April 2009

Software Engineering Lobby

Aleksandr V. Ostapenko
Worcester Polytechnic Institute

Project Number. GFP0809

Software Engineering Virtual Lobby

A Major Qualifying Project Report:

submitted to the faculty of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by:

Alex Ostapenko

Date: April 30, 2009

Approved:

_____

Professor Gary F. Pollice, Major Advisor

1. keyword 1
2. keyword 2
3. keyword 3

Abstract

As new technology emerges, it provides useful tools that can be used for improving different spheres of life. The idea for this project was possible with the creation of an highly expandable and customizable virtual world environment, built using Project Wonderland, which was built by Sun Microsystems Laboratories. The main task was to create a virtual lobby for students taking a Software Engineering class, that will create an additional opportunity for interaction among students, as well as making staff more available for providing help, when needed.

Throughout this report, the results for achieving the goal of creating a basic functional virtual lobby will be explained and their implementation and limitations summarized. The major goal for this MQP involved creation of three team rooms that the students can use in order to work on their version of the project, doors with added security that protected those rooms, and a receptionist, that served as an interface to communicate with the course staff via instant messaging and to schedule meetings with the professor.

Acknowledgements

I would like to thank Professor Pollice for making this project available and demonstrating the features provided in Project Wonderland, as well as providing guidance throughout the duration of this MQP.

I would also like to thank Sun Microsystems and its community for creating Project Wonderland and providing helpful information on how to extend its functionality and answering questions, related to its development.

Table of Contents

List of Illustrations

1. Introduction

As the communication technologies are improving over time, the natural course of development is to try to emulate physical objects and events in virtual environments, both for users' convenience and for the sake of efficiency. Such technologies include the things that we take for granted, such as e-mail and instant messaging, which enable people to communicate almost in real time, whereas such a feat would have been impossible only two centuries ago, where letters might have taken days and even months to arrive to their destination.

Internet is a commodity available to a relatively high percentage of the population and as a result it gives a lot of freedom and opportunities to people, from simple things, such as online shopping to distance education. Even though it is still not very common, but it is possible to take online courses, which do not require the students to all be in one physical class room, but learn from the comfort of their home, no matter where they are physically located on the face of the planet.

The project that we have worked on can theoretically be used for such a purpose. The idea behind it is to provide a virtual lobby that gives a basic foundation to mimic a virtual place, that the students can use as a substitute for physically gathering at one place and working on the project, which is the biggest portion of the software engineering class, as well as getting help from their peers and/or the teaching staff.

To actually implement such an environment, I used the Wonderland project, sponsored by the Sun Microsystems Laboratories. This Java-based open source platform enables creation of virtual worlds and complete freedom of either using pre-made objects

or developing new ones from scratch that are used for voice communication, project collaboration tools, which is only the tip of the iceberg of the features that can be created with this tool.

The rest of this paper will be used to give an overview of virtual worlds and their evolution, as well as a thorough walkthrough of what was managed to be accomplished throughout the duration of this project.

2. Background

The existence of virtual 3D worlds can be traced back to the late 1990's which was the time when MMORPGs (or massively multiplayer online role-playing games) started becoming popular. These, however, were not very popular, because the scope of these games was rather limited and would not attract anyone other than people who actually enjoyed playing computer games. As some time has passed, the idea of virtual worlds became more wide-spread with the emergence of products that were designed to model everyday life and simulate human interaction online. One of the first and most successful of these products is Second Life, which was launched in June of 2003 and served as a base for virtual interactive communication.

This was a major step forward in virtual communication. Before such projects, the users didn't have much choice in the matter, but to use rather simple instant messaging software such as the ones provided by AOL, MSN, Yahoo, etc. where they could simply type some text, that is then transmitted to the other party and replies go back and forth. Although when webcams and VoIP telephony became available features of these messengers, that was essentially as close as one can get to simulate real-time conversations over sometimes very large distances. What makes virtual worlds a lot different from these services is the ability to model the actual human interactions, which might take place when people meet. For instance, instead of just typing "Hi" to the other users, in most virtual world environments it is possible to actually wave and perform a large variety of other actions such as shaking hands, laughing, sitting, walking, etc. Which is a huge step forward in terms of simulating real life and titles such as "Second Life" are actually quite descriptive of the capabilities that they provide.

As was mentioned in the previous paragraph, the main idea behind a virtual environment is for the person to feel like they are actually in that world. This is done through the use of an avatar or in other words the virtual representation of oneself (even though the user is of course given the freedom to choose any alter ego he or she wishes). After that is done, the user can submerge into the world, which can represent anything from a park in a city to any kind of imaginary locations, where the idea is that they will meet and interact with other like-minded individuals.

The problem with most of these projects is that, even though they are technically free, they are not customizable, closed source, and usually require monthly subscriptions to unlock features. That is why things such as Second Life and IMVU (which is another popular 3D instant messenger program) are not exactly ideal for educational purposes. Especially if they cannot be easily modified to suit the needs of the instructor, and that is exactly where Wonderland comes in.

