Considerations

As relevant as the graphic engine are the interface controls. Several studies have shown that not well-planned navigation, complex user action models and annoying conclusions normally slow performance in the real world in the same manner that a 3D interface does [9].

This is the reason why we redesign the environment to integrate TEAL in Wonderland. Figure 6 shows the top view of our new environment. We always keep in mind to give the user in the virtual world fast situation awareness through effective overview of the simulation and we are always engaged to provide a meaningful feedback for user actions.

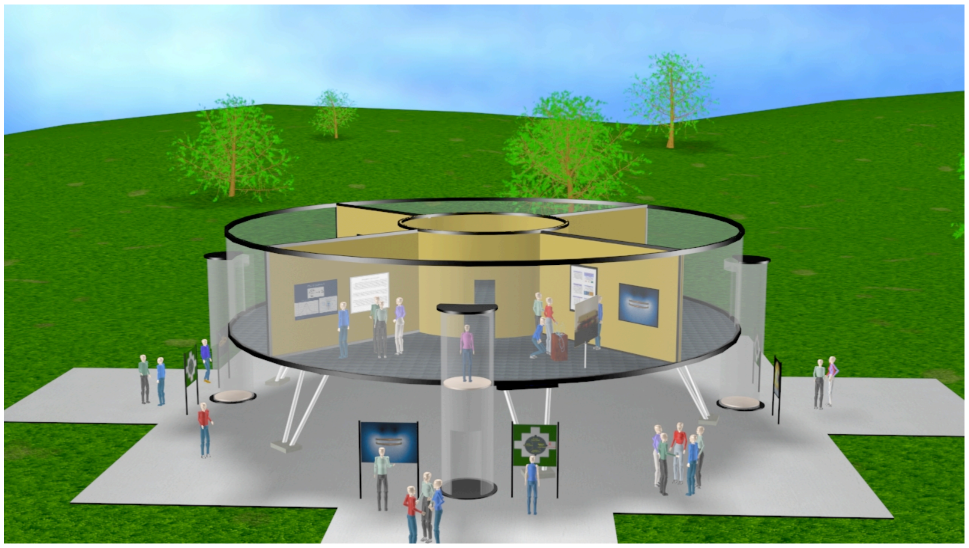

FIGURE 6: THE REDESIGNED COLLABORATIVE VIRTUAL LEARNING ENVIRONMENT IN WONDERLAND INTEGRATED WITH TEAL SIMULATIONS.

The controversy over 3D versus 2D interfaces is also present in Virtual Worlds. In scientific visualization we consider 3D as necessary to the number of tasks involving continuous variables, surfaces and volumes. However for other applications, a better strategy would be to explore variable relationships in two coordinated diagrams to discover trends, outliers or gaps.

Usability testing is more than essential for successful user interface design. So, for the development of a new prototype, which will integrate all the simulations from the TEAL project, we considered findings of our latest evaluation of the proof-of- concept [10]. In this study, we could compare how effective it is to be immersed in a 3D environment for educational purposes in traditional settings, specially related to understanding physics concepts.

We start by grouping some high level objects together. 3D Simulation visualization, the interface controls and the camera were placed next to each other to provide the students an effective overview of the situation. This organization allows students a rapid visual search of any item. We have noticed that by doing so, students could accomplish their tasks with a reduced numbers of movements and clicks, thus reducing navigation complexity. Figure 7 illustrates this organization.

