PORTFOLIO

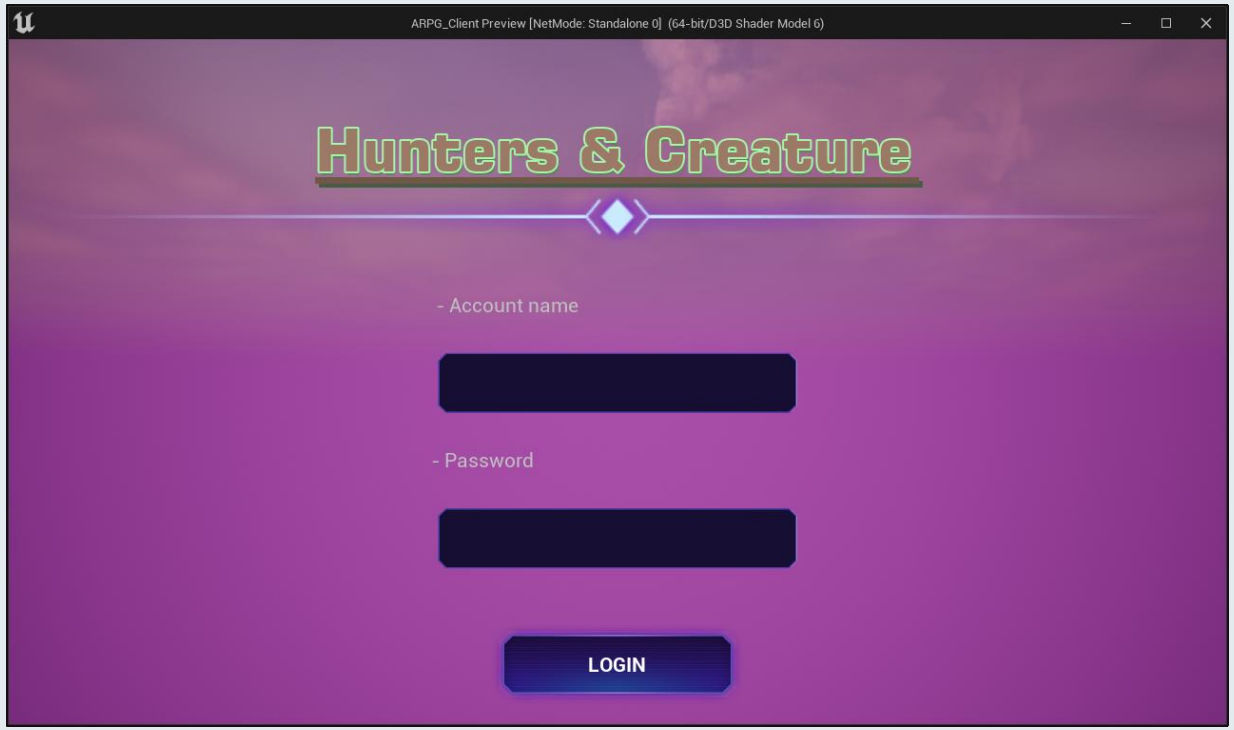
배준수

목차

[PORTFOLIO] 배준수

Hunters & Creature 프로젝트

C++서버와 언리얼로 개발한 MMORPG Shooting 장르의 게임입니다,



▶ 프로젝트 소개

기간	2024.07 ~ 2024.10
인원	1명(서버, 클라이언트)
개발 환경, 개발 툴	Windows visual studio, rider, Mssql, UE5
언어	C++
라이브러리	google::protobuf, Boost::Json
장르	MMORPG, Shooting

