Journal of Korea Game Society 2014 Oct; 14(5): 107-116
http://dx.doi.org/10.7583/JKGS.2014.14.5.107

게임 프로그래밍

클라이언트와 서버가 통합된 단일 모듈을 사용한 온라인 게임 개발 방법론 연구

강민석*, 김경식**
호서대대학원 컴퓨터공학과*, 호서대학교 게임학과**
jusin26@naver.com, kskim@hoseo.edu

A Study on Methodology of Online Game Development using Integrated Single Module with Client and Server

Min-Seok Kang*, KyungSik Kim**
Dept. of Computer, Graduate School of Hoseo University*
Dept. of Game Development, Hoseo University**

요 약

온라인 게임 개발에 있어 전통적인 개발 방법론은 서버, 클라이언트를 별도의 모듈로 개발하는 것이다. 하지만 이 방법은 네트워크 프로토콜 정의, 기획 데이터 관리 등 중복된 모듈이 많아 개발 비용과 유지보수 비용이 증가한다. 이에 서버, 클라이언트 개발 시 각 영역별로 통합된 단일 모듈을 기반으로 개발하는 방법론을 제시하였다. 제시된 방법론을 온라인 게임 "충무공 해상대전"에 적용하여 그 효율성을 보였다. 적용된 사례에서는 서버와 클라이언트의 모듈 통합으로 프로젝트 규모를 15.1% 줄일 수 있었다.

ABSTRACT

The traditional methodology of online game development is separating server development module and cilent development module. However developing and maintaining expenses have been increased a lot due to duplicated modules such as definitions of network protocols and managements of planning data. In this paper, we suggest an advanced methodology of online game development based on the integrated single module with client and server. Its effectiveness was shown by applying the proposed methodology in the development of the online game 'Chungmukong's Battle on the Sea'. In this case, the project size was reduced by 15.1% by using the integrated single module with client and server.

Keywords : Online Game(온라인 게임), Integrated Module(통합 모듈), Unit Test(단위 테스트), Bot(봇), Methodology(방법론)

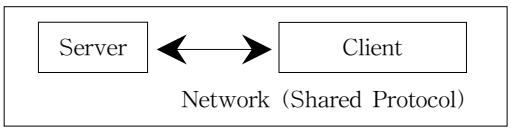
Received: Sept. 12, 2014 Accepted: Oct. 13, 2014
Corresponding Author: KyungSik Kim(Hoseo University)
E-mail: kskim@hoseo.edu

ISSN: 1598-4540 / eISSN: 2287-8211

1. 서 론

1.1 연구 배경

나날이 발전하는 한국 게임 산업에서 가장 중요한 분야는 2012년 전체 매출의 69.7%를 차지한 온라인게임 분야이다[1]. 이러한 온라인 게임 개발에 있어 전통적인 개발 방법론은 서버, 클라이언트를 별도의 모듈로 개발하는 것이다[2]. 서버는 많은 수의 유저들을 관리하는 역할을 주로하게 되고 클라이언트는 유저의 입력을 받아서 서버로 전달하며 서버로부터 전달받은 결과를 화면에 표현하는 역할을 주로하게 된다. 서로 주어진 주요 역할에 따라 그에 맞게 별도로 개발되는 것이다([Fig. 1] 참고).



[Fig. 1] Traditional Division of Online Game Development

그러나 서버, 클라이언트를 별도로 개발할 경우 비슷한 기능을 하는 모듈을 중복 개발하게 되는 경우가 많다. 또한 안정성을 위해 각종 테스트를 할 수 있는 봇(bot)을 개발하게 되는데 봇은 서버, 클라이언트와 비슷한 모듈을 포함하게 된다.

