행위 패턴을 사용한 네트워크 턴 게임 API 설계 및 구현

김종수°, 김태석, 김성우
동의대학교 소프트웨어공학과

Design and Implementation of the Network Turn Game for API Using Behavioral Patterns

Jong-Soo Kim, Tai-Suk Kim, Seong-Woo Kim
Dept. of Software Engineering, Dong-Eui Univ.

요 약

네트워크 게임 제작과 관련된 설계 기술은 기업의 보안 사항에 해당되므로 기술 공유가 어려운 실정이다. 인력과 시간이 많이 드는 네트워크 게임제작에서 기존에 작성된 코드를 재사용이 가능하도록 설계하는 것은 중요한 일인데, 코드의 재사용을 극대화하기 위해서 지금까지 사용되고 있는 여러 가지 디자인 패턴들을 사용하는 것은 소프트웨어 설계에 많은 도움을 준다.

본 논문에서는 네트워크 게임 서버에 로그온한 사용자가 차례를 기다려서 게임하는 턴 게임 설계와 구현을 통해서, GoF(Gang of Four)가 제안한 행위 패턴들이 어떻게 적용될 수 있는지를 연구하고, 자바언어를 이용한 네트워크 게임 개발에 사용될 수 있는 효율적인 API(Application Program Interface)를 제안한다.

1. 서론

첨단 게임은 영화나 오페라를 능가하는 지식기반 종합예술로서 음악, 영상, 시나리오, 디자인 등 다양한 분야의 전문적인 컨텐츠로 구성된다.

현재 우리나라에서도 발달된 네트워크 인프라와 디지털 기술을 바탕으로 SW산업이 기간산업으로 성장할 수 있는 기회를 맞고 있다.

게임 소프트웨어뿐만 아니라 모든 소프트웨어들의 라이프 사이클이 짧아지고 있는 추세이므로, 보다 재미있는 게임을 얼마나 빨리 개발할 수 있느냐 하는 것은 중요한 일이다. 그러나 게임 소프트웨어를 설계하고 개발하기 위한 기법은 기업의 중요한 자산이므로 보안이 철저하고 정보의 공유가 힘들다. 이러한 이유로 게임 개발 기술에 적용되는 설계기법의 효율성에 대한 평가는 거의 없다.

소프트웨어 개발에 있어서 객체 지향 패러다임을 적용하는 이유는 여러 가지가 있지만, 분산된 개발을 가능하게 하고, 기존에 개발된 소프트웨어의 재사용을 쉽게 한다는 장점이 있기 때문이다.

소프트웨어 재사용과 관련되어 연구되고 있는 분야가 디자인 패턴과 관련한 분야인데, 본 논문에서는 자바 언어를 사용한 네트워크 턴 게임 설계와 구현을 통해서, GoF(Gang of Four)가 제안한 행위 패턴 적용에 대해서 연구하고, 네트워크 턴 게임 개발에서 효율적으로 사용할 수 있는 API(Application Program Interface)를 제안한다.

2. 관련연구

본 장에서는 네트워크 턴 게임 구현에 필요한 관련 기술 중 네트워크 게임 엔진과 시스템 구성 그리고 행위 패턴에 대해서 알아본다.

2.1 네트워크 게임 엔진 구성

네트워크 게임 엔진 설계와 제작은 재사용이라는 측면이 고려되어야 하므로, 엔진간의 인터페이스가 서로 종속적이지 않아야 효율적이다. 그림 1 은 일반적인 네트워크 게임에서 엔진의 구성을 보여준다.

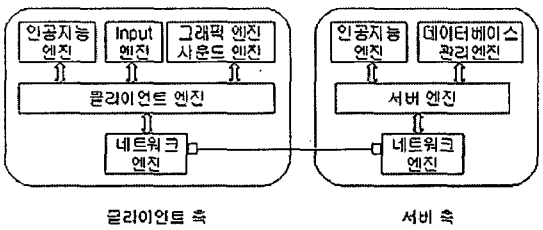

그림 1. 클라이언트/서버 게임의 엔진 구성도

게임 엔진은 크게 클라이언트 부분과 서버 부분으로 나눌 수 있다. 인공지능 엔진은 클라이언트와 서버가 모두 필요하고, Input 엔진과 그래픽, 사운드 엔진은 클라이언트 측, 데이터베이스 관리 엔진은 서버 측에 위치한다.