▶ Github URL

서버 https://github.com/qornwh/MMO_GameServer

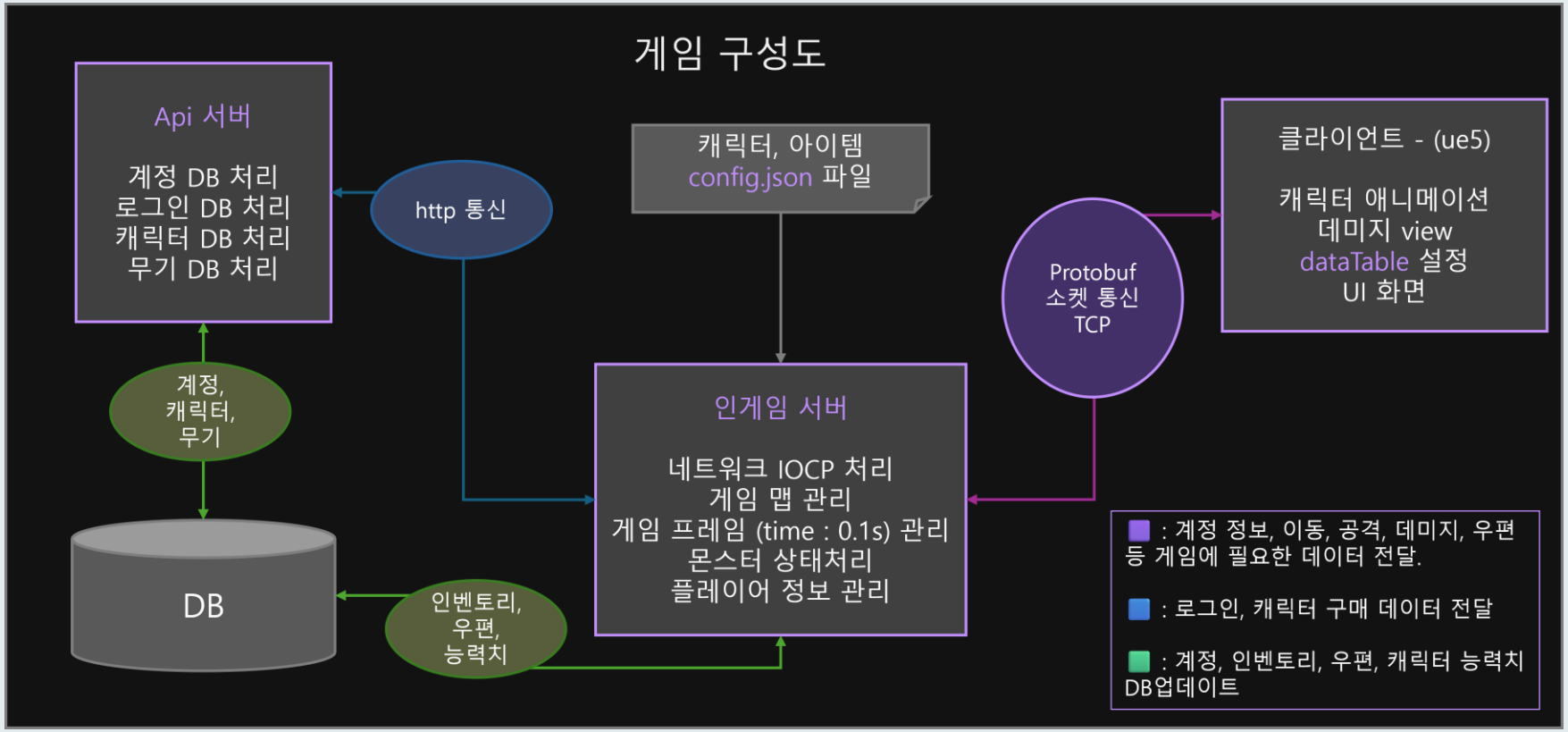
클라 https://github.com/qornwh/MMO_GameClient

▶ 영상 URI https://youtu.be/Z_v4MGjkICA

Hunters & Creature 프로젝트

게임 구성도

전반적인 게임 구성은
인게임 서버 + api 서버 + 클라이언트 + config파일로 구성했습니다.



인게임, api(계정, 로그인 작업등) 서버 분리

– 로그인, 결제등은 인게임에서의 실시간으로 처리되는 작업들과 연관이 없고, 분리해서 관리와 서버의 부담을 덜어주기 위해 분리했습니다.

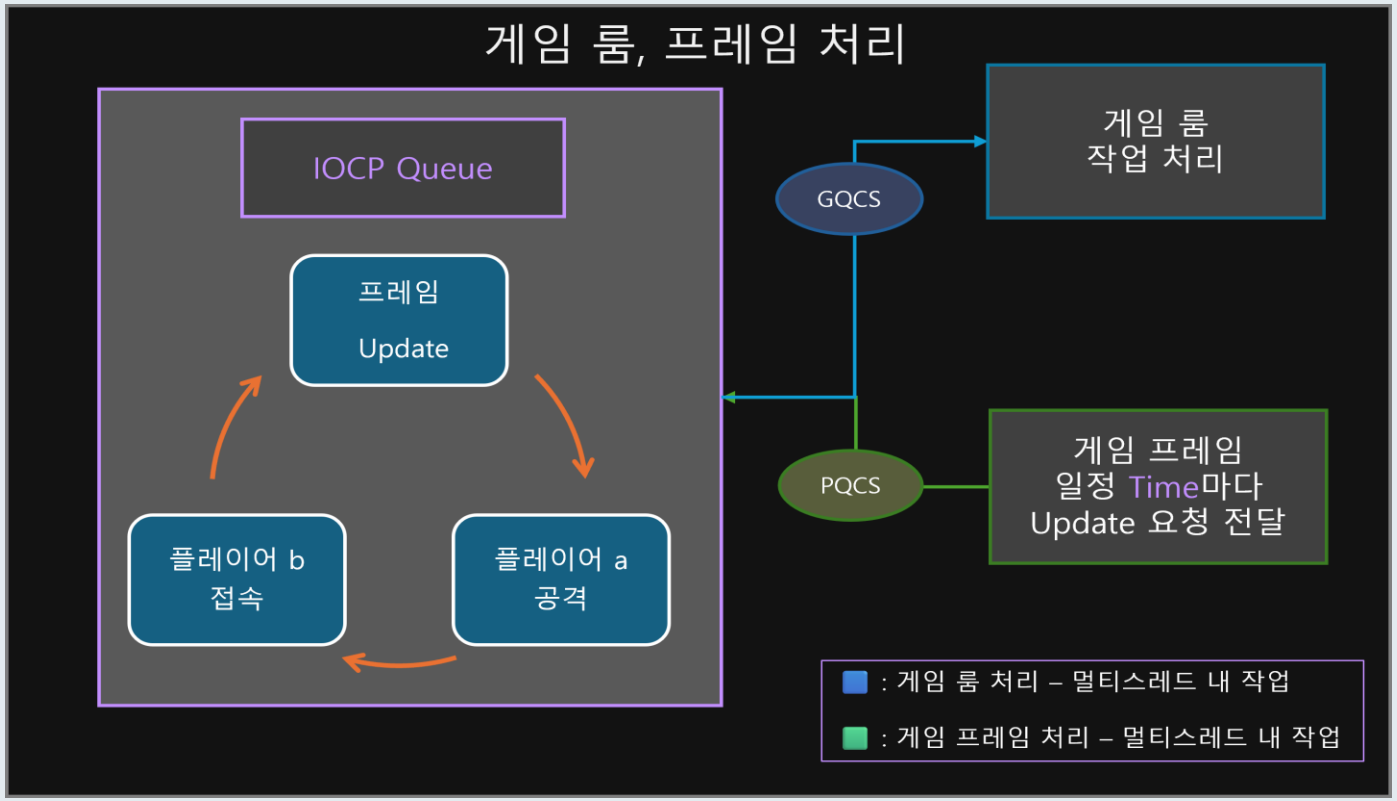
캐릭터, 아이템, 스킬 정보는 config, dataTable로 구성했습니다.

