생성 패턴을 사용한 네트워크 기반 게임 API 설계

김종수, 이종민*, 김태석
동의대학교 소프트웨어공학과

The API Design for the Network-Based Game Using Creational Patterns

Jong-Soo Kim, Jong-Min Lee*, Tai-Suk Kim
Dept. of Software Engineering, Dong-Eui University
E-mail : seatree@dongeui.ac.kr

요 약

네트워크 게임 제작에 있어서, 요즘의 추세를 보면 화려하고 다양한 캐릭터, 애니메이션, 사운드의 지원으로 게임 플레이어에게 현실감을 느끼게 할 수 있는 요소에 치중하고 있다. 특히 요즘에 서비스되고 있는 실시간 네트워크 게임의 경우, 개발 인원도 많이 투입되는 프로젝트이므로, 객체 지향적 설계 방법론을 따르지 않으면, 좋은 어플리케이션 제작이 힘들다. 국내에서 여러 게임 제작업체가 네트워크 게임의 설계와 구현에 있어서 객체지향 패러다임을 적용하여 자체 어플리케이션을 개발하고 있다. 그러나, 회사 자산 보안상의 이유로 게임 설계 기법이 게임의 설계를 배우고자 하는 사람들에게 제공되기는 힘든 실정이다. 만약, 디자인 패턴을 이용한 다양한 설계기법과 그 적용 예가 여러 사람들에게 제공이 된다면, 보다나은 API의 발전이나, Framework의 개발을 가져올 수 있다. 이러한 견지에서, 본 논문에서는 자바 언어를 사용한 네트워크 게임 제작에 있어서, 적용될 수 있는 디자인 패턴들에 대해 연구한다.

1. 서론

인터넷 인프라 발전에 힘입어 다양한 온라인 네트워크 게임의 제작이 나날이 늘어 가고 있다. 통신망이 고속화 대용량화 되면서, 기존에는 거의 불가능할 것으로 예상되었던 실시간 전략 시뮬레이션 게임들도 속속들이 제작되고 있다.

국내 게임 제작 업체들도 다양한 분야의 네트워크 게임을 제작하고 있으며, 일부 기업에서는 유료화에 성공하여 상당한 매출을 올리고 있다. 또한 중국과 일본과 같은 해외시장에서도 호평을 받고 있다.

게임과 관련한 소프트웨어는 쉽고 다양한 방법으로 마케팅이 가능하므로, 게임 제작사들 간에 여러 가지 다양한 방법으로 게임 엔진들이 개발되고 있다. 그리고 객체 지향적으로 게임 소프트웨어를 개발하기 위한 기법은 회사의 중요 자산이므로 아주 철저하게 보완되고 있다.

게임을 개발 단계는 일반적으로 몇 개로 나누어지는데, 각각의 단계는 모두 중요하므로, 문서 작업도 철저히 해야 하고, 실제 작업이 진행되기 전에 사전에 철저히 검토되어야 한다. 이러한 작업적 특성으로 인하여, 게임 개발 기술에 적용되는 객체 지향 설계기법의 효율성에 대한 평가는 거의 없다.

본 논문에서는 네트워크 게임 개발에 있어서, 객체 지향 패러다임을 적용한다. 소프트웨어 개발에 있어서 객체 지향 패러다임을 적용하는 이유는 여러 가지가 있지만, 분산된 개발을 가능하게 하고, 기존에 개발된 소프트에어의 재사용을 쉽게 한다는 장점이 있기 때문이다.

주된 연구 목적은 웹 기반에서 자바 언어를 사용한 네트워크 게임에서 효율적인 객체 지향 설계 기법을 적용하는 것이다. 네트워크 게임 설계에 있어서 유연하고 재사용이 가능한 설계를 처음부터 정확하게 하기란 매우 어렵다. 일반적으로 어플리케이션의 설계는 많은 시간을 투자하여야 하며, 설계를 최종 마무리하

기 전까지도 이미 만들어진 설계를 여러 번에 걸쳐서 재설계를 해야 하는 경우가 많다.

객체 지향 설계에 경험이 많은 사람들은 좋은 설계를 할 수 있지만, 초보자들이 어플리케이션 제작에 있어 객체 지향 개념을 적용하기란 쉽지 않다. 소프트웨어 설계의 전문가들은 모든 문제를 처음 기초 단계에서 해결하려 하지 않고, 전에 사용했던 해결책을 다시 적용해 보고 좋은 방법을 찾아냈다면 계속 사용하게 된다. 설계 전문가들은 이러한 경험을 통해서 많은 객체 지향 시스템에 있어서, 클래스 패턴들이나 객체들 간의 상호작용이 반복됨을 알 수 있다.

