시야 처리

MM4220 게임서버 프로그래밍

정내훈

진행상황 (2019-수목)

- 강의
  - 소개
  - 네트워크
  - IOCP
- 실습
  - MultiThread
  - IOCP

진행상황

- TODO
  - Game Contents
  - DB
  - SCRIPT

게임 서버 구현

- 게임 서버 컨텐츠 구현 기본 이슈
  - Standalone게임과 무엇이 다른가?
  - scale이 다르다 : standalone게임은 한 화면에 보이는 객체만 시뮬레이션 하는데 게임 서버는 월드를 전부 시뮬레이션 해야 한다. (또는 모든 클라이언트에서 보이는 월드들을)

게임 서버 구현(2019-화목)

- 신경 써야 하는 구현 컨텐츠 들
  - 지형 자료구조의 크기 및 검색 속도
    - 2D, 3D?
    - 동굴, 복층, 다리?
  - 충돌의 multithread safeness
    - PC간의 충돌?
    - PC NPC간의 충돌?
  - Drop item으로 인한 부하
    - Item Drop
    - 시체 루팅

게임 서버 구현

- 게임 서버 구현 기본 이슈
  - NPC AI 성능
    - 길 찾기
      - 쫓아오기
    - Script
  - Contents 생산성
    - 전투, 퀘스트, 스킬…

게임 서버 구현

- MultiThread Issue
  - 충돌
    - 뚫고 지나가기?
  - Item rooting
    - 동시성 제어

기본 프로그래밍

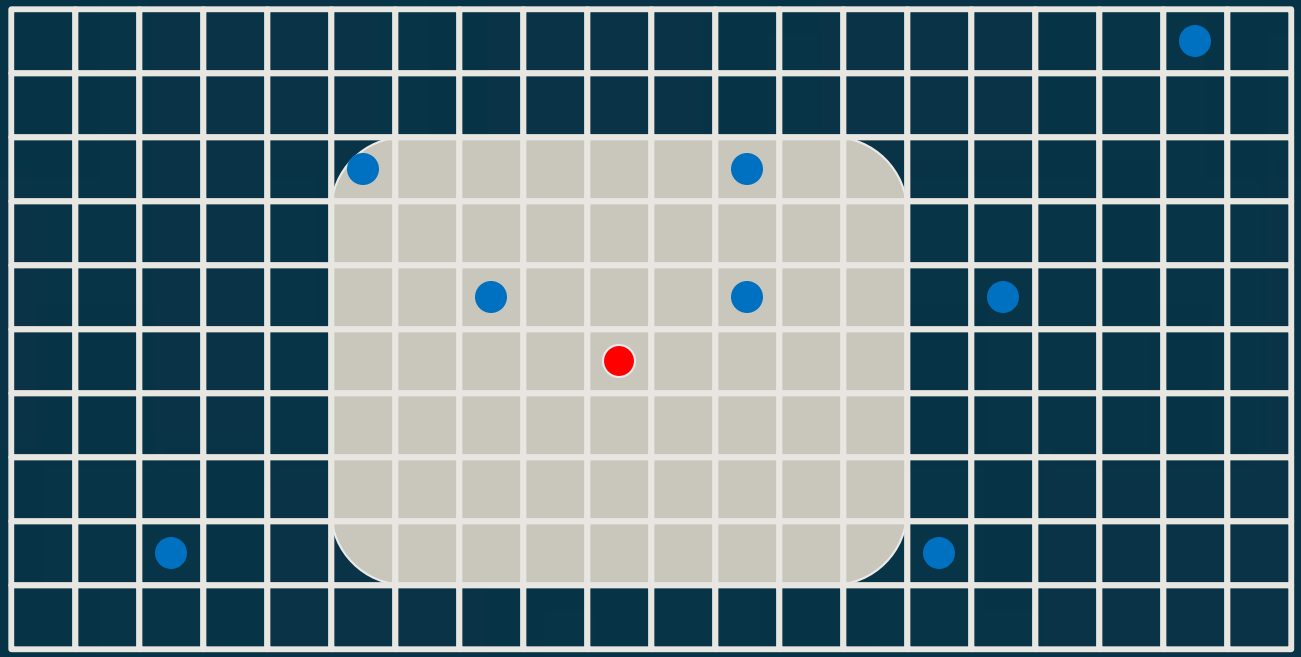
- BroadCasting 문제
  - 클라이언트에서 게임 월드의 상황을 볼 수 있으려면?
    - 서버가 접속한 아바타는 물론 다른 Object들의 상태도 보여 주어야 함
  - 하나의 Object의 상태가 변경되면
    - 주위에 존재하는 모든 아바타에게 변경내용을 전송해야 한다.