As was mentioned before, Project Wonderland is written in Java and uses Project Darkstar (which is a highly scalable infrastructure for massively multiplayer games) as its foundation. As in Second Life, the users choose and customize their avatars any way they want and can wander around in worlds, either already provided or ones that are built from scratch with the world-building tool provided with the project. The main benefit of this project is its open source nature, which means that the code is freely distributed and can be changed by individuals or companies to suit their needs. This also includes creation of custom objects, laying out worlds that represent real offices and so on. Essentially, if the development continues at the same fast pace, this will be an invaluable

utility, not only for people interested in evolving technologies, but also as a help in today's global corporate world for sharing, presentations, and project collaborations.

Because of the lengthy list of features that Project Wonderland provides, it became a suitable candidate for this project. The main idea is to provide a virtual lobby that the students taking the Software Engineering class can use to both gather together in their designated rooms and work on parts of the project together and as an interface for them to schedule meetings with the staff or communicate directly with the professor or the teaching assistants via AOL instant messenger.

3. Methodology

Before any actual work was done on this Major Qualifying Project, the technology that serves as its foundation had to be studied in great detail. Project Wonderland, which is an experimental framework sponsored by Sun Microsystems Laboratories, is in a relatively mature stage of development at the time of this writing. Currently it is in version 0.4, with version 0.5 scheduled to be released later on this year. The main purpose of this software is to enable collaboration among people, for example if people working in the same organization are distributed across the globe and can't physically meet. What Project Wonderland does, is it can be used as a virtual representation of a company's offices, where the employees are able to choose their avatars (i.e. their virtual representations) and work on the same code, participate in presentations, work on the same files, and so on. From a communication point of view, there is also quite a bit of choice, from simply typing into the box, as is common in instant messaging programs, to using VoIP technology to talk to other participants directly. What also makes Wonderland a good choice for this kind of a problem, is the fact that it is built using Darkstar technology, which was originally designed as an infrastructure for helping develop scalable massive virtual worlds and online games.

Figure 1 - Wonderland

https://lg3d.dev.java.net/wonderland/images/2007-12-Wonderland-Various/album/slides/team-room-multi.jpg

Even though, as was mentioned in the previous paragraph, Project Wonderland is primarily designed for collaboration within a corporate world, it is not limited as such. The project is freely distributed and is open-source, which means that anyone with knowledge of Java, can pick it up and either use the existing tools that can be obtained from the Sun website (https://lg3d-wonderland.dev.java.net/) or build new tools from scratch, in order to satisfy the need at hand. Because virtual environments in general are not very common as of now (but they are already quite diverse and taking hold in many different niches), the fact that Project Wonderland is an open-source tool, gives it a lot of interesting potential in terms of research. As a result it is a perfect platform from an educational point of view, providing many ways in which it can be an enhancing feature for a traditional form of education and with just a bit of time and effort, it can be easily made to either mimic a classroom environment or create new ones that will extend the

capabilities through a variety of new tools and objects that can be developed in order to fulfill any particular set of requirements. There is quite a large range of possibilities, which is only limited by the functionality that the Java language provides.

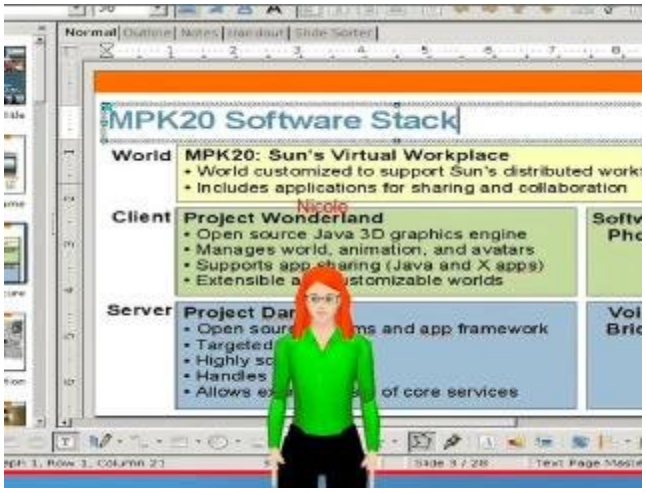

Figure 2 - Example of an object (file viewer)

https://lg3d.dev.java.net/wonderland/images/2007-12-Wonderland-Various/album/slides/MPK20-movie-snap2.jpg

Wonderland can be obtained as a pre-compiled package with an installer or as a source distribution for developers. Since the primary goal for this MQP is to create a lobby room that can be used by the students taking the Software Engineering course, the first option was taken. The complete package consists of three parts – the essential components of Project Wonderland, the additional modules (which include objects and applications developed outside the core components), and finally the textures package, which defines the look of the environment. There are also two types of releases, the standard desktop application and a web application. One of the benefits of the web application is the fact that it can be used to create the blueprints of a world, using the textures provided by the textures package. This will be described in more detail later on.

However, for the majority of functions, both types of releases use Apache Ant to deploy Java packages.

Because of the large number of possibilities the features that Project Wonderland provides, there is quite a steep learning curve. The main project site (http://wiki.java.net/bin/view/Javadesktop/ProjectWonderland) contains several Wiki articles and tutorials that explain the fundamentals of how the Wonderland file system (WFS) works and how new objects can be implemented. Quite a bit of time was spent going through these to get a feel of what Wonderland is capable of and outlining the potential plan of action that was to be taken in order to be able to implement the features that the project requirements asked for.

