Hanbit eBook

Realtime 74

CGSF를 활용한 게임 서버 제작

C++로 온라인 게임 서버 구축하기

박주항 지음

한빛미디어 Hanbit Media, Inc.

CGSF를 활용한
게임 서버 제작

C++로 온라인 게임 서버 구축하기

CGSF를 활용한 게임 서버 제작 C++로 온라인 게임 서버 구축하기

초판발행 2014년 07월 10일

지은이 박주항 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 6월 24일 제10-1779호
ISBN 978-89-6848-670-8 15000 / **정가** 11,000원

책임편집 배용석 / **기획·편집** 정지연
디자인 표지 여동일, 내지 스튜디오 [밈], 조판 최송실
영업 김형진, 김진불, 조유미 / **마케팅** 박상용, 서은옥, 김옥현

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
한빛미디어 홈페이지 www.hanbit.co.kr / **이메일** ask@hanbit.co.kr

Published by HANBIT Media, Inc. Printed in Korea

지금 하지 않으면 할 수 없는 일이 있습니다.
책으로 펴내고 싶은 아이디어나 원고를 메일(ebookwriter@hanbit.co.kr)로 보내주세요.
한빛미디어(주)는 여러분의 소중한 경험과 지식을 기다리고 있습니다.

저자 소개

지은이_박주항

2006년 클라이언트 프로그래머로 게임 회사에 입사하여 회사 사정 때문에 서버 쪽 업무를 맡게 되면서 여러 게임의 온라인 플랫폼을 구축하고 제작해온 개발자다. 프로그래밍 언어로 C++를 주 언어로 사용했지만, 최근에 모바일 플랫폼 관련 회사에서 일하면서 자바 언어를 다루게 되었고, C++ 언어와는 다른 자바만의 매력에 빠져 자바를 보조 언어로 사용하고 있다. 프로그래밍 자체를 좋아하여 운영체제 개발부터 파이썬, 루아 같은 스크립트 언어 활용까지 프로그래밍의 모든 영역에 관심을 두고 있다. 유용한 오픈 소스를 자신의 프로젝트에 활용하는 것을 좋아하여 시간이 날 때마다 여러 오픈 소스 공유 사이트에서 소스 코드를 내려받아 분석하는 것을 취미로 삼고 있다. 또한, 어드벤처 게임을 광적으로 좋아해서 '로라 보우 2 - 태양신의 단도', '스페이스 퀘스트 4' 등 시에라Sierra사 게임의 한글 패치를 제작하기도 했다. 현재 프리랜서로 일하는 중이다.

홈페이지: http://pdpdds.x90x.net/xe

저자 서문

다양한 장르의 온라인 게임이 출시되고 있고 같은 장르라도 그 게임만의 색깔이 나타나긴 하지만, 게임 접속 후 플레이까지의 일련의 과정은 모든 게임이 유사하다는 걸 알 수 있다. 그 이유는 온라인 게임에서 게임 서버와의 연동은 필수고 게임 서버가 구축한 시스템을 거쳐야 하기 때문이다.

특히 우리가 캐주얼 게임이라고 부르는 FPS 게임이나 스포츠 게임 등은 개발하는 과정이 매우 유사하다. 따라서 서버를 구현할 때 공통적인 부분을 미리 모듈화해 둔다면 새 프로젝트를 진행할 때 게임 개발 일정을 대폭 줄일 수 있다. 이런 생각에서 제작된 서버 라이브러리가 CGSF다.

CGSF의 탄생 배경에는 개인적인 이유도 있다. 대학 시절 물리학을 전공하다 보니 근본 원리를 굉장히 중요시하였다. 예를 들어, 전자의 에너지 준위를 생각해보자. 굳이 대학 물리를 언급하지 않더라도 전자의 에너지 준위가 변할 때 불연속적으로 증감이 이루어진다는 사실을 알고 있을 것이다. 또한, 영화도 화면이 연속적으로 움직이는 것처럼 보이지만, 1초에 24~60프레임의 정지 화면을 보여줘서 눈이 착각하는 것에 불과하다. 이렇듯 세상은 연속적으로 보이는 것 같지만, 실제로는 이산적인(discrete) 세상이다. 이런 근본 원리는 시간이 지나도 불변의 원칙이다.

그런데 게임 서버의 개발에서는 그런 근간이 존재하지 않는다. 매번 서버를 백지 상태에서 제작하는 것을 보고 그때마다 "왜 똑같은 내용을 반복하는 거지?"라는 의문이 생겼으며 반복되는 내용을 모듈화하면 코드의 재사용성이 높아질 거라는 생각이 들었다. 이 생각이 그렇게 틀리지는 않다고 본다. 지금까지 수많은 게임 서버가 제작되었지만, 대부분 기억 속에서 잊히고 다시는 재사용되지 못했다. 즉, 이런

서버는 개발 당시를 위해서 제작된 서버일 뿐이다. 물론 게임 서버에서 공통으로 쓰는 부분을 모듈화해서 생명력을 높이는 것은 꽤 어려운 작업이라 생각한다. 게임마다 요구사항이 다르고 어떻게 보면 새로 제작하는 것이 더 효율적일 수도 있다. 그래도 어떻게든 통합형 서버를 제작하고 싶었다. 이런 취지에서 탄생한 것이 CGSF다.

CGSF를 제작한 또 다른 목적은 여러 프로그래머와 교류를 하기 위해서다. 프로그래밍 실력은 스스로 노력해서 얻을 수도 있지만, 다른 프로그래머의 의견이나 조언을 통해서 더 향상될 수 있다. 이 책을 읽고 CGSF의 개발에 참여하고 싶은 분이 있다면 언제라도 환영한다.

끝으로 이 책이 나올 수 있게 많은 도움을 주신 최흥배 님께 감사드리며 CGSF의 제작에 큰 도움이 된 온라인 게임 서버 커뮤니티의 회원분들께도 감사의 마음을 전한다. 또한, 서버 개발의 기초를 알려주신 서정욱 님, 같은 꿈을 향해 달려가는 경빈 님, 일본에서 많은 도움을 주셨던 최영호 님, 호리가미 님, 그리고 제 부족한 필력에 생명을 불어넣어 주신 한빛미디어의 정지연 님께 감사의 마음을 전하고 싶다.

그리고 내가 하고 싶은 대로 살아도 반대하지 않고 항상 지지해 주신 부모님이 오랫동안 건강하시길 바라며 누나, 형들 모두 앞으로의 일이 잘되길 바란다고 전하고 싶다.

이 책을 읽기 전에

온라인 게임 서버 프레임워크의 필요성

다양한 장르의 온라인 게임이 출시되고 있고 같은 장르라도 그 게임만의 색깔이 나타나긴 하지만, 게임 접속 후 플레이하기까지의 일련의 과정은 모든 게임이 유사하다는 걸 알 수 있습니다. 그 이유는 온라인 게임에서 게임 서버와의 연동은 필수고 게임 서버가 구축한 시스템을 거쳐야 하기 때문입니다.

특히 우리가 캐주얼 게임이라고 부르는 FPS 게임이나 스포츠 게임 등은 개발하는 과정이 매우 유사합니다. 따라서 서버를 구현할 때 공통적인 부분을 미리 모듈화해 둔다면 새 프로젝트를 진행할 때 게임 개발 일정을 대폭 줄일 수 있습니다. 하지만 현실은 그렇지 않습니다. 새 프로젝트를 진행하면 서버도 새로 만들고 같은 회사 내 다른 부서에서도 기술 공유를 하지 않아서 매번 반복적인 작업을 수행하는 경우가 대부분입니다.