이 연구의 목적은 게임 개발자들이 네트워크 게임 제작에 있어서 효과적으로 사용할 수 있는 형태의 설계의 패턴을 제시한다. 이미 잘 알려지고 검증된 생성 패턴이 네트워크 게임의 어느 부분에 적용될 수 있는지를 연구하는데 목적이 있다.

본 연구에 사용된 언어는 자바 언어인데, 보다 생동감 있는 게임의 설계와 구현에 있어서, 컴퓨터 하드웨어를 직접적으로 다룰 수 있는 C++와 같은 언어의 사용이 필요할 수도 있다. 그렇지만, UML의 사용이 어떠한 언어를 사용하든지 간에 어느 정도 개발 언어와 무관한 독립성을 보장해 준다.

현재 연구된 디자인 패턴을 이용하면 좋은 설계나 아키텍처를 재사용하기 쉬워진다. 입증된 기술을 디자인 패턴으로 표현해 두면 새로운 시스템 개발자들은 디자인 패턴을 더 유용하게 사용할 수 있다. 디자인 패턴은 설계자로 하여금 재사용 가능한 설계를 선택하게 하고 재사용을 방해하는 설계는 배제하도록 도와준다. 또한 이미 만든 시스템의 유지보수나 문서화도 개선해 주며, 패턴화를 통해서 클래스 명세를 정확하게 하며, 객체 간의 상호작용 또는 설계의 의도 등을 명확하게 정의할 수 있다.

현재도 시스템 개발에 있어서, 여러 가지 다양한 디자인 패턴에 대한 연구가 이루어지고 있지만, 본 연구에 있어서는 검증되지 않은 디자인 패턴을 적용하지 않았다.

2. 시스템 구성

본 시스템의 구현에 있어 가장 큰 구조는 클라이언트/서버 기반의 multi-tier 구조다. 이 구조는 one-tier, two-tier에 비해 광범위한 데이터를 읽기 쉽게 만들어 주고, 네트워크를 통한 접근을 쉽게 하고, 데이터를 은닉을 용이하게 한다는 장점이 있다.

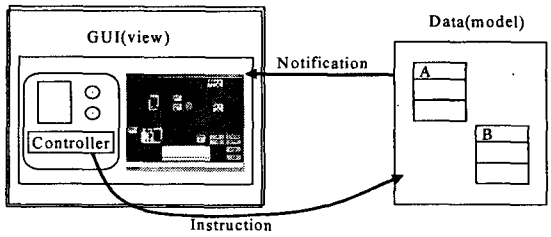


그림 1 MVC(Model/View/Controller) 디자인 패턴

어플리케이션 설계의 기본 구조는 그림1과 같은 MVC(Model/View/Controller) 디자인 패턴을 적용하였다. 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.

자바 언어의 주요 특징은 플랫폼에 독립적이라는 것이다. 그렇지만, 다계층으로 구성된 어플리케이션은 서버 측의 자원에 따라서 다른 API를 사용해야 할 경우도 있으므로 표 1와 같은 개발 자원을 사용하였다.

표 1 클라이언트와 서버의 개발 자원

구분	적용 분야	개발 자원
Server	서버 OS	Windows 2000
	웹서버	인터넷 서비스 관리자
	Database	MS_SQL Server 2000
	언어	JDK 1.3
Client	웹브라우저	Internet Explorer
	언어 및 도구	AppletViewer, JDK 1.1

본 연구는 생성 패턴(Creational Patterns), 구조 패턴(Structural Patterns), 행위 패턴(Behavioral Patterns)로 대표되는 패턴들 중에서, 화투나 트럼프와 같은 카드를 사용하는 네트워크 게임 구현에 있어서, 어떠한 생성 패턴이 적용될 수 있는지를 연구하였고, 주 관심이 되었던 패턴들은 생성 패턴의 종류인 추상 팩토리(Abstract Factory), 빌더(Builder), 팩토리 메소드(Factory Method), 프로토타입(Prototype), 싱글톤(Singleton) 패턴이다. 각각의 패턴들을 일반적으로 다음과 같은 분야에 응용된다.

