Developing a MMORPG game in One Semester

Ilmi Yoon[1], Gary Ng[2]
[1]Computer Science Dept, San Francisco State University
[2]Pacific Ecoinformatics and Computational Ecology Lab

Abstract

Building a game has been used as effective way of teaching computer science. Building a Massively Multiplayer Online Role Playing Game (MMORPG) makes another step and serves as a great comprehensive tool to teach important aspects of technology, teamwork and software engineering principles. A course was taught to design and develop a working MMORPG within one semester. The whole class was structured in to several teams (Game Concept Design Team, Client Team, Server Team, Database Team, Art Support Team, Game Contents Team, Testing Team, Launching Team and IT Support Team) and students needed to join one or two teams. Each team presented their progress, discussed future milestones and troubleshoots, updated documents for clearer communication and utilized SVN throughout the semester. Unlike usual class setting, all students worked collaboratively together like one company to achieve the goal. In one semester, students started from concept design and completed a working MMORPG called "deBugger" (http://thecity.sfsu.edu/~debugger), learning game design, 3D graphics, Game Engine, Server-client architecture, Game Protocol, network programming, database, Software Engineering principles, and large application development as a team project.

Keywords: MMORPG, Game development, Industry style team work, team project taught class

1. Introduction

Game design and development has become a popular computer science course over recent two decades [1]. As students' enthusiasm to build game has been so strong, instructors utilized the eagerness to teach many aspects of computing, including computer graphics, artificial intelligence, human-computer interaction, security, simulation, and software engineering [2]. Parberry [1] studied what game companies commonly want, so their game course to be designed to teach the experience; (1) work on a large project, (2) creation of game demo, (3) team player, and (4) learn independently and taught them in game class. These previous work shows the promising potential that students are willing to take a tough and intensive course work to achieve the goal of building game.

Social gaming and MMORPG are relatively recent game genres as they bloom over the availability of Internet. Social interactions within the game radically increase the excitement of the game further and popular games easily develop communities of multimillion players [3]. Therefore, building a MMORPG game can motivate CS students to focus on the course work and increase learning outcome substantially. Unlike stand-alone game, a MMORPG game keeps virtual world persistent even after players log out, so game server has to run infinitely connecting all the clients and updating game status at database. Therefore, building a MMORPG game covers broad spectrum of computer science technology from computer graphics, 3D modeling, game engine, network programming, client-server architecture, and database, so it helps students exposed to diverse technical components comprehensively. Also a MMORPG game is indeed a large application and building one ensures students to learn the qualities that software companies want.

In section 2, we discuss the course design to teach students to build a working MMORPG game within a semester (15 weeks). In section 3, we present the results and learning outcome. Conclusion and future work are discussed at section 4.

2. Course Design

This class was designed for SFSU CS senior and graduate students who completed programming languages, data structures and Software Development Principles but not necessarily have taken elective courses such Software Engineering, Network nor Database courses. Considering that majority of students find jobs at industry after graduation, the class was organized in more of industrial flavor; a team of students (1) receive tasks and milestones, (2) achieve the milestones by actively looking for solutions from any available resources, (3) interact with other teams collaboratively, (4) and produce documents for clear communications between teams and future extensions.

The class was organized into Game Concept Design team, Game Server team, Game Client team, Game protocol team, Database team, Art Support team, Game Content team, Testing team, and Launching team. Within first 2 weeks, objectives and the responsibility of each team were discussed and then students were assigned into one or two teams based on their interest and backgrounds. From then on, teams worked on in parallel to achieve their milestones, while learning all the required technology for the given task and sharing their learning with other teams.

Summary of each team's task is as below.

Game Concept Design Team: this team has most important initial responsibility. The objective of the whole class was to build an educational MMORPG for Computer Science students, especially for beginners who are taking first programming courses. The concept design team should consider the limitations such as time (10 weeks for actual implementation), lack of artists, and unknown factors like the potential of other team's performance. The team also has to collect everyone's idea, finalize the game concept into several milestones and produce the documentation.