– 캐릭터, 아이템, 스킬의 수치나 정보들을 코드로 관리하면 수정 시 컴파일 시간과 프로그래머가 아니라면 변경하기 어려운 이유로 파일로 처리했습니다.

Hunters & Creature 프로젝트

인게임 내에서 주요 서버 처리

인게임 서버 처리는
클라이언트의 요청(이동, 스킬, 아이템판매등)과 서버 업데이트(몬스터 상태, 충돌판정등) 2의 작업을 처리합니다.



클라이언트 요청(게임 룸으로 작업 전달)

- 플레이어 접속, 이동, 채팅, 공격 등 수신 작업 처리 (네트워크 작업용 IOCP사용)
- 수신된 패킷 내용 처리를 순서대로 1개씩 처리하기 위해 게임룸에서 사용하는 IOCP핸들에 작업을 추가

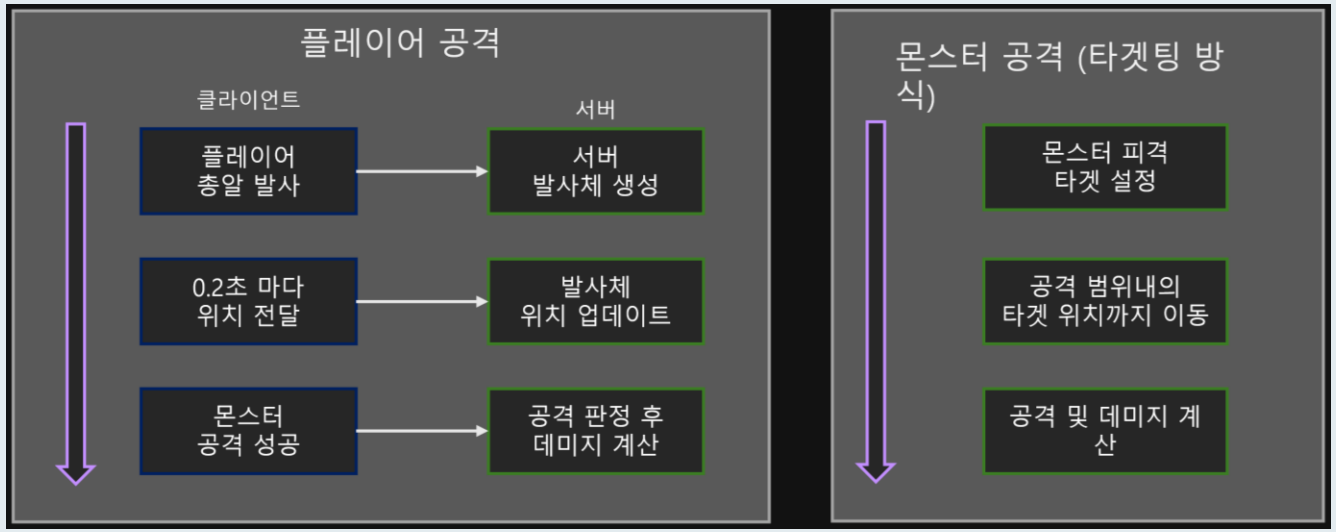
서버 업데이트(게임 룸의 프레임 업데이트)

- 게임룸 IOCP 핸들내의 플레이어가 요청한 작업 처리
- 몬스터 공격, 이동 상태들은 PQCS로 frameTime마다 한 번씩 추가 (멀티스레드에서 다른 lock을 사용하는 것 보다, 해당 방법이 더 관리에 편하므로 이 방법을 선택 코드 링크)

Hunters & Creature 프로젝트

서버 물리 정합성 체크

서버 물리 처리는
클라이언트에서 공격판정에 대한 변조를 막기 위해, 몬스터의 공격을 위한 기능입니다.