The project goals are defined to be straight-forward. Because this is the first phase of the process, before this technology can actually be used in a real teaching environment, the first step is to make an outline for the lobby. Since the class as taught with a term-long project in mind, the plan was to make a large room where the students will be able to interact with each other, and since the class is divided into three groups, the common lobby area is connected to three rooms that the teams can use for virtual meetings, collaboration on code and the like. In order to prevent cheating and ensure privacy, the rooms need to be secured in some way, which is accomplished by password-protected doors that, in theory, would only let people belonging to a given team the access to their appropriate room.

The goals for the project also include facilitation of communication between students and staff. The initial function that the lobby receptionist provides is the ability for students to easily schedule meetings with the professor via a simple form that is emailed

to the professor so that further actions can be coordinated. If the students need to get in touch with the staff (both the professor and the teaching assistants) they need to be able to use the receptionist as a simple instant messaging mechanism that will be implemented using the AIM protocol so that they ask whatever questions they have without having to leave the application (provided, of course, that at least one of the staff members is online at the time).

There is also another requirement - the ability for the students to be able to check important course-related information. This can be anything from mentions of schedule changes to exam and project due dates. Since this feature is already implemented through the modules that are included in the Wonderland package, it will not be covered in much detail. Suffice to say that this is implemented via a PDF viewer module that can be updated by the course staff when there is a need and is displayed right behind where the receptionist is located, in a place that is very easy to see from anywhere in the lobby.

The project began with getting familiar with the capabilities that Project Wonderland provides out of the box. This was important not only because this was my first experience with this particular framework, but also as a starting point for finding out what has already been created and what is still missing, so as not to reinvent the proverbial wheel. For instance, as was already mentioned, Project Wonderland already contains a module that can be used as an information board and very little work needed to be done in order to include it into the world. There were however a couple of choices of the actual implementation – either the whiteboard module or the PDF viewer. My final decision was to use the later, as it would provide a very simple means for the professor and the rest of the course staff to use PDF file that contained as much information as

necessary and could be updated whenever needed, without having to worry about someone else maliciously or accidentally being able to change the information that has the potential of misguiding other students.

The actual design of the floor plan is relatively straight-forward. From what the requirements specify, there has to be a large enough common area where students can talk to each other and have access to the common tools as well as the specifically designated rooms for team collaboration. The working design is shown in the figure below.

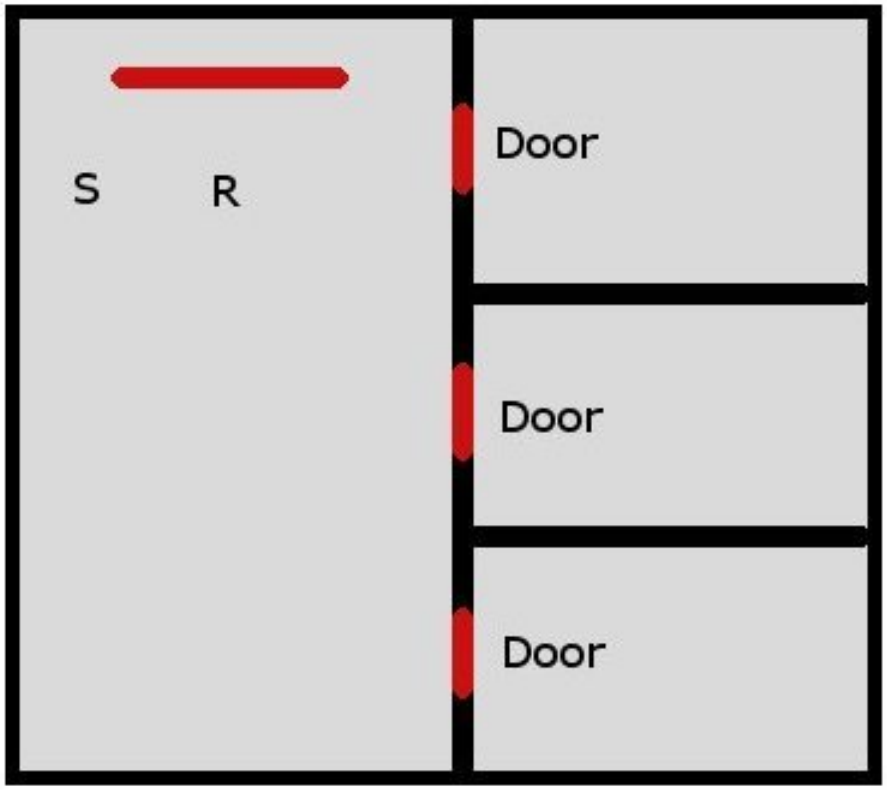

Figure 3 - Floor plan

As can be seen from the figure above, there are four main areas: the lobby and the three team rooms. The long horizontal line in the north side of the lobby is the information board which is conveniently located right next to the spawn point (or in other words, the point where users appear when they first log into the world), marked with “S”. Right next to it (marked with “R”), is the receptionist that can be interacted with as soon as the world loads, also placed there for the students’ convenience. The team rooms can be accessed by going to the appropriate door and providing the right password. Once in the room, the students can utilize the wide variety of tools for project collaboration. As of now though, these rooms are empty, since this isn’t included in the scope of this project.

