The Capella's Song Database

BY MICHELLE CHAMBERLAIN

Mentor – Dr. Barnes
Tbarnes2@uncc.edu

Grad Student Mentor – Amanda Chaffin
Katla@wulfkub.com

Grad Student Mentor – Michael Eagle
mjeagle@uncc.edu

Table Of Contents

Introduction

This is the final report for the database I created for the MMORPG (multi massive online role playing game) Capella's Song as a part of my work on the Game2Learn project with Dr. Barnes for CRAW/REU Summer 2008. Building a database for a game, as I rather quickly discovered, was not exactly simple. There were many considerations that I had to deal with and overcome that are not necessarily the same as other, simpler projects. In this paper, I am going to discuss the formal requirements for the database. I will also discuss the GUIs. Finally, I am going to present my conclusions and discuss possible future work that holds true for both this database project and the Game2Learn project.

The Games2Learn project has various games that teach material from early computer science courses. In order to track the progress of the students playing the games I have designed and implemented a database. I created GUIs that access the database, and run MySQL queries on the database. Finally I created an XML parser that takes an XML file, retrieves data from the file, and passes the data into the database. The GUIs are important because they allow us to access information in the database through a graphical setting. The parser is important because it will allow Dr. Barnes to quickly and efficiently load data into the database without having to do it by hand.

Background

The Games2Learn lab has created several games over the past few years that teach early computer science concepts. For example last year a student named Taylor Dubois created a game called Bunny Arrayser which taught loops. Bunny Arrayser used loops to change evil bunnies into good bunnies. Other game are also created in the Games2Learn lab however. For example last year I was in the lab working on a cultural game called the Kuku-Thaypan fire project. The Kuku-Thaypan fire project taught the aboriginal culture of the northern Australian people. Two girls that are in the lab this year are making a breakdancing game for the culturally situated design tools.

Amanda Chaffin was originally working on the design for the Capella's Song database in one of her classes, and when it wasn't finished she decided to have a REU Student work on it during a summer internship. Amanda had a basic design for how she envisioned the database. Amanda had a entity-relationship design. Amanda

also had prototypes of how the GUIs could look and written ideas on how they should function. Amanda consulted me throughout the entire process of this project.

The XML standard for the parser was originally designed by Michael Eagle. He created the XML standard so that the Games2Learn games data could all be easily passed into the database. The XML files will be passed through the parser I created this summer so that the data would not have to be passed in by hand. Michael consulted me as well throughout the course of this project.

Methods

Database Design:

We began this project by determining the best design for the database. It is important to have a good design for a database because then the data can be more easily accessed. As we were designing I made design prototypes using Microsoft Access. Once the design was complete I implemented the database on the Arena computer in the Game lab using MySQL.

Student Data -

As stated in Appendix A the student data table will consist of an automatically generated ID number, student name, year, major, minor, age, disabilities (if any), race, and sex. We will need this data for bookkeeping and also for running studies on the project. Once the students have filled out the bookkeeping end, they are allowed to select a character name. The character name (Player Name) will be unique to each student and we will heavily encourage students to preserve their anonymity by selecting names that have no bearing in real life. Each student may have one account, which we will check with the student ID. The student ID will be a foreign key in the player data table as well as all of the game data tables, all of the mission tables, all of the interaction tables and all of the activity tables.

Class Data –

For every class there will be a professor, a class name, a class section, and a faction name. The faction name will be a foreign key in the Player Data table.

Player Data –

For each entry in the Student Data table there is an entry in the Player data table. Player data consists of the Student ID, Player Name, Group, Faction Name, and Rank of the student. When the student first signs up using the sign-up sheet GUI the rank will be automatically set to “Apprentice”. Apprentice is the lowest of the possible rank. “Journeyman” is the middle rank. “Master” is the highest rank.

Game Data –

There will be a separate game data table for each individual game. Each game data table will have the student ID of the students whom have played it, a Game Instance, a game description, start time, grade and end time. The game instance is a foreign key in the Mission data tables, the interaction data tables and the activity data tables.

