**IOCP**

Iocp 를 추상 클래스화 하여 iocp 를 상속받으면 iocp로 돌아가게 끔 했습니다.

Iocp 모델을 사용하여 비동기로 동작하도록 했습니다.

완료포트를 생성하고 recv나 send가 발생했을 때 해당 동작은 백그라운드에서 돌아가고

백그라운드에서 동작이 완료되면 완료정보가 getqueuedCompletionStatus로 들어옵니다.

이때 overlap 정보가 들어오는데 overlap\_ex 로 래핑한 구조체의 첫번째 멤버변수로

소유하고 있기 때문에 이 overlap 정보를 overlap\_ex로 캐스팅하여

현재 overlap 동작의 type (recv,send,accept,disconnect) 등을 구분 하고,

연결된 Session 정보를 가져옵니다.

**Send 처리**

Send 는 한번에 여러 번 send를 할 수도 있기 때문에 (아 정확하게 기억이 안난다.)

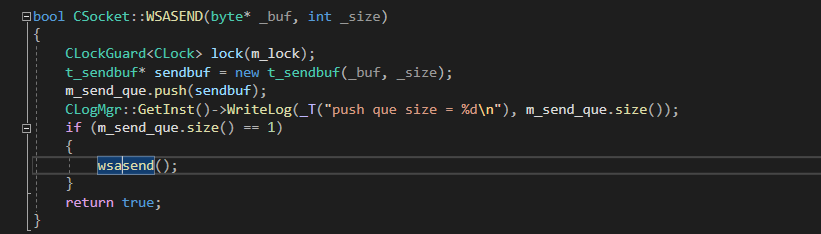
Send event 가 충돌나지 않도록 queue에 넣고 관리한다.?

아니면 순차적으로 보내려고.

만약 아직 보내는중이고 send가 완료되지 않은 패킷이 있다면 큐에 쌓고

앞에 보내던 packet의 전송이 끝나면 해당 packet을 queue에서 없애고

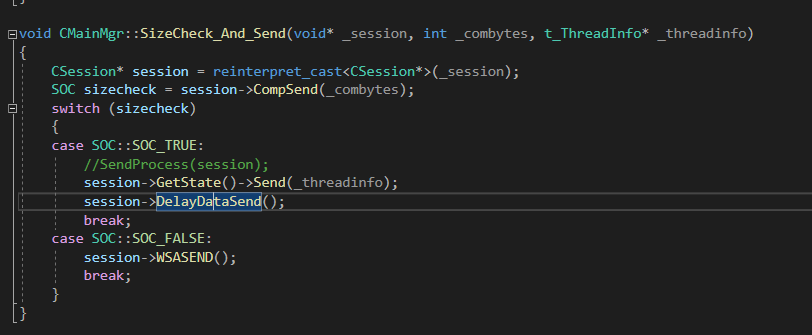
다음 packet을 전송한다.



보낼 당시에 큐에 방금 들어온게 1이면 바로 전송

아니면 걍 큐에 쌓음.

전송이 완료되었을 때

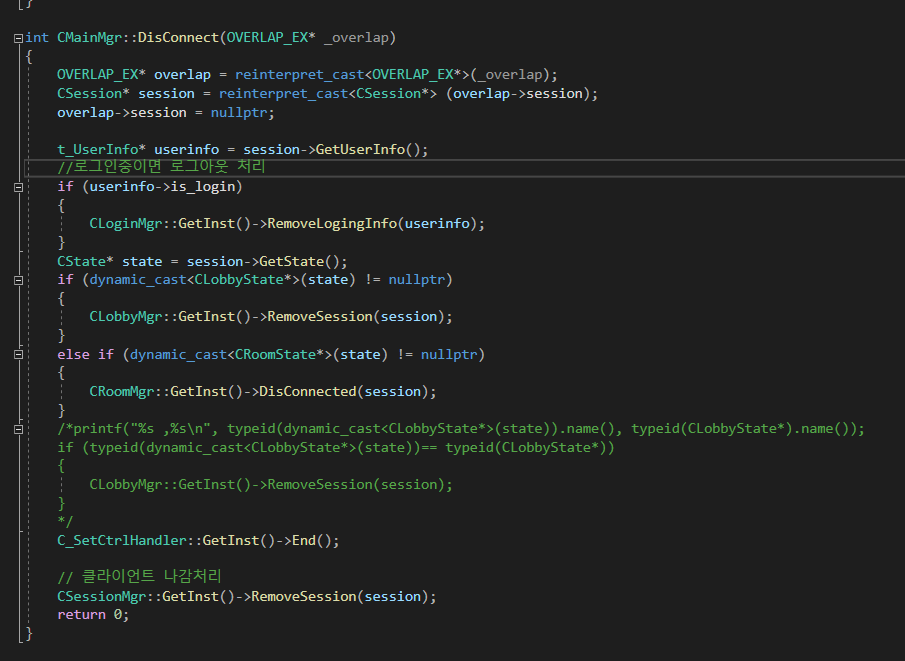


Comsend 함수에서 패킷 전송이 완료됐는지 검사 후

완료되었다면 해당 packet pop 하고

Delaydatasend 에서 다음 패킷 전송.