서버의 물리처리에 대한 문제

1. 서버에서 도트 데미지를 구현해야 되는 문제
 - 일정 범위 내에 도트 데미지를 주어야 하는 상황이 있어, 캡슐형태의 충돌체를 구현으로 해결

2. 서버에서 총알 위치를 업데이트 시 패킷 크기 최적화
 - 총알, 충돌체를 업데이트 하기 위해서는 스킬 코드, 몇 번째 인가, 방향, 위치 정보가 너무 많음
 - 간략화 하기 위해 공격의 ID값을 만들고 위치만 서버로 보내고, 움직이는 총알의 경우 캡슐을 생성해 충돌하는 방향으로 진행
 - 결론은 패킷 크기 감소, 간략한 캡슐 충돌 구현으로 최적화

서버 충돌판정 코드 링크

Hunters & Creature 프로젝트

인벤토리, 우편

인벤토리 – 위치, 장비아이템 – 유니크ID, 우편 – 우편ID를 기준으로 구조를 잡았습니다.

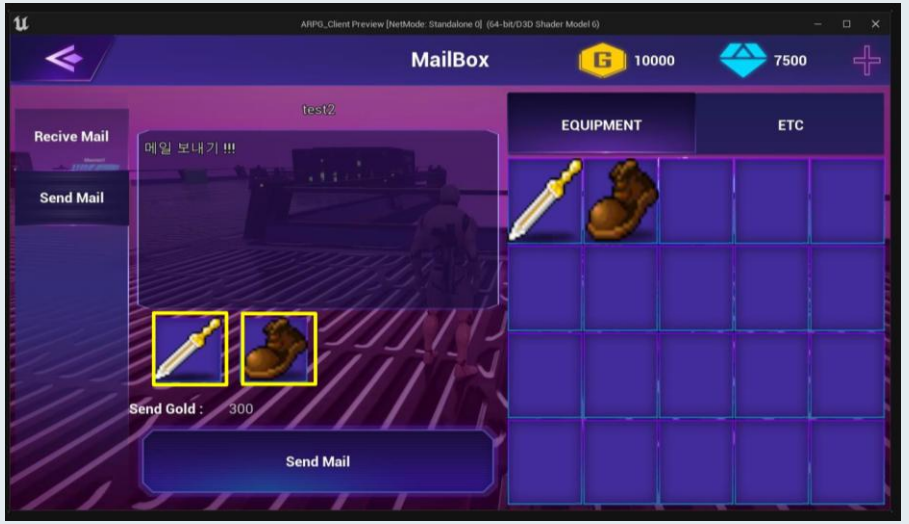
– 인벤토리의 위치가 아이템소켓, 이 위치를 기준으로 판매/획득/우편 첨부되는 방식으로 구현
– 우편은 우편ID가 있고, 우편ID를 참조하는 우편 아이템을 수령/삭제되는 방식으로 구현

인벤토리 구현하기 까지의 문제

1. **장비아이템은 유일하게 1개만 존재해야 됨**
– 초기에는 모두 아이템ID로 관리하다 보니 아이템들이 1개씩 쌓이는 문제가 발생
– 이걸 해결하기 위해 유니크한 ID값이 필요하니, UUID생성 함수를 통해 문제 해결

2. **인벤토리의 위치(아이템소켓) 또한 1개만 존재해야 됨**
– 초기에는 인벤토리에 아이템 판매/획득 시, 맨 끝 인벤토리에 넣어 중간중간에 비는 소켓이 생기는 문제가 발생
– 이를 해결하기 위해 판매/첨부/획득의 작업은 인벤토리의 위치를 기준으로 잡아야 되고, 위치를 기준으로 문제 해결

인벤토리, 우편 추가 내용 github wiki 링크



실제 인게임 화면
아이템 소켓

Hunters & Creature 프로젝트

DB, API서버, 클라이언트 내용

API 서버

- 모든 api는 필요한 파라미터, 응답 데이터들의 변경에 용이하도록 post(body)에 구성
- 2가지의 다른 테이블의 쿼리가 동시에 성공적으로 작동해야 하는 경우 트랜잭션으로 처리 (코드 링크)

C++ 서버와 연동

- 게임서버에서 API서버에 요청하기 위해 cpr라이브러리 사용
- 응답/요청 가독성을 위해 DTO클래스를 구현
- json read/write을 위한 기능 추가

API서버 상세 내용 github wiki 링크

클라이언트

캐릭터, 아이템, 스킬등 관리를 위해 DataTable 사용한 구현
서버와 네트워크 통신(패킷 처리시 protobuf 사용)
메모리 관리를 위해 스마트포인터 사용

