|  |
| --- |
| 한국산업기술대학교 게임공학과 |
| 포트폴리오 |
| 서버 부분 |

|  |
| --- |
| 신윤규(YunKyu Shin)  2017-12-1 |

목차

[1. 백명이 접속 가능한 게임 2](#_Toc500265165)

[기본 설명 2](#_Toc500265166)

[세부 설명 3](#_Toc500265167)

[제작하면서 어려웠던 점. 6](#_Toc500265168)

[2. 멀티로 즐기는 탱크 게임 7](#_Toc500265169)

[기본 설명 7](#_Toc500265170)

[세부 설명 8](#_Toc500265171)

[제작하면서 어려웠던 점. 10](#_Toc500265172)

1. 백명이 접속 가능한 게임



**이미지 설명 -** 다수의 클라이언트가 동시 접속하여, 플레이 하는 모습

기본 설명

**게임 제목**: SimpleMMO

**영상 링크**: https://youtu.be/r57ig-gG6aw

**프로젝트 소스:** https://github.com/YunKyuSHin/YunKyu/tree/master/%ED%8F%AC%ED%8A%B8%ED%8F%B4%EB%A6%AC%EC%98%A4\_1

**사용 언어:** C++언어, LUA사용

**개발 환경**: Visual Studio 15 32bit release

**장르:** MMORPG게임

**제작 인원**: 1인 개발

**제작 & 제작 기간**: 클라이언트, 서버 제작, 4개월 소요

**목표**: 윈도우 소켓, iocp를 활용하여, 100명이 접속 가능한 비동기 게임서버 제작

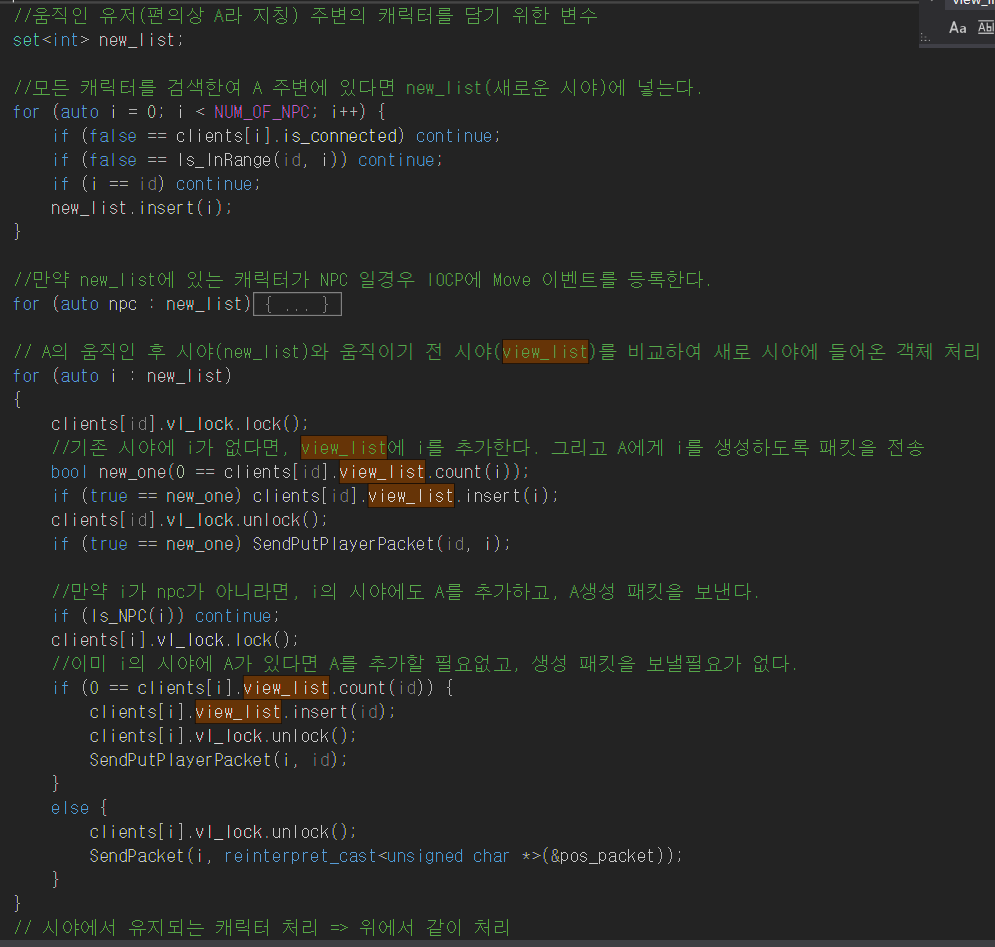
**게임 설정**: 플레이어 - 하늘색 원, 유저, NPC - 하얀색 원, 유저 크기 - 1x1