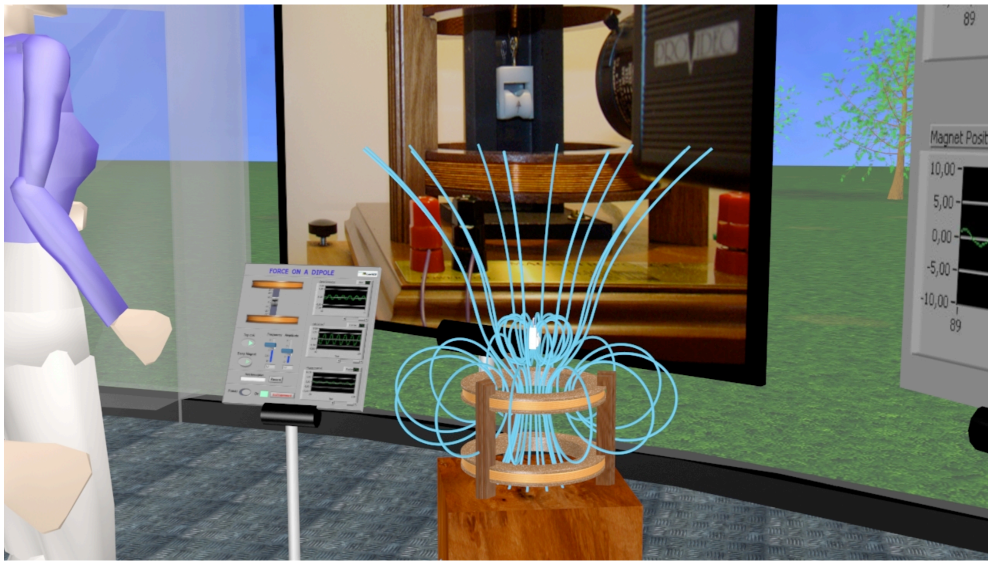

FIGURE 7: THE REDESIGNED FORCE ON A DIPOLE EXPERIMENT IN WONDERLAND.

By the time of the development we were not sure about the implications of implementing either 2D HUD (Heads-up Display) or 3D interfaces in Wonderland cells. On one hand, a 2D version could boost performance and remove clutter to the plan view display. On the other hand, 3DUI can look similar to real world objects leading to a highly immersive environment. Our buttons can appear raised or depressed. Students normally enjoy these interfaces, according to the students preferences documented during the evaluation. In the cognitive domain we believe that students would easily recognize and memorize these objects because they improve spatial memory use [11].

Simulation elements in TEAL can communicate with any other element in the world since all objects must implement TElement interface. Its functionality includes support of Routes and ProperyChangeEvents. In such a manner, simulation objects can communicate with UI components. A property of a simulation object can be manipulated if we connect a UI element such as a slider. In our simulations, the Wonderland swing provided API is wired to the TEAL simulation objects.

Open Wonderland HUD consists of 2D windows, which appear above the 3D scene and they are not shared with other in-world users [5]. They can be either visible or iconified. We developed customs HUDComponents (HUDButtons) using the Java(TM) Swing GUI toolkit to control and change some parameters of the simulation. To display the HUD component the user should select the simulation check box menu item. Since the Wonderland client core itself uses Swing to implement HUD, both are local GUI only and not shared among the users.

Parallel to this implementation, we use the Wonderland API to create 3D objects and make them react to user input events. The interface based on 3D objects can be manipulated by multiple Wonderland users, instead of HUD, which can only be presented to one user. For the Force on a Dipole we developed a set of buttons to control the amplitude and frequency parameters and to turn on and off the simulation and the coil on the top corner. The great advantage of having 3D interfaces is that these buttons are shared among the other Wonderland clients. By organizing all the buttons next to the experiment, we avoid the unnecessary visual clutter caused by the larger VNC panel.

To define how big the 3D buttons are and how they look, we use the jME API. For the dynamic behavior of the 3D objects, such as the button animations we use MT Game processor objects. In order to display 3D interfaces we create a RenderComponent. Otherwise nothing could be displayed. Buttons can only react to user input when they are "pickable".

To make them pickable we need to attach to their nodes a collision component. Every time users press the button, the simulation opens a data socket to the lab view application. The TEALsim simulation engine receives the data through the socket channel and sends it back to update simulation objects to reflect these new values. As we mentioned earlier also part of the simulation engine rule is to inform the viewers rendering engine if there is any visual change necessary to be made. Having this 3D interface near the visualization minimizes the number of navigation steps for students to accomplish their tasks in this environment. This also makes the environment more game-like, being dynamic and enjoyable for students and educators.

3.9 TEALSim & Wonderland Integration

The integration of TEALsim with Wonderland will not significantly change the process of creating simulations, but will provide Wonderland modules and cells that implement the TEAL framework. These modules will provide simulation engines that run within the Wonderland environment, render engines that execute in the Wonderland client, and manage the communication between the server and clients. In Wonderland a module is similar to a plug-in, just by including the TEALsim module and specifying the top level Java simulation class, it will be in your world.