이러한 중복 모듈 개발은 개발 비용이 불필요하게 투자되고 버그를 유발할 가능성이 높아 유지보수 비용이 커지게 되는 문제를 초래한다[3]. 이는 주로 서버와 클라이언트의 개발자가 서로 다른 사람인 경우가 많은 것에 기인한다. 서버의 경우 네트워크 엔진에 기반하고 클라이언트의 경우 렌더링 엔진에 기반하는 것도 원인 중 하나이다[4]. 온라인게임의 특징을 고려하지 않고 사용하는 게임 엔진에 종속적으로 서버와 클라이언트를 개별적으로 개발하게 되는 것이다[5].

게임의 안정성을 더욱 증가시키기 위해 단위 테스트[6]를 도입할 경우 중복 모듈 개발은 단위 테스트 모듈의 중복으로 이어져 문제를 더욱 복잡하게 만든다[7].

따라서 온라인 게임의 서버, 클라이언트, 봇을 개발할 때는 체계적으로 통합된 단일 모듈을 기반으로 개발할 것이 요구된다.

1.2 모듈 간 통합의 필요성

소프트웨어 개발에 있어 프로젝트의 규모가 개발비용을 높이는 것은 일반적으로 알려진 사실이다[8]. 프로젝트 규모가 커지면 오류의 수가 증가하고 생산성이 떨어진다. 따라서 온라인 게임의 경우도 마찬가지로 가능한 프로젝트의 규모를 줄여야 한다. 프로젝트의 규모는 코드 라인수로 판별할 수 있다. 서버, 클라이언트, 봇의 모듈 통합을 통해 중복 코드를 제거하여 프로젝트의 규모를 줄일 수 있다.

또한 통합된 단일모듈에 단위테스트를 개발할 경우 테스트 코드의 중복도 제거되어 개발비용을 줄일 수 있게 된다. 줄어든 테스트코드의 양은 게임 콘텐츠 수정 이후에 발생하는 유지보수 비용도 줄여준다.

1.3 연구 목적

본 연구의 목적은 클라이언트와 서버 모듈이 통합된 단일 모듈과 테스트 모듈을 사용하여 온라인 게임을 개발하는 방법론을 제시하는 것이다. 통합된 모듈을 통해 개발 비용을 절약하고 테스트 모듈을 통해 안정성을 확보하여 온라인게임 개발의 효율성을 높일 수 있다.

첫 번째로 온라인 게임에 사용되는 서버, 클라이언트, 봇을 개발할 때 중복 모듈 없이 통합된 모듈을 기반으로 하는 개발 방법론을 제시한다. 두 번째로 제시된 방법에 테스트 모듈을 추가한 개발 방법론을 제시한다. 세 번째로 사례를 통해 제시된 방법론의 효과를 검증한다.

1.4 관련 연구

게임을 개발할 때 주로 논의되는 방법론은 프로

젝트팀을 운영하는 것과 관련된 방법론이거나 소프트웨어 개발방법론이다.

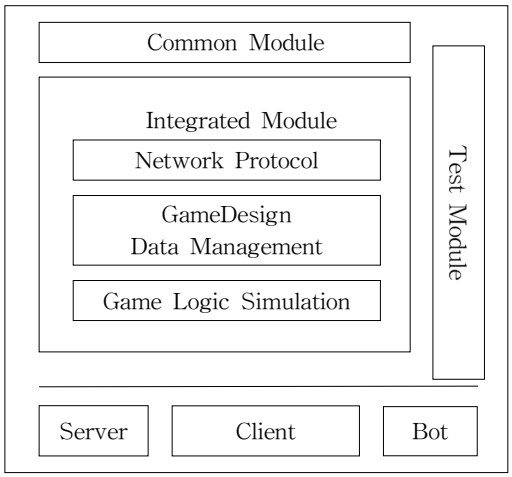
한국게임산업개발원에서 진행한 연구[2]에 따르면 주로 모바일 게임과 같이 짧은 개발 기간을 요구하는 경우에는 폭포수 모델과 XP 위주로 적용을 하고 있으며, 온라인 게임의 경우 프로토타입 모델과 나선형 모델, 단계적 출시 모델을 적용하고 있는 것으로 조사되었다. 조사된 개발방법론에서는 게임 개발 시 서버 프로그램을 설계부터 구현까지 모두 독립적인 영역으로 규정하고 있다. 클라이언트의 경우는 콘텐츠 개발로 규정하고 콘텐츠 개발 프로그래머를 별도로 두었다. 산출물로 별도의 서버 설계도, 서버 구현 플로우차트, 콘텐츠 구조도 등을 규정하고 있다.

