1. Keyboard Input 코드 작성

클라이언트마다 키 입력의 상태를 저장하는 것이 필요하다.

텍스트, 오렌지이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

Old\_keys – 이전 프레임의 키 상태를 가져옴

Present-keys – 현재 프레임의 키 상태

Keys

﹂KeyUp – 키가 떼진 프레임 하나에서 True

﹂KeyDown – 키가 눌린 프레임 하나에서 True

﹂time\_Elapsed – 키가 눌러지는 시간이 저장되고 일정 값 이상 저장하지 않도록 한다.

Key상태를 이용하여 Input 기능 추가

텍스트이(가) 표시된 사진

자동 생성된 설명

KeyDown, KeyUp – 현재 키와 이전 프레임의 키를 비교하여 값 변경

t 값에는 게임 프레임워크에서 elapsed를 넘겨주어 게임 시간으로 넣어야 한다.

키의 time\_Elpased 값이 0보다 크다는 것은 키를 누르고 있다는 의미이고 이것으로 키가 눌려 있는지, 얼마나 눌렀는지 확인 가능하다.

1. 로그인, 로비, 배틀 등 클라이언트의 현재 상황에 따른 데이터 전송 구조 및 방법 구상 – (자세한 패킷 정보는 후에 작성)

현재 필요한 상황들

※ 이후 나오는 게임 화면 사진은 기존의 졸업작품 제안서 ppt로 제작한 게임 예상 진행에서 가져온 것입니다.

1. 텍스트, 나무이(가) 표시된 사진

   자동 생성된 설명로그인 – 사용자는 닉네임을 입력하여 서버 접속

로그인의 경우 닉네임만 입력하여 서버에 접속하기 때문에 많은 정보를 요구하지 않는다.

통신할 데이터 – Accept를 위한 소켓 정보(IP, Port 정보), 입력한 닉네임, 로그인 성공 여부

1. 매치 메이킹 – 비어있는 방 중 가장 많은 플레이어가 있는 방에 참가
2. 방 접속 - 사용자 입력으로 방 접속
3. 방 생성 – IP등 클라이언트 마다의 고유번호를 이용하여 방의 정보로 포함

통신할 데이터

클라이언트 -> 서버 – 입장하려는 방 정보, 플레이어 정보, 채팅, 상태 트리거 요청

서버 -> 클라이언트 – 방 목록, 입장 성공 여부, 플레이어 정보, 채팅, 상태 트리거(걷기, 점프 등 애니메이션 트리거 포함)

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 게임 씬 – 같은 방 번호를 정보로 갖는 클라이언트끼리 게임 상태에 대한 정보를 통신 게임 종료 시 플레이어 입력에 따라 게임 방의 기본 상태(준비)로 복귀

통신할 데이터

클라이언트 -> 서버 – 플레이어 정보, 채팅, 충돌 요청, 게임 종료, 상태 트리거 요청

서버 -> 클라이언트 – 플레이어 정보, 몬스터 정보, 채팅, 모든 클라이언트 충돌 요청 취합 후 결과, 상태 트리거(걷기, 점프 등 애니메이션 트리거 포함)

텍스트, 나무, 표지판, 여러개이(가) 표시된 사진

자동 생성된 설명텍스트, 하늘, 여러개이(가) 표시된 사진

자동 생성된 설명

1. 콘솔 프로젝트 IOCP echo 서버 구현

Iocp 사용 간단 정리

 - IOCP 객체 생성

 - IOCP에 Overlapped I/O 소켓 추가

Add의 두번째 인자로는 소켓 주소가 들어가고 이벤트의 lpCompletionKey 값으로 나오게 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Accept을 걸고 Overlapped I/O 완료 대기하는 것을 표시한다.



Overlapped I/O 소켓에서 나오는 이벤트 대기 (Accept, Recv, Send로 I/O를 걸었을 때 나온다.)

I/O가 완료되면 IocpEvents에 이벤트로 저장된다.



발생한 이벤트를 가져온다.(OVERLAPPED\_ENTRY)



이벤트의 lpCompletionKey로 어떤 소켓의 I/O가 끝났는지 확인한다.

listenSocket은 nullptr을 넣었기 때문에 0이 출력되고 나머지는 소켓의 주소가 출력된다.

ListenSocket이 이벤트로 들어왔을 때



listenSocket이 이벤트로 나왔다는 것은 Accept 연결이 완료된 것이고 Win32 AcceptEx 사용법에 따라 마무리 작업을 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그 후 새 TCP 소켓을 IOCP와 클라이언트를 관리하는 컨테이너에 추가하고 소켓으로 수신할 수 있도록 Recv I/O를 건다.

그 외의 다른 소켓이 이벤트로 들어왔을 때 (Recv, Send 완료 처리)

Receive



컨테이너에서 해당하는 클라이언트를 소켓 주소를 이용하여 찾아낸다.



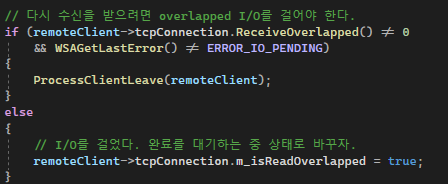
이벤트의 dwNumberOfBytesTransferred는 완료된 바이트 수를 나타낸다.

텍스트이(가) 표시된 사진

자동 생성된 설명

바이트 수가 0일 경우 연결을 해제했다는 의미이고 음수일 경우 에러가 난 상태이니 클라이언트를 해제하면 된다.

바이트 수가 양수일 때는 I/O가 올바르게 완료했다고 판단하고 받아온 데이터를 처리하여 넘겨주면 된다.



데이터를 처리하고 나서는 다시 수신하기 위해 I/O를 걸어주어야 한다.

* 현재 패킷이 잘려서 데이터 왔을 때에는 고려하지 않은 상태이다.
* 여러 패킷이 한번에 도착할 수 있는 경우는 클라이언트 선에서 잘 정리해서 보내고 있으니 서버에서는 걱정하지 않도록 작업해야 한다.

Send

텍스트이(가) 표시된 사진

자동 생성된 설명

Recv로 받아온 데이터를 처리 후 결과를 클라이언트에게 다시 보내주는 SendOverlapped I/O를 건다.

하지만, 이 과정에서 클라이언트에게 여러 번 I/O를 거는 현상이 발견되어 그 이유를 찾고 수정해야 한다. 아마 Send와 Recv 완료 처리를 아직 구분하지 못하는 부분에서 일어나는 오류라고 추측되는데 소켓 통신 과정에 원하는 정보를 찾는 방법을 더 공부해야 한다.

* 한 클라이언트에서 하나의 접속만 허용한다는 가정하에 Send에서 데이터 버퍼와 Overlapped를 하나만 갖도록 하게 했기 때문에 다중 접속을 위해서라면 이를 동적으로 할당해야 한다.
* 성능을 위해 Pool을 고려해야 한다.