기본 프로그래밍

- BroadCasting 문제
  - 자신이외의 다른 Object들의 상태를 보여 주는 것
  - 동접 N 일때 N * N의 패킷 필요
    - 1초에 1번 이동
    - 동접 5000 => 25M packet / sec
    - 1000 CPU cycle per packet!!! => 25GHz필요
    - 패킷당 20byte => 4Gbps bandwith!!!!
  - 해결 ?
    - 근처의 object만 보여준다.

기본 프로그래밍

- BroadCasting
  - 근처의 Object?



기본 프로그래밍

- BroadCasting
  - 근처의 Object?

```
for (i=0; i<MAX_AVATAR;++i)
   if (RANGE >= Distance(avatar[i], me))
      SendState(i, me);
```

  - 효율적인 검색이 필요.

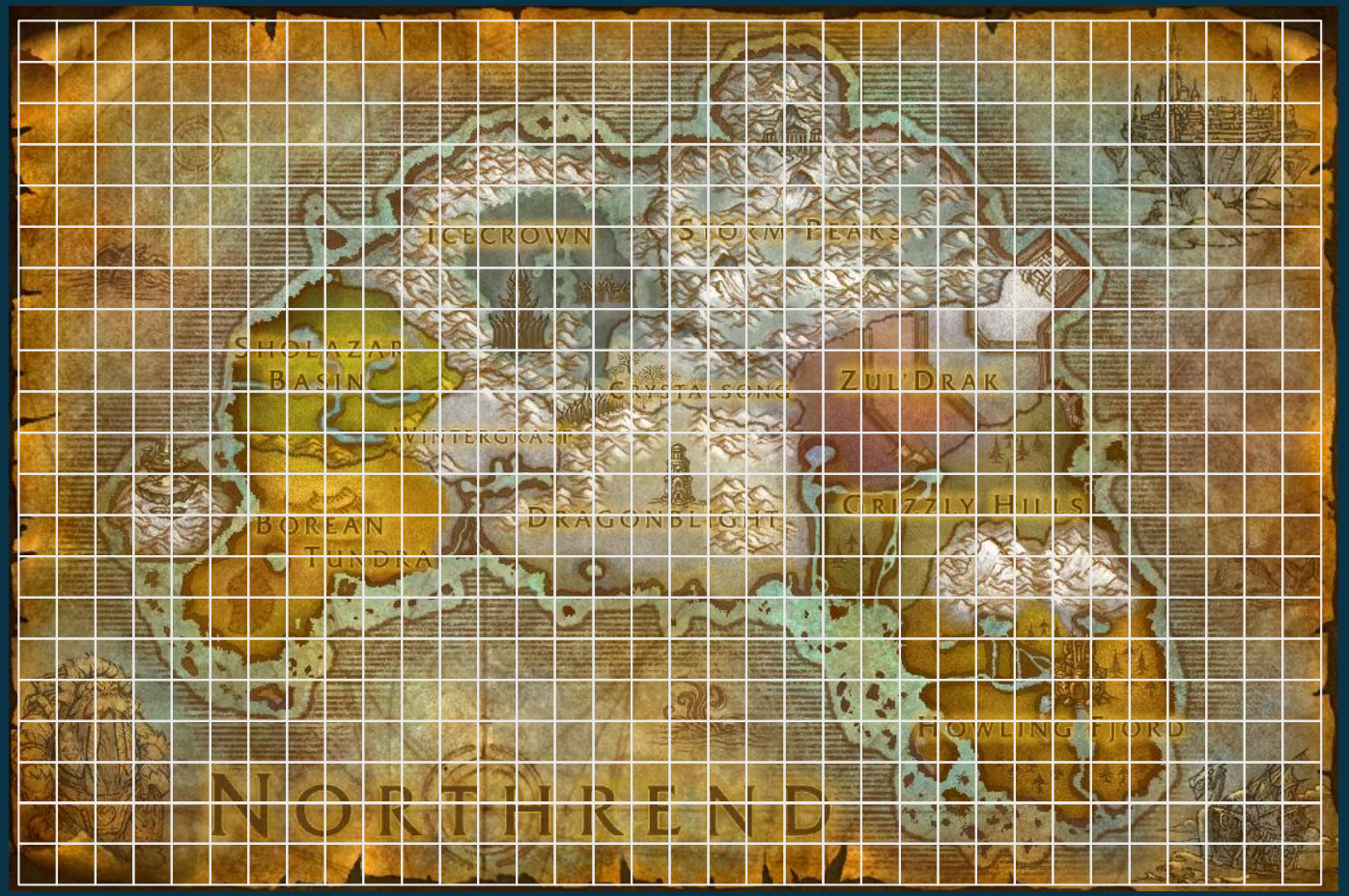
기본 프로그래밍

- BroadCasting
  - Zone (용어는 게임마다 다름)
    - 효율적인 검색의 기본
    - 전체 월드를 큰 논리적인 단위로 쪼개기
      - 보통 차원이 다르거나, 바다가 가로막고 있거나…
      - Seemless하지 않아서 이동 시 로딩이 필요
    - 대륙, 월드, 인던
      - 서로 볼일이 절대 없음
  - Sector (용어는 게임마다 다름)
    - Zone도 너무 크다. 더 줄이자.