- Abstract Factory : 구체적인 클래스를 지정하지 않고 관련성을 갖는 객체들의 집합을 생성하거나 서로 독립적인 객체들의 집합을 생성할 수 있는 인터페이스를 제공한다.
- Builder : 복합 객체의 생성 과정과 표현 방법을

분리함으로써 동일한 생성 공정이 서로 다른 표현을 만들 수 있게 한다.

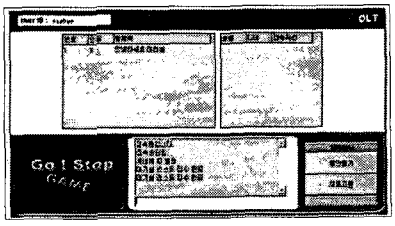
- Factory Method : 객체를 생성하는 인터페이스를 제공하지만, 인스턴스를 만들 클래스의 결정은 서브 클래스가 한다. Factory Method 패턴에서는 클래스의 인스턴스를 만드는 시점을 서브클래스로 미룬다.
- Prototype : 프로토타입의 인스턴스를 이용해서 생성할 객체의 종류를 명세하고 이 프로토타입을 복사해서 새로운 객체를 생성한다.
- Singleton : 클래스의 인스턴스는 오직 하나임을 보장하며 이 인스턴스에 접근할 수 있는 방법을 제공한다.

3. 어플리케이션 설계

카드류를 사용하는 네트워크 게임에서 일반적인 게임은 사용자가 게임 서버에 로그인하고, 대기실에 참여한 후, 게임이 진행되고 있는 방을 찾거나 새로 만들어서 다른 사용자와 같이 게임을 진행한다. 우선적으로 관심의 대상이 되는 것은 MVC디자인 패턴에 따른 각각의 클래스들이 어떻게 재 사용되어 질 수 있는가 하는 것으로 View와 관련된 GUI설계, 프로토콜의 정의와 관련된 Controller의 설계, Data와 관련된 Model의 설계부분을 위주로 연구되었다.

3.1 대기실 GUI설계에서 생성 패턴의 적용

아래의 그림 2는 일반인들에게 잘 알려진 카드를 사용하는 게임들의 대기실 GUI설계의 예를 보여 주고 있다. 두 그림을 비교해 보면, 게임에서의 대기실에서 거의 같은 GUI 디자인 패턴을 가지고 있다는 것을 볼 수 있다.



(a) 고스톱 게임 대기실 GUI 설계

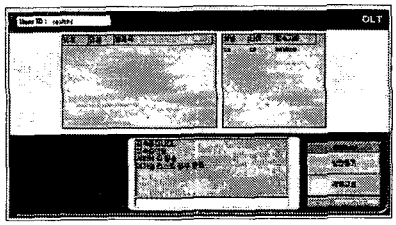


그림 2 (b) 홀라 게임 대기실 GUI 설계

위의 예에서 살펴보면, 객체지향 언어를 사용해서 클래스를 얼마나 효율적으로 설계하느냐에 따라서 충분히 클래스의 재사용과 같은 편의성을 제공해 줄 수 있다는 가능성을 보여 주고 있다.

그림 3의 추상화 팩토리 패턴이 두 개 또는 여러 개의 GUI 디자인에서 연관된 객체의 클래스들 중의 하나를 반환하고자 할 때 사용할 수 있다.

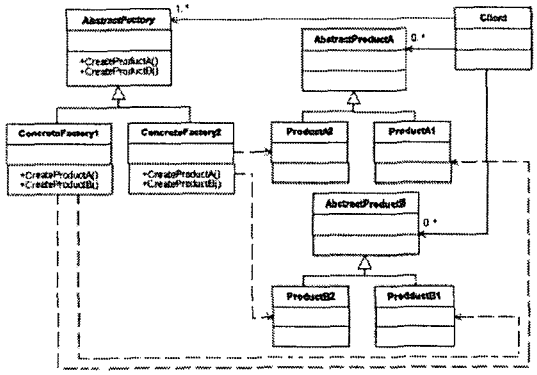


그림 3 추상 팩토리(Abstract Factory) 패턴 UML

추상화 팩토리는 여러 팩토리들 중의 하나를 반환하는 팩토리 객체로 사용되는데, 추상화 팩토리가 적용된 실례로 컴퓨터시스템에서 다중 "룩앤필(look and feel)"을 지원하는 경우가 있다. 게임의 제작에 있어서, 대기실의 GUI도 룩앤필과 같이 해석될 수 있다.

