네트워크 기반 다자간 아바타 채팅 시스템의 구현

김종수, 권오준, 김태석
동의대학교 소프트웨어공학과

An Implementation of the Avatar Chatting System for Network-Based Multi-User Environment

Jong-Soo Kim, Oh-Jun Kwon, Tai-Suk Kim
Dept. of Software Engineering, Dong-Eui University
E-mail : saso01@hanmail.net

요 약

다자간 채팅은 인터넷 기반에서 진보적인 발전을 한 분야이고, 네트워크 게임과 같은 어플리케이션 제작에 있어 필수적인 기술이다. 본 논문에서는 자바 기술을 이용하여 one-tier, two-tier 구조가 가진 불편한 유지보수, 데이터 은닉의 문제점을 해결하기 위하여 multi-tier architecture로 아바타를 사용하여 다자간 채팅시스템을 구현하였다. 또한 채팅에서 흥미를 유발하기 위한 캐릭터 전송기술, 다양한 문자를 보내기 위한 문자 포맷 기술을 구현하였다.

1. 서론

오늘날 네트워크상에서 캐릭터를 이용한 어플리케이션은 2D에서 3D로 발전하고 있으며, 아주 일반화되어 가고 있다. 많은 어플리케이션 사용자들은 자신만의 아바타를 가지기를 원하고 있고, 이에 부응하는 산업이 점차 활성화되고 있는 추세다.

인터넷이 현재 매우 대중화되어 있고, 많은 회사들이 웹에서 다양한 컨텐츠를 서비스하고 있다. 인터넷상에서 대규모 커뮤니티를 형성하기 위한 가장 좋은 방법이 네트워크 게임 서비스이다. 현재 국내의 유명한 몇몇 사이트들이 회원 유치를 목적으로 캐릭터와 아바타를 응용한 고품질의 네트워크 기반 어플리케이션을 서비스하고 있으며, 일부 업체에서는 서비스의 유료화에 성공하여 상업화로 정착되고 있다.

본 논문에서는 웹 기반에서 자바 언어로 구현된 기존 문자 기반의 채팅 시스템에서 발전하여 아바타와 이미지를 이용하여, 사용자간의 흥미를 유발하기 위한 다자간 채팅 시스템을 구현을 목표로 하고 있다.

본 시스템은 채팅을 위한 클라이언트측 구현, 로그인한 클라이언트를 그룹화 하는 서버측 구현 및 데이터베이스를 통해 회원의 정보를 조작하기 위한 데이터베이스측 구현의 세 부분으로 구성되어 있다.

구현된 어플리케이션에서 자바 기술이 사용되었기 때문에 향후 모바일 인터넷 환경에서 사용되는 PDA나 Mobile Phone으로 전이가 용이하다는 장점이 있다. 또한 사용자 정보와 같은 중요한 자원을 보호하기 위해 multi-tier 구조를 채택하였고, 향후 소프트웨어 유지 보수에 유리하도록 MVC(Model-View Controller)디자인 패턴을 채택하였다. UML(Unified Modeling Language)을 바탕으로 설계하여 객체지향 언어인 자바가 가지는 장점을 이용하기 위해서 잘 설계된 객체지향 구조들이 가지는 패턴을 이용하였다.

객체지향 시스템의 품질을 측정하는 방법들 중 하나는 개발자들이 객체들 사이의 협력 방법에 얼마나 주의를 기울였느냐 하는 것을 판단하는 것이다. 본 연구에서는 UML로 구현되는 다양한 디자인 패턴에 대한 기법들을 사용하여 시스템 구조가 간단하며 이해하기 쉽게 구현하였다.

2. 시스템 구성

본 어플리케이션의 구현에 있어 가장 큰 골격은 클라이언트/서버 기반의 multi-tier 구조다. 이 구조는 one-tier, two-tier에 비해 다음과 같은 장점이 있다.

- 광범위한 데이터를 읽기 쉽게 만들어 준다.

- 네트워크를 통해 쉽게 접근하게 해준다.
- 데이터를 쉽게 은닉할 수 있도록 한다.

다음의 표1 에서 각 tier의 소프트웨어 계층과 물리 계층에 대해서 요약하였다.