2.2 네트워크 턴 게임 시스템 구성

어플리케이션 설계의 기본 구조는 그림 2와 같은 MVC(Model/View/Controller) 디자인 패턴을 적용하였다.

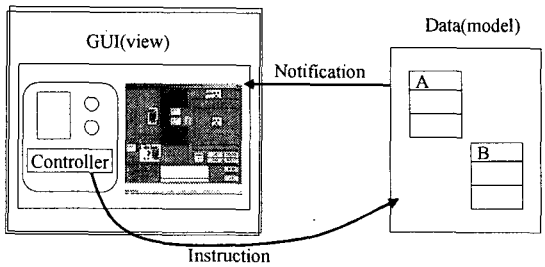

그림 2. MVC(Model/View/Controller) 디자인 패턴

MVC는 기존에 개발된 클래스의 유연성과 재사용성을 증대시킨다. MVC에서 뷰(View)와 모델(Model) 간에 자료를 요청하고 보내는 프로토콜을 만들어 종속성을 없앴으며, 뷰는 외형이 모델의 상태를 반영하도록 작성하였다.

그리고 본 시스템은 클라이언트/서버 기반의 다계층 구조를 가진다. 이 구조는 one-tier, two-tier 에 비해 광범위한 데이터를 읽기 쉽게 만들어 주고, 네트워크를 통한 접근을 쉽게 하고, 데이터 은닉을 용이하게 한다는 장점이 있다.

2.3 GoF의 행위 패턴 고찰

본 연구는 생성 패턴(Creational Patterns), 구조 패턴(Structural Patterns), 행위 패턴(Behavioral Patterns)으로 대표되는 GoF의 패턴들 중에서, 턴 게임 구현에 어떠한 행위 패턴이 적용될 수 있는지 연구하였고, 주 관심이 되었던 패턴들은 다음과 같다.

① Chain of Responsibility

요청을 처리할 수 있는 기회를 하나 이상의 객체에게 부여함으로써 요청하는 객체와 처리하는 객체 사이의 결합도를 없앤다. 요청을 해결할 객체를 만날 때까지 객체 고리를 따라서 요청을 전달한다.

② Iterator

내부 표현 방법을 노출하지 않고 복합 객체의 원소를 순차적으로 접근할 수 있는 방법을 제공한다

③ Observer

객체 사이에 일 대 다의 종속성을 정의하고 한 객체의 상태가 변하면 종속된 다른 객체에 통보가 가고 자동으로 수정이 일어나게 한다.

④ State

객체의 내부 상태에 따라 행위를 변경할 수 있게 한다. 이렇게 하면 객체는 마치 클래스를 바꾸는 것처럼 보인다.

3. 행위 패턴을 적용한 네트워크 턴 게임 구현 및 고찰

본 장에서는 훌라라고 불리는 네트워크 턴 게임 구현을 통해서, 앞장에서 제시한 행위 패턴이 게임 설계에 어떻게 적용될 수 있는지 알아본다.

3.1 어플리케이션 설계 및 구현

전체 어플리케이션은 클라이언트 측으로 로그온한 사용자의 정보를 보여주는 대기실, 2명에서 5명의 사용자들이 동시에 게임을 하기위한 게임 방으로 구성되었고, 서버측은 클라이언트들의 정보와 데이터베이스에서 자료를 가져오기 위한 부분으로 구성되었다.

3.1.1 클라이언트 측 대기실 설계 및 구현

다른 게임 어플리케이션에서와 같이 대기실은 본격적인 게임을 하기 위한 준비 단계다. 본 시스템의 대기실은 그림 3과 같이 설계되었다.

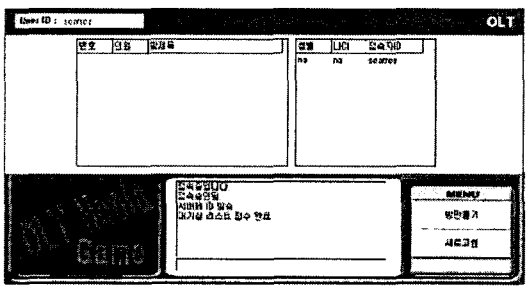

그림 3. 클라이언트측 대기실 GUI 설계

대기실은 개설된 게임방의 정보를 보여주는 리스트와 로그온한 사용자의 정보를 보여주는 리스트, 로그온한 사용자들이 서로 대화할 수 있는 대화창으로 구성되었고, 게임방을 만들고, 서버측에 갱신된 새로운 정보를 요청할 수 있는 메뉴로 구성되었다.