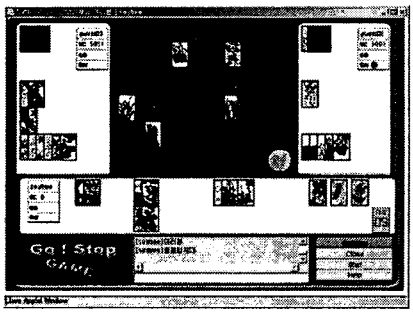
여기서 팩토리는 사용하는 시스템이 고스톱이면 고스톱 대기실과 관련된 GUI, 홀라이면 홀라 대기실과 관련된 GUI를 반환할 수 있음을 볼 수 있다. 즉 두개의 클라이언트 측 GUI는 추상화 팩토리(Abstract Factory)패턴이 아주 적절하게 적용될 수 있음을 볼 수 있다.

추상화 팩토리 패턴을 사용함으로써, 추후 추가될 수 있는 또 다른 어플리케이션을 쉽게 작성할 수 있게 하거나, 여러 개의 어플리케이션을 한 클라이언트

에게 서비스하는 경우 같은 자원을 공유할 수 있으며, 자원의 효율을 극대화 할 수 있을 것이다. 대기실의 GUI를 구성하고 있는 여러 가지 컴포넌트들 중에서 같은 역할을 하는 컴포넌트도 추상화 팩토리 패턴이 적용될 수 있다.

3.2 게임룸 GUI설계에서 생성 패턴의 적용

두 개의 게임 룸의 설계는 앞에서 본 그림 2와 많은 차이가 있지만, 객체 지향적 설계 기법의 기본이 되는 추상화의 개념을 적용해 보면, 여러 가지 같은 클래스의 형태가 있을 수 있다는 것을 발견할 수 있다.



(a) 고스톱 게임룸 GUI 설계

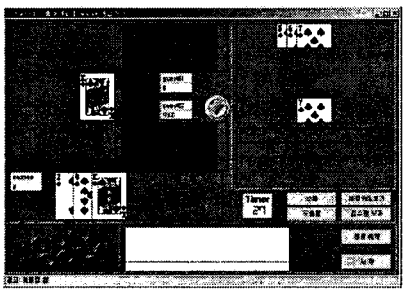


그림 4 (b) 홀라 게임룸 GUI 설계

게임 룸의 설계는 게임에 참여하는 사용자들에게 얼마만큼 재미있는 요소를 제공해 줄 것인가 하는 것이 중요한 사항이다. 그림 4는 각각 다른 게임의 게임 룸 설계의 예를 보여준다.

게임룸의 설계에서도 마찬가지로 그림 3에서 보았던 추상화 팩토리 패턴을 사용할 수 있다. GUI구성하고 있는 각각의 요소를 살펴보면, 여러 가지 생성패턴들이 적용될 수 있는 부분을 살펴볼 수 있다.

게임룸을 구성하는 객체들 중에는 디자인 패턴이 적용될 수 있는 객체들이 많이 있다. 그 중에 화투패나 카드는 그림 5와 같은 빌더(Builder) 패턴을 적용할 수 있다.

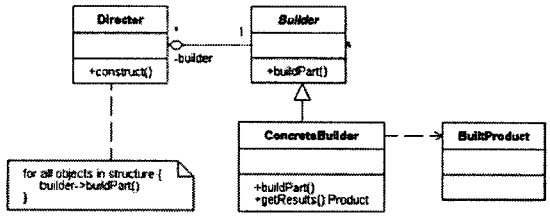


그림 5 빌더(Builder) 패턴 UML

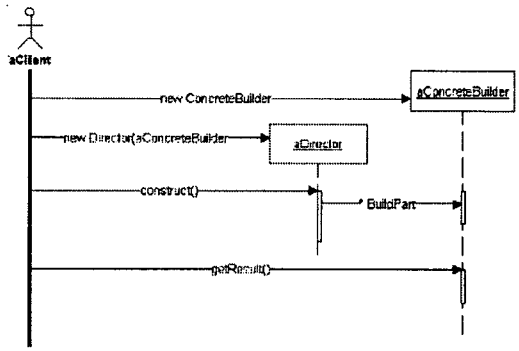


그림 6 빌더(Builder) 패턴 협력도

그림 6은 빌더 패턴에 있어 각 객체들의 협력도를 나타낸다. 빌더 패턴은 여러 개의 객체로 생성될 수 있는 카드에서, 각각을 클래스로 관리할 경우 클래스의 개수가 많아 질 수 있으므로, 복잡한 객체를 생성하는 방법과 표현하는 방법을 정의하는 클래스를 별도로 분리하여 서로 다른 형태의 카드를 생성할 수 있는 동일한 구축 공정을 제공할 수 있다는 장점이 있다.