Mission Data –

There will be a mission data table for each individual game in the system. Each mission will have the student’s ID, game instance, mission instance, start time, and end time. The mission instance will be a foreign key in the interaction tables

Interaction Data –

Each game will have its own interaction table. The interaction tables will consist of the student ID, game instance, mission instance, interaction instance, and message.

Activity Data -

Each game will have its own activity table. The activity tables will consist of student ID, game instance, mission instance, start time, skills, level, question, answer, response, grade, and end time. The skills are a small list of skills taught in the activity. The question is a test question that corresponds to the skills taught in the activity. The answer is the correct answer to the question. The response is how the player responded to the question.

GUI Design:

While I was working on the database design I was also working on the GUIs in Microsoft Visual C# (See Appendix A for screenshots of the GUIs and full descriptions of each GUI). The first GUI I created was the start screen for student which asks if the student is an existing user or if they need to sign up. The second GUI I created was the Existing user login screen, which checks the novell username and password against information in the database in order to log the student in so they can play the Games2Learn games. Next I worked on the sign-up sheet GUI which takes information about the student and automatically enters it into the database. The sign-up sheet GUI

also checks if the passwords match in the "Enter Password" and "Confirm Password" boxes and if the passwords do not match then the GUI will bring up an error message and erase both passwords. The Sign Up Sheet will also bring up an error if any of the "Required" fields are missing. Next I worked on the report generating GUI which generates a MySQL query based upon which information the user checks that they wish to view. The reports generating GUI will allow Dr. Barnes to run studies on the students whom play the Games2Learn games. The final GUI I worked on is the common queries GUI, which allows the user to pick a pre-made MySQL query and run it. The common queries GUI will make it so that Dr. Barnes does not always have to create a GUI using the reports generator, the more common ones that she wants to run can be done in the common queries GUI.

XML Parser:

I was also required to make an XML parser (See Appendix B for a sample of the XML schema). The XML parser takes data from an XML file and puts the data into the database. The Parser was originally going to be made using Java, however not far into the creation of the Parser I discovered that the Arena computer does not have the correct drivers to connect a Java project to the MySQL database. We therefore decided to convert the Parser into Microsoft Visual C#.

Problems Encountered

One of the problems I encountered was the overall design of the database. I couldn't decide at first whether or not I wanted separate game data, mission data, interaction data and activity data for each game. The main reason I was having problems with making that decision was because if I did not have separate tables then I would not have to have it so that for each new game you have to add another game data, mission data, interaction data, and activity data table to the database. However in the end I decided that the best course of action would be to have separate tables for each game because it makes it easier to access the data in the database.

Another problem I encountered during the course of this project was the creation of the parser. Originally the parser was written in Java because then it could be run from any system which would make it more convenient because then it could be run from any computer. However we discovered that the Arena computer does not have the correct drivers to connect any Java applications to the database. We decided to write the parser in C# instead at this point. Converting the parser from Java to C# presented few problems and only took a few days.

Another problem I encountered during this project involved the report generating GUI. The output for the report generating GUI is supposed to go into an excel sheet, however whenever I would run a query no output would show up on the excel sheet. I have not, as of yet, found a solution to this problem. I will however continue trying to fix it in my free time.

Impact

The database will hold valuable information that will help us understand where students have trouble with early computer science concepts. The database will hold important information such as how many times a student plays a Games2Learn game, how long it takes a student to complete a task in a Games2Learn game, and demographical information about the students playing the Games2Learn games. The database will allow us to run studies on which computer science concepts students have the most trouble with. The database will allow us to gain a better understanding of how long it takes a student to understand a question they are reading so we can see what does work and what does not work when it comes to teaching computer science.

The GUIs used to access the database will allow Professor Barnes to run studies on the data within the database without having to go through the data herself. This will allow her to get more work done faster and with more efficiency than she would if she did have to go through all of the data by hand. The GUIs can also be used to teach MySQL to people who have no knowledge of it, because you can choose to see the SQL statenments being used to run a query.

The parser will make it so that Dr. Barnes doesn't have to enter all of the data for the database by hand. It will pass the data into the database quickly and efficiently through reading what is between the XML tags and putting the information into the proper place in the database. All you have to do in order to enter new data is to put the XML file into the proper place and run the XML parser on it and all of the data in that file is automatically entered into the database.