3.1.2 서버 측 게임방 설계 및 구현

본 시스템에서 게임 방 설계는 그림 4와 같이 구현되었다. 사용자의 흥미를 유발하기 위해서 게임방은 여러 가지 다양한 기능이 구현되어야 한다.

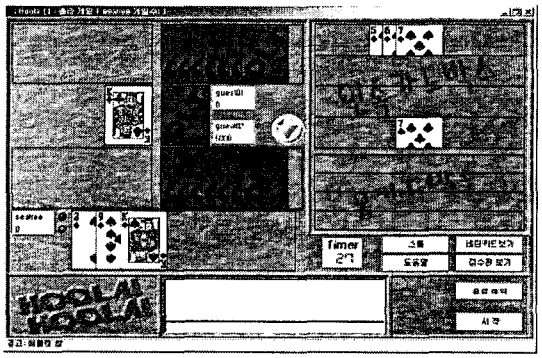

그림 4. 클라이언트측 게임 방 GUI 설계

게임 방은 2명에서 5명의 참여자가 동시에 게임에 참여할 수 있도록 구성되었다. 게임 참가자는 30초 내에서 자신이 하고자 하는 플레이를 하면 된다. 그리고 게임 참여자들이 대화할 수 있는 대화창과 게임 진행 상태를 보기 위한 메뉴, 게임 진행과 관계된 스톱, 종료예약 그리고 시작 버튼으로 구성되었다.

3.1.3 네트워크 게임 서버 설계 및 구현

네트워크 게임 서버 설계는 게임에 참여하는 사용자에게 공유 자원을 효율적으로 분배해주고, 게임 참여자들을 서로 연결해 준다는 측면에서 중요한 기술이다. 그리고 안정성 있는 서버의 구현과 재미있는 게임 요소를 위한 인공지능의 추가라는 면에서 항상 개선의 여지가 많다.

그림 5는 본 연구에서 구현된 네트워크 게임 서

버이다.

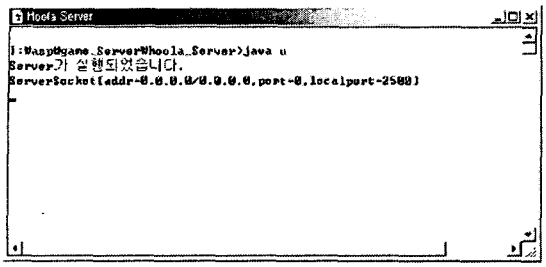

그림 5. 네트워크 게임 서버의 설계

서버는 로그온한 사용자를 관리하기 위한 Users 클래스와 게임 방의 정보를 관리하기 위한 Rooms 클래스와 JDBC-ODBC 브릿지를 이용하여 SQL_SERVER에 접속한 후, 데이터베이스를 관리하는 클래스로 구성되었다. 그리고 클라이언트와 서버는 TCP/IP프로토콜을 사용하여 2500포트로 통신한다.

3.2 Chain of Responsibility 패턴 적용

Chain of Responsibility 패턴은 그림 6과 같이, 메시지를 보내는 객체와 이를 받아 처리하는 객체들 간의 결합도를 없애기 위한 패턴으로 어떤 객체에서 이벤트 처리를 요청하면, 이 처리는 실제 서비스를 제공하는 객체를 만날 때까지 정의된 연결 고리를 따라서 계속 전달된다.

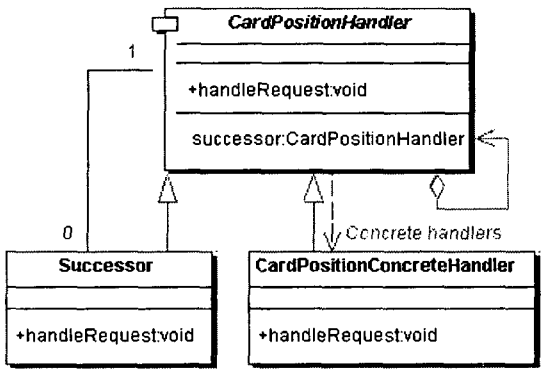

그림 6. Chain of Responsibility 패턴 다이어그램

이 패턴은 게임룸에서 각각의 카드 위치와 고유한 아이디가 부여된 Card 객체를 가지는 User 객체를 결정하는데 사용되었는데, 각각의 사용자에게 카드를 배분하는 게임 초기시에 사용되었다. 그리고 사용자가 새로운 카드를 가져올 때, 카드를 소유할 수 있는 user 객체는 현재 게임에 참여하고 있는 User 클래스의 인스턴스 user1, user2, user3, user4, user5 중의 하나가 된다. 카드가 어느 사용자한테 갈 것인가 하는 것은 현재 자기 차례를 나타내는 플래그(flag)가 참으로 설정된 사용자다.