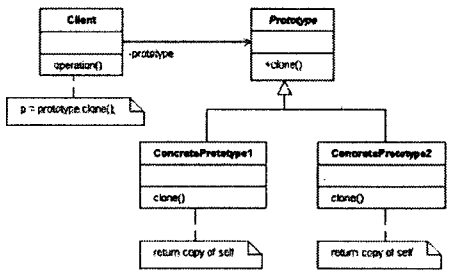


그림 7 프로토타입(Prototype) 패턴 UML

카드 객체의 생성은 그림 7과 같은 프로토타입

(Prototype) 패턴에 의해서 생성할 수 있다. 프로토타입은 추상 팩토리와 빌더 패턴이 가지는 비슷한 결과를 가지지만, 클라이언트에게 어떠한 객체가 생성되어 있는지 감출 수 있기 때문에, 클라이언트가 상대해야 하는 클래스의 수가 적어지게 된다는 장점이 있고, 수정이 없이 애플리케이션에 종속된 클래스들과 동작할 수 있다. 특히 런타임시에 새로운 카드를 삽입하고 삭제할 수 있는 기능의 구현에 아주 유용하고, 이것의 응용에 따라 좀더 재미있는 게임의 구현이 가능하다.

게임룸을 구성하는 객체 중에서, 게임에 참여하는 사용자 클래스의 정의 시에 고려해 볼 수 있는 패턴은 그림 8과 같은 팩토리 메소드 패턴이다.

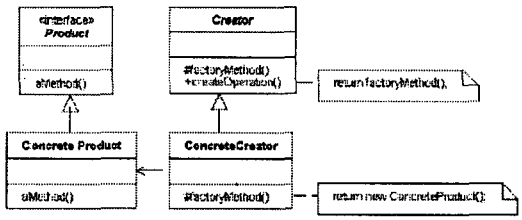


그림 8 팩토리 메소드(Factory Method) 패턴 UML

일반적으로 게임룸에서 3종류의 유저들이 있다. 각각은 게임의 진행을 맡게 되는 유저와 게임에 참여하는 유저, 게임을 관람하는 유저다.

각각의 게임룸은 게임마다 다양한 형태를 가질 수 있다. 다양한 게임룸의 설계를 위해 전체적인 구조는 추상 팩토리 패턴을 사용하고, 구현은 팩토리 메소드 패턴을 이용하여 구현하는 것이 바람직하다고 할 수 있다.

3.3 게임 서버 설계에서 생성 패턴의 적용

네트워크 게임에서 여러 명의 공유 자원을 효율적으로 분배해주고, 게임 개발자 간에 커뮤니티를 형성하게 해준다는 측면에서 중요한 기술이고, 확장과 개선의 여지가 많은 부분이다. 그러므로 특히 객체들 간의 상호의존성을 배제할 수 있는 설계 기법이 중요하다고 볼 수 있다.

본 연구에서 구현된 게임 서버는 그림 9와 그림 10에서 보는 것과 같이 실행 방법과 결과는 거의 같다. 그림 10의 (b)는 카드게임의 상태를 효율적으로 관찰할 수 있는 약간의 GUI가 추가된 게임 서버의 형태를 보여주고 있다.

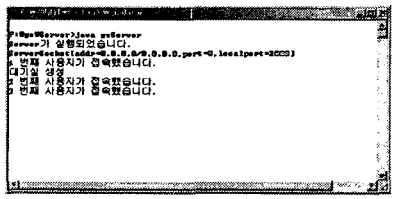
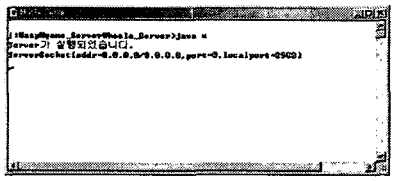
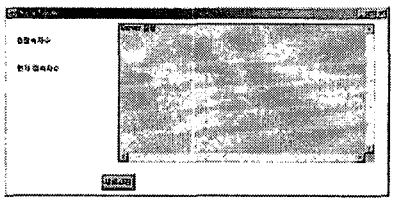


그림 9 고스톱 게임 서버의 동작



(a) 훌라 게임 서버의 동작



(b) 훌라 게임 서버의 GUI 설계

그림 10 훌라 게임 서버

게임 서버의 설계에서도 추상 팩토리 패턴을 적용할 수 있는 부분이 많이 있다. 그리고, 추가적으로 생각해 보아야 할 디자인 패턴은 그림 11과 같은 싱글톤(Singleton) 패턴이다.