The next figure gives a better overview of what the world looks like once the user is logged in (Note: this angle was chosen to show the lobby in its entirety). As can be seen, the lobby provides more than enough room to hold the whole class, if needed. The receptionist and the information board are clearly visible from any point in the lobby and are easily accessible.

Figure 4 - Actual lobby view

Because of the way Wonderland is designed, in order to implement a new object, several things need to be done. First of all, there is a server class that is responsible for storing information about a class globally, so that no matter how many clients are connected to the server, the information will remain the same throughout (of course with the exception of cases where such behavior is undesirable). Secondly, there is a client class component that is responsible with how the information about the object is presented to the user; here also is stored all the information about how the users can communicate with objects, such as what happens when the user clicks on the object, walks into it, etc. For convenience and clarity of code, a third class is used, with all the setup information that is needed for both the client and the server. Visually it can be represented as a simple client-server type of model, shown in the diagram below.

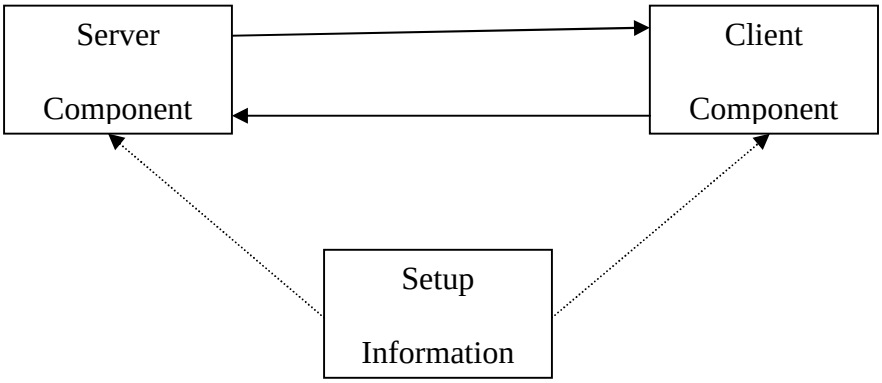

Figure 5 – Class model

As was mentioned before, the first goal is to accomplish the creation of virtual doors to protect team rooms. The tutorial provided on the main Project Wonderland site's Wiki pages was a great starting point, demonstrating on practice how the objects operated inside of the virtual world. The tutorial worked around an object that either turned into a sphere or a cube when the user clicked on it. This was essentially enough to provide a foundation for the door class, with the other features added on top.

The spirit of object oriented programming prevails in Project Wonderland mainly through the availability of the base classes that can be extended in order to provide new ones. The server component class is not especially interesting in terms of functionality. Its only real purpose for the implementation of the door is to hold some global information, so that the multiple clients that are going to be using it have access to that information, without the freedom to change it. And that is understandable, because the doors are designed specifically to be interacted with directly by the users.

Below an abridged version of the UML diagram of the Door class is provided. A lot of the functions are inherited from the more basic classes that are responsible for the actual display of the objects in the world and how the objects change based on events occurring in that world, so they are not shown here, but can be studied by either looking at the Javadoc or the actual source code of the supporting classes.

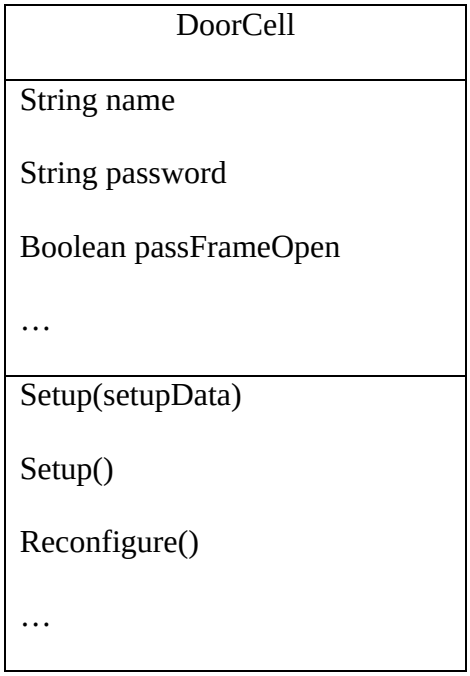

Figure 6 – Class diagram

Inside of the DoorCell.java file, there is also another class defined – MouseButtonListener, that is responsible for reacting to various events, more specifically events occurring because of the user input. As the name of the class implies, the exact nature of events is manipulation by the mouse.

The class-specific variables shown in the figure above are given the exact role that their name implies. The name variable is responsible for holding the name of the

door, which can be customized through the XML file, controlling it, same with the password variable. Because this is the initial phase of this long-term project, the passwords are not encrypted, however it is possible to create a function that will look for that specific string in the XML file and encrypt and mark that it is encrypted every time it is changed while the world is being initialized. That would, of course, also require that the client be able to decrypt the password in order to check if what the user has entered is actually the same as the set password. However, this is part of the plan for future work. The Boolean value passFrameOpen plays a more mechanical role and prevents multiple copies of Java frames being open, due to the fact that for an unknown reason the mouse events are registered several times and would consequently show several dialogs to appear for the user to input the password, which is a highly undesirable behavior.