구분	논리적 소프트웨어 계층		물리 계층
1 tier	Presentation	데이터를 받거나, 가공(User Interf_ace, Window, Screen)	Client
2 tier	Business logic	비즈니스 룰 수행	Application Server
3 tier	Data & Application	데이터 저장과 액세스	Database Server

표1. 각 tier의 소프트웨어 계층과 물리 계층

본 논문에서 구현된 어플리케이션은 그림1과 같이 3개의 tier로 구성되며 다음과 같은 역할을 한다.

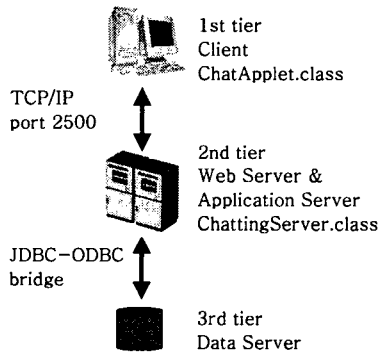

그림1. Multi-tier architecture 설계

클라이언트는 네트워크 서버의 ServerSocket이 accept() 메쏘드를 이용하여 기다리고 있는 특정 포트에 접속하기 위한 소켓(Socket)을 생성함으로써, 네트워크 서버에 대한 연결 설정을 시도한다. ServerSocket을 이용하여 특정 포트에서 기다리고(accept) 있다가, 클라이언트가 연결 설정을 시도하면, 해당 클라이언트와 연결을 위한 소켓을 생성하고, 다음으로 이 클라이언트를 관리하기 위한 객체의 thread 클래스의 객체를 생성함으로써, 클라이언트와 서버의 네트워킹이 가능하도록 한다.

어플리케이션 설계의 기본 전략은 GUI와 프로토콜을 독립적으로 개발할 수 있는 전략인 Model View Controller(MVC) 디자인 패턴을 적용하였다.

자바 언어의 주요 특징은 플랫폼에 독립적이라는 것이다. 그렇지만, 다계층으로 구성된 어플리케이션은 서버 측의 자원에 따라서 다른 API를 사용해야 할 경우도 있으므로 다음과 같은 환경에서 시스템을 구성한다.

- 서버측 환경
  - 서버 : Windows 2000
  - 웹 서버 : IIS
  - 데이타베이스 : MS_SQL Server 2000
  - 개발도구 : JDK1.3

- 클라이언트측 환경
  - 웹 브라우저 : Internet Explorer
  - 테스트 및 개발 : AppletViewer
  - 개발도구 : JDK1.1

3. 어플리케이션 설계

흥미로운 채팅을 구성하기 위한 GUI설계는 전체 어플리케이션 설계에 있어 중요한 부분 중의 하나이다. 사용자 인터페이스는 여러 부분이 있을 수 있으나 채팅 어플리케이션에서 반드시 있어야 할 것이 어떠한 방이 채팅 방으로 개설이 되었다는 정보(이하 대기실)와 실제 채팅에 참가한 사용자들이 사용하는 채팅 방의 설계이다.

대기실은 본격적인 채팅을 하기 위한 준비 단계로써 다른 채팅 어플리케이션과 크게 다르지 않다. 대기실의 설계는 그림2와 같다.

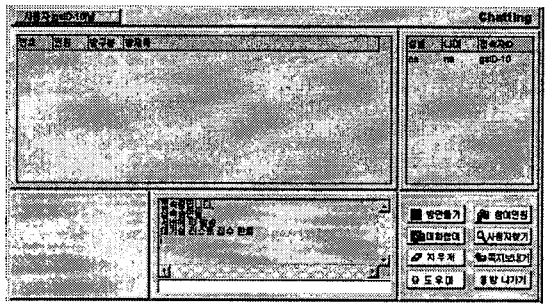

그림2. 클라이언트측 대기실 GUI 설계

채팅방의 설계는 언제나 개선의 여지가 필요하며 사용자들의 흥미를 유발하기 위한 아주 중요한 부분이다. 채팅방에서 구현된 기능 여하에 따라서 채팅관련 어플리케이션의 성능이 좌우된다. 본 어플리케이션은 사용자의 흥미를 유발하기 위해 기존의 채팅 기능을 수용하고 사용자가 개별적으로 소유할 있는 아바타에 대한 욕구를 충족시키기 위해 그림3과 같이 설

계하였다.