As we mentioned, the TEALSim rendering engine is implemented as a Java3D based Viewer, which also handles the explicit rendering of the scene. Since Open Wonderland in its version 0.5 uses jME as its main rendering engine, we considered also a redesign of the TealSim 3D viewer. For many reasons, the viewer is tightly coupled to the SimEngine. Every time a simulation object has changed, the simulation engine has to inform the Viewer. Additionally, each of these elements has an associate visual representation, now as jME geometry. In the other way, when the user manipulates a visual object in the viewer, this should report back to the simulation.

4. Technical Evaluation

From the technical point of view, there have been some client architectural changes in the project. In the last section, we mentioned that until Wonderland's fourth version, rendering was primarily done using Java3D APIs. The 5th version of Wonderland toolkit has as graphic subsystem jMonkeyEngine (jME), which is, like Java3D, a scene graph-based engine. Its 3D game engine, written in Java, provides core graphic API and some graphic primitives. The feature list of jME is well extensive and includes, among others, embedded integration of Java Applets and SWT (Standard Widget Toolkit).

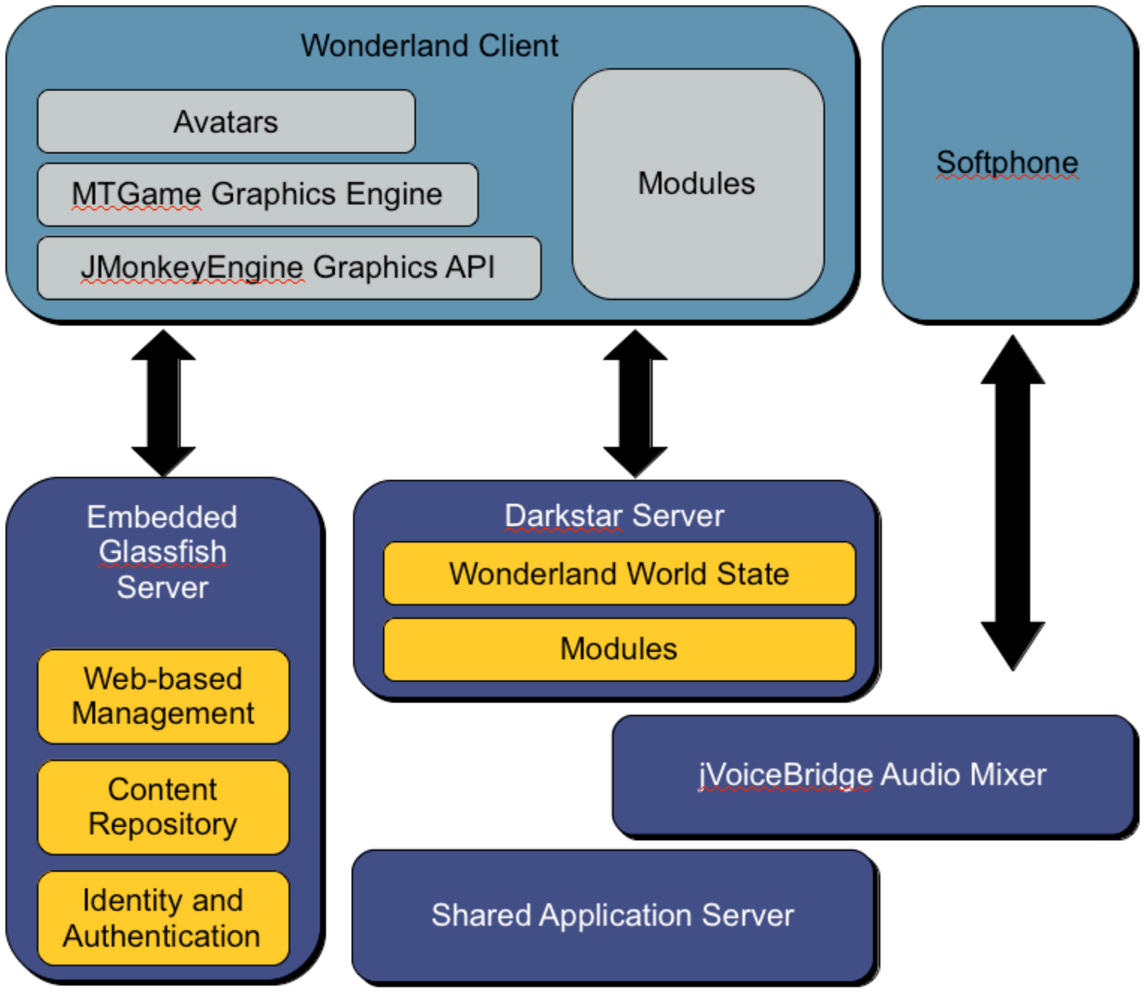

FIGURE 8: 4. WONDERLAND CLIENT ARCHITECTURE. [5]

In order to support multi-threading, some extensions in Wonderland graphics system were developed. Although jME is a robust scene graph-based rendering engine, it does not provide everything necessary for rendering 3D real time simulation in Wonderland. MT Game is a full performance Multi-Threaded Game engine, which fulfills this need and also takes advantage of new multicore client systems. It provides a fully parallel processing model and its rendering systems supports all major rendering techniques. In the Figure 8 we can see where jME and MTGame fit in Wonderland client architecture.

Besides the gain in performance, another important feature of version 0.5 is the possibility of embedded Swing components for user interface development in world. Wonderland HUD (Heads-up Display) consists of an area that includes the whole scene where 2D windows appear above the 3D scene and are not shared with other in-world users [5]. They can be either visible or iconified.

Scalability depends on the bandwidth between server and its clients and also on the resources available to the server. During Open Wonderland community sessions, we noticed some performance issues in meetings between more than 30 avatars.