방법론에 대한 다른 방향의 연구는 기반이 되는 엔진을 선택하는 방법들과 연관되어 있다. 엔진 선택이 방법론과 연관된 이유는 엔진의 제약사항에 따라 개발 방법론이 달라질 수 있기 때문이다.

엔진 선택은 주로 개발사의 규모와 어떤 장르의 게임을 개발하느냐에 따라 달라진다[4,5]. 개발사에서 상용개발 엔진을 사용할 경우 엔진의 특징에 개발 방법이 종속되게 된다. 예를 들어 네트워크 엔진이 서버 측 로직에 관여하는 방식이라면 클라이언트에서는 같은 모듈을 사용할 수 없으므로 서버와 클라이언트의 통합이 불가능해지는 것이다. 반대로 3D 렌더링 엔진이 클라이언트의 설계를 제한하게 된다면 서버에서는 같은 모듈을 사용할 수 없을 수도 있다.

2. 본 론

온라인게임에서 사용되는 서버와 클라이언트, 봇을 세 단계로 통합 할 수 있다. 세 단계는 각각 공용 기반 모듈 통합과 서버, 클라이언트, 봇의 모듈 통합, 테스트 모듈 통합을 말한다([Fig. 2] 참고).



[Fig. 2] Hierarchy after Integrated Module

2.1 공용 기반 모듈 통합

온라인게임이 소프트웨어로서 일반적으로 가져야할 기능들의 집합이다. 디버깅이나 상태 파악을 위해 로그를 출력하는 모듈, 게임 내 네트워크 송수신을 위한 모듈, 데이터 로딩을 위한 파일 입출력 모듈 등이 포함된다.

2.2 서버, 클라이언트, 봇 통합

서버, 클라이언트 간의 통합은 크게 세 가지 영역으로 나뉠 수 있다. 네트워크 프로토콜 영역, 기획 데이터 관리 영역, 게임 로직 시뮬레이션 영역으로 나눌 수 있다.

2.2.1 네트워크 프로토콜 영역

네트워크 프로토콜 영역은 서버, 클라이언트 간의 교신을 위한 패킷 정의 집합을 말한다. 프로토콜은 주로 두 가지 방식으로 정의되는데 정의 정보를 통해 규약을 공유하는 방식[Table 1]과 모듈을 직접 공유 하는 방식[Fig. 3]으로 나뉜다.

[Table 1] Example of Shared Definition-Information

Name	Size (Byte)	Description
Packet Size	2	Packet Total Size (Byte)
Packet Type	1	Packet의 Type - Login Request = 1
UserUID	4	Connected User's Unique Identity Key
...	...	...

```
enum EPacketType { LoginReq = 1, };
struct SLoginReq
{
        WORD size;
        BYTE type;
        DWORD userUID;
        ...
};
```

[Fig. 3] Example of Shared Module Directly

두 가지 방식 중 코드를 직접 공유하는 모듈 공유 방식으로 통합 하여야 한다. 정의 정보 공유 방식은 서버, 클라이언트, 봇에 각각의 코드를 개발하게 되어 중복 코드가 생성되기 때문이다.