A sample walkthrough of how the doors are used can be seen below:

1. The user enters the world.
2. The user might interact with people or objects in the lobby first
3. The user walks up to the door
4. The user uses the left mouse button to click on the door
5. If there are no previous door dialogs open for this user, a password dialog appears
6. The user enters the password
7. If the password is correct, the user is transported inside of the room, guarded by the door. If the password is incorrect, the user can enter the password again, unlimited number of tries as of the time of this writing.

This can be better demonstrated by a diagram:

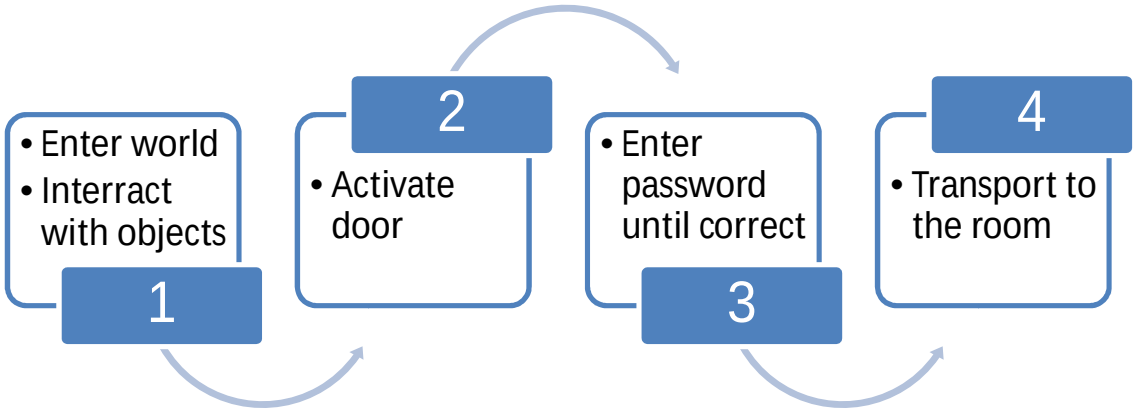

Figure 7 – Interaction with doors

What happens behind the scenes after the user enters the password is relatively straight-forward – there has to be a mechanism to check the entered password against the actual password stored in the door settings. However, there is a small difficulty for when the password is correct. Because the doors are meant to keep people out, there has to be a mechanism that checks if this is the case, so as not to ask the users for a password in order to be able to leave a room. Such behavior would not make much sense in the first place and would probably not please the users of the system. This problem is partially solved by checking the user's current coordinates in relation to the door's coordinates and thus figuring out if the user is inside or outside of the room. This works fine for doors that are positioned in the north-south orientation, with the "outside" being on the south side, however a new variable has to be added to the door XML description file that shows where the outside of the door is facing, so that the right set of coordinates are checked to determine where the user is located.

Figure 8 - Activated door

The doors are currently displayed through a texture that was created and exported using Blender, and can be changed in there. They also operate a bit differently than doors do in real life. They can actually be more accurately described as teleporters, rather than doors, because they don't open to let the user in, but rather transport the user both ways. The reasoning behind this is that if the door actually opened and closed, it would create a security hazard, where more than one user might be let inside of the room where only one is actually allowed to be in. The way it is designed now, it would be impossible to exploit such an approach, because every single user has to have a valid password to gain access.

The other object that was developed for this project is the receptionist. Visually the receptionist is represented as an animated model of a person. The initial version of the receptionist was designed to be responsible for two main roles: meeting scheduling and instant messaging communication with the staff. The former is accomplished through a form that is converted into an email message, while the latter is accomplished through the open-source SDK that AOL Instant Messaging.

When the receptionist is first initialized (by left-clicking on it) and the menu dialog appears:

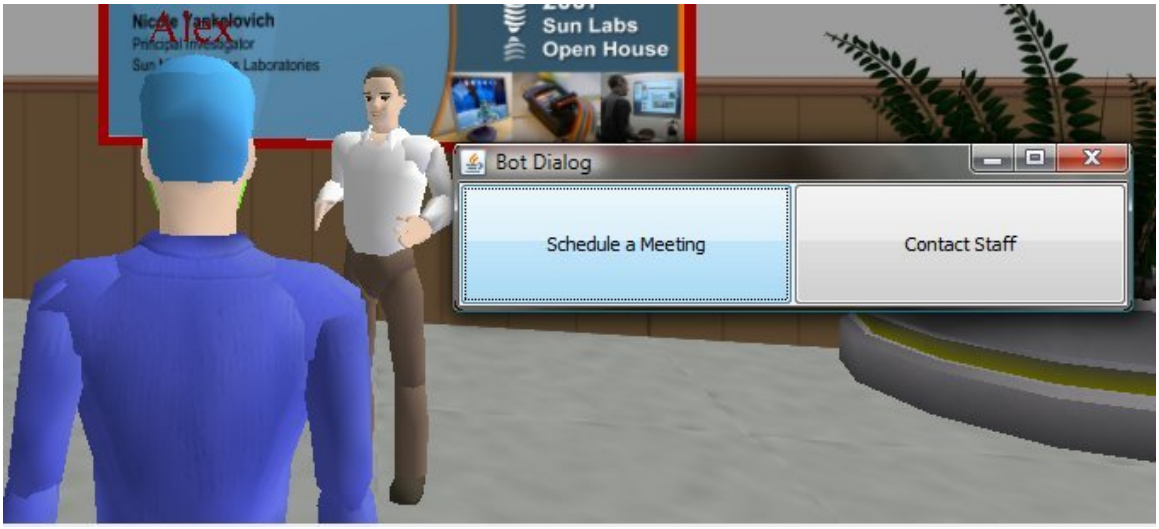

Figure 9 - Initial receptionist dialog

Whenever the user selects the first option in the receptionist dialog “Schedule a Meeting” a new dialog appears on the screen. This dialog is a form with fields for several relevant pieces of information that are usually required to schedule meetings. These include the person’s name, email address, date when the meeting is desirable and the time. The user should also indicate the reason for scheduling the meeting to make it easier for the professor. This form can be seen in the figure below.