클라이언트 코드/상세 내용 github wiki 링크

DB 커넥션 풀링

- 메모리 관리를 위해 커넥션은 shared_ptr로 만들었고, 레퍼런스 카운트가 0이면 다시 pool에 쌓는 구조로 구현

bindParam, bindCol함수 래핑

- int, char, wchar등 많이 사용되는 자료구조에 맞춰서 모든 함수들을 가독성을 위해 래핑 작업 진행

DB 처리 코드/상세 내용 github wiki 링크

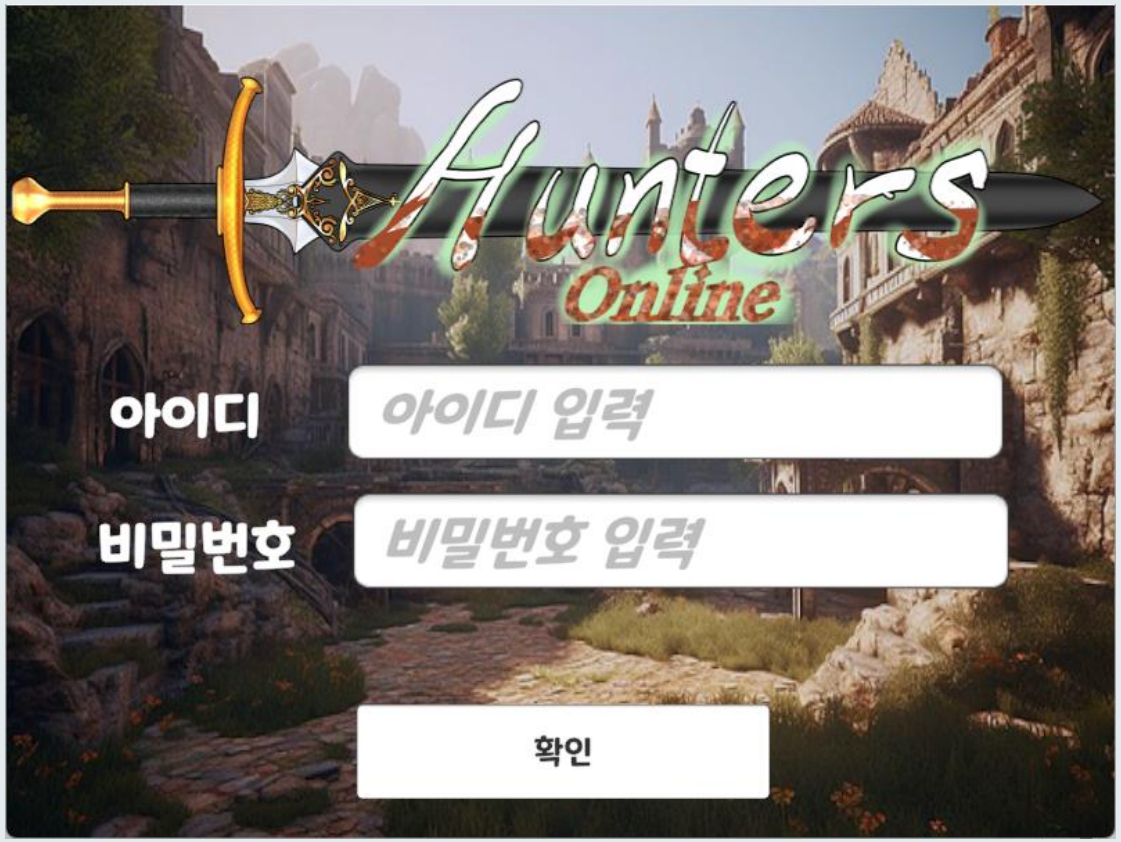
네트워크 구성 블로그 링크

더미 클라이언트 테스트 링크

[PORTFOLIO] 배준수

Hunters Online

C++서버와 유니티로 2명이서 개발한 MMORPG 장르의 게임입니다,



프로젝트 소개

기간	2023.12 ~ 2023.05
인원	2명(서버 담당, 클라이언트 네트워크 코드 담당)
개발 환경, 개발 툴, 언어	Windows, visual studio, rider, Mssql C++(IOCP, Boost::Asio), Unity, google::protobuf
장르	MMORPG