2.2.2 기획 데이터 관리 영역

기획 데이터 관리 영역은 데이터 드리븐 방식에 따라 프로그램 외부에서 정의된 데이터들을 말한다. 주로 xml 이나 jason, lua table 같은 표준 문서 양식을 띈다[9]. xml 로 구성된 기획 데이터는 [Fig. 4]와 같다.

```
ex. Item.xml
<Item ID="1" Name="Item_1" Price="100" />
<Item ID="2" Name="Item_2" Price="100" />
<Item ID="3" Name="Item_3" Price="100" />
```

[Fig. 4] Example of Designer's xml data file

기획 데이터를 파일로부터 입력받고 필요한 데이터를 질의(Query)에 따라 반환해 줄 수 있어야 한다. 이 때 서버와 클라이언트가 같은 기획데이터 파일로부터 같은 모듈로 해당 기능을 수행하여야 한다.

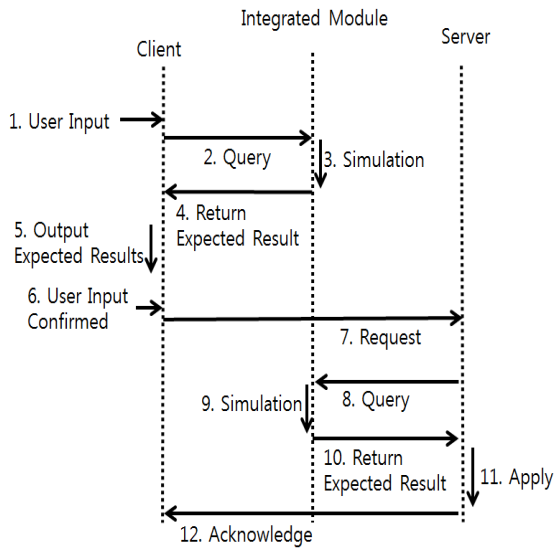
2.2.3 게임 로직 시뮬레이션 영역

게임 로직 시뮬레이션 영역은 입력받은 기획 데이터를 사용하여 실제 게임을 동작시키는 영역이다. 서버와 클라이언트는 같은 동작을 각자 다른 방식으로 활용하게 된다. 서버는 주로 클라이언트의 입력에 따라 동작 후 결과를 클라이언트로 전송하게 된다. 클라이언트는 유저에게 예상 결과를 안내하거나 빠른 반응을 위해 서버로부터 결과가 오기 전에 미리 보여주는 용도로 활용한다.

2.3 통합 모듈을 기반으로 서버, 클라이언트 개별 구현

서버는 여러 유저를 관리하고 클라이언트로부터 전달받은 요청을 처리한다. 아이템 거래, 파티플레이 등 여러 유저들 간에 상호작용이 필요한 기능을 담당하며 보안 문제로 특별히 서버에만 존재하여야 하는 기능도 담당한다.

클라이언트는 유저의 입력에 반응하고 필요한 요청을 서버로 전달하며 화면에 정보를 출력하는 것을 담당한다. 유저가이드 등 서버와 연관이 필요 없는 모듈도 담당한다([Fig. 5] 참고).



[Fig. 5] Data Flow

2.4 봇, 단위테스트 통합

봇은 서버를 테스트하기 위한 전용 프로그램을 말한다. 일반적으로 서버의 기능테스트와 성능테스트를 위해 개발하게 된다. 이 중 기능테스트를 위한 모듈을 단위테스트와 통합할 수 있다. 단위테스트 중 네트워크 명령을 직접 실행하거나 가상으로 시뮬레이션 하는 테스트들을 모아 봇에서 기능테스트로 사용한다. 또 기능테스트 모듈을 동시에 반복적으로 실행하는 방법으로 성능테스트를 진행할 수 있다.