Wonderland Community has yet to agree on a set of minimum system requirements. However, in order to have a smooth experience with wonderland, some general technical requirements must be met. First, you should have a game-like hardware on the client side. A typical client configuration should contain a ATI or nVIDIA graphic card with 128MB video memory, 3D accelerated graphics and at least 2 GB RAM.

Regarding the Java Version, in case it’s not present already, Java SE 6 should be installed on your system.

For developers, besides having Java SE 6 installed on their system, should install two more tools: Ant Version 1.7.1 and Subversion.

Open wonderland provides a very good performance table containing client specifications and their respective results after the performance test. Our goal is to have consistent, smooth motion. In order to possess meaningful performance information, we focus on frames per second (FPS) measurements tool to display the frame rate in the default garden arches world. FPS strongly influences user experience in a virtual environment. In video games, they refer to the speed of image refresh. All game interaction relies on an effective impression of movement, which a low FPS cannot deliver. Rates between 30 fps and 110 fps are considered acceptable among gamers. Since FPS varies depending on the current computational load, and that is what currently happens at a given moment, from one second to the other, we have to consider average FPS values under certain conditions. Animation can become choppy if a process is running on the background. The following table illustrates one system tested by the community.

Wonderland version	0.5-dev (rev. 4425)
Computer make/model	Apple MacBook Pro (MacBookPro4,1)
Graphics card make/model	NVIDIA GeForce 8600M GT (512MB VRAM)
CPU	Intel Core 2 Duo (2.6GHz)
Memory	4GB
Operating System	Mac OS X 10.6.3
Java version	1.6.0_17-b04-248-10M3025
Average FPS	24.6

TABLE 1: SYSTEM EXAMPLE TESTED BY WONDERLAND COMMUNITY.

5. Current Status and Future Work

We plan to finish the TEALSim integration into Wonderland at the end of the Summer 2010. A stable version of this module should be running by April 2010. Depending on time and the level of integration of the game engine at this time, we will start to experiment with the NPC avatars or semi- autonomous tutors, making them intelligently react according to data stream.

Additionally, we are involved in the development of 3D interactive games, where students should be able to understand how magnetic forces and fields behave in nature.

6. Conclusion

This research investigates the learning benefit for students and educators by using avatars in virtual worlds for collaboration and educational undertakings. We strongly believe that CLVE plays an important role converging collaborative technologies and tools such as video, graphics and real time simulations.