Game Client Team: this team is responsible to implement 3D game client using Panda3D. Students have to learn 3D graphics, Panda3D game engine and Python. Also client communicates with server using the game protocol to update the status (position, motions, and levels, etc) of other clients concurrently in the same virtual space.

Game Server/Protocol Team: this team is responsible to implement game server that connects to all the clients, synchronize every client status and update the Database. Team did not start from scratch as there was a JAVA game server written for a similar MMORPG game called "NurseTown" by students of Internet Application course. The server team had to read and revise the game server to work for "deBugger" game. Also the server team has to develop an effective set of protocols used between clients and server. Students in this team have to learn socket programming, client-server architecture, and Network protocols.

Game Database Team: this team is responsible to design the DB schema, install mySQL server and deploy the DB for every activity to maintain virtual persistent world. Students in this team have to learn SQL, Database scheme design and mySQL server management.

Art Support Team: this team is responsible to provide 3D models to create the virtual world and characters in the game. As there are no artists in the class, this team has to find freely available 3D models and convert the format to work with Panda3D. Students in this team have to learn basic of 3D modeling tools such as Blender or Maya and converting 3D model formats, and real time rendering performance trade-off (visual quality vs. rendering speed).

Game Content Team: "debugger" game is a unique MMORPG game for educational purpose. Therefore, the educational game contents are as important as other game assets. This team develops educational contents. As senior or graduate students of computer science, this team is already familiar with educational contents that beginner programmers learn which is good and bad. This team needs to interview beginner students, observe the troubles that they probably forgot and produce effective educational contents.

Game Testing Team: this team is responsible to create scenarios for testing the stability of client, server and protocols. One main task is to develop an autonomous client that connects to the server and simulates an ordinary client. Then the scaling of such clients can be used to test the server and protocols. The students in this team have to learn Panda3D, network protocols and practices in software testing planning.

Game Launching Team: this team is responsible to develop a web site where client installer can be easily downloaded and players can get simple instructions about how to get started with games – game rules, help, hot keys, etc. Also this team works on packaging the client program into an easy installer program.

IT Support Team: this team is responsible for helping every team to function without any system/software issues. For example, this team sets up SVN and creates account for each student. This team also helps teams to set up development (IDE) environment.

There were a little over 20 students in the class, so each student choose one primary team and one secondary team, so each team has at least 2 to 3 students. By having students in a few different teams, students could learn more than one technical area and teams can communicate and update the progress easily. Each team has to work very collaboratively as the whole class works for one project together.

Important concepts and technologies were taught to the whole class by several lectures, but most of class meetings were used as presentation of milestones, progress, trouble shootings of problems on hand, so the whole class stays in a same page and gets exposed to the problem and solutions.

Software Engineering principles have been practiced throughout the whole course as teams realized the importance of the inter-team communications due to the rapid developments in parallel. Changes in one team did propagate changes in milestones or requirements in other teams. SVN was actively used.

3. Results

The whole class successfully completed a working MMORPG game. Each team achieved milestones close to the initial (very ambitious) goal. **Game concept team** came up with the title, "deBugger" and the theme of the game; fighting against bugs within inside of computer, inspired by the origin of the word and the significance of the word to computer science students. Team also prioritized concepts in to several milestones, so kept the milestones doable for one semester, but left many interesting ideas for future extensions.

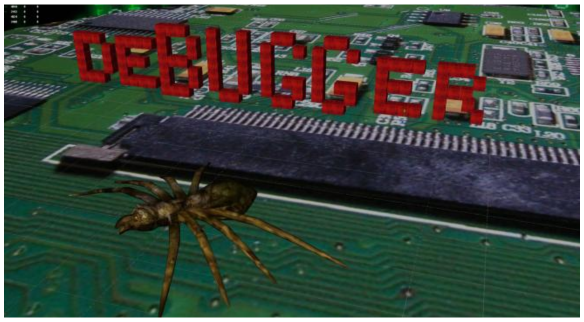


Fig. 1 – Theme of the game.