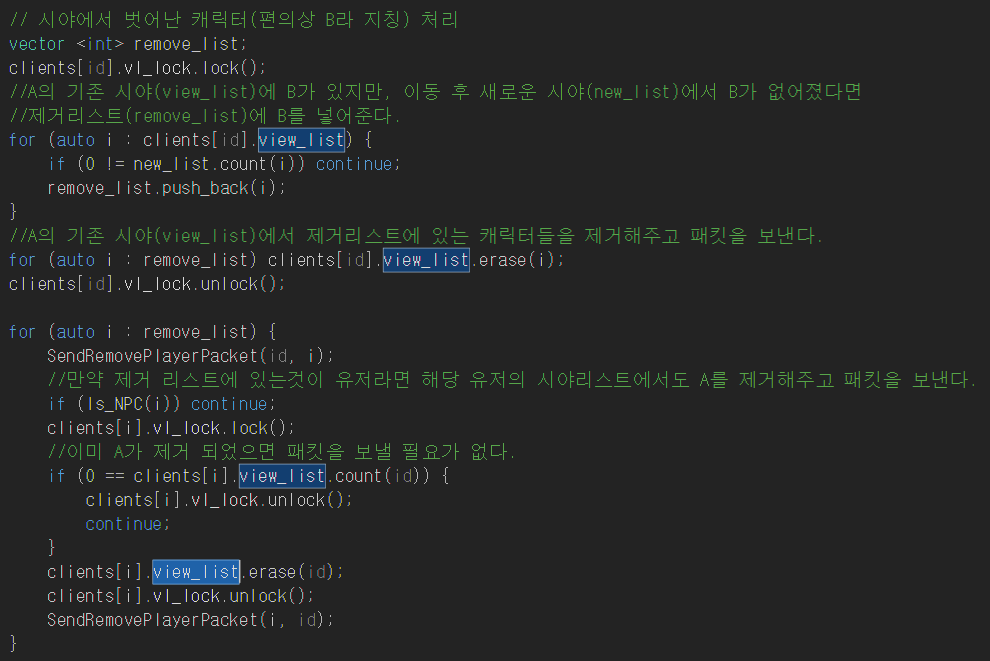
장애물 – 자주색 사각형, 맵 크기 – 250x250, 유저의 시야 범위 – 7