Lessons Learned

I learned a great deal over the course of this summer. I learned more about the process that goes into database design. I learned more about the process of implementing a database. I learned more about MySQL and how it works. I learned how to create Windows Forms using Microsoft Visual C#. I learned more about the Microsoft Visual C# programming language. I learned how to connect a C# program or windows form to a MySQL database that is on another computer. I learned how to mount and use a USB drive on a linux computer. I learned more about some previous work done in databases that are used in games, most especially in World of Warcraft (See Appendix C for related works). I learned some of the similarities and differences between a linux and unix computer. I learned some of the similarities and differences between a windows and linux computer. I learned how to create an XML parser that will put data from an XMNL file into a MySQL database.

Conclusions

Database:

- For each new game a new game data table, mission data table, interaction data table, and activity data table will have to be added to the database.
- The database will allow Dr. Barnes to run important studies on how well students do in various aspects of early computer science courses
- The database will allow Dr. Barnes to figure out which Games2Learn games questions students are having trouble with and to therefore adapt the questions in order to make them more understandable to students in early computer science courses

GUIs:

- Creating the GUIs in Microsoft Visual C# Windows Forms was ideal because it allowed easy access to the database with a graphical interface.

- Separating the Reports Generator into four forms, one for choosing keys, one for choosing fields, one for choosing tables, and one for choosing joins made the GUI more efficient for creating MySQL queries and running them. Doing this also made the report generator easier to read which made it easier to use.
- The common queries GUI is useful for getting data quickly and efficiently without having to fill out the report generating GUI forms or having to write the MySQL statement out yourself. Since it shows you the MySQL statements this also make it a wonderful learning tool for anyone who would like to learn how to use MySQL.
- Hooking the sign-up sheet to the database so that all new students are automatically entered into it made the database more efficient.

Parser:

- The parser will allow Dr. Barnes to enter data into the database without having to do it manually which will make entering data much easier than it would be otherwise.
- Java was not ideal for building the parser because the Arena computer does not have the correct drivers int order to allow it to enter data into the database. While I tried to find out how I could put in the correct driver, I was unsuccessful.
- Using Microsoft Visual Basic C# was ideal for building the parser because it allows you to quickly and efficiently enter data into the database without having to do it all manually.

Future Work

In the future the report generating section of the GUI should be completed. This is important because it will allow Dr. Barnes to run studies on the database so that she can learn more about things such as how long a student took to get a particular question in one of the Games2Learn games right or what the average grade for a particular question on a Games2Learn game is. The XML parser should be altered as needed for Dr. Barnes. This is important because someone might make a game for the Capella's Song database that uses a different XML schema, For all new Games2Learn games there will have to be a new Game Data, Mission Data, Interaction Data and Activity Data

table added to the database. This is important because it is the only way to keep track of data associated with a game because that's the way the database was designed. The database should have more test data inserted into it for each game in the database. This is important because in order to make sure the database is working correctly it should have a much test data as possible. The database, GUIs and parser should be tested further in order to ensure maximum efficiency of all aspects of the project. Once they are fully ready for it, studies should also be run on the GUIs, the parser and the database so Dr. Barnes can get information for her studies.

Appendix A

Capella's Song Database

Design and Development by
Michelle Chamberlain

MChamberlain84@yahoo.com

Mentor – Dr. Barnes
Tbarnes2@uncc.edu

Grad Student Mentor – Amanda Chaffin
Katla@wulfkub.com

Grad Student Mentor – Michael Eagle
mjeagle@uncc.edu

Introduction

This is the initial report for the database I created for the MMORPG (multi massive online role playing game) Capella's Song as a part of my work on the Game2Learn project with Dr. Barnes for CRAW/REU Summer 2008. Building a database for a game, as I rather quickly discovered, was not exactly simple. There were many considerations that I had to deal with and overcome that are not necessarily the same as other, simpler projects. In this paper, I am going to discuss the formal requirements for the database, the ER diagrams that were created for the project. Finally, I am going to present my conclusions and discuss possible future work that holds true for both this database project and the Game2Learn project.