Github URL

서버 https://github.com/qornwh/GameServerProject

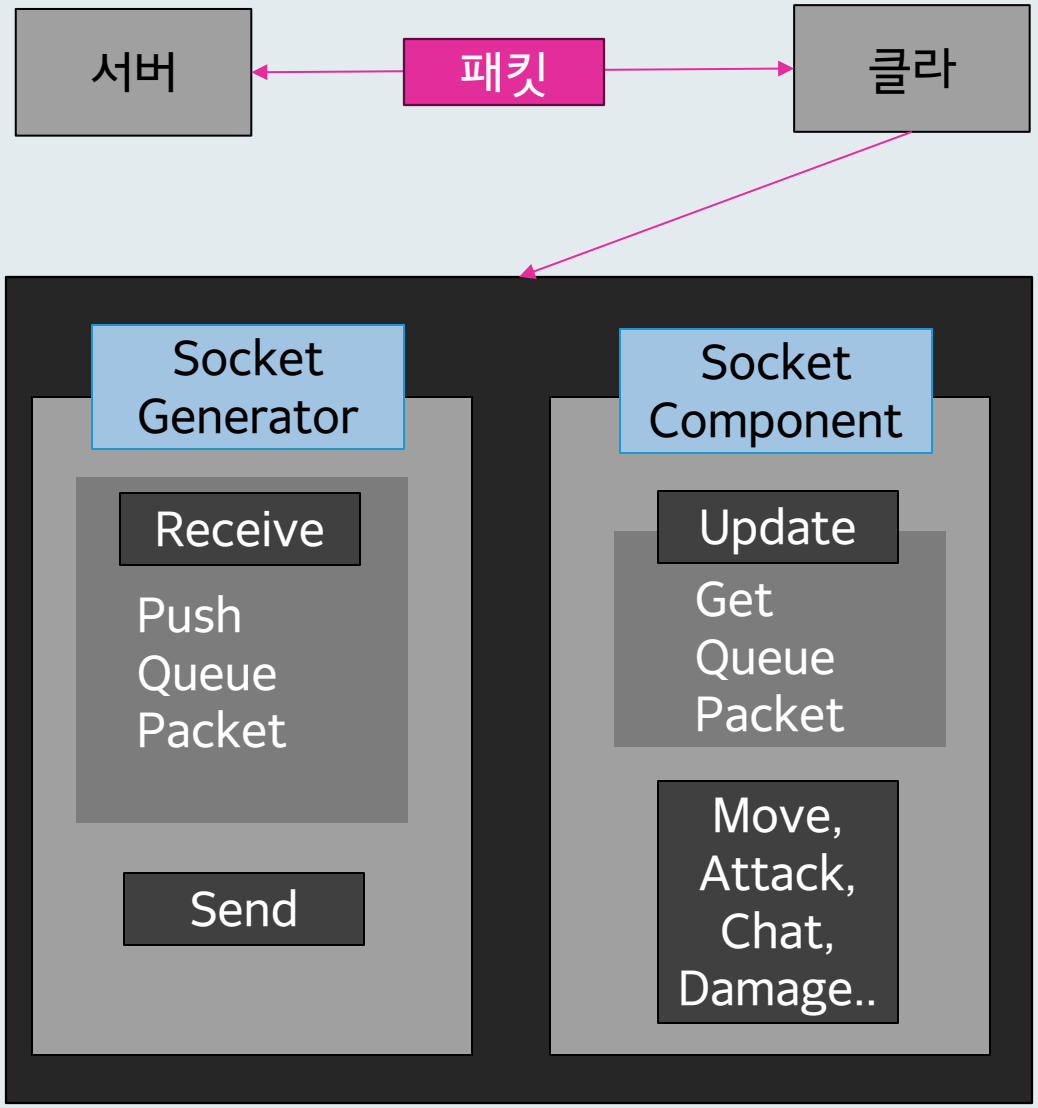
클라 https://github.com/Theta08/RpgProject

영상 URI

https://youtu.be/flfkXGUXLOc

Hunters & Creature 프로젝트

서버-유니티 클라이언트 통신



클라이언트의 통신

1. 월드 입장 시 SocketGenerator에서 월드에 통신담당의 SocketComponent를 배치
2. 패킷 일기/쓰기를 위해 protobuf로 직렬화 역직렬화
3. C#에서 지원되는 비동기 구조의 소켓모델의 SocketAsyncEventArgs사용
4. 패킷 수신할 때 프레임 순서를 맞추기 위해 처리할 패킷을 queue에 담고 SocketComponent가 update호출 시 모든 패킷 처리