[Fig. 6]은 단위테스트 모듈과 봇-기능테스트를 통합하는 것에 관한 예시이다. 봇-성능테스트도 같은 모듈을 통해 진행하는 것을 볼 수 있다.

```
// UnitTest
bool Test_BuyItem(User user)
{
    Item item = ItemManager.Random()
    if( item.Price > user.CurrentMoney )
    {
        //TestResult: Failed
        bool bResult = BuyItem(user, item)
        if( bResult == true )
            return false
    }
    else
    {
        //TestResult: Succeed
        bool bResult = BuyItem(user, item)
        if( bResult == false )
            return false
    }

    return true
}
```

```
// Bot-Function Test
User user = new BotUser
user.ConnectToServer()
if( Test_BuyItem(user) == false )
    Error
...
// Bot-Performance Test
for( count < n )
{
    User user = GetBotUser(n)
    Test_BuyItem(user)
}
```

[Fig. 6] Example of Integrated Bot-UnitTest

2.5 설계 단계에서의 통합

이러한 통합은 설계 단계에서 미리 고려되거나 초기 개발 단계에서부터 점진적으로 통합되어야 효과가 크다. 이미 별도로 개발된 모듈을 통합하는 경우에는 유지보수 비용은 줄일 수 있으나 개발비용은 줄일 수 없기 때문이다. 통합된 모듈들의 핵심 역할을 정리하자면 [Table 2]와 같다.

[Table 2] Integrated Modules Summary

Part Name	Summary
Common Module	A Set of Software features ex. Output Log, Networking, File IO, ...
Network Protocol	A Set of Packet Definition
GameDesign Data Management	Input/Output GameDesign Data And Query
Game Logic Simulation	Action Game using the GameDesign Data
UnitTest Bot-Function Test	Perform a functional test of the unit tests gathered from the bot goes through the network test
UnitTest Bot-Performance Test	Measuring the performance of the server as a single module to perform multiple times repeatedly, a functional test

3. 사 례

통합된 모듈을 사용하여 개발하게 될 경우의 효과성을 검증하고자 한다. 이는 통합되어 제거된 코드의 양으로 판단할 수 있다.

3.1 프로젝트 소개

사례로 소개할 “충무공 해상대전”은 아산시의 지원 하에 개발된 MO-RTS(Multiplayer Online Real Time Strategy) 게임이다[10,11]. 2008.9 - 2009.2까지 개발 되었고 2009.4월말 “아산시 성웅 이순신 축제”에서 게임대회를 가졌다. 이후 매년 기능 업그레이드를 진행하였고 대회를 개최하였다.

게임의 구성요소는 유닛(배), 빌딩(건물), 자원으로 나뉘며 각 플레이어는 자원을 수집하여 빌딩을 건설하고 유닛을 생산한 후 생산된 유닛으로 다른 플레이어와 전투를 하게 된다.

본 프로젝트의 경우 봇이 별도로 제작되어 있지 않아 서버와 클라이언트의 통합 모듈 적용에 대해서만 효과를 검증하였다.

[Fig. 7] Chungmukong's Battle on the Sea

3.2 프로젝트에 적용

공용 기반 모듈을 먼저 통합한다. 공용 기반 모듈에는 디버그를 위한 로그 출력, 유한상태기계(Finite State Machine), 메모리풀, 파일 입출력, 시간 관리자 등이 포함되어 있다.

네트워크 프로토콜 영역에는 로그인, 로그아웃, 방생성, 삭제, 빌딩건설, 유닛생산 등의 구조체가

포함되고 기획 데이터 관리 영역에는 자원데이터 관리자, 빌딩데이터 관리자, 유닛데이터 관리자가 포함된다. 게임 로직 시뮬레이션 영역에는 RTS 게임의 동기화를 위한 게임턴 관리자[12]와 각 플레이어 별로 생성한 빌딩, 유닛을 관리하기 위한 관리자 등이 포함되어 있다.

[Fig. 8]은 기획 데이터 관리 영역에서 유닛데이터 관리자의 중복을 보여주는 예시이다.