Table of Contents

Formal Requirements for the Game2Learn *Capella's Song* Database

Student Data -

Players need to register first with the MMORPG in order to log onto the game server. The authentication server will be handled separately from the game database. However, the registration information will be passed into the database. The information will consist of an automatically generated ID number, student name, year, major, minor, age, disabilities (if any), race, and sex. We will need this data for bookkeeping and also for running studies on the project. Once the students have filled out the bookkeeping end, they are allowed to select a character name. The character name (Player Name) will be unique to each student and we will heavily encourage students to preserve their anonymity by selecting names that have no bearing in real life. Each student may have one account, which we will check with the student ID. The student ID will be a foreign key in the player data table as well as all of the game data tables, all of the mission tables, all of the interaction tables and all of the activity tables.

Class Data –

For every class there will be a professor, a class name, a class section, and a faction name. The faction name will be a foreign key in the Player Data table.

Player Data –

For each entry in the Student Data table there is an entry in the Player data table. Player data consists of the Student ID, Player Name, Group, Faction Name, and Rank of the student.

Game Data –

There will be a separate game data table for each individual game. Each game data table will have the student ID of the students whom have played it, a Game Instance, a game description, start time, grade and end time. The game instance is a foreign key in the Mission data tables, the interaction data tables and the activity data tables.

Mission Data –

There will be a mission data table for each individual game in the system. Each mission will have the student's ID, game instance, mission instance, start time, and end time. The mission instance will be a foreign key in the interaction tables

Interaction Data –

Each game will have its own interaction table. The interaction tables will consist of the student ID, game instance, mission instance, interaction instance, and message.

Activity Data -

Each game will have its own activity table. The activity tables will consist of student ID, game instance, mission instance, start time, skills, level, question, answer, response, grade, and end time. The skills are a small list of skills taught in the activity. The question is a test question that corresponds to the skills taught in the activity. The answer is the correct answer to the question. The response is how the player responded to the question.

Student Interface –

The only interface the students will have with the database will be via the game HUD as listed above as well as the new user account registration.

Professor Interface –

Professors will be able to log into the database and pull up all the student information for the class. They will not have access to the game data. They will only be able to view the data and not make changes to it.