Primary milestone for Fall 2009 was to build virtual world where players can explore and fight with bugs by solving multiple choice questions. Health drops under bug's attack and game items can be used to shield from the bug's attack (delay the health damage or clear out options from the multiple selections). Players can level up by solving questions required by each level. Also player should be able to chat, create friends' list and maintain their inventory (trading or giving gifts).

Art Support Team achieved beyond the requirement. Students learned 3D modeling tools and created 3D environments for the game. We used 3D character models from Panda3D repository, but all the environments (figure 2, 3 & 4) were created by art support team and theme was inspired by movie Matrix and Tron during inspirational discussions at class.

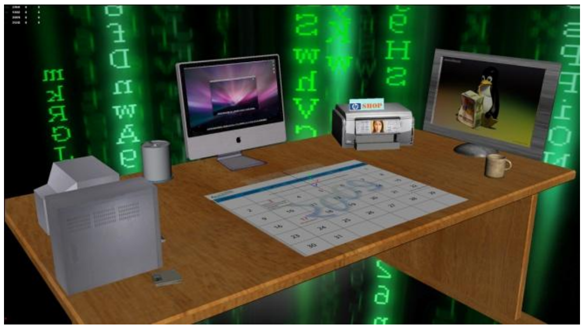


Fig. 2 – Full view of the desk space. This is a default/initial space where players get started.

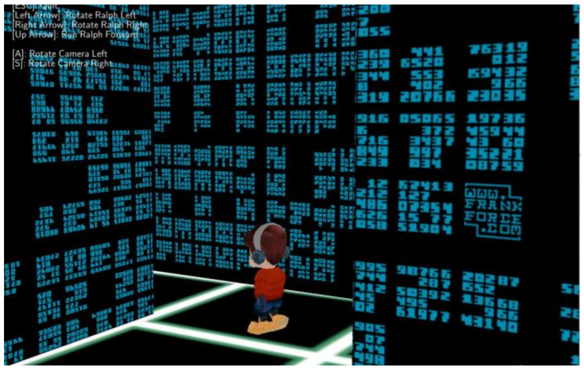

[ESC]: Quit
[Left Arrow]: Rotate Ralph Left
[Right Arrow]: Rotate Ralph Right
[Up Arrow]: Run Ralph Forward
[A]: Rotate Camera Left
[S]: Rotate Camera Right

Fig. 3 – A corner of motherboard. This is a dungeon where lots of bugs live. Players can teleport into this space from the initial desk space.

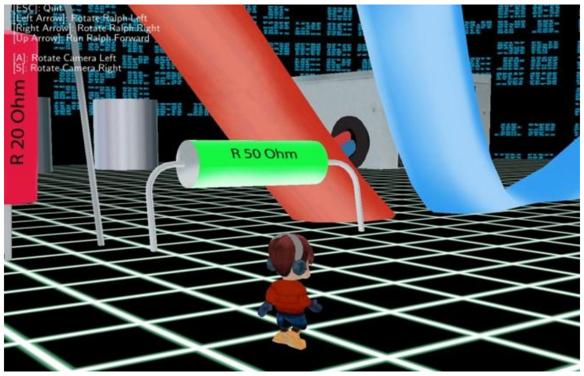

[ESC]: Quit
[Left Arrow]: Rotate Ralph Left
[Right Arrow]: Rotate Ralph Right
[Up Arrow]: Run Ralph Forward
[A]: Rotate Camera Left
[S]: Rotate Camera Right
R 20 Ohm
R 50 Ohm

Fig. 4 – Another corner of motherboard.