Such a game like design, keeps students interest even though they are physically remote. Combined with peer cooperation, the real time visualization helps students fully understand the dynamics of electromagnetism [2]. More and more, students and educators are agreeing on the additional value of CLVE as an educational tool, fulfilling students and educators needs for an environment for remote communication and collaboration. Once the students see the same behavior at the same time, it is easier to cooperate on misunderstood concepts. Other than 2D applications, here users could explore our 3D space analyzing TEAL simulations from different locations in the room.

Moreover, the combination of a collaborative learning environment with internet-accessible iLabs is a less expensive solution for educators, because both are based in open source technology and sharing of resources. Some of the experiments and simulations could be very costly in traditional settings. Both are easy to use and are intuitive.

We are also planning to evaluate the different user interfaces approach considering student's cognitive process, relevant features and entertainment. Through such a study we can polish our design and generate new guidelines.

These are the first steps integrating TEAL Simulations in our collaborative virtual learning environment. We will continue to research ways CLVE can increase even more its pedagogical value fostering the learning process. Scientific collaboration may happen in the future primarily in Virtual Environments.

Acknowledgement

We thank the Center for Educational Computing Initiatives (CECI) at MIT for hosting and supporting this research. We also thank the faculty, staff and students involved in the TEAL Project and the team of developers from Sun Microsystems. Special thanks to Mark Bessette and Tina Scheucher. This research has been partially supported by the Austrian Marshall Plan Foundation.

REFERENCES

[1] K. Jaa-aro, “Reconsidering the avatar: From user mirror to interaction locus” Ph.D. dissertation, KTH Numerical Analysis and Computer Science, Stockholm, Sweden 2004.

[2] B. Scheucher, P. Bailey, C. Guetl, V. Harward, “ Collaborative virtual 3D environment for internet-accessible physics experiments” The International Journal on Emerging Technologies in Learning, 2009.

[3] Y.D. Dori, J Belcher, “How does technology-enabled active learning affect undergraduate student’s understanding of electromagnetism concepts?” The Journal of the Learning Sciences, 14(2), pp. 243-279., 2005.

[4] V. Judson Harward et al, “iLab: A scalable architecture for sharing online experiments” International Conference on Engineering Education, Gainesville, Florida, 2004.

[5] Open Wonderland (2009)[Online]. Available: http://openwonderland.org/

[6] D.Twilleager (2008), “MTGame Programming Guide” Sun Microsystems. Available: https://lg3d-wonderland.dev.java.net

[7] jMonkeyEngine (2009) “jMonkeyEngine User Guide” Available: http://www.jmonkeyengine.com/

[8] J.Belcher, A. Mckinney, P. Bailey, M. Danzinger “TEALSim: A Guide to the Java 3D Software.” Massachusetts Institute of Technology, Cambridge, MA, Dec. 2007.

[9] B. Schneiderman “Why not make interfaces better than 3D Reality?” IEEE Comput. Graph. Appl. Nov/Dec 2003.

[10] B. Scheucher, J. Belcher, P. Bailey, F. Dos Santos, C. Guetl “Evaluation results of a 3D virtual environment for internet-accessible physics experiments” Interactive Computer Aided Learning Conference, Villach, Austria, 2009.

[11] W. Ark et al., “Representation Matters: The effect of 3D objects and a spatial metaphor in a graphical user interface” Proc. Human -Computer Interaction Conf. People and Computers XIII, Springer Verlag, 1998, pp. 209-219.

[12] Baker H. (2004). Computer Graphics with OpenGL. (3rd ed.): Pearson Prentice Hall.

[13] Gerardo Viedma et al. “A Web-Based Linear-Systems iLab” American Control Conference, 2005.

[14] V. Judson Harward et al. “iLab: A Scalable Architecture for Sharing Online Experiments” International Conference on Engineering Education, Florida, USA, 2005.

[15] F. Santos, P. Bailey, C. Guetl, V. Harward “Dynamic Virtual Environment for Multiple Physics Experiments in Higher Education” IEEE EDUCON Education Engineering, Madrid, Spain, 2010.