Game2Learn Interface –

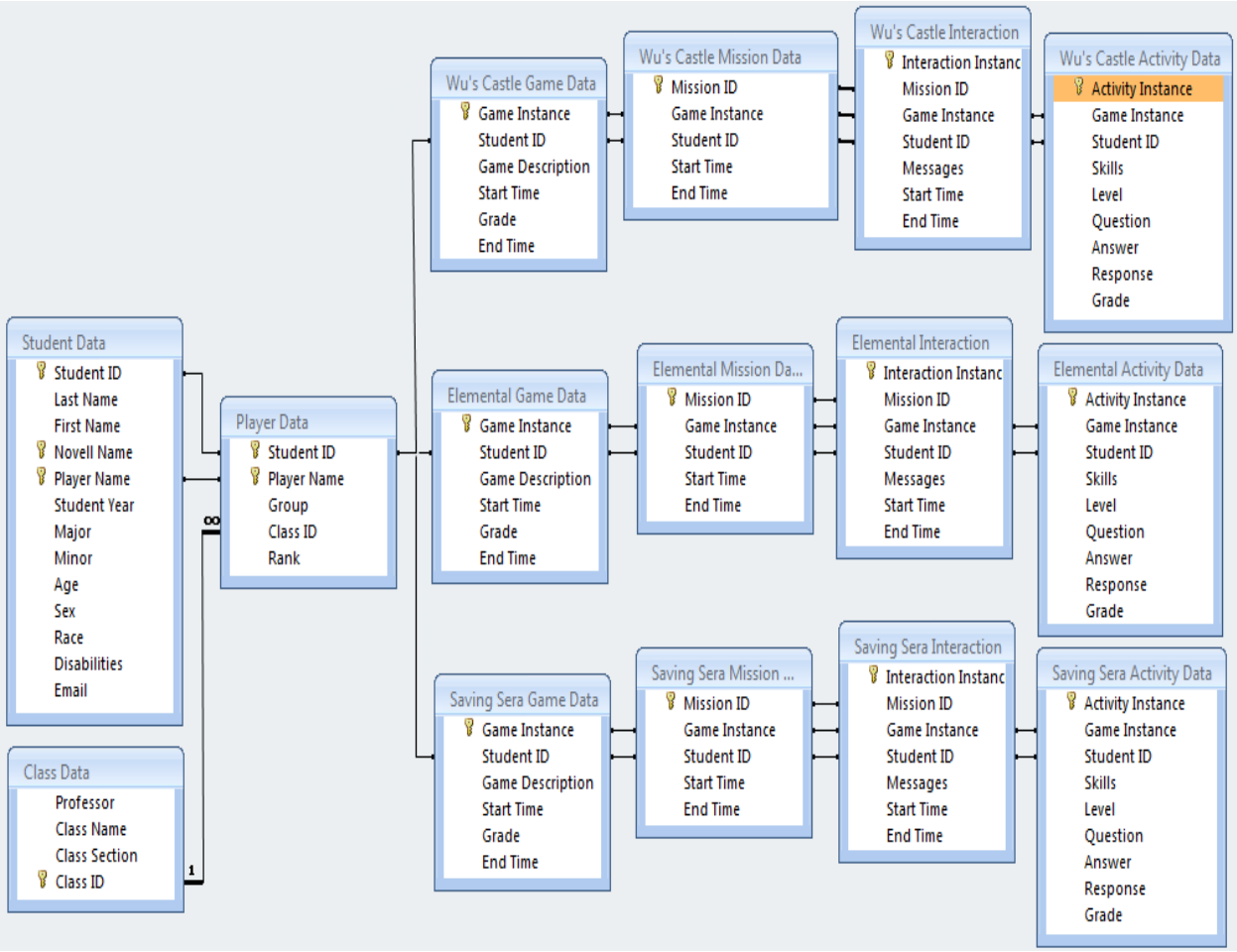
The Game2Learn group will have full access database access with permission to remove, add, and move any of the data in the databases or create a new one. Furthermore the Game2Learn group will be able to create, alter, or remove the forms connected to the database.

New Games –

For each new game that is put into the database you will need to enter 5 new tables a game data table, a mission data table, and interaction data table, an activity table and a message table.

ER Design

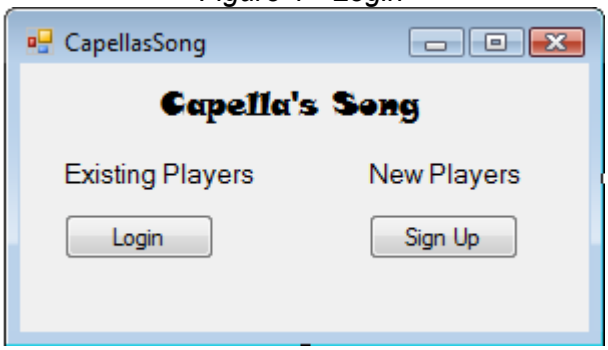
Complete Entity-Relationship Diagram



GUI Design

Since the GUI for this database is going into a game launcher, it was imperative that it be designed functionality and with an eye towards game design. The game GUI has several distinct stages to it because I needed to handle both new and existing users. The first thing the player sees is Figure 1 where they are asked if they are a new or existing user.