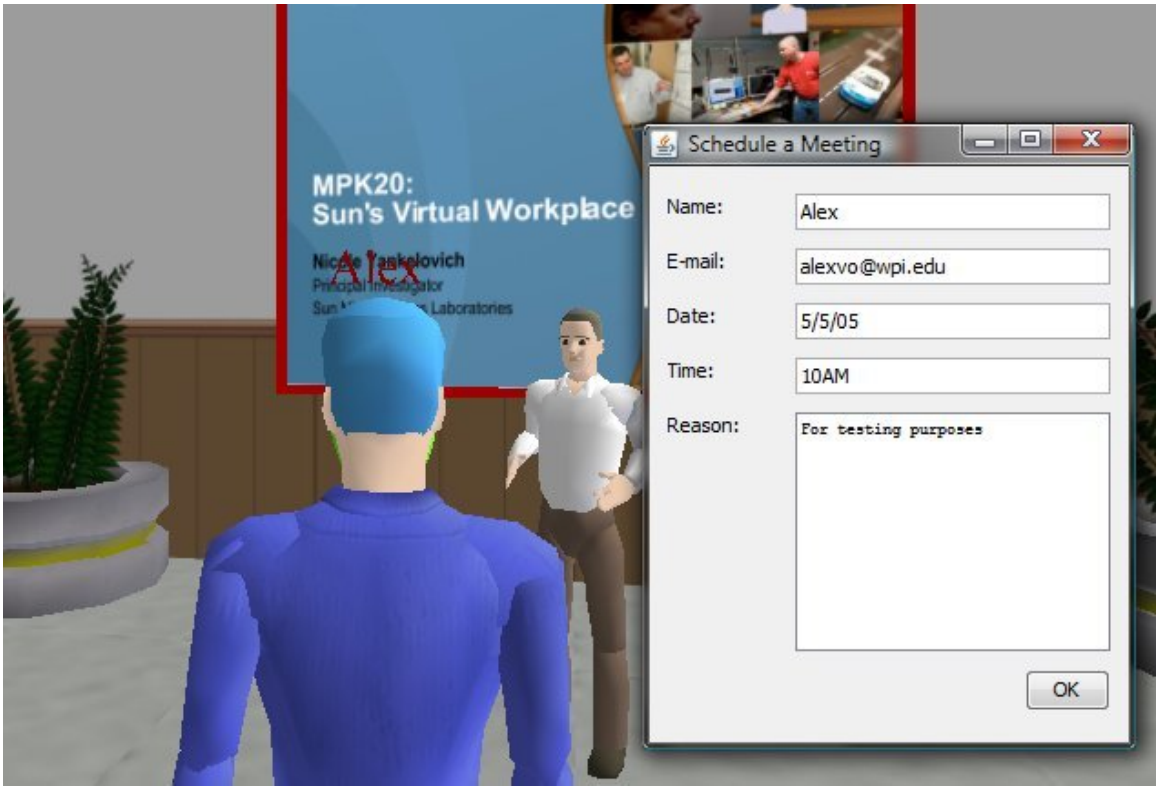

Figure 10 - Scheduling a meeting

When the form is complete, it is converted into an email message, which is formatted in the same way as the form appears. This message is then sent to the professor, whose email address is configured through the XML file that is storing all the receptionist settings, and to the person whose email was provided in the form. All further communication, however, has to be done through email or direct contact with the professor, but this facilitates matters quite a bit since the users who are logged into Wonderland don't even need to open any other software in order to send a request for a meeting.

The second option that is available in the receptionist menu is labeled "Contact Staff" and allows the users to do just that. When this option is activated, a typical instant

messaging window appears on the screen that also provides a list of names that the students might be able to contact. These include the professor and the teaching assistants if they have an AIM account. As soon as this window appears, an account that is tied to the receptionist (called wllobbybot) is signed in. Through this account the communication is actually possible.