Client Team was the largest team and they were divided into sub teams to be responsible for specific tasks in parallel. List of such tasks are as below.

i. Registration Process
  a. Avatar Selection
ii. Login Process
iii. Chat
iv. Bubbles (translucent panel)
  a. Chat Bubbles
  b. Character Name and Damage Bubbles
v. Friends
vi. Inventory (figure 5)
vii. Character Info
viii. Hot keys
ix. Camera control
x. Character Movements
  a. Interface (mouse, Keyboard)
  b. Collision
xi. Battle System (figure 6)
xii. NPCs (Non Player Character) (figure 7)
xiii. Bug
xiv. Mini maps
xv. Mouse Picker

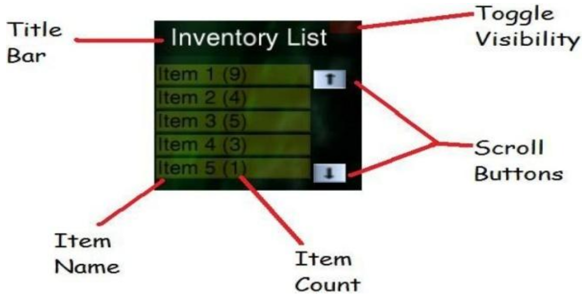
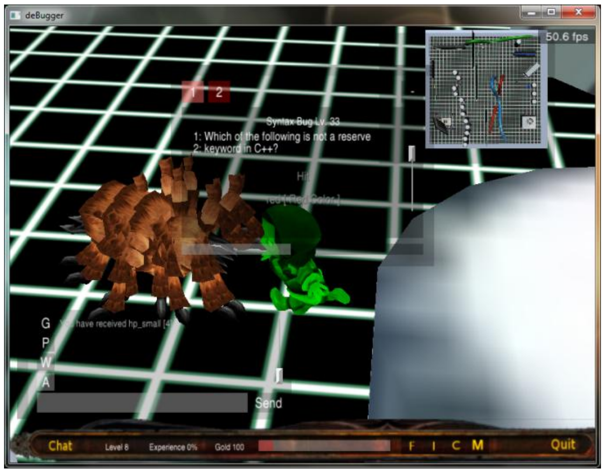


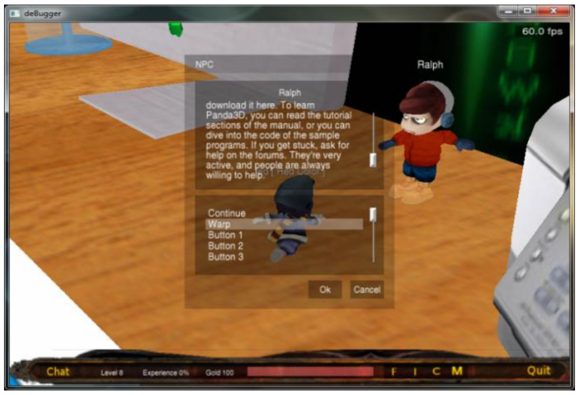
Fig. 5 – Inventory of a player.



Syntax Bug Lv. 33
1: Which of the following is not a reserve
2: keyword in C++?

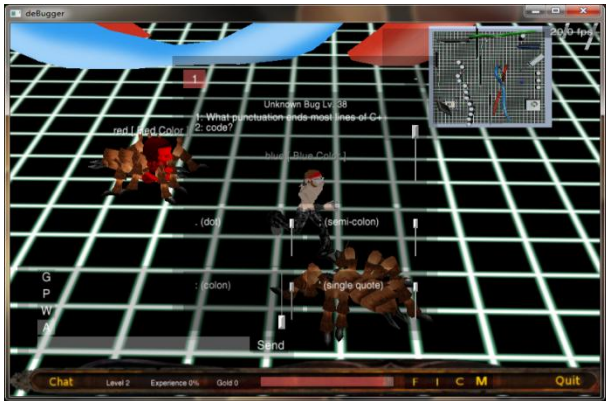
You have received hp_small [4]

Fig. 6 – A player is fighting with a bug inside of computer by solving the quiz. On the right upper corner, there is a mini-map showing the current location of the player. At the bottom, there is a bar where user can click to activate several game functions such as chat box or user inventory.



