Middleware for creating peer to peer instanced duengons in server client mmorpgs

The goal of this project is to explore the feasabilty of using the peer to peer architecture in augmenting the server client model used in mmorpgs.
Server client has been the backbone of the mmorpg market since its inception and the reasons for this have been well researched.
The main advantages are security, simplicaty and speed.
The peer to peer model on the other hand has some definetsiy in this area, meaning that a solely peer to peer mmorpg would require a lot more work than a server client one.

I aim to combat the weaknesses in the peer to peer model by using a hybrid peer to peer and server client model, using peer to peer for the main gameplay packets and the server client to add security and anti cheat measures to the mmorpg.
I have chosen to create instanced dunengons as these are self contained areas of the world in which usually 5 players interact. This simplfiys the peer to peer process as the clients involved are static.

In order to create a middleware solution I need an engine to work with, so I have decided to create a simple mmorpg as a test bed. It will be required to have a game world with the player represented as an avatar with some simple combat to create a complex enough game on which to test my middleware.

Once i've created a simple mmorpg I will work on creating the peer to peer middleware to plug into the software.

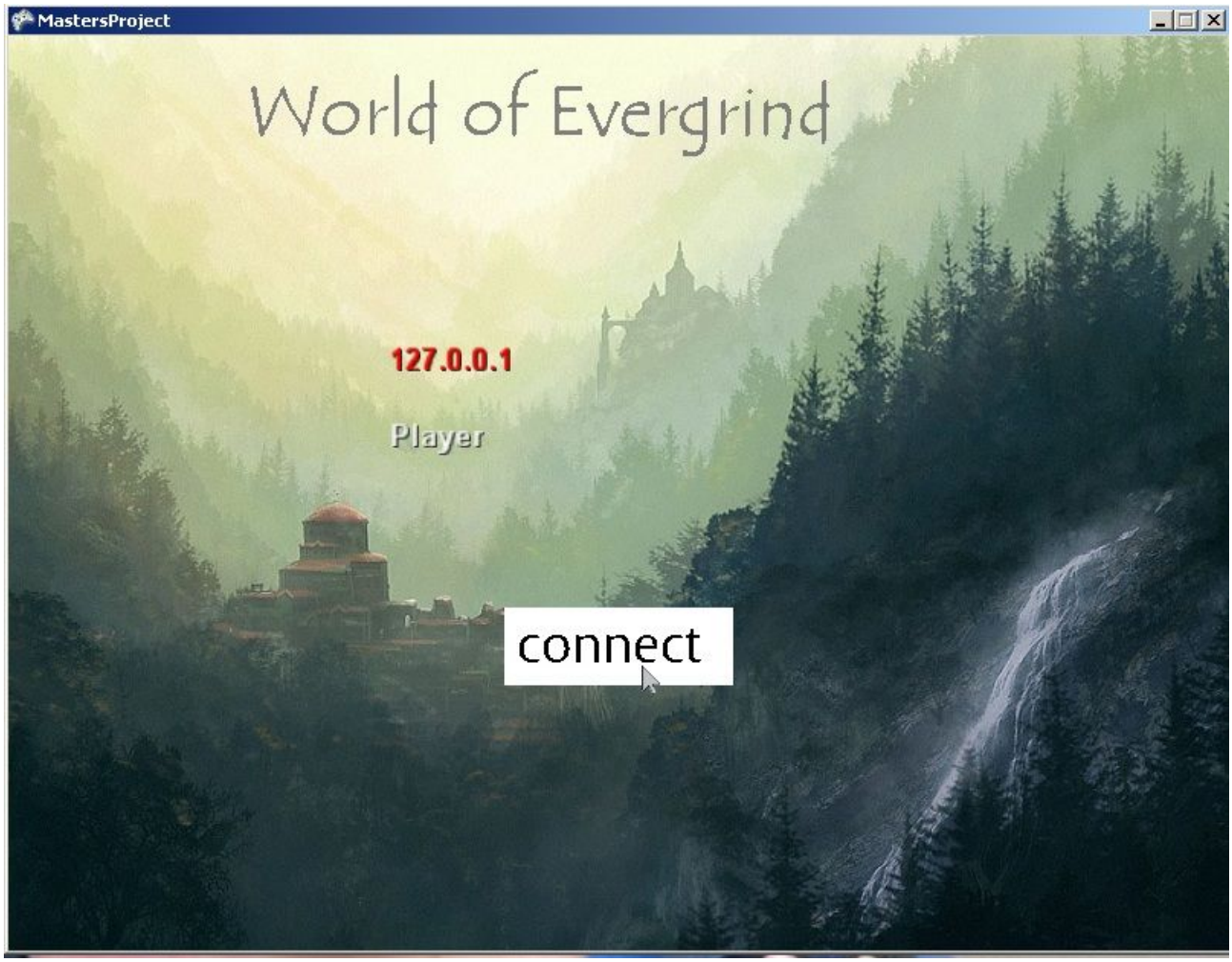
I've decided to create a seperate server from the game world which will handle all the instances in the game world in a lightwieght manner, allowing for a single server to run what would have taken many.
This server which I have nicknamed the “Arbiter” will be responsible for setting up the world state and distributing it the the players in the instance, then it will casually watch the progress in each instance and resolve any disputes.
By having a server which is always right we avoid the “Byzantine Generals Problem”.