Figure 1 - Login



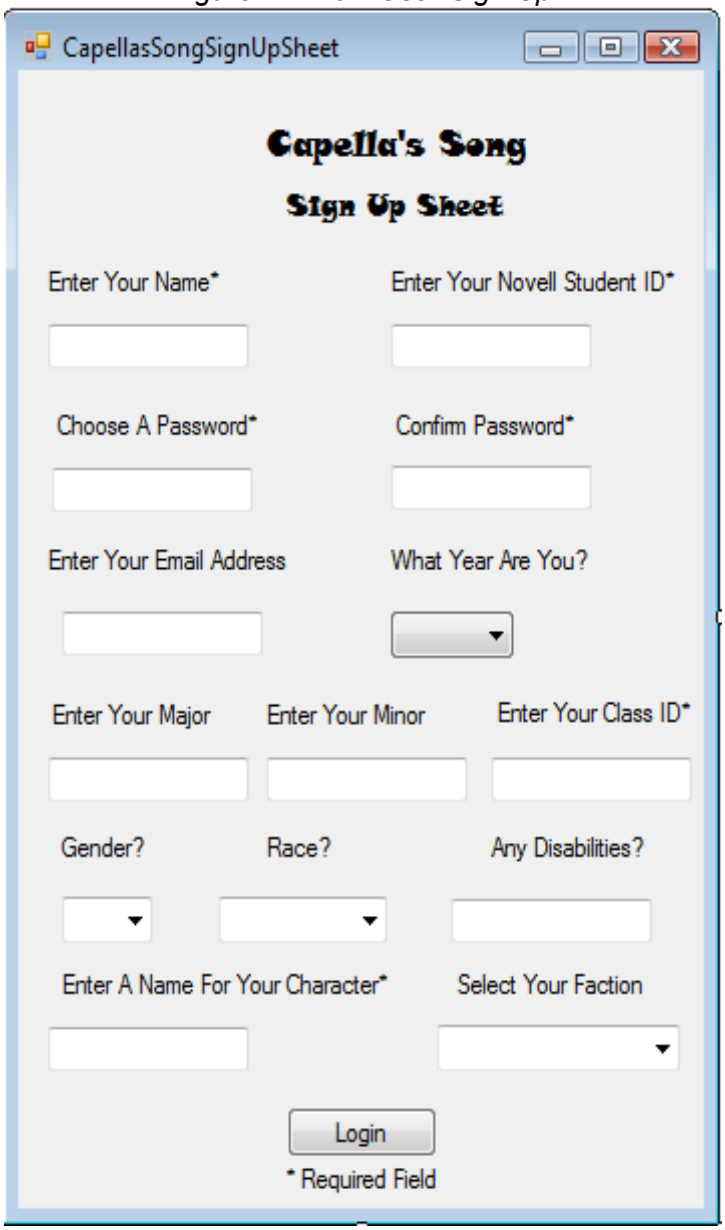
If the player logs in as an existing user, they are taken to the existing user screen where they are asked for their username and password which is checked against the StudentData table. Figure 2 shows the login screen and Figure 3 shows the login failure screen.

Figure 2 – Existing User Login

Figure 3 – Existing User Login Failed

If the player is a new user, they are taken to the new user sign up screen (Figure 4).

Figure 4 – New User Sign Up



If the new users choose a password field and confirm password field do not match up then a prompt comes up to let them know, and then clears the passwords so the user can try again (Figure 5). An important thing to note here is that the password comes up as a group of circles, and not as the letters the player actually uses in the passwords so the user needs Figure 5 in order to be able to tell when they make this error.

Figure 5 – Passwords Did Not Match

Furthermore error messages also come up if any of the fields marked as required fields are left empty. All of these error messages are in a similar fashion as figures 3 and 5.

Appendix B

<CSLog>

<PlayerName>Alan Morri</PlayerName>

<GameName>Elemental</GameName>

<GameDescription>XNA Game Studio</GameDescription>

<StartTime>July 3 2008 10:00 AM</StartTime>

<Quest>

<QuestDescription> Started the game. </QuestDescription>

<GameInteraction>

<Message>Ele: Wh-wh-where am I? What's going on?? </Message>

<StartTime>July 3 2008 10:00 AM</StartTime>

<EndTime>July 3 2008 10:01 AM</EndTime>

</GameInteraction>

<GameInteraction>

<Activity>

<StartTime>July 3 2008 10:25 AM</StartTime>

<Skill>For Loop, Array</Skill>

<Level>Application</Level>

<Question>Make this array: 1111111111111111111</Question>

<Answer>1111111111111111111</Answer>

<Responce>

for(i=0;i<=19;i = i + 1)