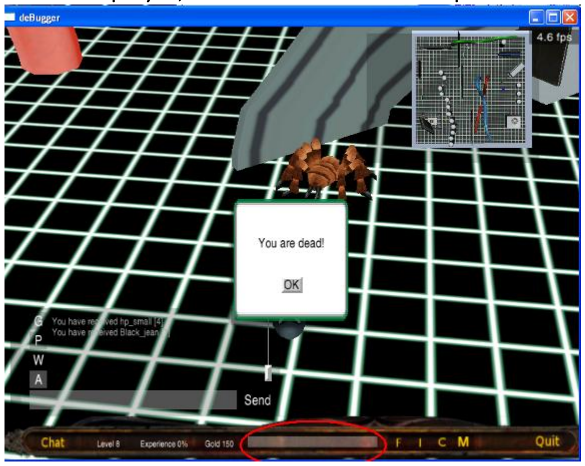
download it here. To learn Panda3D, you can read the tutorial sections of the manual, or you can dive into the code of the sample programs. If you get stuck, ask for help on the forums. They're very active, and people are always willing to help.

Fig. 7 – NPC (Non Player Character) requires simple AI to interact with players; guide players to follow rules of game or inform about events to participate.



Unknown Bug Lv. 38
1: What punctuation ends most lines of C++
2: code?

Fig. 8 – Bugs are attacking players. Bugs have AI to chase the player, handle collision and re-spawn.



You are dead!

You have received hp_small [4]
You have received Black_jean [1]

Fig. 9 – Health bar shows the damage from the bug's attack. This player lost all the health.

Client team also achieved impressive progress of completing all the requirements and the resulting game client was reliable and capable of very smooth rendering. Each task listed above took intensive effort to understand the technology (3D graphics) to manipulate Panda3D game engine.

Server team worked hard to make MMORPG game alive. Server team was able to reuse the majority of server code built for a game called, "Nursetown" by Prof. Ilmi Yoon and her graduate students. Server team had to understand the communication mechanism (figure 10 & 11) and modified DB schema and added lots of new game protocols.

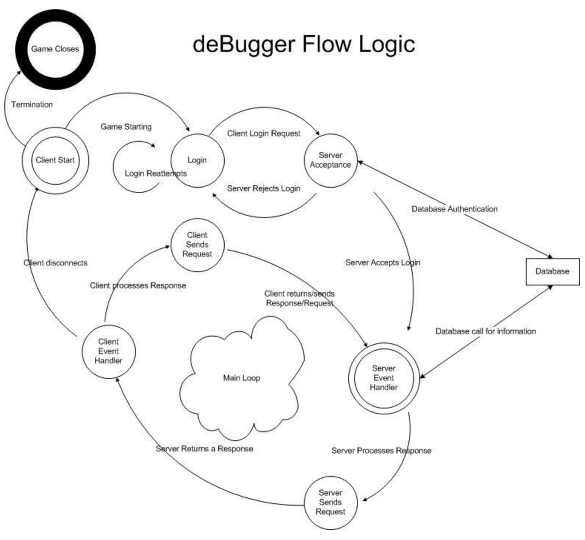


Fig. 10 - deBugger Flow Logic used to show the cycle of information being passed between *Client* and *Server*