그림 7. 사용자 객체 탐색 순서

그림 7은 선택된 사용자를 찾기 위해 카드가 위치할 객체를 탐색하는 순서를 보여준다. 이러한 설계를 채택함으로써, 사용자가 가변적으로 변할 수 있는 턴 게임에서 카드 객체와 유저 객체의 결합도를 없앨 수 있다.

3.3 Iterator 패턴 적용

게임 서버 자원의 관리에 있어서, 그림 8에서 보는 Iterator 패턴은 대기실에서 로그인한 사용자의 정보를 관리하기 위한 Users 클래스와 대기실과 게임방에서 현재 개설중인 게임방의 정보를 관리하는 Rooms 클래스에서 사용되었다.

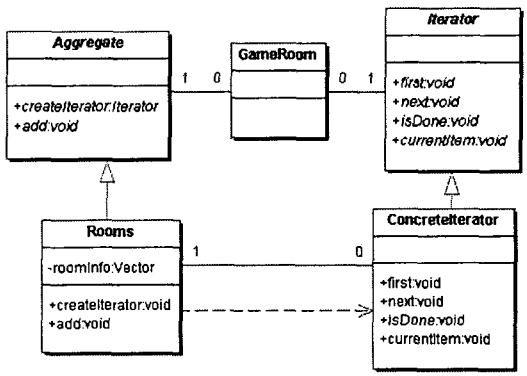

그림 8. Iterator 패턴 다이어그램

현재 개설된 방의 정보와 네트워크를 통해서 로

그온하는 사용자의 상세한 정보는 데이터베이스 서버로부터 가져와서 게임 서버가 관리한다. 또한 실시간으로 개설되고 있는 게임방에 대한 정보도 게임 서버가 관리한다.

본 시스템에서 구현된 Users와 Rooms 클래스는 게임 시스템의 개선이나 확장이 있을 경우, 추가적인 정보를 관리하는 속성이 필요하고 클래스를 수정하는 번거로움 생긴다. 이러한 작업을 줄일 수 있는 방법으로 클라이언트 측에서는 속성의 추가 없이 사용할 수 있는 Vector 클래스를 사용하여 Users와 Rooms 클래스를 구현하였고, 애플릿의 제약을 벗어나는 서버측에서는 탐색과 삽입 삭제를 개선한 Collection API를 사용하였다.

3.3 Observer 패턴 적용

객체의 일관성을 유지하고 결합도를 높이기 위해서 그림 9의 Observer 패턴이 사용될 수 있다.

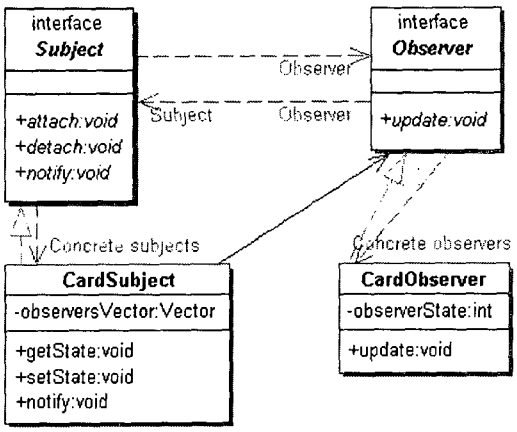

그림 9. Observer패턴 다이어그램

자바에서는 GUI 컴포넌트가 발생하는 이벤트를 관찰하기 위한 메커니즘을 제공한다. 본 시스템의 구현에 있어서도 다양한 객체가 발생하는 이벤트를 처리하기 위해서 Observer 패턴이 사용되었다.

3.4 State 패턴 적용

클라이언트측과 서버측에서 TCP/IP 프로토콜을 사용하여 자료를 주고 받을 때, 자바에서 지원하는 다양한 스트림 모델을 사용할 수 있다. 본 시스템에서는 I/O를 다루기 위한 스트림 모델로 ObjectInputStream, ObjectOutputStream을 사용하였다.