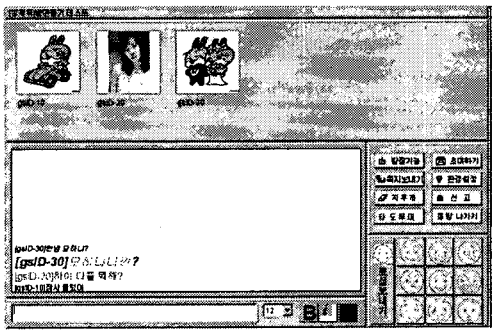

그림3. 클라이언트측 채팅 방 GUI 설계

주요 클래스 설계시에 초점을 둔 부분은 자바 언어가 가진 객체지향의 장점을 최대한 살리면서, 추후로 개발될 새로운 어플리케이션의 대해 기존에 개발된 클래스의 재사용성을 높이기 위한 부분에 초점을 두었다. 여기서는 자바 언어를 이용하여 다중사용자를 위한 네트워크 채팅 어플리케이션을 작성한다. 클라이언트 서버 모두 썬 마이크로 시스템이 제공하는 자바 개발 도구인 JDK를 사용하였다.

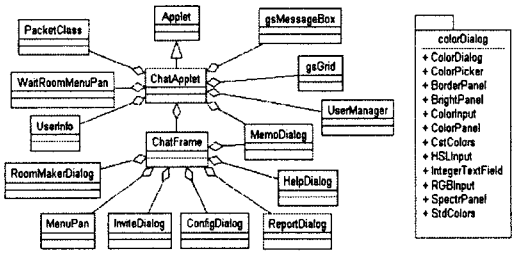

그림4. 클라이언트측 UML 설계

채팅 어플리케이션 서버의 주요 기능은 여러 명의 클라이언트가 같은 방에서 채팅할 수 있도록 구성해 주는 것이다. 또한 안정적인 서비스를 할 수 있어야 좋은 채팅 어플리케이션 서버라고 할 수 있다.

채팅 서버는 Applet 기반인 클라이언트와는 다르게 JDK1.1의 제약을 벗어난다. 그래서 안정적인 서버를 구축하기 위해서, JDK 1.2 버전부터 제공이 되는 자바 기술 중 Collection API를 사용하여, thread를 사용함으로써 얻게 되는 서버의 부하를 감소시켰다.

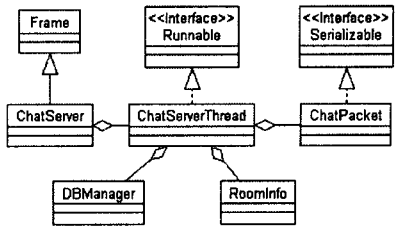

그림5. 서버측 UML 설계

- ChatServer

클라이언트의 연결 설정을 기다리는 클래스이다. (ServerSocket의 포트에서 accept() 메소드를 이용하여 클라이언트가 접속하기를 기다린다) 클라이언트가 접속을 시도하면, 클라이언트와 네트워킹하기 위한 소켓을 생성하고, 이 소켓을 ChatServerThread의 객체에 매개 변수로 전달하여 클라이언트와 직접 패킷을 주고받을 수 있도록 한다.

- RoomInfo

대화방을 관리하기 위한 클래스이다. 클라이언트의 방 개설 요청이 있으면, 방 번호, 방 아이디, 방 타이틀, 방의 사용자 아이디, 사용자의 소켓을 관리한다.

- ChatServerThread

클라이언트와 네트워킹을 전담하기 위한 클래스이다. 각 클라이언트와 연결된 소켓을 가지고 있다. 클라이언트가 방 개설을 신청하면 RoomInfo객체를 사용하여 방의 정보를 저장한다.

- DBManager

클라이언트의 접속 시 클라이언트에 대한 정보를 가져오고, 접속 종료 시, 클라이언트에 대한 정보를 갱신한다.

그림6은 앞에서 설명한 주요 클래스로 구성된 채팅서버의 수행을 보여 주고 있다.