클라이언트 비동기 처리 함수

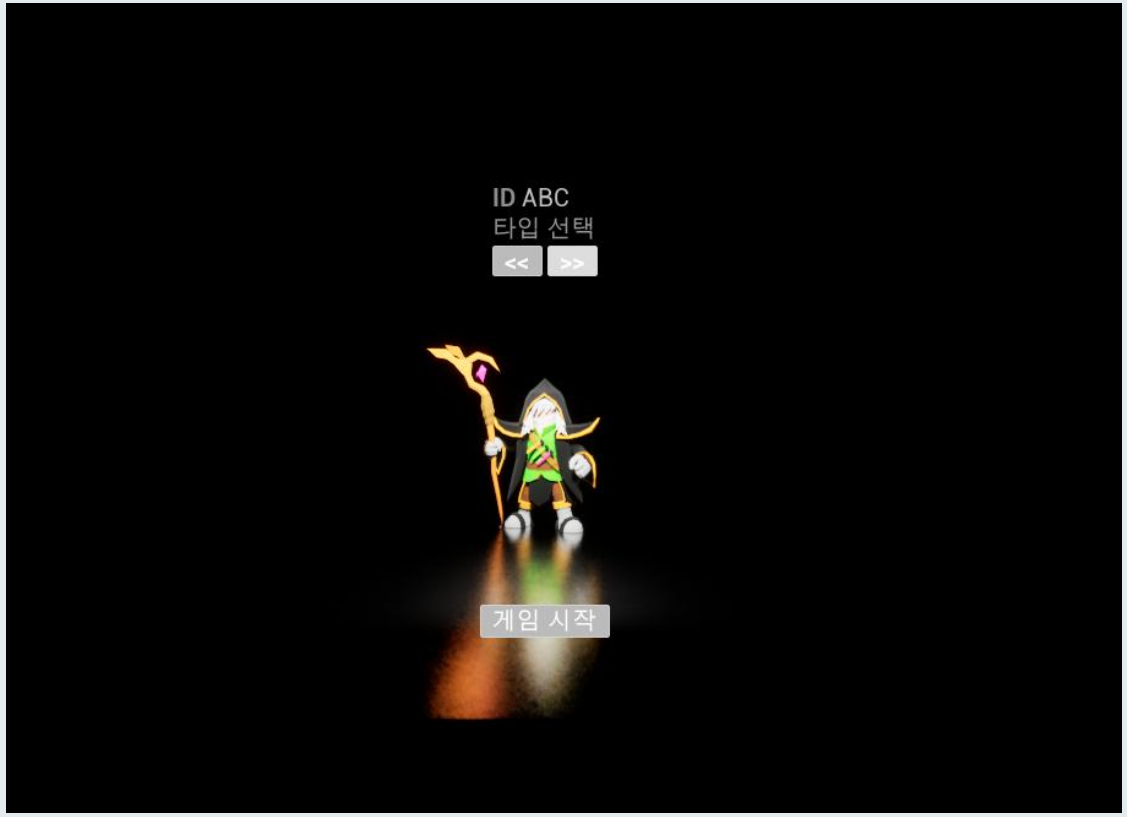
```
_socket.BeginReceive(_recvBuffer.GetBuffer(), _recvBuffer.ReadPos(), _recvBuffer.FreeSize(),
    SocketFlags.None,
    new AsyncCallback(OnReciveHandler),
    null);
```

```
public void Send(IMessage msg, UInt16 id)
{
    byte[] buffer = PacketHandler.MakePacketHandler(msg, id);
    _socket.BeginSend(buffer, 0, buffer.Length, SocketFlags.None, new AsyncCallback(OnSendHandler), null);
}
```

[PORTFOLIO] 배준수

MMORPG Simulation

C++서버와 언리얼로 혼자 개발한 MMORPG 시뮬레이션 입니다.



▶ 프로젝트 소개

기간	2023.11 ~ 2024.3
인원	1명(서버 담당, 클라이언트 담당)
개발 환경, 개발 툴, 언어	Linux(Ubuntu), visual studio, vscode C++(Epoll), Unreal, google::protobuf
장르	MMORPG

▶ Github URL

서버 https://github.com/qornwh/BSGameServer

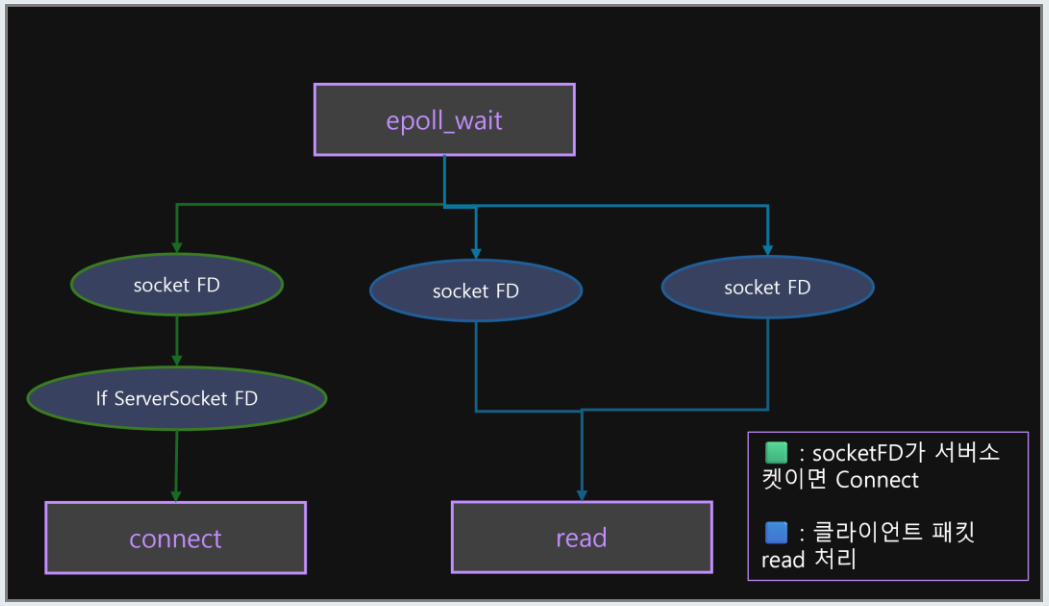
클라 https://github.com/qornwh/BSGameClientUE5