이 책에서는 게임 서버, 특히 캐주얼 게임 서버에서 나타나는 공통적인 부분을 모듈화하여 새 프로젝트를 진행할 때 빠르게 개발하는 것에 역점을 두었습니다. 또한, 온라인 게임 서버 프로그래밍에 필요한 기본적인 개념을 익히고 여러 라이브러리의 사용법을 배우며 간단한 게임을 제작하여 온라인 게임 서버 시스템을 구축할 수 있도록 합니다. 이 책에서 소개할 CGSF[Casual Game Server Framework]는 오픈 소스를 최대한 활용한 서버 라이브러리입니다. 최종 사용자[End-user]가 사용하기 쉽게 설계하는 것을 목표로 하며 아울러 온라인 게임 서버의 구조를 파악하기 쉽게 만들어서 입맛에 맞게 커스터마이징할 수 있도록 구현하는 것을 최종 목표로 합니다.

컴퓨터 과학 VS 소프트웨어 공학

게임 프로그래머는 컴퓨터학과 또는 전산학과 출신이 많습니다. 그런데 이 프로그래머들이 공통으로 경험하는 난감한 문제가 있습니다. 그것은 게임 프로그래밍이 대부분 컴퓨터 과학의 영역이 아니라 소프트웨어 공학의 영역이라는 것입니다. 예를 들어, 어떤 프로그램에서 Quick Sort가 필요할 때 컴퓨터 과학에서는 Quick Sort를 직접 구현하지만, 소프트웨어 공학에서는 이미 구현된 Quick Sort를 사용하려고 합니다. 또 다른 예로 사운드 관련 라이브러리가 필요하면 컴퓨터 과학에서는 사운드 라이브러리를 직접 개발해서 MP3의 헤더 정보를 파싱하는 등의 처리를 직접 구현하지만, 소프트웨어 공학에서는 그냥 기존의 사운드 라이브러리(Miles Sound System, Bass, 또는 FMOD)를 가져다 씁니다.

이렇게 두 영역은 극명하게 차이가 납니다. 그렇다면 소프트웨어 공학에 가까운 게임 프로그래밍의 이점에는 어떤 것이 존재할까요? 제 생각으로는 창의력의 발현, 전체적인 그림을 그릴 수 있는 능력의 획득이라고 봅니다.

자기 생각을 프로그래밍으로 구현할 수 있다는 것은 정말 매력적인 부분이라고 생각합니다. 그래서 과거에는 없던 정말로 옛날 사람들이 보면 이상하게 생각할 기현상, 즉 온종일 컴퓨터 모니터에서 눈을 떼지 않는 패턴에 수긍하고 몰입할 수 있는 것입니다. 또한, 전체적인 그림을 그리는 것, 게임을 만들기 위해 렌더링 엔진의 준비, 사운드 라이브러리의 선택, 재질 효과를 위한 쉐이더 코드 작성, 길 찾기 알고리즘, 물리 엔진의 적용, 해킹을 막기 위한 보안 라이브러리의 적용 등 딴 분야에서는 절대 맛볼 수 없는 여러 영역을 게임 프로그래머는 맛볼 수 있습니다. 물론 특정 영역의 깊이는 얕을 수 있겠지만, 아니 그보다 그 깊이를 파야 할 필요성이 굳이 느

껴지진 않겠지만, 프로그래밍 자체를 즐기는 사람이라면 호기심에서 그 깊이를 파헤쳐 볼 것이며 이를 통해 게임 프로그래밍을 하면서 느꼈던 아쉬운 부분을 해소할 수 있을 것입니다.

이 책에서는 온라인 게임 시스템 구축을 소프트웨어 공학 측면에서 설명하려고 합니다. 어떻게 보면 앞에서 언급한 바와 같이 그 깊이는 얕을지도 모르겠습니다. 그러나 전체를 보는 능력에는 이 책이 큰 도움이 된다고 생각합니다. 빌딩을 건설하기 위해 자재를 준비하는 것과 고층을 어떻게 완성해 나갈지 생각하는 것은 완전히 다른 영역입니다. 필자는 후자의 입장에서 책을 서술하고 있으며 이 책이 온라인 게임 서버의 전반적인 이해에 대해 큰 도움이 될 수 있을 거라 확신하는 바입니다.

또한, 책의 내용의 깊이가 얕을지도 모른다고 언급했지만, 서버 엔진의 전반적인 내용을 대부분 설명하고 있으므로 어떻게 보면 깊이가 얕은 것도 아닙니다. 특히 서버 프로그래밍은 시스템 프로그래밍에도 익숙해야 하므로 컴퓨터 과학 측면과 소프트웨어 공학 측면을 모두 다뤄야 하는 영역이라고 할 수 있습니다. 이런 매력적인 요소를 지닌 게임 서버 프로그래밍에 도전을 진정으로 환영하는 바입니다.

오픈 소스의 매력

앞에서 언급하였지만, 게임을 개발할 때 클라이언트이든 서버이든 간에 소프트웨어 공학 측면에서 접근할 필요가 있습니다. 로그를 남기기 위해 로그 라이브러리를 개발하기보다 기존의 로그 라이브러리를 활용하는 것이 현명한 행동입니다. 또한, 프로세스 덤프를 남기기 위해 직접 덤프 모듈을 개발하기보다 이전에 검증받은 덤프 모듈을 활용하는 것이 생산성을 더 높일 수 있겠죠. 게다가 C++는 아직 주요한

언어이긴 하지만, 자바 등의 다른 프로그래밍 언어와 비교하면 생산성이 떨어지는 것은 부인할 수 없는 사실입니다(물론 그만큼 자유도는 매우 높습니다).

이런 상황에서 오픈 소스를 활용하는 것은 매우 생산적인 행위이며 원하는 제품의 개발 시간을 획기적으로 당겨줄 수 있습니다. 물론 그 안정성을 검증받지 못한 오픈 소스가 부지기수입니다. 이런 경우에는 수많은 테스트와 디버깅을 통해 그 신뢰성을 확보하면서 프로젝트에 적용하면 크게 문제가 되지 않습니다. 또한, 부스트 Boost나 ACE 라이브러리 같은 이미 수많은 고급 프로그래머에 의해 검증된 오픈 소스도 굉장히 많이 존재합니다. 이런 검증받은 오픈 소스를 사용하면 프로그램의 잠재적인 버그에 대한 리스크를 크게 줄일 수 있습니다.

CGSF는 이미 검증된 수많은 오픈 소스 라이브러리를 사용해서 구축하였으며 다음 책에서 이런 좋은 오픈 소스 라이브러리를 소개하겠습니다.

책의 구성

이 책은 두 권으로 구성되어 있습니다. 1권에서는 에코 서버, 채팅 서버 제작을 통해 범용 서버를 제작하는 방법을 다루고, 서버를 제작하는 데 도움을 주는 CGSF 라이브러리의 활용 방법을 소개합니다. 또한, 세븐 게임 프로젝트와 FPS 게임 프로젝트를 통해서 게임 서버를 쉽게 구축하는 방법에 관해서도 다룹니다. 2권에서는 CGSF 라이브러리의 활용 방법을 넘어서 CGSF의 내부 구조와 구현 원리를 설명하고 모바일 클라이언트와의 연동 방법에 대해 간단히 설명합니다.

