Chapter 7 Conclusions and Future Works

7.1 Conclusions

Middleware plays an important role in the development of next generation MMOGs. In the dissertation, I surveyed some of the current technologies that focus on providing a scalable, reliable, fault-tolerance, lower cost, load balance, single sign-on, and secure framework to build a seamless virtual world. However, a MMOG platform needs to address not only the service aspect, but also code maintainability and development for programmers, especially ease of development, deployment, maintenance, and versioning.

After a careful analysis, I found that MMOG’s behavior is generally modeled as driven by events. Second, in most cases in client, many codes can be executed in parallel. Furthermore, message-oriented Middleware (MOM) technology has the advantage of allowing the platform to decouple the versioning relationship for both the client and the server. We then described the building of our easy, compact and high performance message-oriented middleware with a code-generation programming model to answer most of the above-mentioned problems.

In summary, this dissertation provides several contributions for those who try to build the essential framework to support a MMOG.

(1) To provide a scalable MMOG service, an efficient message-based network engine plays a important role. Moreover, the efficient and customizable message-based network engine reduces complexity for application programmers designing the protocol. This dissertation tried to stick to the idea that “simple is better.” In
addition, the network engine helps developers focus on creating simple, high-impact protocol descriptions without worrying about network programming. Define the protocol precisely also makes the generated protocol handlers more compact, thus increasing network-transmission performance without as much overhead as will encounter in CORBA.

(2) To make development, deployment, and change of content easier, a code generation programming model is introduced here to help content provider builds a MMOG rapidly. The model reduces the work load of the programmers. In addition, protocol designer can modify the message protocol by change the message attributes’ order in the description document, and generate the code rapidly. This code generation model also address security and content-updating issues, our platform models the content-oriented protocols as sets of message fields; the code-generator engine randomly shuffles the fields, thus increasing protection against hacking message-oriented protocols and faking messages. Game vendors undergo this automatic updating process periodically and easily, and it shouldn’t give them, or players, too much trouble. Although this won’t provide total protection from hacking, it makes attacks more difficult. Moreover, including both messaging protocol and encryption algorithms, such as Secure Sockets Layer (SSL), can help prevent attacks.

(3) To Leverage future management concepts, a n-tiers hierarchy model is introduced here to provide a scalable management architecture in MMOG environment. The code generation model helping programmers build a manageable object rapidly and easily. The framework here can scale from small objects to a large distributed computing environment and all the services can be dynamically loaded, unloaded, or updated in the management infrastructure. In the development of the MMOG management system, JMX is extended to solve the network communication problems and an
Mbean server is used to implement the n-tier management server architecture. The method should make the management task more flexible convenient.

(4) To build a rapid development environment, separating out the various tasks of the system administration, application programming and communication configuration is an important job. In a large scale distributed computing environment, it is important to separate out the various tasks in the software engineering process. In the model presented here, the system administrators/analyzers can define only the interfaces necessary for management tasks. The programmers who employ the interface generated by the code generation engine do not need to know the detail of the communication technology.

7.2 Future Works

Building a MMOG middleware is a great challenge. Some of the future goals of a MMOG middleware, which are not addressed in this dissertation, are set out below.

**High Availability**

To build a modern MMOG middleware, middleware with distributed technologies always plays an important role. The architecture purposed in this dissertation is the same. In order to handle large amounts of avatars and game content simultaneously, MMOG is always hosted on lots of different hardware such as multiple servers, gateways and network devices. However these sophisticated devices are not as truly reliable as people desire. Each crash will cause serious damages to players’ benefit and MMOG companies’ profit. How to build a high available MMOG with middleware support is very desirable.
Beside the availability problem caused by hardware crash, another aspect of high availability can be done by adopting dynamic load balance scheme. Building a seamless game world requires tightly dynamic collaboration between cell servers. The processing of game interaction, for example, boundary update, and user state migration, is spread across a network of server farms. Although such a scheme is very important for next generation MMOGs, it is not featured in many current models. It helps the server maintain quality of service whenever the numbers of online players increase, and also helps server scalability when vendors need to add more servers to a cluster and build a seamless virtual world. However, since the geography of player states is an essential in the virtual world, a good dynamic load balance scheme is very desirable.

**Security**

Security is always a problem of pressing importance for Internet services, and for MMOGS, the needs are for both network and in-game application security. While most security needs can be met by industry standard firewalls, software and hardware keys, in-game (application level) cheating remains a troubling question.

**Persistent**

In order to reduce the developing effort and provide a better MMOG environment, persistence components are necessary. They will not only handle manipulate relation-database but also provide a better method to use. For the reason, those components should fulfill the requirement: easy-to-use, high-performance and transparent persistent.

Generally speaking, there are always thousands of people, even ten thousands of people online at the same time in a MMOG. So, the performance issue is very
important to us. But, how can we respond these plenty of uses in a short time? It is a
very important issue for us to solve. Also, relational database is a common commodity
nowadays. If programmers design and program to save game data in object point of
view instead of tables, we think this will reduce many addition works for
programmers.