[CreateUnit Logic - Client Side]
0. Prepare: Load UnitData
1. Request: CreateUnit (Click Button)
2. Check: Enough resource
3. Action: Send Packet to Server

[CreateUnit Logic - Server Side]
0. Prepare: Load UnitData
1. Request: CreateUnit (Recv Packet)
2. Check: Enough resource
3. Action: Broadcast to Client

[Fig. 8] Example of Duplication-Logic

3.3 통합 효과 측정

서버와 클라이언트의 모듈 통합 후 Microsoft Visual Studio 2005(Visual C++)에서 코드라인 수를 측정하였다. 서버의 경우 네트워크 엔진과 관련된 코드는 제외하였고 클라이언트의 경우 3D 렌더링 엔진과 관련된 코드는 제외하고 측정하였다.

[Table 3] Server Code Measurements after Integration

Part Name	Code Line Count	Percent
Common Module	**1,060**	**17.8**
Network Protocol	**328**	**5.5**
GameDesign Data Management	**487**	**8.2**
Game Logic Simulation	**1,743**	**29.4**
Game Server	2,307	38.9
Total	5,925	100

[Table 4] Client Code Measurements after Integration

Part Name	Code Line Count	Percent
Common Module	**1,060**	**5.8**
Non-integrated Common Module	2,360	13
Network Protocol	**328**	**1.8**
GameDesign Data Management	**487**	**2.7**
Game Logic Simulation	**1,743**	**9.6**
Non-integrated Game Logic	5,367	29.7
Game Client	6,679	37
Total	18,024	100

공용 기반 영역에서 1,060 Line을 통합할 수 있었고 네트워크 프로토콜 영역에서 328Line, 기획 데이터 관리 영역에서 487Line을 통합할 수 있었다. 게임 로직 시뮬레이션 영역에서는 1,743Line을 통합하여 총 3,618Line을 줄일 수 있었다. 따라서 서버와 클라이언트가 통합된 사례에서는 전체 23,949Line 중 15.1%의 코드가 줄어드는 효과가 생겼다. 이는 중복개발을 없애 개발비용이 줄어들고 프로젝트의 규모가 작아져 유지보수 비용이 적어졌다는 것을 의미한다.

서버는 전체 코드의 61.1%가 통합된 모듈을 사용하게 되었고 클라이언트는 전체 코드의 19.9%가 통합된 모듈을 사용하게 되었다. 이는 두 프로그램 간에 중복코드로 인해 발생할 수 있는 유지보수 비용이 감소되었다는 것을 뜻한다.

통합 후 성능측정에서 속도는 동일하게 유지되었지만 메모리 사용량은 조금 증가하였다. 속도부분에서는 속도의 관건인 게임 로직 시뮬레이션 영역에서 서버와 클라이언트가 통합 전에도 동일한 알고리즘을 사용하였기 때문에 차이가 생기지 않았다. 메모리 부분에서는 기획 데이터 관리 영역에서 각각 서로에게 필요 없는 데이터를 메모리에 저장하고 있는 경우가 발생하여 사용량이 증가하였다.

대표적인 예는 클라이언트만이 사용하는 데이터를 서버에서도 메모리에 저장하고 있는 경우였다. 하지만 기획 데이터 관리 영역의 메모리 사용량이 작기 때문에 차이가 유의미하진 않다.

결과적으로 설계 단계에서부터 미리 통합되어 개발 되었다면 전체 개발 비용과 유지보수 비용의 15.1% 가 절약될 수 있다는 것을 말한다. 따라서 온라인 게임을 개발할 때 통합된 단일 모듈을 기반으로 개발하는 것이 독립적으로 개발할 때보다 더 효율적이다.

4. 결 론

본 연구에서는 온라인 게임에 사용되는 서버, 클라이언트, 봇을 개발 시 통합 모듈을 기반으로 개발하는 방법론을 제시하였다.