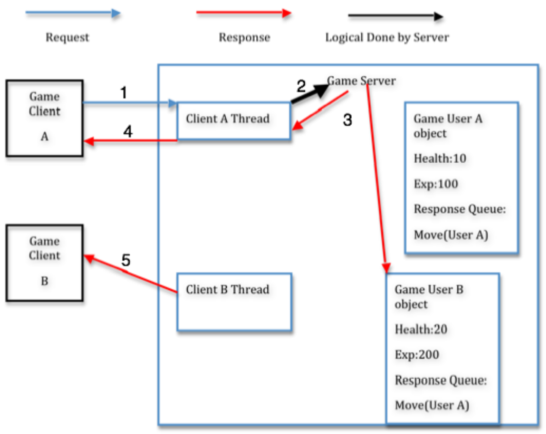


Fig. 11 - Handling of Request and Response from Game Client to Game Server. In the picture, starting at 1 the game client A sends the game server a request. The game server receives the request from its game client object. At 2 the game server handles the request and creates the corresponding response. At 3, a response is sent back to the original requestor (client A), and also same response is created for client B to be updated about client A. At 4 the move response is sent back to the player. At 5 the client B sends a heartbeat (10 times per sec) request and receives the update of client A's move.

Game Contents team focused on adding educational contents into the game. They attended "Introduction to Computer Programming" class to refresh the memory of what they learnt at the beginning level and interviewed students of the course. Game contents team structured pedagogy of computer programming learning and created quizzes in different levels for such purpose.

Data Base team studied design of efficient DB schema and developed it for debugger game. Team also installed mySQL server and deploy the data sets.

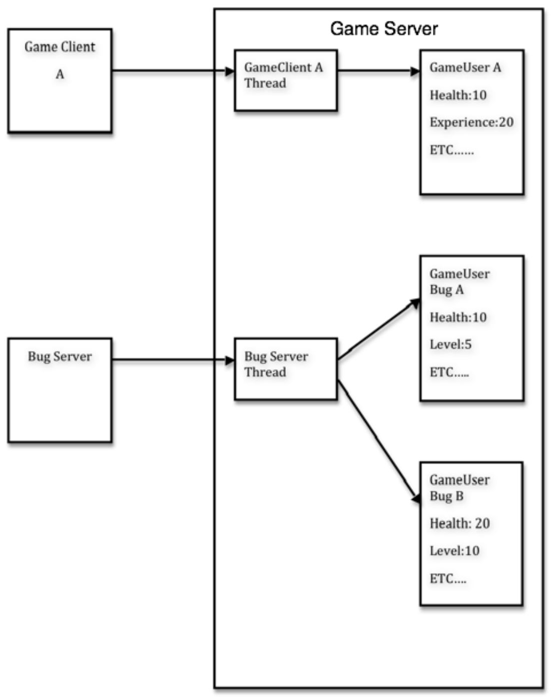


Fig. 12 - Difference between Player and Bug Server overview

Testing team was developing a test client that has a minimal AI to wander around, acts like a usual client and save the communication log, so we can use the test client to measure the scalability of the server. In the middle of the semester, we found that the debugger game needs a bug server that controls all the bugs that has AI to attack the player and wander around the debugger world. Testing team took the role and successfully completed the creation of the bug server with the close collaboration with the client team (figure 12).

Launching team studied popular MMORPG hosting company's web site and designed a web site for the possible full version of the debugger web site and implemented core set of it to be able to launch the game for programming class students (figure 13). The launching team made a simple web site (http://thecity.sfsu.edu/~debugger) with link to download client application (self-extracting installer for windows), game rules, information of how to get started, and screen captures.

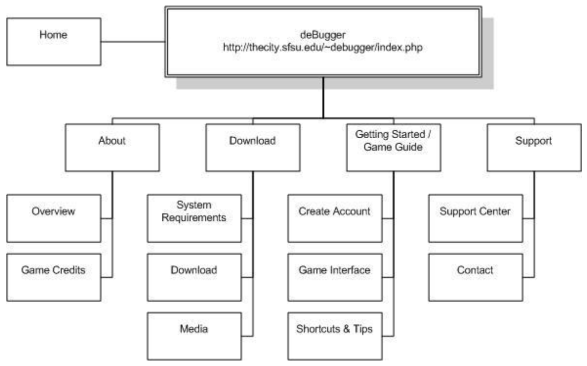


Fig. 13 – debugger web site map : debugger launching team studies other MMORPG site and designed a core functionality in similar style.