The main advantages that I forsee in using this system are the speed benefit geographally similar players will enjoy, meaning even if they are on the other side of the world from the server they should still enjoy good latency. Also the lightened load on the server should allow for less servers to be used to run the mmorpg.

The work done so far

Since the start of the project I have gather various papers on the subject area which I feel will be useful to reference.
I've also create a simple mmorpg using XNA and .Net which allows for players to join the server and explore the world and fight mobs with simple AI.
Chat is also implemented in the game, allowing text messages to be broadcast to anyone playing the game.
For development purposes i've implementated a level editor in the client along with level loading.
On the server there is a database stored in an external file for all the player information such as there location and level.
The server is multi-threaded with each thread handling a single client.
Simple combat is functional and being handled server side.

Screenshots:



Logon screen for the game, allowing you to enter the servers IP and your player name.
Player name is linked to you character and it acts as your unique identifier.

Sample game screen. Shows the tilemap, enemies, player and the health bar in the top left.
The server window is the server application showing the console.
Console currently only outputs client connection messages.

problems encountered, solutions attempted

As I created the base network design for my mmorpg I ran into the problem of trying to use a loopback address with UDP. There doesn't seem to be a simple method of loopback so I decided to use TCP for all network communication. This is not the standard however as i've not always got access to more than one machine for development I felt that it was a acceptable compromise.

While developing my mmorpg i've had to go back and rework the network model a few times, I feel that this was expected as this is the first time i've tried to develop an mmorpg so i've come across deficiencies as i've added more features.

skills acquired

I've learned a lot from creating this half of the project, from network connections to the importantance of server side design.
I feel a lot more confident in working with networking in general and the basics of mmorpg creation.
As I move onto the peer to peer part of this project I feel i'll learn a whole lot more and i'm looking forward to it.

prioritised list of what needs to be done

1. Finish the leveling/XP system
   Once I have finished this system off I feel that I will have completed the first part of my project and I will be ready to start creating the middleware side of the project.

2. Data hooks for the middleware
   Next on my list will be creating a base middleware dll which will have functions for taking in the required data from the server and client in order to create the instance.

3. Networking functionailty for the middleware
   The main work remains in creating the peer to peer networking model in my game, this is quite a large task but I feel I should be able to complete it in the required time.

4. Proofs
   The final stage will be to create proformance measures and cheating functions to test the secruity and functionality of the program.

proposed “chapter plan” for the final report

Introduction

brief description of MMORPGS
Setting the scene with a description of general mmorpg's and giving the reader an idea of the market.

Server client model
Describing the server client model used in most mmorpgs, along with the strengths and weaknesses

Peer to peer networking
A look at how peer to peer networking has been used in games, along with the strengths and weaknessees

Proposed Idea
Describing how creating the hybrid model would help an mmorpg and a detailed explanation of what I intend to do.

Creation of the testbed mmorpg
Descussing the creation of the testbed mmorpg, including design choices and implementation.

Implementation of the peer to peer middleware
Descussing the creation of the peer to peer middleware, including design choices and implementation.

Testing
Stress and functionailty tests, hopefully showing some improvement.
Discuss results**.**

Conclusion
Wrap up the paper discussing the success and failures of the project and how it could be expanded/improved.