계층화된 형태의 공용 기반 모듈, 네트워크 프로토콜, 기획 데이터 관리 모듈, 게임 로직 시뮬레이션 모듈, 봇-기능테스트 모듈을 서버, 클라이언트, 봇이 공동으로 사용하도록 개발하는 방법이다. 제시된 방법론을 적용한 실제 사례를 통해 효과성을 입증하였다. 통합된 모듈을 사용하여 개발할 경우 더 적은 개발 비용과 유지 보유 비용을 가져 개발 효율성을 높일 수 있었다. 또한 통합된 모듈을 바탕으로 테스트 모듈도 통합하여 적은 테스트로도 안정성을 높일 수 있었다.

본 연구를 통해 그 동안 온라인 게임을 개발함에 있어 개발자들이 감각적으로 설계하던 부분을 체계화할 수 있을 것으로 기대된다. 특히 개발기간이 길고 유지보수 기간이 길수록 효과가 더욱 커질 것으로 기대된다.

향후 온라인 게임의 보안에도 활용하는 방법을 연구할 수 있을 것으로 기대된다. 온라인 게임 보안을 위해서는 클라이언트에서 독립적으로 처리하던 부분을 줄이고 서버에서 처리하는 것이 좋다. 이 때 서버와 클라이언트의 모듈이 중복되는 경우가 많이 발생하는데 앞으로 연구가 필요할 것이다.

REFERENCES

[1] 2013 White Paper on Korean Games, Guide to Korean Games Industry and Culture, Korea Content Agency, 2013. 10.

[2] Survey Methodology of Game Development, Korea Content Agency, 2005. 04.

[3] Robert C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship, Pearson Education, 2009.

[4] Myoun-Jae Lee, Kyoung-Nam Kim, “A Study on the technical consideration in initial game software planning of online game”, Journal of Korea Game Society, Vol. 3, No 2, pp.3-11, 2003. 09.

[5] Yong-Ho Chang, Won-Jo Joung, “An Impact of Technological Regime of Game Engine upon Game Development Performance”, Journal of Korea Game Society, Vol. 10, No 1, pp.79-92, 2010. 02.

[6] Kent Beck, Test-Driven Development: By Example, Addison-Wesley Longman, 2002.

[7] Gerard Meszaros, xUnit Test Pattern, Addison Wesley, 2010.,

[8] Steve McConnell, Code Complete 2nd Edition, Microsoft, 2004.

[9] Sang-Wan Lee, Hye-Young Kim, “An Efficient Message Management Scheme in Game Engine”, Journal of Korea Game Society, Vol. 8, No 2, pp.77-84, 2008. 05.

[10] http://game.asan.go.kr

[11] Min-Seok Kang, WooSeock Lee, KyungSik Kim, SeongSeock Oh, NaeHyun Lee, JungJun Lee, “Development of the Online Game ‘Chungmukong's Battle on the Sea’ (For the Purpose of Education of History)”, 2009 Autumn Conference of Korea Game Society, pp.241-247, 2009.

[12] Sam Kweon Oh, Min Seok Kang, “Performance Evaluation of Synchronization Algorithms for Multi-play Real-time Strategy Simulation Games”, Hoseo The Journal of Research Institute for Engineering & Technology, Vol. 32, No 1, 2013. 06.

강 민 석 (Kang, Min Seok)

2005년 호서대학교 게임공학과 (학사)
2008년 호서첨단정보기술대학원 게임애니메이션과(석사)
2008년-현재 호서대학교 컴퓨터공학과 (박사과정)
2009년-현재 KOG

관심분야 : 온라인 게임, 서버/DB 플밍

김 경 식 (Kim, Kyung Sik)

1982년 서울대학교 전산기공학과 (학사)
1984년 서울대학교 전산기공학과 (석사)
1990년 서울대학교 컴퓨터공학과 (박사)
1984년-1991년 한국전자통신연구원 선임연구원
1991년-현재 호서대학교 게임학과 교수

관심분야 : 기능성 게임, 게임 교육, 서버/DB 플밍