대상 독자

CGSF는 온라인 게임 서버 시스템을 최대한 빨리 구현해서 프로토타입을 구축하려는 의도로 제작되었습니다. 현업에서 게임 서버를 개발했을 때 구축했던 구조를 담고 있기 때문에 온라인 게임 제작에 관심이 많거나, 게임 서버 프로그래밍에 관심이 많은 학생에게 좋은 가이드가 될 것입니다. 그리고 서버 개발에 대한 세부 지식 없이 게임 서버를 구축하려는 분들께도 하나의 대안이 될 수 있습니다. 아울러 서버 엔진을 직접 개발해 보려는 분들께도 이 책은 좋은 안내서가 될 것입니다. 현업 개발자에게도 온라인 게임 서버의 리뷰 차원에서 이 책은 가치가 있을 것입니다. 또한, CGSF는 다양한 오픈 소스를 활용하고 있고, 조금만 커스터마이징한다면 일반적인 서버로도 활용할 수 있으므로 네트워크에 관심 있는 분이라면 모두 대상 독자에 속한다고 볼 수 있습니다.

용어 표기 기준

이 책의 용어들은 표준국어대사전[01]의 한글 맞춤법과 표준어 규정, 외래어 표기법을 따릅니다. 또한, 표준국어대사전 규정에 없는 용어들은 한국정보통신기술협회 정보통신용어사전[02]을 참고하였습니다.

예) Thread → 스레드
Release → 릴리스

---

01 http://stdweb2.korean.go.kr/main.jsp
02 http://word.tta.or.kr/terms/terms.jsp

사전 지식 및 준비사항

CGSF는 윈도우 기반 라이브러리입니다. 크로스 플랫폼 지원은 현재 고려하고 있지 않습니다. 서버는 클라이언트보다 크로스 플랫폼을 크게 고려할 필요가 없으며 클라우드 서버는 윈도우 기반으로 많이 지원하므로 활용하는 데 별다른 문제가 없을 것입니다. 다만, 클라이언트는 모바일 환경에도 대응해야 한다고 생각하고 있기 때문에 나중에 모바일용 라이브러리를 제작할 계획입니다.

CGSF를 활용해서 게임 서버 및 응용 서버를 제작하려면 다음과 같은 도구와 지식이 필요합니다.

Visual Studio 2013

CGSF는 Visual Studio 2013으로 제작되었습니다. 따라서 윈도우 환경에서만 개발할 수 있으며 윈도우 운영체제에서만 서버 운영이 가능합니다. Visual Studio 2013은 무료 버전인 Express 버전이 존재하므로 Express 버전을 설치하면 CGSF의 빌드가 가능합니다. Visual Studio 2013 Express 버전은 다음 사이트에서 내려받으실 수 있습니다.

http://www.microsoft.com/ko-kr/download/details.aspx?id=40787

링크를 따라가면 DVD 이미지를 받아서 설치할 것인지 웹에서 내려받아 설치할 것인지를 묻는 화면이 나옵니다. 편의성을 생각해서 wdexpress_full.exe 파일을 내려받아서 설치합니다.

C++

CGSF는 전체를 C++로 구현하였습니다. 따라서 C++의 기본적인 문법은 숙지해

야 합니다. 해당 모듈을 활용만 한다면 C++의 기본적인 지식으로도 충분히 활용할 수 있지만, CGSF의 엔진 구조를 이해하기 위해서는 템플릿이나 추상 인터페이스 등의 개념을 제대로 이해하고 있어야 합니다. 또한, C++는 객체 지향 언어이므로 객체 지향에 대한 개념을 또한 알고 있어야 합니다.

디자인 패턴

CGSF에서는 단순하지만, 여러 가지 패턴을 활용하고 있습니다. 디자인 패턴 관련 서적을 살펴본 분이라면 무리 없이 책의 내용을 소화할 수 있을 것으로 생각합니다. 디자인 패턴에 대해 잘 모른다 하더라도 내용이 그렇게 어렵지 않으므로 모르는 패턴이 나올 때마다 웹에서 관련 자료를 참조한다면 이해하는 데 크게 무리가 없을 것입니다.

한빛 eBook 리얼타임

한빛 eBook 리얼타임은 IT 개발자를 위한 eBook입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾는 것도 쉽지 않습니다. 또한 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

1. eBook Only - 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다.

   500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 수 있는 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내는 선배, 전문가, 고수 분에게는 보다 쉽게 집필할 수 있는 기회가 될 수 있으리라 생각합니다. 또한 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 하고 있습니다.

2. 무료로 업데이트되는 전자책 전용 서비스입니다.

   종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나 저자(역자)와 독자가 소통하면서 보완하여 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

3. 독자의 편의를 위해 DRM-Free로 제공합니다.

구매한 전자책을 다양한 IT 기기에서 자유롭게 활용할 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해 독자 여러분이 언제 어디서 어떤 기기를 사용하더라도 편리하게 전자책을 볼 수 있도록 하기 위함입니다.

4. 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다.

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 힘든 전자책이 아니라, 전자책의 환경에 가능한 한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가도록 하겠습니다.

차례

PART 1 시작하기

PART 2 게임 서버 제작

PART 1
시작하기

1 | CGSF란

CGSF[Casual Game Server Framework]는 온라인 게임 서버를 제작할 때 공통으로 쓰이는 부분을 모듈화하여 초기 서버 제작 진입 비용을 최소화하기 위해 제작한 서버 프레임워크다. 캐주얼 게임, 즉 FPS나 스포츠 게임, 턴제 게임 등은 그 서버 구조가 유사하여 공통 부분을 모듈화하면 어떨까 하는 생각해서 CGSF가 탄생하게 되었다. 이후 기능을 보강하여 캐주얼 온라인 게임 서버뿐만 아니라 커스터마이징 가능한 서버 프레임워크로 발전해 왔다. CGSF를 활용하면 프로토타입 제작이 매우 쉽고 자신만의 독특한 서버를 개발하기가 쉽다는 것을 느낄 것이다.

지금까지 수많은 게임 서버가 제작되었지만, 현재는 기억 속에서 잊혔고 아마도 다시는 재사용되지 못할 것이다. 즉, 이런 서버는 그 당시를 위해서 제작된 서버일 뿐이다. 물론 게임 서버에서 공통으로 쓰는 부분을 모듈화하는 것은 꽤 어려운 작업이다. 게임마다 요구사항이 다르고 어떻게 보면 새롭게 제작하는 것이 더 효율적일 수도 있다. 그러나 필자는 반복적인 내용을 다시 개발하는 것은 싫었기 때문에 어떻게든 통합형 서버를 제작하고 싶었다. 이런 취지에서 탄생한 것이 CGSF다.[01]

그렇다면 CGSF를 쓰면 어떤 점에서 이득이 있을까?

"CGSF는 100% C++로 제작된 오픈 소스다." 이것은 한마디로 개발자가 100% 소프트웨어의 모든 영역을 다룰 수 있다는 것을 의미한다. 자바나 C#은 가상 머신에서 동작하므로 프로그래머가 메모리 관리에 직접 관여할 수 없다. 하지만 C++로 개발하면 메모리 관리 방식을 직접 변경할 수 있고, 수준에 따라 메모리 사용량을 줄일 수도 있으며 메모리 할당과 관련해서 속도 향상을 도모할 수도 있다. 또한, CGSF는 구현 세부사항을 몰라도 서버를 쉽게 제작할 수 있어서 간단하게 생각하

01 CGSF에 대한 배경 지식은 이 책 앞부분의 「이 책을 읽기 전에」에 자세히 나와 있으므로 참고하기 바란다.

면 정말로 단순한 구조라고 할 수 있다. 즉, CGSF는 고급 프로그래머뿐만 아니라 초급 프로그래머도 접근할 수 있는 게임 서버다. 이것이 가능한 이유는 네이티브 C++의 장점 때문이다. 언어 자체가 하드웨어적인 부분과 밀접하게 닿아 있고, C 언어보다 높은 생산성을 가진 객체 지향 언어라서 저수준/고수준 레벨의 프로그래밍을 모두 다룰 수 있다.