**강제종료 DisConnect**



**Session 구조**

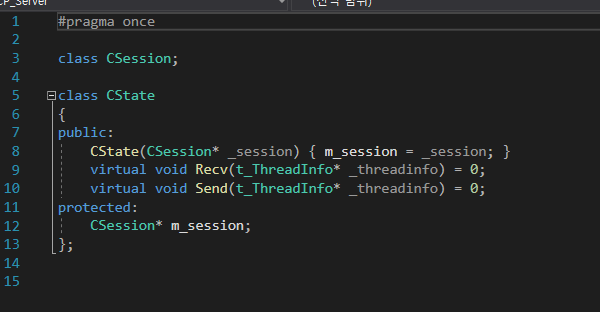
Socket 을 packet이 상속 받고 packet을 session이 상속 받음.

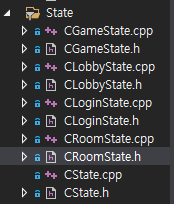
Session은 자주 생성 삭제가 일어날 객체이기 때문에

Memorypool을 상속시켜주어 메모리 재사용하도록 함.

**State 패턴**

State 패턴을 이용한 상태에 따른 다른 동작을 구현.

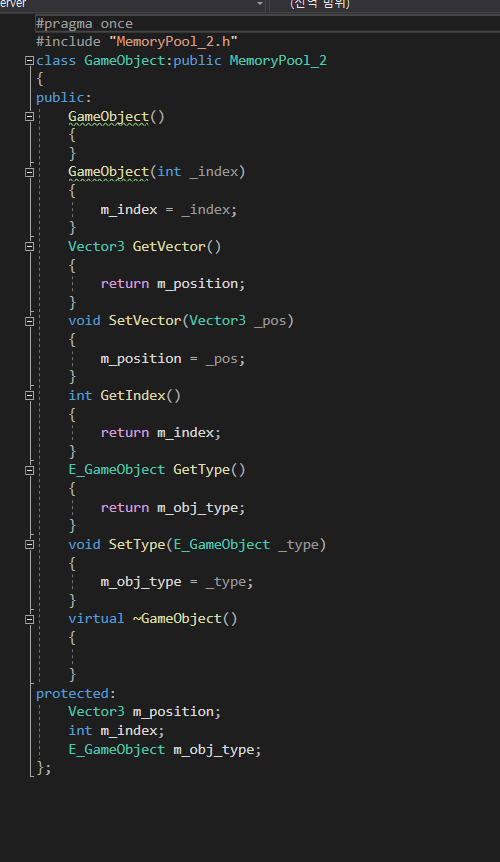




**MemoryPool**

자주 생성, 삭제되는 오브젝트들은 모두 memory pool을 상속받게 했고

Gameobject 라는 클래스를 상속 받게 했다.



Ex



+메모리풀 동작 방식 설명

**Sectormanager**

쿼드 트리로 구현 (서버에서 충돌 검사를 하게 될 것을 생각해서 만들어놨습니다.)

Vector 값으로 sector 를 찾을때는 쿼드트리 사용.

만약 sector id 값으로 찾는다면 트리 전부 순회하는것보다

Sector 값들을 vector로 들고 있고 [] 로 찾는게 더 편하고 빠를 것 같아서

둘다 해놨습니다.

현재 구조

Sector -> tile 정보 소유

Tile -> object 정보 소유

+ viewsector + viewtile 테스트 이미지 .

**LRU\_Queue 구현**

최대한 메모리 반납. 렌더링 우선순위를 정합니다.

현재 viewsector 내의 tile 정보가 들어가는데 우선순위 기준은 time입니다.

만약 capacity가 20이라면 이전 타일 10 과 새로운 타일 20일 때 (중복이 없다고 치면)

30의 렌더링 해야할 타일 정보가 쌓일텐데 time 갱신이 된지 오래된 순으로 렌더링 정보에서

제외시킵니다. 그럼 결국 큐에는 새로갱신된 20개의 타일만 존재!

**Map 구현 (red-black tree)**

**Dump**

Threadinfo 로 thread정보들을 들고 있다가 에러 발생 시

Dump 파일,log txt 파일 생성되도록 함.

서버 포트폴리오

https://developstudy.tistory.com/72