    - Cluster라고도 함.

Zone 구성

가상 분할

- Sector



노스랜드 (From World of Warcraft)

Sector 구성

- Sector는 서버 내부에서 검색효율성을 위해 도입한 개념
  - 자기 주변에 있는 object의 리스트를 빨리 얻어내기 위해 사용
  - Sector의 크기는 최대 시야에 비례한다. 즉 적당한 개수의 Sector검색으로 시야 범위 내의 모든 object를 찾을 수 있어야 한다.
    - 너무 크면 : 시야 범위 밖의 개체가 많이 검색됨
      - 병렬성이 떨어진다. (여러 스레드가 한 섹터를 공유)
    - 너무 작으면 : 많은 sector를 검색해야 한다.
      - 이동시 잦은 섹터 변경 오버헤드

Sector 구성

- 클라이언트와는 아무런 상관이 없는 개념이다.
- Sector마다 Sector에 존재하는 object의 목록을 관리하는 자료구조가 존재한다.
  - g_ObjectListSector[Row][Col]
- 모든 object의 이동/생성/소멸 시 자신이 속한 Sector의 object목록을 업데이트 하여야 한다.
  - 멀티쓰레드 자료구조 필요

시야 처리

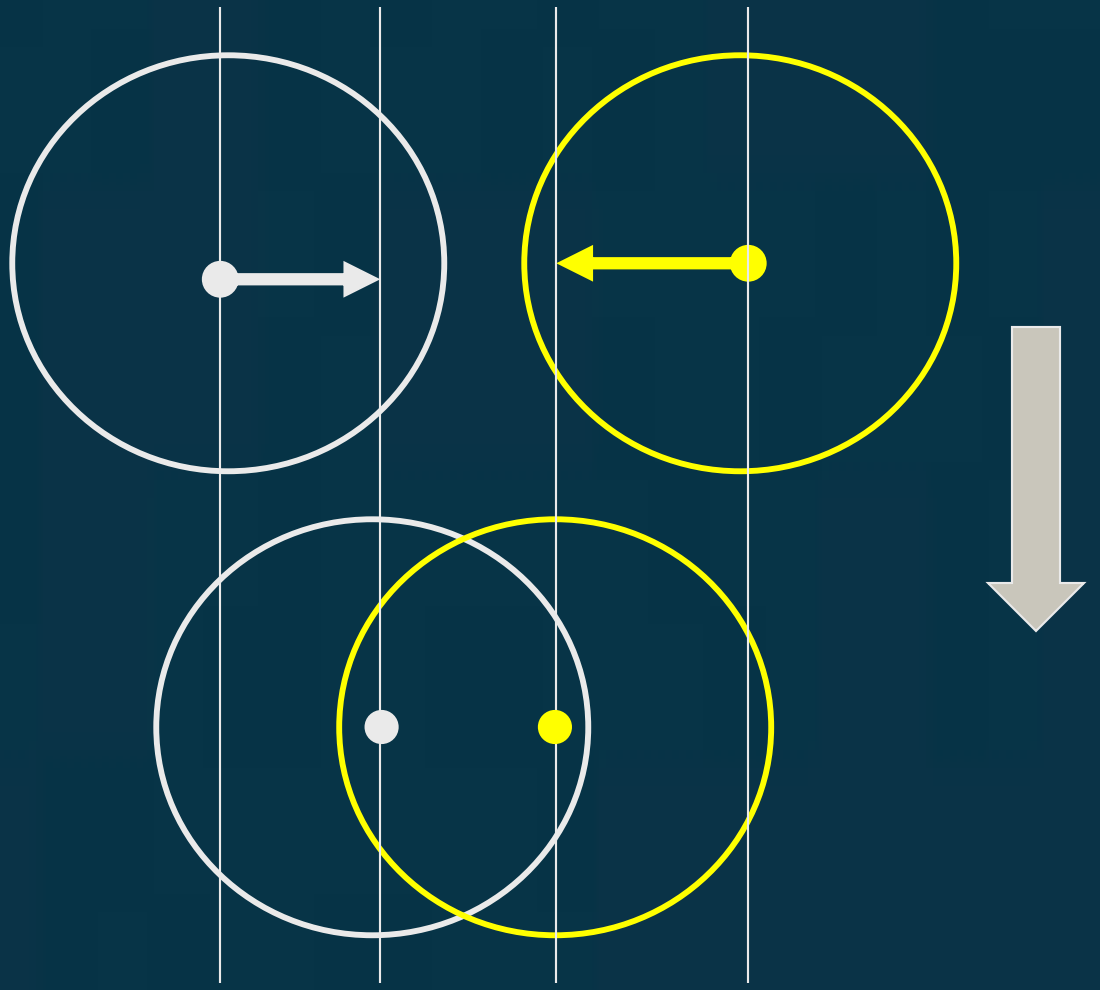
- 모든 플레이어는 시야라는 개념을 갖고 있다.
  - 시야 밖의 object에 대한 정보는 클라이언트에 전송되지 않는다.
  - 서버 성능 향상 및 네트워크 전송량 감소에 많은 도움을 준다.