하지만 C++는 자바나 파이썬, C# 등에 비해 생산성 측면에서 경쟁력을 잃어가는 추세다. 게다가 하나의 모듈을 제작할 때마다 처음부터 개발해야 한다면 C++는 더욱더 경쟁력을 잃어가게 될 것이다. 이런 상황을 타개하려면 오픈 소스를 최대한 활용하는 것이 필수라고 생각한다. 지엽적인 부분에 시간을 낭비하지 않고 서버 구조와 콘텐츠에만 집중할 수만 있다면 서버를 개발하는 언어로서 C++도 경쟁력 있는 언어로 살아남을 수 있다. 이를 위해 CGSF는 수많은 오픈 소스를 활용하고 있다.

1.1 특징

앞에서 CGSF의 탄생 배경과 대략적인 특징을 설명하였다. 좀 더 구체적으로 CGSF가 어떤 특징이 있으며 CGSF로 서버를 개발할 때 장점이 무엇인지 살펴보자.

윈도우 기반의 C++ 서버

- CGSF는 윈도우 운영체제에서 동작하는 게임 서버다. 윈도우 XP를 포함한 상위 버전의 운영체제에서 동작한다.
- 32비트/64비트 모두 대응한다.
- 운영체제에 종속적인 기능을 활용하고 있다(IOCP, 인라인 어셈블리).

손쉬운 서버 제작

- 사용자가 서버에 대한 별다른 지식이 없어도 쉽게 서버를 구축할 수 있다. 이것이 가능하도록 네트워크와 관련된 기능은 최대한 래핑해서 외부 노출을 최소화했다.

- 서버 제작을 위한 다양한 샘플 프로젝트가 준비되어 있다.

기능 모듈화 및 모듈 교체 용이

- 네트워크 기능을 담당하는 모듈을 DLL 형태로 제작하였으며 CGSF에서 제공하는 인터페이스로 DLL을 제작하면 기존 네트워크 DLL을 대체할 수 있다.
- P2P 기능을 담당하는 모듈을 DLL 형태로 제공한다.
- 데이터베이스 처리에 관련된 내용이 모듈화되었다.

기존 게임 서버에서 나타나는 공통 부분 모듈화

- 캐주얼 온라인 게임에서 나타나는 아웃게임/인게임 구조를 모듈화(로비, 방)하여 코드의 재사용성을 높였다.

개발 편의성 도모

- 멀티 스레드 문제를 신경 쓰지 않고 로직 구현에 집중할 수 있도록 설계하였다.
- 조립해서 만드는 서버다(DB 기능이 필요하면 DB 모듈을, P2P 기능이 필요하면 P2P 모듈을 추가하면 된다).

성능을 고려한 서버 설계

- 현업에서 사용하고 있는 게임 서버 모델이다.
- DB 처리를 비동기로 구현하였다.

손쉬운 커스터마이징

- 게임 서버 모델의 변경이 쉽다.
- 다양한 패킷 프로토콜을 지원하고, 패킷 프로토콜이 모듈화되어 있어서 손쉽게 다른 패킷 프로토콜로 변경할 수 있다.

물론 CGSF는 오픈 소스이므로 누구나 마음껏 가져다 쓸 수 있다.

1.2 빌드 환경 설정

1.2.1 소스 코드 레이아웃

다음 링크에 접속해서 소스 코드를 내려받고[02] 압축을 풀면 [표 1-1]과 같은 폴더를 확인할 수 있다.

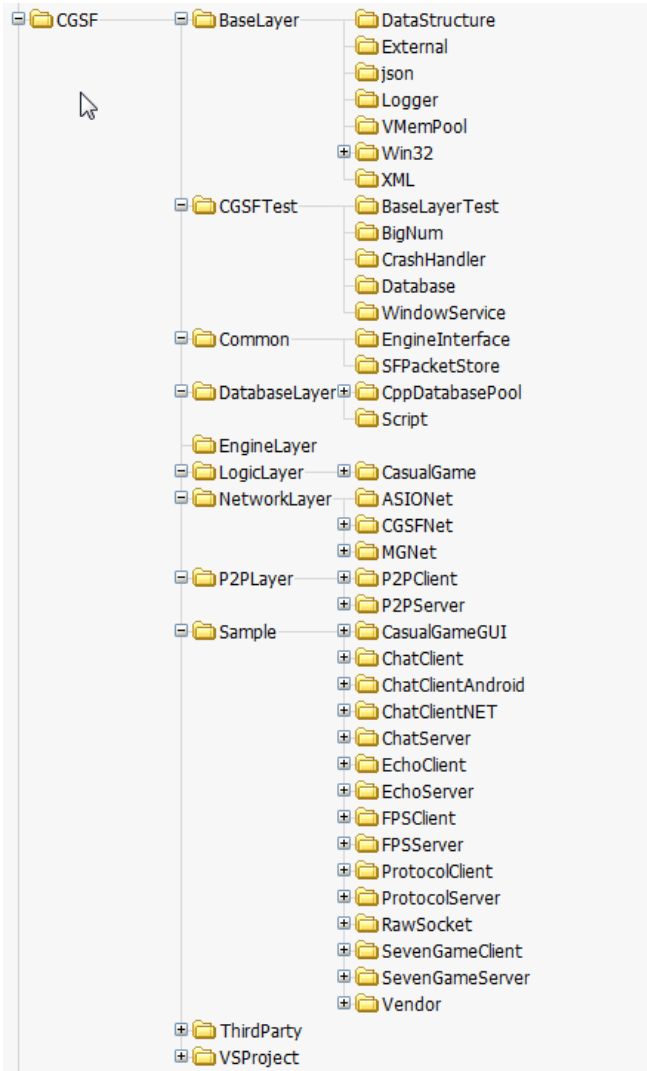
https://github.com/pdpdds/CGSF

[표 1-1] CGSF 소스 폴더

폴더	설명
BaseLayer	CGSF에서 공통으로 사용하는 라이브러리를 모은 프로젝트다.
CGSFTest	CGSF의 베이스 라이브러리 테스트 코드가 들어 있다.
Common	서버와 클라이언트가 공유하는 코드가 들어 있다.
DatabaseLayer	데이터베이스 처리를 위한 라이브러리를 담고 있다.I
EngineLayer	CGSF 시스템을 구축하는 핵심 코어다. 네트워크 레이어와 로직 레이어의 가교 역할을 담당한다.
LogicLayer	캐주얼 온라인 게임 서버 제작을 위해 제작한 라이브러리다. 만약 캐주얼 온라인 게임 서버가 아니라 MO나 MMO 서버를 제작하려고 한다면 새로운 로직 레이어를 개발하면 된다.
NetworkLayer	실제 네트워크에서 들어오는 데이터를 처리하는 모듈 단이다. 유저가 보낸 데이터, 즉 패킷을 생성해서 엔진 레이어에 넘길 책임이 있다.
P2PLayer	P2P 시스템을 개발하기 위한 모듈이 포함되어 있다.
Sample	CGSF 라이브러리를 활용해서 제작한 각종 샘플 프로젝트가 들어 있다.
ThirdParty	각종 오픈 소스 프로젝트의 모음 폴더다.
VSProject	솔루션 파일 및 바이너리 실행을 위한 환경 설정 파일들이 들어 있다.

02 현재 개발 중인 코드를 내려받아도 문제없지만, 책에 기술한 내용이 변경되었을 가능성이 있으므로 홈페이지의 release 항목에 있는 파일을 받거나 링크(http://goo.gl/bqAC6I)에서 전체 소스를 내려받길 바란다.