세부 설명

1. 게임 구조:   
   윈도우 소켓, IOCP를 활용한 비동기 TCP 게임 서버 제작.  
   구조적으로 100명의 유저를 받을 수 있으며, 1000명의 NPC가 존재합니다.  
   NPC는 유저가 가까이 접근 시, 무작위 방향으로 움직이거나, 공격을 합니다.  
   유저의 hp는 100이며, NPC에게 피격 시 hp가 1 감소하며, 일정 시간 후,  
   hp가 1씩 회복 합니다.
2. 서버 상세설명:
3. 비동기 IOCP 사용 이유:   
   다수의 스레드 수는 스레드들간의 컨텍스트 스위칭으로 인하여 운영체제 오버헤드를 발생하는데, 이를 줄이기 위하여 IOCP를 사용하였습니다. 또한 동기로 인하여 스레드가 대기상태에서 기다리지 않기 위해서 비동기를 사용하였습니다.
4. 스레드 종류와 수
   1. Accept\_threads(1개): 유저의 접속만 받아들이는 스레드  
      소켓을 IOCP에 연결해주고, 수신대기상태로 만든다.
   2. Worker\_threads(6개): 서버의 실제 작업을 하는 스레드  
      역할: 유저와 통신을 주고 받고 패킷의 해석 및 처리, NPC의 이벤트를 처리.  
      IOCP로부터 I/O 결과를 받아와 알맞은 이벤트를 처리한다.
   3. Timer\_threads(1개): 시간에 맞춰 발생 할 이벤트가 있음을 알려주는 스레드  
      역할: 유저의 HP회복 알림, NPC 행동 알림  
      Timer\_threads내에 있는 변수 p\_queue는 우선순위 큐이며, 인자 값(ID, 이벤트 시간, 발생할 이벤트)을 가집니다. 발생 시간이 되면, IOCP에 이벤트를 추가하고, worker\_threads에서 해당 이벤트를 처리합니다.
5. 패킷 : 패킷 구조체의 처음 2바이트는 고정이고, 나머지는 구조체 마다 다릅니다. 처음 1BYTE는 패킷의 사이즈, 다음 1BYTE는 패킷의 타입이 들어갑니다.

1. OverlapEx구조체 : WSAOVERLAPPED구조체를 확장하여 사용한 구조체
   1. Operation – IOCP의 이벤트를 설정
   2. Paraml – NPC에서만 사용, 인자값 전달을 위해 사용
   3. Recv\_buf – WSARecv(), WSASend()에 인자로 들어갈 구조체
   4. Socket\_buf –
   5. Packet\_size – 패킷의 크기를 저장, 만약 패킷이 아직 다 도착하지 못한 경우 패킷 조립을 위해 필요함
2. LUA사용 : 스크립트 언어 LUA를 활용하여, NPC가 움직이는데 필요한 함수들을 미리 제작하고, 다른 사람이 해당 함수를 이용하여, NPC에게 명령을 내릴 수 있습니다.
3. 유저 시야 처리 : 시야 밖의 오브젝트의 정보를 전송하지 않음으로 써, 서버 성능 향상 및 네트워크 전송량 감소를 위하여 사용했습니다. 시야 내의 유저와 NPC 정보를 저장하기 위하여 view\_list의 이름을 가진 변수를 사용하였습니다. 자료형은 검색속도가 빠른 set자료구조를 사용했습니다.
   1. 문제 발생 : 단순히 캐릭터와 캐릭터의 거리로 view\_list를 만들면, 동시에 캐릭터가 접근 시, 병렬 처리로 인하여 view\_list에 캐릭터가 없지만 실제로는 존재하는 문제가 발생합니다. 이러한 문제를 해결 하기 위하여, mutex락(변수명: vl\_locK )을 사용하여, 아래 사진과 같이 해결하였습니다.   
      해당 부분 코드: https://github.com/YunKyuSHin/YunKyu/blob/ab0dad2b2f5ad0491f1ec29ffdc3a46fb03dfa5f/%ED%8F%AC%ED%8A%B8%ED%8F%B4%EB%A6%AC%EC%98%A4\_1/%EC%84%9C%EB%B2%84/%EC%84%9C%EB%B2%84ICOP/Sever\_main.cpp#L653





제작하면서 어려웠던 점.

1. 데이터 레이스 발생
   1. 자료형이 set인 A의 변수가 있을 때, 프로그램이 멀티스레드로 진행되면서 한 쪽 스레드에서는 A의 원소를 이미 제거했는데, 다른 스레드에서는 A의 원소를 읽으려는 하는 문제가 발생하였습니다. 이 문제는 lock를 잘 못 걸어서 발생한 것으로 lock이 잘 못 걸린 부분을 찾는 부분이 어려웠습니다.
2. 멀티로 즐기는 탱크 게임