The window looks as displayed on the screenshot below:

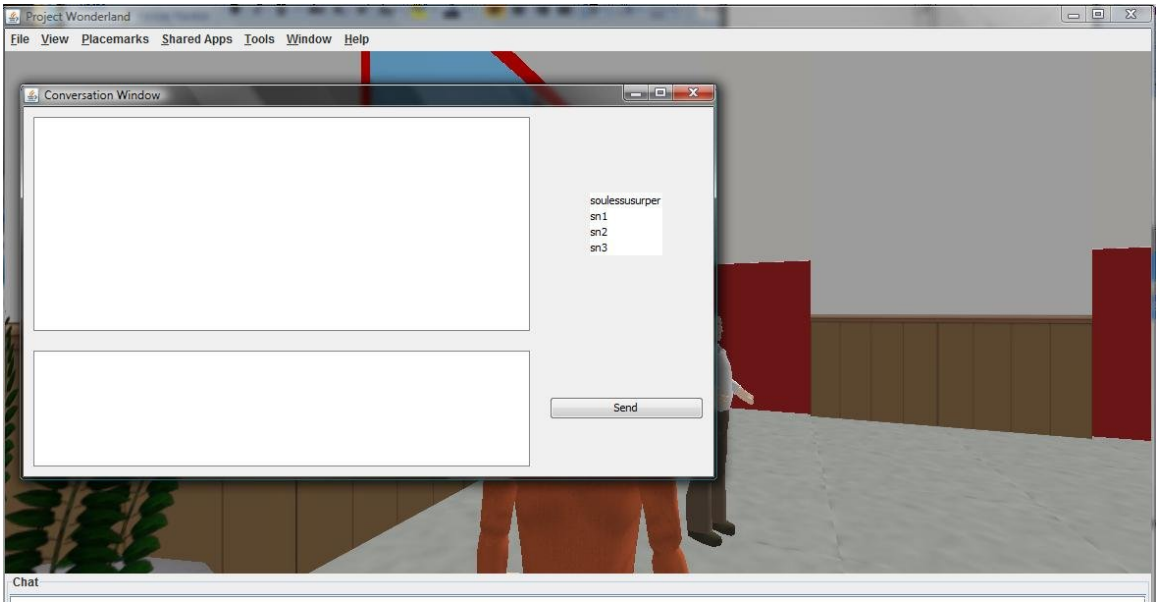

Figure 11 - Instant messaging window

As can be seen in the figure above, the window is very similar to what should be familiar to most people using any kind of instant messaging clients such as AIM , Windows Live, and YM. The top field serves the role of a message log, the bottom – an input field, while the contacts are displayed on the right-hand side.

There are 6 classes on the client side that are responsible for implementing the receptionist functionality. AIMConvo.java contains all the functions that are needed to

work with the AIM protocol, supported by AIMConvoWindow.java which contains all the GUI parts, responsible for showing the user the messaging window. The same is true for the meeting scheduling functionality – MeetingScheduler.java provides the functionality, whereas MeetingSchedulerWindow.java is used only to allow the users access to the form dialog and manipulating the input data to send an email message.

4. Results and Analysis

Throughout the course of this project several things needed to be accomplished. The first and most basic of these was to design a virtual environment that mimics a lobby. As was explained in the first section of this paper, the lobby plays the role of a general area where students can communicate to each other and ask for help from staff, similar to what might happen in a real classroom environment. The lobby is also connected to the three rooms that are currently empty, since this project's purpose was to create a groundwork that could then be expanded by introducing even more useful items into the world, as well as populating the team rooms with various types of collaboration tools.

The next goal of creating an information board was successfully accomplished essentially by default, because there was already a module developed for viewing PDF files that can be used to post announcements. It serves its purpose by first of all being visible, as it takes up quite a lot of space and is conveniently located right near the point where users appear whenever they log into the system so that they are practically looking directly at it, so that chances of missing important information are quite low. Secondly it is secure enough, so that the information is very hard to temper with, provided that the PDF file is stored in a secure location, which implies that chances of someone deciding to give the rest of the students any kind of false information are very low.

The meeting scheduling component of the receptionist works as promised. It creates an email message out of input provided through the form and sends it to the professor, whose email address needs to be set in the XML file. Except for the cases where there is no connection to the email server, there should not be any problems with this component. The only minor problem encountered was an error message about a

forceful closure of connection by the email at the end of the Wonderland server session. However, there doesn't appear to be any way to gracefully close all the connections to it, as the email session is being closed as soon as it is sent.

I did not get a chance to test the AIM communication dialog under a more serious load. The problems that might arise with it is if too many users are trying to access it at the same time, the server might complain about too much activity (i.e. signing on and off) and it might have to be extended to several different login names, so that if one fails, another one takes its place. The text size and font were left as default, which is a possible future feature that might make it easier to use this interface since the text seems to be a little too small on a large screen resolution, although that might just be a matter of preference. Overall, though, the messaging window does enable the user to send messages back and forth to different people that are displayed on the right side of the window and simultaneously receive their replies.

The doors should keep the team rooms relatively safe. As a security measure, one of the main Wonderland menu items was disabled. It provides the users with the ability to transport to any given set of coordinates. Such freedom would ruin the whole purpose of the function that the doors provide, because there would be no single entry point to any given room. This is however not very important, since the virtual world is small enough so that it wouldn't be possible to get lost and it doesn't take very long to get from one corner to another. Because the doors are not using any kind of encryption for the passwords, it is provides a security hole, although it might not be a major one, provided that no one besides the course staff is given access to the XML configuration files on the server.

The overall testing was done by exploring whether or not the functions that are presented in the project requirements are achievable. This means that the most common approaches to what was hypothesized a typical user might do, were considered and the preventive steps taken, such as checking how many instances of a particular door password dialog are open, because too many of those might be used to overload the memory and cause the application to crash.