The class as one whole team completed a working game (version 0.9) by Dec. 2009. The game contained core MMORPG game features. The client and server connect and communicate in stable manner. DB stores persistent data. User can create an account with options of a few characters. And then user can log in at the location where they logged out last time. Player can explore the virtual world starting from a desktop space into the inside computer world. User can find if friends are logged in and chat with friends or other players within the same space. Players can check their game items at the inventory box, health bar and the current level as well as their current position at the mini-map. Most importantly, players can battle with bugs using the multiple choice questions and the health bar and level were properly updated.
At the same time, whole class produced one giant document (437 pages, available at http://thecity.sfsu.edu/~debugger/download_documentation.php) of each team's work for future developers who will extend this game to the next level.

4. Conclusion

Students were proud of what they achieved in one semester (15 weeks). Considering that the first few weeks were used to the introduction to the class and setting up the team, students had about 2.5 months to work on the game. To be able to make the MMORPG game work, each team has to deliver what they were expected in milestones. It serves as a good pressure on each team and students collaborated across team enthusiastically. Most students put efforts beyond requirements, resulting in good learning outcome.

This class can serve as a good team project course for senior students in Computer Science, so they can taste the industry style problem solving approach and team work in academic environment. Also as a by-product, the class produced a usable game that is currently used in CSc 210 Introduction to Computer Programming course at SFSU. The objective of using this game for educating CS freshman seemed to give pressure to students to deliver a smoothly working game as their clients are sitting right next to them, so it lets them alert about their end user.

To be able to replicate the same class in the future, course material needs to be better collected and organized. There are hundreds of discussion threads at iLearn news group that were effective while being used, but not so organized, so may not be transferrable to next year class nicely. Inter-team communications were well captured to be used in the class for each team progress presentation and documentations. But inner team communication and resources were not saved. Each team created very useful resources to be shared within team to learn the required technology for each team for example, client team learned Panda3D, Python, collision handling together helping each other. Same for each team. Vast amount of self study materials were created by each team. These can be better managed to be created at wiki, so they can produce more re-usable study resources.

Acknowledgement

Big thanks goes to Zoran Millic and Supakit who developed Nursetown game and provided kind support to CSc 631/831 students to understand the key technical components. Sergey Sergeev as an art support team leader went beyond the requirement and provided creative 3D models to the game. Jason

D'Silva and Gary Ng put tremendous hours to lead the client team to produce the client with all the required features. Alan Nguyen worked hard to make the bug server possible. The rest of the client team that includes Alexander Bonilla, Julio Benitez, Nicholas Byrd, Vivekanand Rao and Yi Liu contributed great effort to implement several important features used in the client. The database team that includes Alvin Wan, Calvin Kuang and Tun Win worked great together to design and implement a database used specifically for the server. Ali Ugur, Jordan Mangini and Sara Tily produced an interesting game concept that we all could follow. And Kaven Tan for contributing the best he could to get the server running.

This project is being funded by National Science Foundation Div. Of Biological Infrastructure, Biological Databases and Information, NSF DBI-0543614

5. References

[1]. R. M. Jones. Design and implementation of computer games: A capstone course for undergraduate computer science education. In Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education, pages 260–264. ACM Press, 2000.

[2] I. Parberry. Introduction to Computer Game Programming with DirectX 8.0. Wordware Publishing, 2001.

[3] R. Moser. A fantasy adventure game as a learning environment: Why learning to program is so difficult and what can be done about it. In Proceedings of the 2nd Conference on Integrating Technology into Computer Science Education, pages 114–116. ACM Press, 1997.