[그림 1-1] CGSF 소스 폴더 트리



1.2.2 Third-party 설정

CGSF는 많은 오픈 소스를 활용한다. 이런 서드 파티의 오픈 소스를 일일이 컴파일하는 부담을 덜기 위해 미리 빌드한 바이너리를 프로젝트 내에 포함하였다. 단, 부스트Boost 라이브러리와 DirectX SDK 2010 6월 버전은 로컬에 직접 설치해야 한다.

부스트 라이브러리 설치

부스트 라이브러리 공식 사이트 Current Release 페이지[03]에 접속한다(현재 1.55.0 버전이다). 부스트를 빌드하려면 시간이 오래 걸리므로 소스 코드를 내려받는 대신 OTHER DOWNLOADS의 Windows binaries를 클릭한다. 연결되는 페이지[04]에서 boost_1_55_0_msvc-12.0-64.exe와 boost_1_55_0_msvc-12.0-32.exe 두 파일을 내려받고 로컬에 설치한다.

DirectX SDK 2010 설치

마이크로소프트 홈페이지 Download Center[05]에서 DirectX SDK(현재 2010년 6월 버전)를 내려받은 후 로컬에 설치한다. 파일에는 32비트와 64비트 버전의 라이브러리가 모두 포함되어 있다.

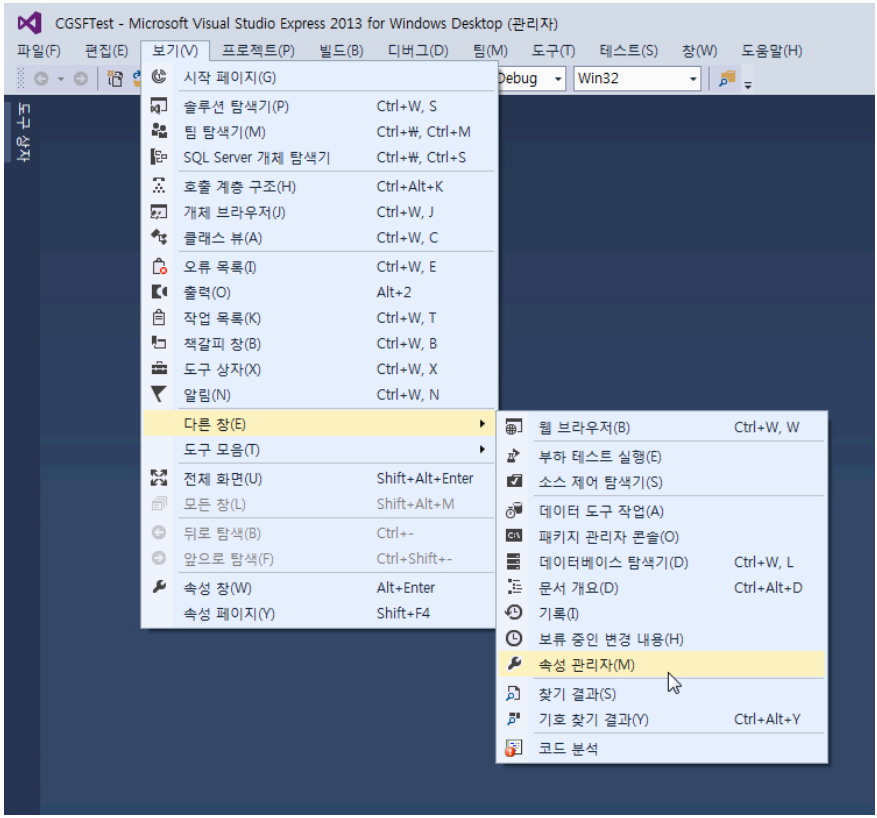
1.2.3 프로젝트 빌드

VSProject 폴더의 AllProject.sln 파일을 실행한다. 포함 경로와 라이브러리 경로에 부스트 라이브러리와 DirectX SDK가 설치된 곳을 지정한다. 두 라이브러리가 모든 프로젝트에 적용되도록 전역으로 설정한다.

---

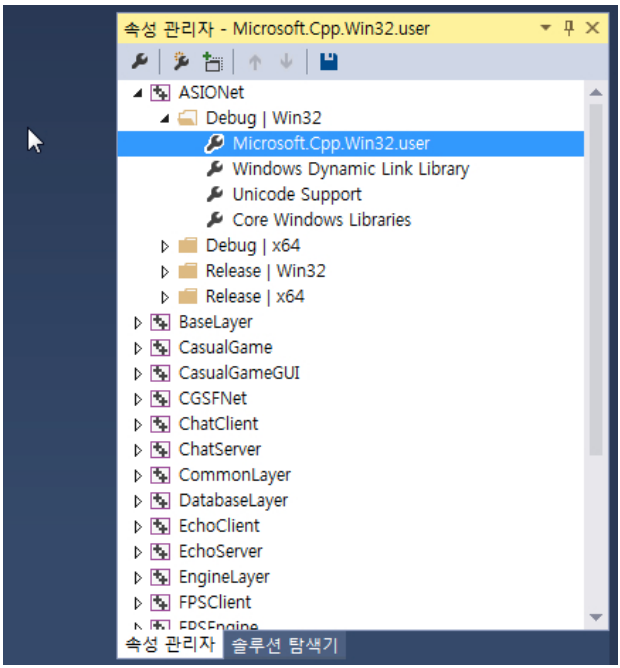
03 http://www.boost.org/users/history/version_1_55_0.html
04 http://sourceforge.net/projects/boost/files/boost-binaries/1.55.0/
05 http://www.microsoft.com/en-us/download/details.aspx?id=6812

[그림 1-2] 속성 관리자 창



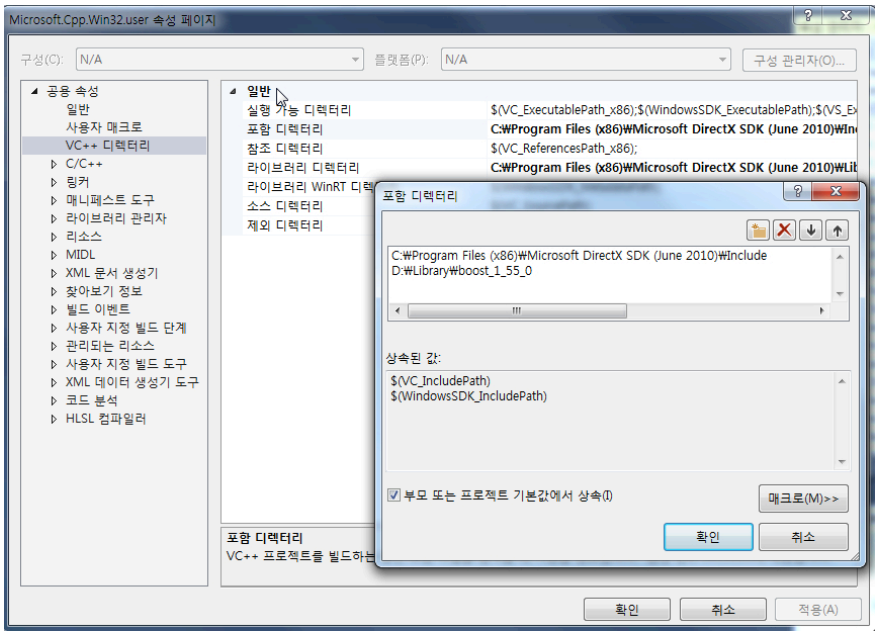
보기 → 다른 창 → 속성 관리자를 선택하여 속성 관리자 창을 연다.