만약 다른 형태의 카드 게임을 같이 관리하는 게임 서버의 구성이 필요하다고 가정한다면, 이 게임 서버는 컴퓨터 자원의 상당한 부분을 차지하게 된다.

이러한 이유로 게임을 총괄하는 서버를 구성하는 클래스에서 서버의 인스턴스는 하나일 필요가 있다. 이러한 요구를 충족시키기 위해 적용할 수 있는 패턴은 싱글톤이다.

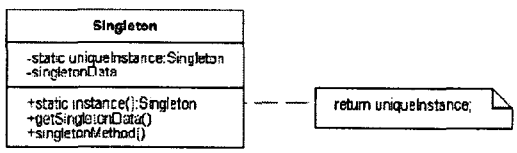


그림 11 싱글톤(Singleton) 디자인 패턴 UML

또한 실시간 RPG게임에서처럼, 서버가 구성해 주는 여러 가지 다양한 형태의 게임 룸을 생성해야 한

다면 앞에서 살펴본 그림 5와 같은 빌더 패턴을 적용할 수 있다.

3.4 프로토콜 구현에서 생성 패턴의 적용

네트워크 게임은 특성상 실시간 자료 전송이 필수적이고, 또 클라이언트와 서버간의 자료 전송이 분명히 이루어져야 하므로, TCP/IP기반 프로토콜을 사용해야 한다. 자바에서 사용할 수 있는 다양한 스트림 모델이 있는데, 본 연구에서는 구현된 게임은 객체를 주고받을 수 있도록 구성된 스트림 모델 중 ObjectInputStream과 ObjectOutputStream을 사용하였다. 클라이언트와 서버간의 실제 통신에서는 Serializable 인터페이스와 협력하는 패킷 클래스의 객체를 주고받는다.

이러한 특성을 감안해 볼 때, 프로토콜의 구현에서는 다양한 객체가 정보를 달리하여 생성되어야 하기 때문에 그림 7과 같은 프로토타입 패턴을 적용할 수 있다.

4. 결론

본 연구에서 카드를 사용하는 두개의 게임에 대한 GUI를 분석한 것을 바탕으로 하여, 디자인 패턴의 적용에 대해 연구하였다. 게임 API 설계의 여러 부분에서 이미 검증된 다양한 디자인 패턴이 적용될 수 있다는 것을 살펴보았다.

본 연구에서는 생략되었지만, 추후의 연구에서는 생성 패턴을 제외한 구조 패턴과 행위 패턴의 적용 부분이 추가되어야 한다. 또한 사용자가 자기 차례를 기다려서 게임하는 턴 게임 방식의 네트워크 게임에 대한 디자인 패턴의 적용 외에도 실시간 스토리를 풀어가면서 진행하는 롤 플레잉 게임 분야의 설계에 있어서 디자인 패턴의 적용을 연구할 예정이다.

본 연구에서 자바를 이용한 네트워크 게임의 설계에서 객체 지향 개념이 어떻게 적용될 있는지에 대한 여러 가지 대안을 제시하고 있지만, 사용자에게 흥미있고 생동감있는 게임을 제공하기 위해서 다음과 같은 부분들을 향후에 연구할 예정이다.

- 마이크로소프트사가 제공하는 DirectX API를 사용하는 게임 설계에서 디자인 패턴 적용에 대한 연구
- .net 플랫폼을 이용한 네트워크 게임의 개발 연구
- DirectX 기반의 기본 컴포넌트 제작에 있어서, 디자인 패턴의 적용에 대한 연구
- 네트워크 게임 제작에 있어서, 생성 패턴, 구조 패턴, 행위 패턴의 적용에 대한 연구
- 네트워크 게임에서 효율적인 설계를 위한 새로운 디자인 패턴에 대한 연구

위에서 제시한 내용 외에도, 게임 API의 설계를 위해서 동시성, 분산 프로그래밍, 실시간 프로그래밍에 관한 패턴 그리고, 게임 어플리케이션 제작에 종속적인 디자인 패턴에 대한 연구가 필요하다.

[참고문헌]

[1] Barry Geipel, University of California, Irvine University Extension, *Design Patterns for Java*, 2001(269 pages)

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

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

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

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

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

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

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

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