1. 멀티 플레이어 접속에 대한 로직과 패킷 전송 방법 구상

고려해야 할 점

1. 방금 접속한 클라이언트에게 많은 클라이언트 정보를 줄 수 있다.
2. 방금 접속한 클라이언트와 기존에 서버에 있는 클라이언트의 데이터가 달라야 한다.

(방금 접속한 클라이언트는 모든 클라이언트 정보, 나머지는 방금 들어온 클라이언트 정보)

1. 서버에서 보내는 다른 클라이언트의 연결에 대한 정보가 중복하여 들어오지 않도록 한다.

고려해야 할 점 해결 과정

1번.

방금 접속한 클라이언트는 현재 서버에 접속된 많은 양의 클라이언트를 받을 수 있다.

이 과정에서 패킷이 잘려오는 데이터 처리, 또는 버퍼의 크기를 동적으로 할당해야 하는 필요가 있을 수 있지만 다음의 이유로 일단은 고려하지 않는다.

첫째, 우리 게임에서 한 클라이언트가 많은 클라이언트의 연결 정보를 받는 양이 많지 않다.

우리 게임에서 한 클라이언트 연결 상태에 대한 정보를 나타내는 데이터는 32바이트에서 약간의 편차로 표현할 수 있을 것이라 생각한다. 그리고 게임 씬에서 동시에 4명이 연결하여 진행하기 때문에 사실 상 받아올 클라이언트의 연결 정보는 크기 않다는 것을 알 수 있다.

둘째, 이 과정이 현재 필수적인 것은 아니다.

데이터가 잘려오는 경우 이를 표시해야 할 데이터가 추가되어야 하고 잘린 데이터를 담아뒀다가 나중에 처리하는 코드도 추가해야 한다. 이 과정에서 데이터를 최대한 압축해서 보내는 장점이 있지만 지금은 자원 문제가 일어나지 않는 상태이며 드라마틱한 효과를 보진 못한다고 생각하고 코드도 복잡하게 될 것이다.

지금은 시간을 효율적으로 사용하여 좋은 작품을 내는 것을 목적으로 하고 있다. 후에 필요할 시기가 오면 그때 처리해도 문제가 없는 부분이라 생각하기에 지금은 중요한 일부터 하는 것이 먼저라 생각한다.

2번.

1번의 내용의 결론으로, 방금 접속한 클라이언트에게 서버가 연관된 클라이언트를 찾아 모두 보내주면 된다. 이 과정에서 클라이언트 개수, 각 클라이언트의 정보 등 포함된다.

다른 클라이언트 들에게는 방금 접속한 클라이언트 정보 하나가 전달된다.

클라이언트 A – 다른 클라이언트

클라이언트 B – 방금 접속한 클라이언트

텍스트이(가) 표시된 사진

자동 생성된 설명

서버 -> 클라이언트 A

방금 접속한 클라이언트의 ID를 담아 보낸다.

텍스트이(가) 표시된 사진

자동 생성된 설명

서버 -> 클라이언트 B

위의 버퍼에 클라이언트 수와 ID들을 담아 보낸다.

텍스트이(가) 표시된 사진

자동 생성된 설명

클라이언트 B

받은 데이터에서 클라이언트 수와 ID들을 이용해 새로운 플레이어를 만든다.

3번.

2번과 내용이 올바르게 실행되면 클라이언트 정보가 중복되지 않는다.

1. 콘솔 프로젝트 멀티 스레드 IOCP 서버 – (1) Data Race 구분

Data Race의 후보들을 미리 정해두고 이를 이벤트나 뮤텍스 등으로 여러 스레드에서 동시에 접근하지 못하도록 막아야 한다.

No Data Race의 조건

1. 둘 이상의 스레드가 동시에 접근하지 않는 데이터이거나
2. 데이터가 한 번 값을 갖고 변경되지 않는 경우

크게 두 가지로 보았다.

그 조건에 부합하지 않는 데이터는 Data Race로 판단한다.

전역 변수



Data Race

클라이언트 생성, 삭제나 데이터를 처리할 때 여러 스레드에서 동시에 접근할 수 있다.

RemoteClient 클래스



Data Race

Thread 주소는 많은 스레드에서 접근하는 것 자체가 위험하다고 판단한다.

따라서 접근한다 해도 메인 스레드에서 종료를 기다리는 작업 등의 수준으로 접근해야 한다.



Socket 클래스에 멤버 변수마다 경우가 다르기 때문에 따로 설명이 필요하다.



Data Race

한 스레드에서 이벤트가 들어온 클라이언트가 다른 클라이언트의 플레이어에 접근할 수 있고 이는 다른 작업과 동시에 접근할 수 있다.

Socket 클래스



Date Race

i/o 이벤트를 확인할 때와 데이터 처리를 위해 진행 중이 아니라고 표시할 때 이 변수가 동시에 접근이 될 수 있다.





No Data Race

위의 모든 변수들은 초기 단계에서 한 번 값이 적히고 변경하지 않거나



No Data Race이지만 경우에 따라 Data Race 일 수 있음

멀티 스레드에서 클라이언트 이벤트 정보가 여러 스레드에 분산되지 않는다면 위의 변수들은 Data Race가 아니다.

또한 i/o 진행중이라는 것을 위의 m\_isReadOverlapped 변수로 확인하기 때문에 이 변수에서 컨텍스트 스위치에 관한 문제가 일어날 가능성은 적다.

후에 Data Race의 lock 관련하여 교착 상태(Deadlock)과 기아 상태(Starvation)가 일어나지 않도록 주의하며 작업해야 하고 본격적으로 콘솔 프로젝트를 멀티 스레드 IOCP로 만들어 문제점을 확인해야 한다.