[그림 1-3] Microsoft.Cpp.Win32.user



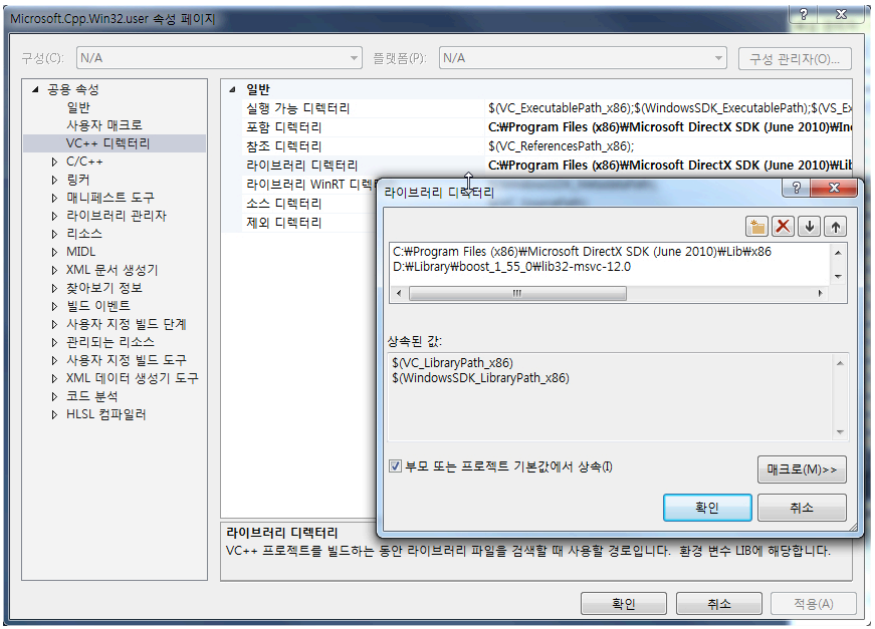
속성 관리자 창에서 아무 프로젝트나 선택한 다음 Microsoft.Cpp.Win32.user를 더블 클릭한다.

[그림 1-4] 포함 디렉터리 경로 지정



공용 속성 → VC++ 디렉터리 → 포함 디렉터리를 편집해서 부스트와 DirectX 경로를 설정한다(해당 폴더 위치는 각자 설치한 경로 위치로 지정한다). 64비트도 동일하게 적용한다.

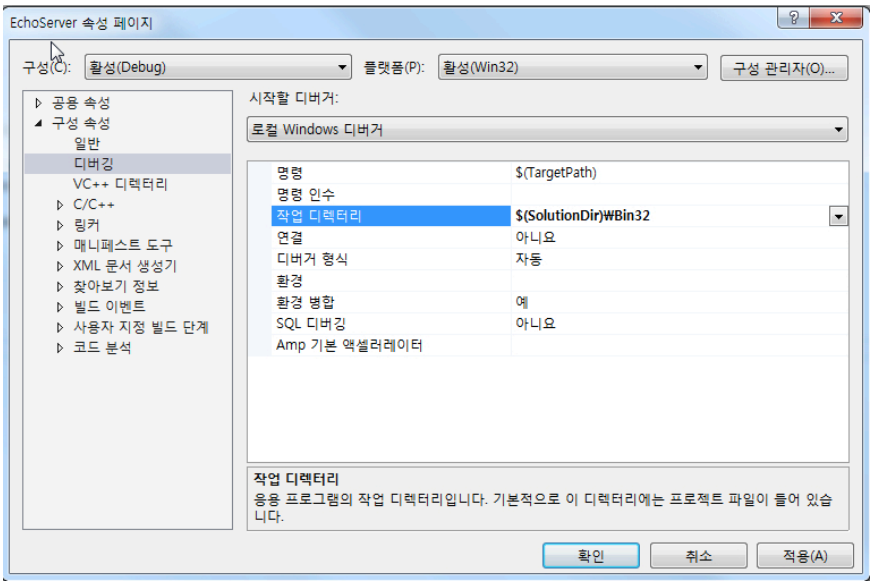
[그림 1-5] 라이브러리 디렉터리 경로 지정



공용 속성 → VC++ 디렉터리 → 라이브 디렉터리를 편집해서 부스트와 DirectX 라이브러리 경로를 설정한다. 64비트는 64비트 라이브러리 경로로 설정한다.

이제 솔루션을 빌드해서 x86/x64 디버그/릴리스 모드로 컴파일되는 것을 확인한다. 32비트 바이너리는 Bin32 폴더에, 64비트 바이너리는 Bin64 폴더에 생성된다. 마지막으로 디버깅을 위해 디버깅 경로를 지정한다.

[그림 1-6] 디버깅을 위한 디버그 경로 지정(EchoServer 프로젝트)



솔루션 탐색기 → 속성 → 구성속성 → 디버깅 → 작업 디렉토리를 선택해서 경로를 $(SolutionDir)\Bin32로 설정한다(64비트는 $(SolutionDir)\Bin64). 다른 프로젝트를 디버깅할 때에도 같은 방법으로 디버깅 경로를 설정한다.

이제 CGSF를 사용할 준비가 끝났다. 32비트 및 64비트로 컴파일되는지 꼭 확인한다. 이후 내용은 32비트 디버깅 모드를 기준으로 설명한다. Bin32/Bin64 폴더의 서드 파티 dll 파일들은 모두 디버그로 빌드된 파일이다. 릴리스로 배포할 때에는 릴리스 버전의 dll로 배포하기 바란다. 서드 파티의 dll 파일들은 ThirdParty 폴더에서 찾을 수 있다.

1.3 정리

1장에서는 게임 서버 제작에 들어가기 전에 기본으로 알아야 할 CGSF 특징과 환경 설정 등을 살펴보았다. 다음 장부터는 본격적으로 CGSF를 사용해서 서버 제작의 기본이 되는 에코 서버와 채팅 서버를 제작해 보고 클라이언트와 서버가 주고받는 패킷 프로토콜 설정 방법을 살펴본다. 또한, 서버 제작에 도움이 되는 CGSF 라이브러리의 활용 방법과 세븐 게임 프로젝트와 FPS 게임 프로젝트를 통해서 게임 서버를 쉽게 구축하는 방법에 관해서도 살펴보겠다.

2 | 에코 프로젝트

모든 서버의 기초는 에코 서버에서 시작된다고 할 수 있다. 산에서 큰소리로 외치면 자신의 소리가 메아리가 되어서 되돌아오듯이 에코 서버도 클라이언트가 보낸 정보를 그대로 되돌려 주는 역할을 한다.

에코 서버와 에코 클라이언트 프로젝트는 Sample 폴더에 존재한다. VSProject 폴더에서 AllProject.sln 또는 Sample.sln 파일을 열어서 EchoServer와 EchoClient 프로젝트를 확인한다.[01]

2.1 에코 서버 구현

우선 에코 서버 프로젝트부터 살펴보자.

2.1.1 에코 서버 메인 엔트리

[코드 2-1] EchoServer.cpp

```
int _tmain(int argc, _TCHAR* argv[])
{
	EchoLogicEntry* pLogicEntry = new EchoLogicEntry();

	SFEngine::GetInstance()->Intialize(pLogicEntry, new
SFPacketProtocol<SFJsonProtocol>);
	SFEngine::GetInstance()->Start();

	google::FlushLogFiles(google::GLOG_INFO);

	getchar();
```

01 디버깅 경로 설정을 잊지 말자. 다른 프로젝트를 디버깅할 때도 마찬가지다.

```
    SFEngine::GetInstance()->ShutDown();

    return 0;
}
```

EchoLogicEntry 클래스는 에코 서버의 메인 부분으로, 패킷 처리를 담당하는 클래스다. 해당 클래스의 객체를 생성한 뒤 SFEngine 객체 초기화 메소드의 첫 번째 인자로 넣어준다. SFEngine 클래스의 Intialize 메소드 두 번째 인자는 패킷 프로토콜에 관련된 것으로, 에코 서버에서는 JSON(JavaScript Object Notation) 프로토콜을 사용하므로 JSON 패킷 프로토콜 객체를 생성해서 인자로 넘겨준다.