  Snow_Array = 1</Responce>

<Grade>100</Grade>

<EndTime>July 3 2008 10:26 AM</EndTime>

</Activity>

</GameInteraction>

</Quest>

<FinalGrade>100</FinalGrade>

<EndTime>July 3 2008 10:40 AM</EndTime>

</CSLog>

Appendix C

Related Works

Caitlin Kelleher, Randy Pausch, and Sara Kiesler in Storytelling Alice Motivates Middle School Girls to Learn Computer Programming[i] wanted to see if there was any difference in the level of learning computer science amongst middle school girls between Storytelling Alice and Generic Alice. They tested this through various four hour long workshops in which they would split a group of girls in two assigning one half to using generic Alice and the other half to using storytelling Alice. Each group of girls was also given different tasks to perform since there are some differences between what you are capable of doing with Generic Alice and Storytelling Alice.

Caitlin Kelleher, Randy Pausch, and Sara Kiesler tested 200 girls over the course of their two-year long study. They found no difference between how much was learned about basic computer programming between the groups of girls that used Storytelling Alice, and the group of girls using Generic Alice.

This is related to my work because the database I have created is going to be used with games that have been made to teach college students about how to program. My database will make us more able to understand how much the student learns from the games made in the Game2Learn lab at UNC Charlotte.

Ducheneaut, Yee, Nickell and Moore in the paper The Life And Death of Online Gaming Communities: A Look At Guilds In World of Warcraft[ii], wanted to study group dynamics in Massively multiplayer Online games (MMORPG). They wanted to study the structures of guilds in the MMORPG World of Warcraft (WoW). They conducted their study by looking through five servers from the game WoW. All together they watched over three hundred thousand unique game characters.

Ducheneaut, Yee, Nickell and Moore found that bigger guilds tended to gain more members over time. They also found that guilds with smaller subgroups tended to last longer because of less clashing over things like who got the best treasure from a defeated enemy. Furthermore they found that guilds that have people of varying levels last longer, guilds that have more interaction between the members last longer and subgroups that work in the more complex game areas last longer since they tend to interact more due to having to strategize what they do.

This is related to my work because I have created a database where there will be many players and those players will possibly be in guilds or “factions”. It is important for us to understand how various people in guilds interact with one another so that when we look through the data in the database we understand more of what we are seeing in the interactions between various students.

Wadley and Sobell wanted to know in the paper Using A Simple MMORPG to Teach Multi-user, Client-server Database Development[iii] if creating a simple game and database could be used to teach an advanced database design class. They did this through using SQL servers and a “black box solution” that showed the student how the database and game should work without giving them any code.

Radu Privantu wrote a guide for people whom wanted to begin creating their own massively multiplayer online role playing game (MMORPG) entitled A Beginners Guide to Creating an MMORPG[iv]. This is not a “classical” paper in that there are no methods or results mentioned or any statistics of any kind. This is just a guide. However Privantu does give some helpful suggestions.

First he suggests that you have some basic skills that are needed in order to create an MMORPG. For example it’s a good idea that you know a programming language that will allow you to create an MMORPG. He also suggests choosing graphics and networking libraries that will be of help to you. He also suggests having some previous game creation experience.

Privantu also suggests making certain decisions that will have a large impact early in the design stages. One thing that should be decided upon early is if you will save data in files or databases. Another thing that should be decided upon early is whether you will make the game threaded or non-threaded. Another very important aspect you should decide upon early is whether your game will be 2D or 3D.

Privantu further suggests that you have a good security system in place on your MMORPG because otherwise the users will hack into your system and change everything that they can. He also suggests making a well balanced team to create the MMORPG. Privantu then mentions that some of the myths surrounding how the experience of creating an MMORPG is. Such as that only big companies can make an MMORPG.

This is related to my work because I have created a database that will store information from the Game2Learn games in the UNC Charlotte Game Lab. These games may someday be played online by students who are learning computer science. It is important to know which skills are needed so that if I have any problems with this database I can use ideas from those fields in order to improve it.

Yee wanted to see some empirical data in Motivations for Play in Online Games[v] about what kinds of player types there are. He used a 40 question survey with question related to Bartle’s Player Type taxonomy. They also used questions based off of previous surveys taken by people whom play massively multiplayer online role playing games (MMORPGs). He surveyed 3,000 people from different MMORPGs.

