1. 패킷 전송 오류 수정

서버 -> 클라이언트



Socket 클래스의 Overlapped I/O Send를 진행하는 함수

텍스트이(가) 표시된 사진

자동 생성된 설명

수정 전

리스트를 for문으로 loop하여 필요한 객체를 찾는다.

이전엔 처음 메시지를 보낼 때 for문 안에 있는 if문이 객체의 NULL로 인해 알 수 없는 값이 전송되었다.

텍스트, 스크린샷, 화면이(가) 표시된 사진

자동 생성된 설명

수정 후

Iterator를 이용하기 때문에 기존의 문제가 해결되며

C++ Standard Algorithm로 효율적이고 신뢰성 있는 방법으로 loop하고 작업한다.

클라이언트의 처리



서버에 자신과 관련된 플레이어 리스트를 요청할 때

텍스트이(가) 표시된 사진

자동 생성된 설명

수정 전

Memcpy()의 인자를 거꾸로 놓았었다.. 사소한 실수 하나에 큰 오류가 있었던 것

또한 r\_id를 플레이어 리스트에 추가해야 하는데 받아온 본인 id를 추가하고 있다.

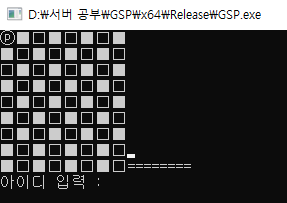
텍스트이(가) 표시된 사진

자동 생성된 설명

수정 후

Memcpy()의 값을 올바르게 바꾸고 혹시나 다른 id가 요청한 메시지로 들어오면 차단하게 끔 본인의 id일 경우로 if문을 추가했다.

1. 멀티 스레드로 변환 전 싱글 스레드로 구현한 콘솔 프로젝트 소개



로그인

아이디를 입력하면 서버에 접속하여 서버에 클라이언트 리스트에 접속한 클라이언트가 추가된다.

빠른 작업을 위해 서버 IP는 Local Loop으로 설정해둔 상태이다.(127.0.0.1)



클라이언트 진행

여러 클라이언트가 접속하게 되면 서로의 접속을 확인하고 각자 관리하는 플레이어 리스트에 추가되며 ‘말의 이동’과 같은 데이터 메시지가 들어오면 플레이어 리스트에서 맞는 플레이어를 확인하고 시행하게 된다.

클라이언트가 올바른 메시지를 주고 받는지 확인하기 위해 서버에선 송신할 데이터 정보를 출력하고 클라이언트는 최근에 받은 데이터의 ID를 출력한다.

후에 간단한 연산들은 서버에서 시행하게 하고 충돌처리와 같은 복잡할 수 있는 연산들은 클라이언트에서 연산한 뒤 서버로 결과를 전송하여 서버는 결과를 취합해 가장 적합한 결과를 조정하는 식으로 진행할 예정이다.

현재는 가벼운 클라이언트이기 때문에 서버에서 연산을 담당하고 있다.

1. 멀티 스레드 생성/종료

텍스트, 오렌지, 스크린샷이(가) 표시된 사진

자동 생성된 설명

N\_THREAD – 생성 할 스레드 수

Worker\_threads – 스레드 포인터들을 관리하는 vector 컨테이너

Worker\_Thread() – 스레드가 loop하며 수행 할 작업들 (Accept, Recv/Send)

스레드를 N\_THREAD만큼 생성하고 프로그램을 종료하기 전 오류가 나지 않게 모든 서브 스레드의 종료를 확인하고 메인 스레드를 종료한다.

1. C++ 뮤텍스, 잠금 생성(lock\_guard)

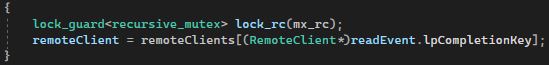
경쟁 상태를 막기위한 동기화 방법 중 recursive\_mutex를 이용한 잠금으로 구현하였다.



mx\_rc - mutex 객체

사용 예시

remoteClients 접근



listenSocket에 Accept 이벤트 처리



lock\_guard – mutex 객체을 템플릿으로 받아 생성하면 그 객체가 소멸할 때 까지 임계영역이 생성된다.

mutex와 recursive\_mutex 차이

같은 스레드에서 한 mutex의 잠금을 여러 번 호출 할 때 mutex 객체에서는 오류가 나게 되지만 recursive\_mutex 객체를 사용하면 잠금 횟수에 관한 count를 증가시켜 중복하여 잠금을 해도 오류가 나지 않게 된다.

1. 멀티 스레드 구현 과정에서 Socket bind 오류 수정









빌드하는 과정에서 listenSocket를 bind() 할 때 에러가 나며 빌드가 안되는 상황이 있었다.

기존에는 메인 스레드 하나에서 listenSocket을 지역 변수로 두고 생성하여 사용했는데 스레드를 여러 개로 만들어 사용할 때는 하나의 listenSocket만 만들기 위해 전역 변수로 두고 메인 스레드에서 생성해야 했다. 하지만 그 과정에서 왜인지 모르게 bind()가 실행되지 않는 상황이 일어났고

1. 생성자가 제대로 불러지지 않는다.
2. ListenSocket이 데이터(Data) 영역에 생성되면 안된다.
3. 다른 스레드에서 모종의 이유(불필요한 복사 등)로 불러올 수 없다.

그 원인으로 위의 3가지를 추측했다.

해결 방안은 전역 변수를 포인터 형식으로 두고 소켓을 동적 할당하여 포인터로 통신하는 방법을 사용했고 안전한 해제를 위해 스마트 포인터로 구현했다.