엔진을 초기화한 뒤 Start 메소드를 호출하여 서버를 구동한다. 정상적으로 서버가 시작되었다면 결과값으로 true가 반환된다. 이후 네트워크 이벤트가 발생하면 EchoLogicEntry 객체가 그 이벤트를 다룬다. 서버 구동에 관련된 코드는 이것이 전부다.

2.1.2 EchoLogicEntry 클래스

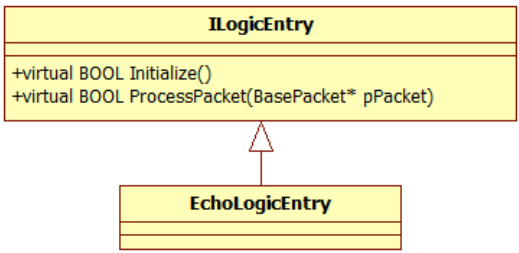
EchoLogicEntry는 프로그래머가 직접 작성해야 할 클래스이므로 자세히 살펴보자.

[코드 2-2]

```
class EchoLogicEntry : public ILogicEntry
{
public:
    EchoLogicEntry(void);
    virtual ~EchoLogicEntry(void);

    virtual BOOL Initialize() override;
    virtual BOOL ProcessPacket(BasePacket* pBasePacket) override;
};
```

[그림 2-1] EchoLogicEntry 클래스 다이어그램



EchoLogicEntry 클래스는 ILogicEntry 인터페이스를 구현한 클래스로, Initialize 메소드와 ProcessPacket 메소드를 반드시 구현해야 한다. Initialize 메소드는 서버가 처음 구동될 때 초기화 작업을 수행하는 메소드로, DB 접속이 필요한 서버를 만든다면 해당 메소드에서 기능의 초기화 작업을 수행하면 된다. ProcessPacket 메소드는 실제 네트워크 이벤트를 다루는 메소드다.

[코드 2-3]

```
bool EchoLogicEntry::ProcessPacket(BasePacket* pPacket )
{
      switch (pPacket->GetPacketType())
      {
      case SFPACKET_DATA:
         SFEngine::GetInstance()->SendRequest(pPacket);
         break;
      }

      return true;
}}
```

패킷을 분석할 필요가 없으므로 어떤 패킷 프로토콜을 쓰든지 신경 쓰지 않고 그대로 패킷을 클라이언트에게 되돌려 주는 코드다. CGSF에서 사용되는 패킷은 모두 BasePacket 클래스를 상속받아 사용한다. 패킷 이벤트에는 여러 종류가 있는데, 유저가 보낸 데이터는 패킷 유형으로 SFPACKET_DATA가 설정된다. 해당 유저에게 데이터를 보낼 때는 SFEngine 클래스의 SendRequest 메소드를 사용한다.

2.1.3 디자인 패턴

코드에서 크게 어려운 부분은 없지만, 몇 가지 중요한 디자인 패턴이 사용되었다.

싱글턴 패턴

SFEngine 클래스는 객체가 하나만 생성이 되도록 싱글턴 패턴을 사용했다. 싱글턴 패턴에 대해서는 링크[02]를 참조하기 바란다.

단위 전략 패턴

SFPacketProtocol<SFJsonProtocol>처럼 패킷 프로토콜을 지정하기 위해 템플릿으로 인자를 넘긴다. JSON을 사용하지 않고 다른 패킷 프로토콜을 사용하고 싶다면 해당 템플릿에 다른 프로토콜을 담으면 된다. 이렇게 상황에 따라 전략을 선택할 수 있는 형태를 단위 전략 패턴이라고 부른다. 단위 전략 패턴의 좋은 점은 단위 전략을 바꿔도 컴파일할 때 전혀 문제가 없으며(호출되는 메소드를 제대로 구현했을 때) 가상 함수를 쓰지 않으므로 함수 호출이 컴파일 타임에 바인딩되어 성능에 이점이 있다.[03]

---

02 http://have1824.blog.me/110149560453

03 최근 모바일 게임 서버들이 별문제 없이 잘 돌아가는 것을 보면 여기서 언급한 성능상의 이점이 과연 이점일까라는 생각이 들기도 한다.

2.2 에코 클라이언트 구현

에코 서버에 접속하기 위한 에코 클라이언트를 살펴보자.

2.2.1 에코 클라이언트 메인 엔트리

[코드 2-4] EchoClient.cpp

```
int _tmain(int argc, _TCHAR* argv[])
{
	EchoCallback* pCallback = new EchoCallback();

	SFNetworkEntry::GetInstance()->Initialize(pCallback, new
SFPacketProtocol<SFJsonProtocol>);
	SFNetworkEntry::GetInstance()->Run();

	ProcessInput();

	SFNetworkEntry::GetInstance()->ShutDown();

	return 0;
}
```

서버 코드와 크게 다르지는 않지만, 엔진 클래스로 SFEngine이 아니라 SFNetwork Entry 클래스를 사용하고 있다. SFNetworkEntry 클래스는 내부적으로 SFEngine 객체를 사용하고 있으며 클라이언트를 위한 기능이 추가된 래핑 클래스다. 서버와 마찬가지로 패킷 프로토콜을 지정하고 콜백 객체를 생성해서 SFNetworkEntry 객체에 넘겨준다. 그런 다음 Run 메소드를 호출하여 네트워크 연결을 시도한다. 여기서 ProcessInput 함수는 유저의 입력을 처리하고 네트워크 이벤트를 가져오기 위해 임시로 구현한 함수다.

[코드 2-5] 에코 클라이언트 유저 입력 처리 스레드 생성 및 네트워크 이벤트 가져오기

```
void ProcessInput()
{
      int inputThreadID = ACE_Thread_Manager::instance()->spawn_n(1,
(ACE_THR_FUNC)EchoInputThread, NULL, THR_NEW_LWP,
ACE_DEFAULT_THREAD_PRIORITY);

      SFASSERT(inputThreadID != -1);

      while (SFNetworkEntry::GetInstance()->IsConnected())
      {
         SFNetworkEntry::GetInstance()->Update();
         Sleep(1);
      }

      ACE_Thread_Manager::instance()->wait_grp(inputThreadID);
}
```

네트워크 이벤트를 가져오려면 SFNetworkEntry 클래스의 Update 메소드가 호출되어야 한다. 메소드 호출은 메인 스레드에서 하므로 서버로 데이터를 전송하기 위해서는 별도의 스레드를 생성해야 한다. 별도의 스레드를 생성하기 위해 ACE 라이브러리[04]를 활용한다. 루프 부분에 Sleep(1)을 넣은 이유는 CPU 낭비를 막기 위해서다. 당연한 이야기지만, GUI가 들어간 코드에서는 Sleep 함수가 없어야 한다.

이번 에코 클라이언트 예제에서는 ACE의 스레드 제어 기능을 활용했다. ACE_Thread_Manager라는 스레드 관리 싱글턴 객체를 통해서 스레드를 생성하며 시작 지점은 EchoInputThread다.