****

**이미지 설명 –** npc 제거 미션을 수행하는 모습

기본 설명

**게임 제목**: 탱크 게임

**영상 링크**: -----

**프로젝트 소스:** https://github.com/YunKyuSHin/YunKyu/tree/master/%ED%8F%AC%ED%8A%B8%ED%8F%B4%EB%A6%AC%EC%98%A4\_2

**사용 언어:** C++언어, UE4 블루프린트

**개발 환경**: UE4 4.14 & 4.16버전 64bit

**장르:** 온라인 탱크 시뮬레이션

**제작 인원**: 3명

**역할:** 서버 제작, 클라이언트의 통신을 위한 클래스 제작

**제작 기간**: 약8개월 소요

**목표**: 언리얼4에서 윈도우소켓을 활용한 클라이언트 네트워크 구현 및 Visual Studio17 64bit을 사용한 윈도우 서버 제작

**게임 내용**: 유저 vs NPC 대결, 유저 vs 유저 대결 의 2가지 모드 존재

세부 설명

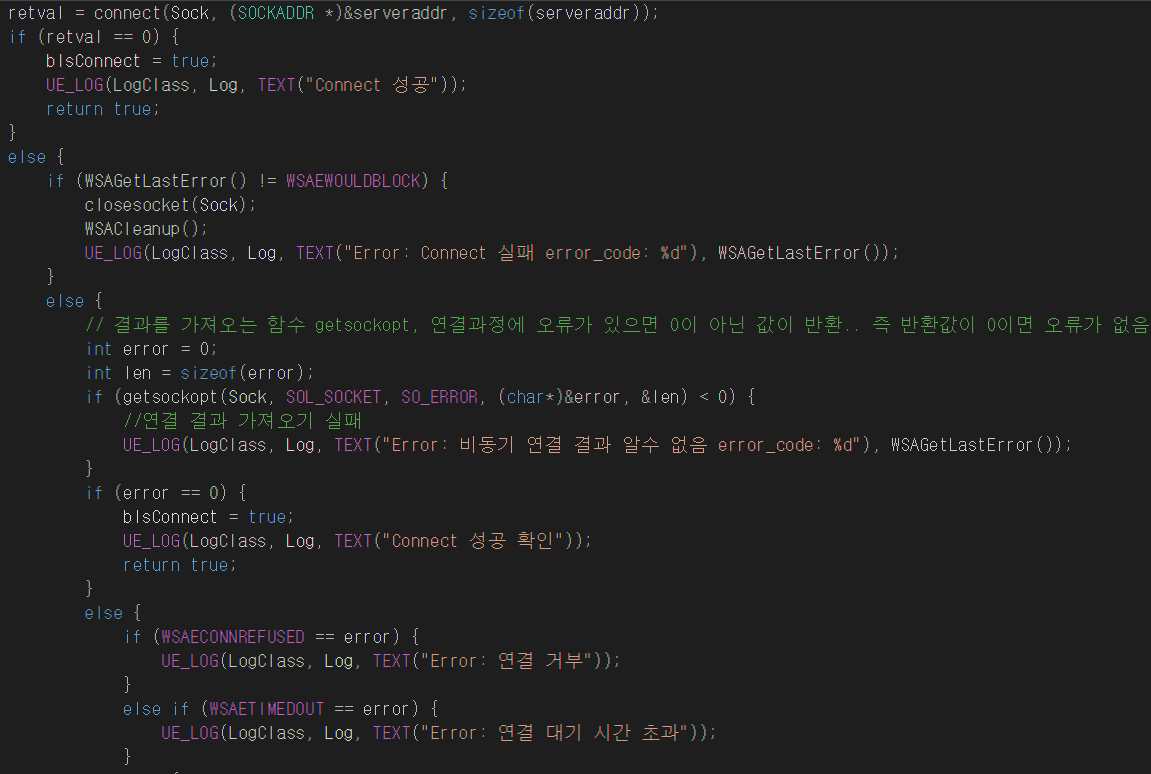
1. 시스템 구조:

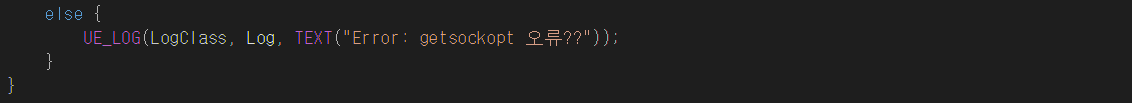
언리얼4에서 윈도우소켓을 활용한 클라이언트 및 IOCP를 활용한 비동기 TCP 게임 서버 제작.  
서버의 프레임 워크는 **‘백명이 접속 가능한 2D MMO RPG게임 서버’** 와 비슷합니다.  
구조적으로 10개의 룸이 존재하며, 룸은 기본적으로 최대 10명의 유저를 받을 수 있습니다.  
룸은 클라이언트에서 볼 수 없으며, 게임 시작 시 서버로 연결되며, 자동으로 게임이 시작되지 않은 룸으로 이동합니다. 게임이 시작 된 룸은 접근이 불가능합니다.  
NPC는 유저가 일정 거리에 가까이 접근 시 도망가며, 멀어질 시 유저에 접근하여 공격합니다.  
유저의 hp는 10이며, NPC의 hp는 3입니다. 포탄의 공격력은 1입니다.

1. 클라이언트 상세 설명:
2. AMainServer 클래스

해당 클래스는 윈도우 서버로 접속하여 통신을 하기 위한 클래스 입니다. 향후 다른 프로젝트에서도 재사용이 가능하도록 제작하였습니다. Connect, Send, Recv, 오류 출력 기능이 있습니다.

1. ATankGameGameModeBase 클래스
   1. 약 3프레임마다 AMainServer::Recv() 함수를 실행하며, 패킷 해석을 진행합니다.
   2. GameInstance를 사용하여 GameModeBase가 재 시작 되더라도 게임 내에 필요한 변수들이 초기화 되는 것을 막았습니다.
   3. 클래스의 함수와 변수들에 UFUNCTION와 UPROPERTY를 사용하여 C++코드가 아닌 블루프린트에서도 접근 가능하도록 하여, 작업의 효율을 높였습니다
2. 넌블러킹 연결 Connect.
   1. Connect함수는 블록킹 함수입니다. 그렇기에 서버가 미 오픈 시, Connect함수를 호출하면 게임이 멈춰버리는 현상이 발생합니다. 그래서 Connect함수를 호출 하기 전에 소켓을 논블록킹소켓으로 소켓으로 설정하여, 논블록으로 만들었습니다.
   2. 논블록으로 Connect를 실행 시, 서버와 연결이 성공 혹은 실패인지 알 수 없던 문제가 있습니다. 그렇기 때문에 getsockopt함수를 사용하여 연결 결과를 가져와서, 오류가 없을 시, 서버와 연결되었음을 판단하여 진행하였습니다.





제작하면서 어려웠던 점.

1. **언리얼 맵 정보를 받아 활용하기**
   1. NPC의 경사진 곳에서 이동속도를 제어하기 위하여, 언리얼 클라이언트에서 제작한 맵의 지형 높이를 추출을 시도하였습니다. 맵의 높이 정보를 .RAW파일로 추출하고, 파일 정보가 16bit로 되어있는 것까지는 확인했습니다. 하지만 해당 정보를 읽는데 실패하여 아쉬웠습니다.
2. 맵을 변경 시, GameModeBase의 변수들이 전부 초기화 되고, 다시 실행하는 문제가 발생하였습니다. 처음 이 문제를 접하였을 때, 서버로 Recv하는 것 또한 멈추어서, 서버에 버그가 있는 것으로 착각하여, 오랜 시간 고생하였습니다. 삽질의 결과로 GameModeBase가 다시 시작된다는 것을 알고 GameInstance를 사용 할 수 있었습니다.
3. 언리얼4.14버전에 Winsock2.h 등 윈도우 헤더를 사용 못하게 한 문제.
   1. 처음에는 언리얼이 윈도우 헤더의 변수를 인식 못하는 것으로 판단하여, 필요한 헤더를 추가하고 변경해주는 작업을 하다가 작업량이 너무 많아 다른 방법을 찾던 중, Winsock2.h만 인식하지 못하는 것이 아니라, 다른 윈도우 헤더도 인식하지 못하는 것을 찾고 그와 관련한 해결방법을 찾다가 간단하게 마크로를 사용하여 해결하는 방법을 찾았습니다.