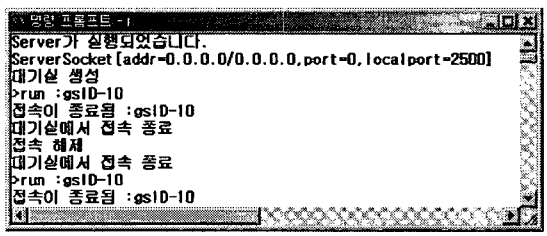

그림6. 채팅 서버 실행

채팅은 Web에서 실시간 자료의 전송이 가능해야

하므로, TCP/IP(Transmission Control Protocol/Internet Protocol)기반의 프로토콜을 채택하였다. 자바언어는 ServerSocket과 Socket에 기반으로 패킷을 주고받는다. 다양한 스트림 모델을 제공하는데, 본 논문에서는 객체를 주고받을 수 있도록 ObjectInputStream과 ObjectOutputStream을 사용하였고, 패킷의 전송에 Serializable 인터페이스와 협력하는 ChatPacket클래스의 객체를 클라이언트와 서버가 주고받으면서 통신한다. 표2에 클라이언트/서버간의 프로토콜을 정의하였다.

Protocol	기 능
ICR	권한 확인
IDF	신상 명세 발송
SRR	방만들기 요청
SRC	방 리스트 접수
GMR	리스트 접수 완료
MRD	방만들기 불허
OVU	정원 초과 메시지
RET	없어진 방
SRR	방만들기 요청
RGD	초기 게임 데이터 접수
RRE	방 참가 요청
ERM	방 나가기 메시지 발송
ACD	접속 승인 요청
CHT	대기실 채팅 메시지
SUL	대기실 유저 리스트 접수
STC	방/대기실 리스트 접수
MRA	방만들기 허용 접수
ARE	방 참가 허용 접수
RID	룸 아이디 다름
RLL	방 참가자 리스트 갱신
RUL	방 참가 리스트 후 초기화
RCM	방 채팅 메시지
RFS	새로 고침
GDT	게임 데이터 접수
END	애플릿 종료

표2. 클라이언트와 서버간의 프로토콜

4. 결론

본 연구에서 multi-tier 구조에 기초하여 아바타를 이용한 채팅을 구현하였다. 아바타를 이용한 채팅은 사이트의 일반 사용자 측면에서 볼 때, 기존의 문자에 기초한 채팅보다 흥미 있는 채팅을 할 수 있다는 장점이 있다. 또한, 채팅을 서비스하는 기업의 측면에서 볼 때는 보다 많은 회원을 모집할 수 있으며, 사이트에서 회원들의 커뮤니티를 형성하는데 도움이 될 수 있다. 그리고 아바타를 사용자에게 판매함으로써 직접적인 수익을 마련할 수도 있다.

이번의 구현은 아바타를 추가한 네트워크 기반의 채팅 솔루션의 기능을 중심으로 구현이 되었다. 향후 지속적인 유지보수를 통하여 좀더 사용자 편의적인 어플리케이션으로 연구할 예정이다.

본 연구가 아바타를 이용한 채팅에 있어 많은 부분의 기능 구현했지만 사용자에게 흥미로운 채팅을 제공하기 위해서 다음과 같은 부분들이 향후 연구되어야 할 것이다.

- 편리한 유지 보수를 위한 MVC 디자인 패턴의 강화
- 3D 아바타의 구현
- 다이나믹한 채팅을 위한 애니메이션 보강
- PDA와 모바일 솔루션으로의 전이

위에서 제시한 내용 외에도 흥미로운 채팅을 위한 추가적인 사항이 더 있을 수 있지만, 어플리케이션의 구현에 따르는 많은 시간과 비용이 필요하므로 본 논문에서는 생략하였다.

[참고문헌]

[1] Patrick Naughton, Herbert Schildt, *The Complete Reference JAVA 1.1*, Osborne Mc Graw Hill, 1998(1028 pages)

[2] Patrick Chan, Rosanna Lee, doublas Kramer, *The Java Class Libraries Second Edition Volume1*, Addison Wesley, 1988(2050 pages)

[3] James Gosling, Frank Yellin, Java Team, *The Java Application Programming Interface Volume1*, Addison Wesley, 1996(494 pages)

[4] John Lewis, William Loftus, *Java Software Solutions Foundations of Program Design*, Addison Wesley, 1998(857 pages)

[5] Sun microsystems, *sun educational serviecs Java Programming SL-275, SunSoft Press, 2000*(720 pages)

[6] Sun microsystems, *sun educational serviecs Advanced Java Programming SL-300, SunSoft Press, 2000*(400 pages)

[7] Erich Gamma, Richard Helm Ralph Johnson, Hohm Vissides 공저 Gof의 디자인 패턴, Pearson education Korea(440 pages)