04 CGSF 전반에서 ACE 라이브러리를 활용하고 있으므로 어떤 기능이 있는지 살펴보길 바란다.

[코드 2-6] 유저 입력 처리

```
void EchoInputThread(void* Args)
{
      std::string input;

      while (SFNetworkEntry::GetInstance()->IsConnected())
      {
         std::cin >> input;

         if(input.compare("exit") == 0)
            break;

         SFJsonPacket packet(ECHO_PACKET_ID);
         packet.GetData().Add( " ECHO " , input.c_str());

         SFNetworkEntry::GetInstance()->TCPSend(&packet);
      }
}
```

앞의 코드는 유저의 입력을 받아서 서버로 패킷을 보낸다. 패킷 프로토콜을 JSON으로 선언하였으므로 SFJsonPacket 객체를 생성하여 유저의 입력을 버퍼에 담고, SFNetworkEntry 객체의 TCPSend 메소드로 패킷을 서버에 전송한다. 서버는 받은 데이터를 그대로 클라이언트에 전달하고, 클라이언트는 콜백 객체인 EchoCallback에서 서버가 보낸 데이터를 처리한다.

2.2.2 EchoCallback 클래스

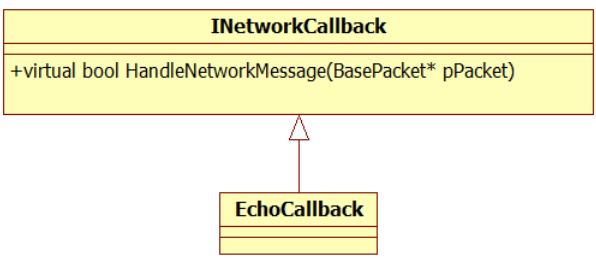
[코드 2-7] EchoCallback 클래스

```
class EchoCallback : public INetworkCallback
{
public:
```

```
    EchoCallback(void);
    virtual ~EchoCallback(void);

    virtual bool HandleNetworkMessage(BasePacket* pPacket) override;
};
```

[그림 2-2] EchoCallback 클래스 다이어그램



서버 쪽 네트워크 이벤트 처리 클래스는 ILogicEntry 클래스를 상속받았지만, 클라이언트 쪽은 INetworkCallback 인터페이스를 상속받는다. 네트워크 이벤트는 HandleNetworkMessage 메소드에서 처리할 수 있다.

[코드 2-8] EchoCallback 클래스의 네트워크 이벤트 처리

```
bool EchoCallback::HandleNetworkMessage(BasePacket* pPacket)
{
    SFJsonPacket* pJsonPacket = (SFJsonPacket*)pPacket;

    std::cout << "Received : " << pJsonPacket->GetData().GetValue<tstring>
("ECHO") << std::endl;

    return true;
}
```

에코 클라이언트는 서버로부터 받은 데이터를 콘솔 창에 그대로 출력한다.

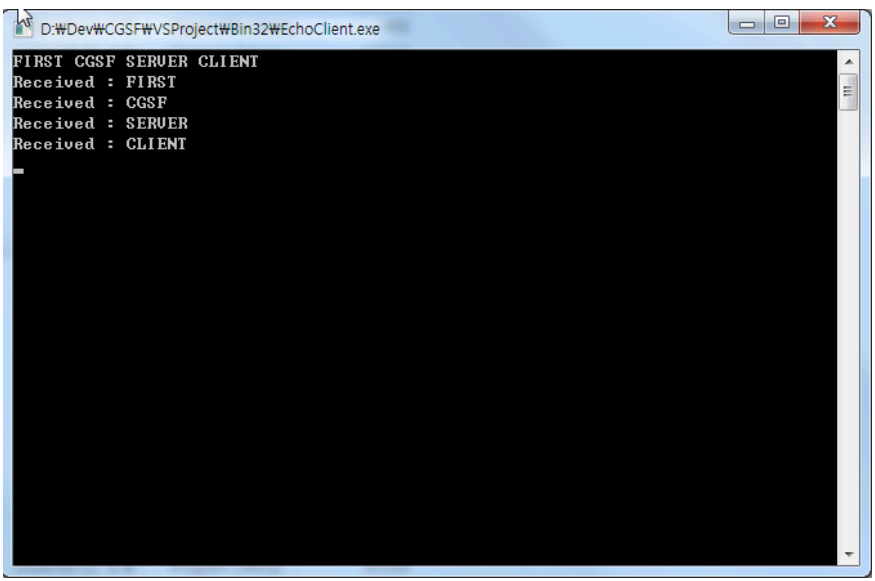
NOTE

JSON 포맷의 유효성 검증과 빠른 편집을 위해 구글 크롬의 애플리케이션 중 하나인 JSON Editor[05]를 설치해서 활용해 보자. JSON에 대해서 잘 모른다면 큰 도움이 된다.

2.2.3 결과 확인 및 설정 파일

에코 서버를 제작하는 데 필요한 코드는 앞에 설명한 코드가 전부다. 비록 간단한 서버와 클라이언트지만, 서버의 모든 기능을 갖추고 있다.

[그림 2-3] 에코 클라이언트 결과 화면



여기서 서버 접속 부분에 관해 기술하지 않아서 의아하게 생각할 수도 있다. 클라이언트에서는 Connection.ini 파일의 정보를 이용해서 서버에 접속하고, 서버는 EngineConfig.xml이란 파일의 정보를 토대로 구동한다.

05 https://chrome.google.com/webstore/detail/json-editor/lhkmoheomjbkfloacpgllgjcamhihfaj

[Connection.ini]

```
[ServerInfo]
IP=127.0.0.1
PORT=25251
[UDPInfo]
IP=127.0.0.1
PORT=30700
[Engine]
NAME=CGSFNet.dll
[EngineConfig.xml]
<?xml version="1.0" encoding="utf-8"?>
<EngineConfig.xml type="Struct">
    <EngineName type="wstring" value="CGSFNet.dll" />
    <ServerIP type="wstring" value="127.0.0.1" />
    <ServerPort type="uint16" value="25251" />
    <HostName type="wstring" value="host" />
    <TimerList type="array">
      <item0 type="uint32" value="4" />
    </TimerList>
    <LogDirectory type="wstring" value="d:\cgsflog\" />
</EngineConfig.xml>
```

두 설정 파일에서 중요한 것은 같은 네트워크 엔진을 사용해야 한다는 점이다. 여기서는 두 파일 모두 네트워크 엔진으로 CGSFNet.dll 엔진을 사용하지만, 만약 한쪽이 AsioNet.dll 엔진을 사용한다면 서버와 클라이언트 간 통신이 제대로 되지 않는다.

서버 설정 파일은 로그를 남기기 위해 경로를 지정할 수 있으며 디폴트로 바이너리 경로의 서브 폴더인 Log 폴더에 로그가 남도록 하였다. 로그는 구글의 로그 라이브러리인 glog[06]를 사용하였다.

06 https://code.google.com/p/google-glog/

2.3 정리

에코 서버와 에코 클라이언트는 CGSF를 활용한 서버 제작의 기초가 되므로 내용을 확실히 이해하고 넘어갈 필요가 있다. 서버와 클라이언트를 동시에 디버그해서 네트워크 흐름을 확인하도록 하자.

IDE를 각각 띄워서 디버깅을 할 수 있겠지만, 하나의 IDE로도 디버깅하는 것이 가능하다. 에코 클라이언트를 디버깅한다고 가정하고 여기에 에코 서버까지 동시에 디버깅하는 방법을 설명하겠다. 일단 에코 서버 바이너리를 실행하자. 그런 다음 IDE에서 '디버그 → 프로세스에 연결' 또는 '도구 → 프로세스에 연결'을 클릭하면 여러 프로세스가 나열되는데, 여기서 EchoServer.exe를 선택하면 에코 서버와 에코 클라이언트를 동시에 디버깅할 수 있다.