시야 처리

- 모든 object의 이동시 시야 정보를 업데이트 해주어야 한다.
  - ViewList라는 객체를 모든 플레이어가 관리 한다.
  - ViewList는 클라이언트 측의 객체의 생성(PutObj)와 소멸(RemoveObj)의 기준이 된다.
    - 단순히 거리만으로 판단할 경우 서로 동시에 움직이므로 Put과 Remove가 어긋 날 수 있다.

시야 처리

- 어긋남
  - 이동과 시야 리스트 업데이트는 Atomic하지 않음



시야 처리

- 이동 시 시야처리
  - 시야 리스트 업데이트
  - 시야에서 사라진 player에게 removeObj전송
  - 새로 시야에 들어온 player에게 PutObj전송
  - 기존 시야리스트에 있는 obj에게 moveObj전송

시야 처리

- 이동 시 시야처리 순서
  - sector 검색 후 Near List 생성
  - Near의 모든 객체에 대해
    - viewlist에 없으면
      - viewlist에 추가
      - 나<-putObj(상대)
      - 상대 viewlist에 있으면
        - 상대 -> MoveObj(나)
      - 상대 viewlist에 없으면
        - 상대 viewlist에 추가
        - 상대-> putObj(나)
    - viewlist에 있으면
      - 상대 viewlist에 있으면
        - 상대->MoveObj(나)
      - 상대 viewlist에 없으면
        - 상대 viewlist에 추가
        - 상대 -> putObj(나)
  - ViewList에 있는 모든 객체에 대해
    - near에 없으면 removedID list에 추가
    - near에 있으면 => 아까 처리 했음
  - RevmovedID list의 모든 객체에 대해
    - viewlist에서 제거
    - 나<-RemoveObj(상대)
    - 상대 viewlist에 있으면
      - 상대 viewlist에서 제거
      - 상대<-RemoveObj(나)

시야 처리

- 최적화
  - ViewList의 Copy
    - lock이 걸려 있는 시간을 최소화 하여 lock으로 인한 성능저하 최소화
  - RemovedID List
    - 이중 Lock으로 인한 Deadlock 방지
  - 일반적인 mutex lock 대신 RWLOCK을 사용할 것
    - Read Operation을 많이 사용하고 있음.

숙제 (#4)

- 게임 서버/클라이언트 프로그램 작성
  - 내용
    - 숙제 (#3)의 프로그램의 확장
      - IOCP로 서버 구현
      - 전체 지도는 100 x 100
      - 클라이언트는 자기 말 주위 11x11 표시
      - 간격 표시자 (8칸마다), 좌표표시 클라이언트 화면
      - 자기 화면 주위의 플레이어만 broadcast (시야 7*7)
  - 목적
    - 클라이언트 영역 개념 이해
  - 제약
    - Windows에서 Visual Studio로 작성 할 것
  - 제출
    - 4월 18일(목요일) 오전 11시
      - 2019 게임서버[화목] 학번 이름 숙제 4번
      - Zip으로 소스를 묶어서 e-mail로 제출

다음시간

- 시야처리 실습

중간고사

- [화목반] 4월 23일 화요일
  - 4월 25일 수업
    - NDC당첨 -> 휴강
    - NDC탈락 -> 정상수업 -> NDC 참석인증자 출석인정
- [수목반] 5월 1일 수요일
  - 4월 24일, 4월 25일 수업
    - NDC당첨 -> 휴강
    - NDC탈락 -> 정상수업 -> NDC 참석인증자 출석인정
-