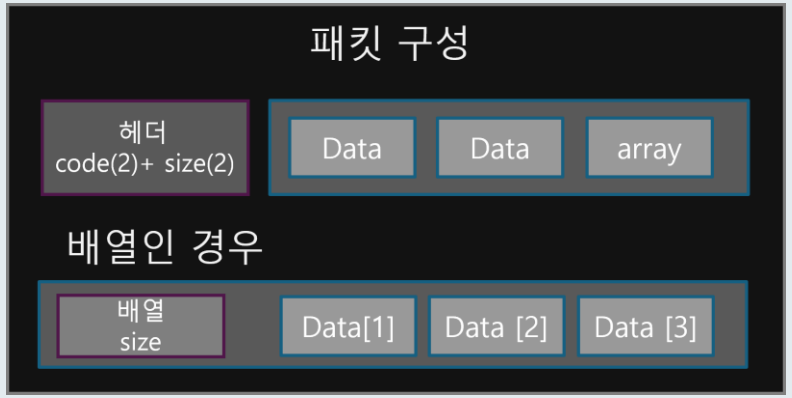
▶ 영상 URI

https://youtu.be/NIEy5bpEhBk?si=LbHDJnopb3nIc4fo

MMORPG Simulation 프로젝트

리눅스 서버 구성





리눅스 서버(Epoll) 통신

1. epoll_wait로 socket FD로 연결/패킷수신 판단.
2. serverSocket일 경우 conn/ 아닐경우 read

- 패킷 헤더(code 2바이트 + size 2바이트 => 4바이트) + 데이터로 구성
- 배열과 문자열을 처리하기 위해서 size+내용을 한 쌍으로 구현

파싱 진행
- 패킷 헤더 길이 확인(최소 4byte 이상)
- 패킷 헤더의 size의 길이와 수신된 패킷 길이 확인
- 패킷 코드에 맞는 클래스로 역직렬화 (바이트 위치를 position만큼 늘리면서 진행)

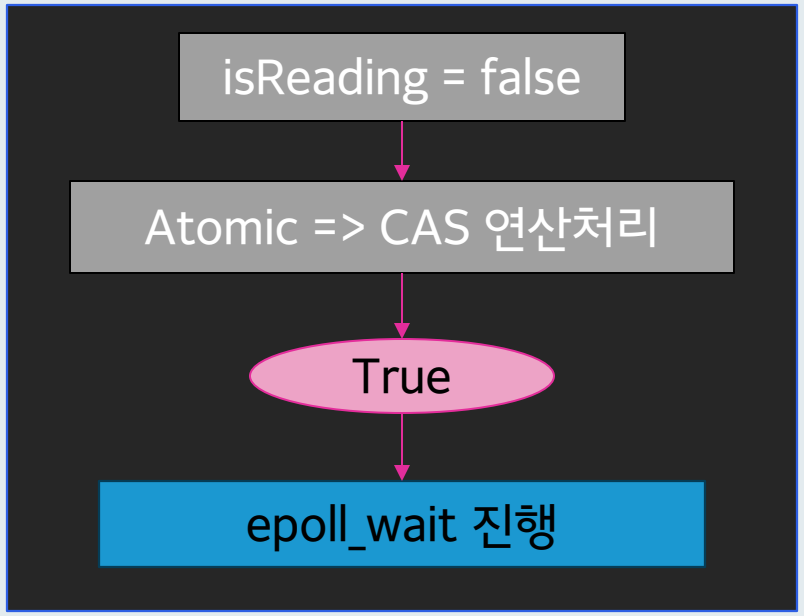
MMORPG Simulation 프로젝트

리눅스 서버 문제 해결

해당 프로젝트에서 발생된 문제는 FD의 변경감지가 될 때 입니다.

원인 : 멀티스레드에서 epoll_wait로 접근할 때 2개이상의 동일한 스레드가 socketFD에 접근할 경우 thread-safe하지 못한다.

해결 : 해당 접근 시 thread-safe하도록 만들기 위해 atomic으로 lock-free한 구조를 만들어 해결



서버에서 epoll_wait에서 변경감지된 socketFD들이 있으면 Connect/Read하는 코드입니다.

```
void ServerService::Dispatch()
{
	// atomic으로 compare_exchange_strong을 한 이유.
	// 멀티스레드에서 동시에 읽어오는 이슈가 있어서 사용. fd읽는 도중에 또 같은 fd를 읽어버림 !!!
	int isReading = _reading;
	if (!_reading && CheckReading(isReading))
	{
		int eventCount = EventCount();

		ASSERT_CRASH(eventCount > -1);

		for (int i = 0; i < eventCount; i++)
		{
			if (isServerFd(i))
			{
				AcceptClient(i);
			}
			else
			{
				ReadClient(i);
			}
		}
		OffReading();
	}
}
```