There are about 1600+ lines of code that was done for this project distributed among the following classes:

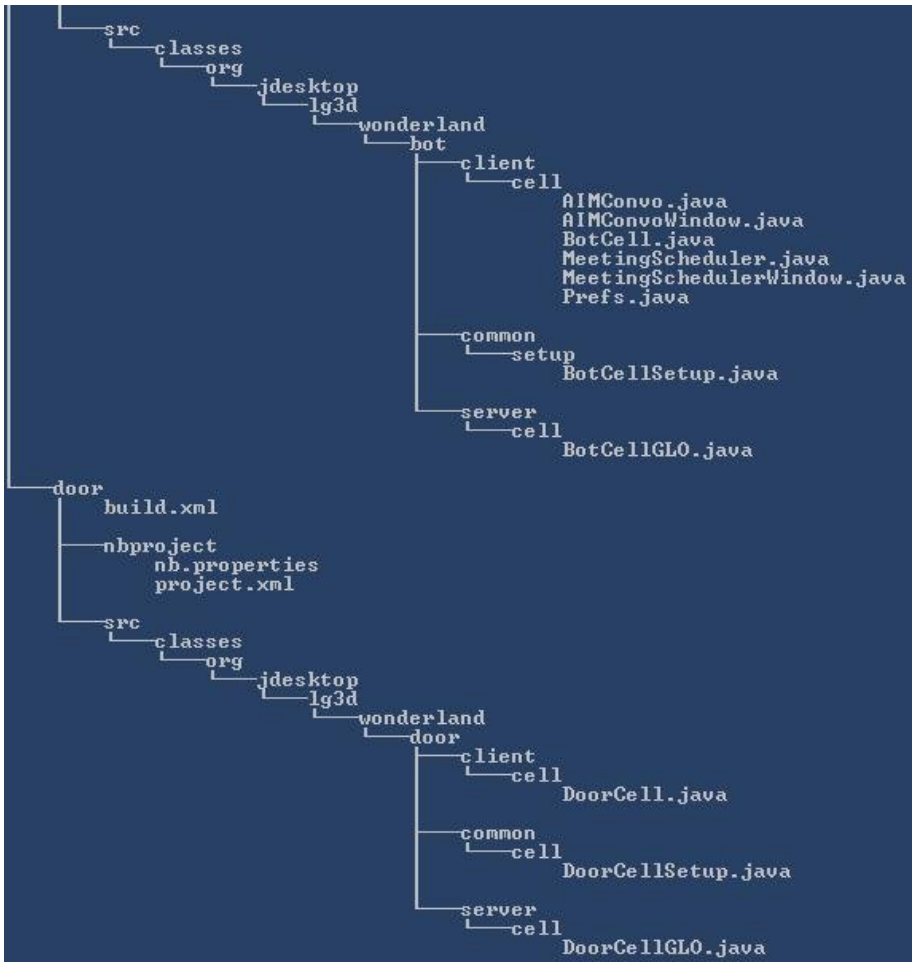

Figure 12 - Project folder structure

5. Future Work and Conclusions

Since a developer release version of Project Wonderland v0.5 is available at the time of this writing, the next logical continuation for this project is to upgrade to that version, when the final release is available. There should not be any conflicts during this transfer. As a matter of fact, some of the features that were not implemented during this project will be a lot easier accomplished in the next version of Project Wonderland. One of these features is a sign component for the door objects. They already have a name variable associated with them, but it would be beneficial for this name to be displayed in within the virtual world, so that navigating can be facilitated, especially in a large-enough world with multiple doors, so that the users are not getting confused. As mentioned before, the door passwords will need to be encrypted and decrypted in the future to prevent cheating, in case that the students are given access to the XML files stored on the server. It will also make sense to move all the dialogs inside of the application, when better support for Swing components is introduced, so as not to clutter the taskbar.

Another area that needs more exploring is the receptionist interface. While there are only two tasks that the receptionist is responsible for, the receptionist menu is fine the way it is now. However as more and more options are added, this menu will become too cluttered. As an idea, it might be worthwhile to make it into a list menu in the form

1. <action name>
2. <action name>
3. …

Or create a command-line-like interface so that users can type in more or less human-readable commands such as “>schedule meeting” and the meeting request dialog form

will appear, although simply clicking on menu items seems to be the better choice of the two, because it requires less work on the part of the users.

The goal of this project was to implement a virtual lobby for the software engineering course. In order for it to be useful for students, an interface for communicating with the course staff was required. As is described above, such an interface was successfully implemented in the form of a simple instant messaging mechanism that would allow communication with the professor and the teaching assistants, as well as the ability to schedule a meeting with the professor without leaving the virtual world. The design of the room, created as the first step for the project, allows team collaboration on the course project. A new object was introduced in the form of virtual doors that will protect the individual team rooms. This phase of the virtual lobby has provided the base for both future enhancements and introduction of new objects that will help with the students of CS3733 to successfully fulfill the course requirements.

6. Works Cited

http://secondlife.com/. (2009, April 29). Retrieved April 14, 2009, from Second Life: http://secondlife.com/

http://www.imvu.com/. (2009, April 29). Retrieved April 14, 2009, from IMVU: http://www.imvu.com/

https://lg3d-wonderland.dev.java.net/. (2009, April 29). Retrieved April 14, 2009, from Project Wonderland: https://lg3d-wonderland.dev.java.net/