Yee found that there are three basic archetypes of players with three sub-types to each archetype. The archetypes are achievement, social and immersion. The sub-types for achievement are advancement, mechanics and competition. The sub-types for social are socializing, relationships and teamwork. The sub-types for immersion are discovery, role-playing and customization.

The relates to my work in that it is very likely the database I have created will at some point be taking information from online games from the Game2Learn project, and it is important for me to know what kinds of players there are besides students that may decide to someday play the games which are connected to the database.

In Pre-games: Games Designed To Introduce CS1 and CS2 Programming Assignments[vi], Giguette was concerned about whether or not games for educational purposes are effective. He lists a few problems involved with determining whether or not this is true. The first problem is that people playing games do not always follow the instructions. They will often skip as much as possible to get into the "meat" of a game. Another problem is getting the students to actually work through the problems. The students will try and ask for assistance working though anything that they do not think they can understand. Yet another problem addressed is that students do not understand what designing an algorithm is.

However Giguette believes that there are some advantages to using games for educational purposes. For example games provide a concrete example, it gives a more solid and visual way to think of programming assignments. Another example of this is that doing this allows for experimentation since the students know ahead of time that games are hard but still try to overcome the obstacles within it. Also students regard algorithms in a game environment as a strategy which is yet another advantage.

Giguette has not done any research to see whether or not his ideas are true. As a result I cannot give you any percentages as to his accuracy, or who learned the most from his techniques or anything else that you may wish to know about his research.

This relates to my work in that the games from the Game2Learn project all involve teaching students about computer science and the information taken from those games will go into the database that I have created.

Assiotis and Tzanov are interested in ways of improving server performance near borders in Massive Multiplayer Online Role Playing Games (MMORPGs). In A Distributed Architecture for MMORPG[vii] they propose that splitting a few large servers into many smaller servers will help to solve some of the problems found within large servers.

Lin, Kemme, Patino-Martinez, and Jimenez-Peris are interested in the effects of replicating databases for multiplayer online games (MOGs). In Applying Database Replication to Multi-player Online Games[viii] Lin, Kemme, Patino-Martinez, and Jimenez-Peris discuss their proposed solution for the problems that become apparent when using database replication with an MOG.

This is related to the work I am doing because at some point there may be enough information in the database I am implementing that someone may decide they need to have

replicas of the database. So it is important to know the different ways of implementing replicas so all of the information in the replicas match the database itself.

[I] Kelleher, C., Pausch, R., and Kiesler, S. Storytelling Alice motivates middle school girls to learn computer programming, In Proc. CHI 2007, ACM Press (2007), 1455--1464.
[II] Ducheneaut, N., Yee, N., Nickell, E. and Moore, R.J., The life and death of online gaming communities: a look at guilds in world of warcraft. In Proc. CHI 2007, ACM Press (2007), 839 -- 848.
[III] *Using a simple MMORPG to teach multi-user, client-server database development*
Authors: Greg Wadley, Jason Sobell
[IV] A Beginner's Guide to Creating a MMORPG by Radu Privantu 06/08/2004 © 2003-2004 DevMaster.net. All Rights Reserved
[V] Yee, N. (2006). Motivations for Play in Online Games. CyberPsychology and Behavior, Vol. 9, No. 6: 772-775.
[VI] Ray Giguette, Pre-games: games designed to introduce CS1 and CS2 programming assignments, Proceedings of the 34th SIGCSE technical symposium on Computer science education, February 19-23, 2003, Reno, Nevada, USA
[VII] M. Assiotis and V. Tzanov. A distributed architecture for MMORPG. In *Proceedings of NetGames '06*, page 4, October 2006.
[VIII] Lin, Y., Kemme, B., Patino-Martinez, M., and Jimenez-Peris, R. 2006. Applying database replication to multi-player online games. In *Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support For Games* (Singapore, October 30 - 31, 2006). NetGames '06. ACM, New York, NY, 15. DOI= http://doi.acm.org/10.1145/1230040.1230080