클라이언트와 서버는 그림 10에서 보는 바와 같이 직렬화된 Result 클래스의 객체를 주고 받는다. State 인터페이스를 수행하는 Result클래스의 개수는 클라이언트와 서버가 자료를 주고 받기 위해 필요한 프로토콜이 개수와 동일하다.

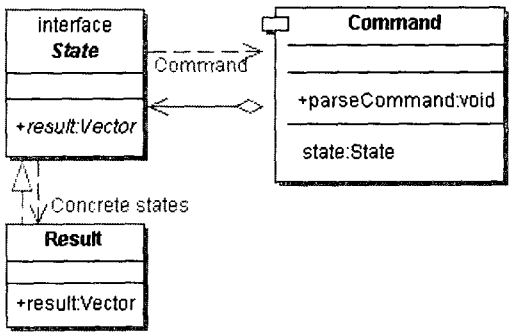

그림 10. State 패턴 다이어그램

패킷의 송신과 수신은 Command 클래스가 가진 parseCommand() 메소드가 담당한다. MVC 패턴에 따라서 클라이언트와 서버간의 프로토콜 정의를 Command 클래스가 담당하였다. 이 방법은 외형인 View와 프로토콜의 송수신에 따른 Controller의 개발을 분산할 수 있는 장점이 있다. 또한 데이터베이스 관리 모듈에서도 State 패턴이 적용되었다.

4. 결론

본 연구에서 다계층 구조에 기초한 네트워크 게임을 구현하였다. 시스템의 구현에 있어서 GoF 의 행위패턴을 적용하였는데, 관심이 되었던 패턴은 Chain of Responsibility, Iterator, Observer, State 패턴이였다.

각각의 패턴은 게임 어플리케이션 구현에 있어서 다음과 같은 부분에 적절하게 적용될 수 있었다.

① Chain of Responsibility : 게임 룸에서 카드 객체와 사용자 객체의 결합도를 없애고 카드를 자유롭게 분배하기 위해 적용
② Iterator : 클라이언트와 서버 프로그램에서 사용자 정보를 관리하기 위한 Users 클래스와 방정보 관리하기 위한 Rooms 클래스 구현에 적용
③ Observer : 다양한 객체가 발생하는 이벤트를 처리하기 위해서 적용
④ State : 클라이언트와 서버가 송수신하는 패킷의 구현과 해석에 적용되었고, 데이터베이스의 관리 모듈에 적용

본 논문에서는 자바를 이용한 네트워크 턴 게임 시스템 설계와 구현에 효율적으로 적용될 수 있는 행위 패턴에 대해서 제시하였고, 연구된 패턴들은 소프트웨어 재사용과 분산작업에 효율적이라는 것을 알 수 있었다.

향후에는 사용자에게 생동감 있는 게임을 제공하기 위해서 DirectX 를 이용한 게임 설계에 디자인 패턴을 적용할 예정이다.

[참고문헌]

[1] 김종수(2003) " 웹을 기반으로 한 JAVA 네트워크 게임 시스템의 설계와 구현" , 부산외국어대학교 대학원, 석사학위논문.

[2] Sun Microsystems, *sun educational services Java Programming SL-275, SunSoft Press, 2000*(720 pages)

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

[4] 김종수, 권오준, 김태석 " 네트워크 기반 다자간 아바타 채팅 시스템의 구현" , 한국멀티미디어학회 춘계학술발표논문집, pp.171-174, 2003.

[5] Sun Microsystems, *sun educational services Advanced Java Programming SL-300, SunSoft Press, 2000*(400 pages)

[6] Sun Microsystems, *sun educational services Java Programming SL-110, SunSoft Press, 2000*(615 pages)

[7] 김종수, 이종민, 김태석 " 생성 패턴을 사용한 네트워크 기반 게임 API 설계" , 한국멀티미디어학회 추계학술발표대회논문집, pp.669-674, 2003.

[8] Soon-Kak Kwon, Jong-Soo Kim, Tai-Suk Kim, "An Implementation Avatar Chatting System for Network-Based Multi-User Environment" EALPIIT2003 National University of Mongolia, Ulaanbaatar, Mongolia July 6-9, 2003 pp.119-123, 2003.

[9] GRAIG LARMAN 원저 " UML과 패턴의 적용" 홍릉과학출판사(705 pages)