서버 개발 과제들

질문이나 요구 사항으로 던지고 함께 생각해 보려 한다. 무엇을 모르는 지 아는 것이 아는 것이라는 소크라테스의 관점에서 질문은 가장 중요하다. 질문에 대한 의문도 중요하다. 다양한 아이디어와 혁신의 출발점은 현상에 대해 의문을 갖고 질문을 하는 데서 시작한다. 앞으로 많은 질문들이 나오고 다양한 답이 제시되길 기대한다.

여기서는 MMORPG를 대상으로 한다. 다른 형태의 게임들은 몇 가지 해결해야 할 다른 문제들이 있긴 하나 주로 확장성 이슈이므로 별도로 묻는 형태로 포괄할 수 있다.

개발 절차

기획

- 구현을 시작하고 결과를 검증할 수 있는 정도의 기획 스펙을 확정하기 위한 가장 효율적인 절차는 무엇일까?
- 기획 없이 개발이 가능할까? 기획 문서가 없는 것과 기획이 없는 것의 차이가 있을까?
- 회의로 기획을 결정할 수 있을까?
- 게임 전체, 게임 서비스 시간 전체에 대한 상(이미지) 없이 기획이 가능할까?
- 기획을 이해하지 못 하고 구현이 가능할까?
- 개발 게임을 플레이 하지 않는 프로그래머가 구현을 잘 할 수 있을까?
- 프로그래머는 게임을 어느 정도 어디까지 이해해야 할까?

설계

- 분리된 설계 절차가 구현 품질을 올리는데 필수적인가? 아니라면 불필요한가? 필요하다면 어느 정도로 진행해야 하는가?
- 설계란 무엇인가?
- 설계 리뷰를 할 필요가 있는가? 설계 리뷰를 항상 해야 하는가?
- 설계를 UML로 한다고 할 때 어디까지 진행해야 할까? 서버에서 프로토콜 설계가 시퀀스 다이어그램으로 충분할까?
- 개별 컨텐츠 내 타잎 설계는 어떻게, 어디까지 해야 할까?

코드

- 코드의 품질은 무엇으로 결정되는가?
- 정확한 코드라고 확신할 수 있는 방법은 무엇인가?
- 코드 리뷰가 필요한가? 코드 리뷰로 개선이 가능한가?
- 코드 리뷰가 효과가 있다면 잘 적용이 안 되는 이유는 무엇인가?
- 시간이 없어 코드 리뷰를 할 수 없다면 코드 리뷰로 찾을 수 있는 버그와 품질 개선이 시간을 절약할 수 없는가?

테스트

- 단위 테스트가 MMORPG 개발에서 얼마나 가능할까?
- 어느 정도로 단위 테스트를 해야 하는가?
- 단위 테스트 되지 않은 기능에 대해 정확성은 어떻게 보증할 수 있는가?
- 단위 테스트로 검증 가능한 품질 수준은 어디까지인가?
- 100% 단위 테스트로 코드 커버리지 된 코드와 그렇지 않은 코드의 품질 차이는 얼마나 될까?
- 기능 테스트 (봇 테스트)로 가능한 검증 범위는 어느 정도인가? 기능 테스트 프레임워크는 어느 정도로 개발되어야 하는가? 부하 테스트를 포함한 기능 테스트와 손쉬운 테스트 개발은 어떻게 가능할까?

버전 관리

- 코드에 들어가는 버전별 세부적인 기능 차이를 어떻게 관리해야 할까?
- 큰 시스템 단위 컨텐츠는 어떻게 관리해야 할까?
- 여러 지역의 버전별 다른 차이는 어떻게 관리해야 할까?
- git과 같은 분산 형상 관리 도구의 request / merge 방식이 게임에도 동작할까?
- 코드, 기획 데이터, 그래픽 애셋은 어떻게 통합 관리해야 할까?
- define으로 기능을 관리하면 코드가 복잡하고 지저분하게 된다. 대부분의 게임이 이와 같이 하고 있는데 레전드는 서비스 단위별 브랜치와 병합을 사용하고 있다. 어느 쪽이 더 나을까? 다른 대안은 없을까?

QA

- QA에서 검증 가능한 범위는 어디까지일까?
- 클라이언트 중심으로 테스트 하는 현재 방식이 어디까지 유효한가? 한계는 무엇일까?
- QA 과정에 리뷰, 단위 테스트를 포함해야 하지 않을까? 이런 과정까지 절차로 만드는 게 적합할까?

디버깅

- 서비스 중에 발생하는 다양한 이슈들을 빠르게 디버깅하기 위한 방법은 무엇일까?
- 어떤 기능을 구현해 두어야 가능할까?
- 서비스 중에 자주 발생하는 이슈는 무엇일까?
- 서버의 덤프를 뜨고 디버깅할 때 디버깅이 잘 되려면 코딩을 어떻게 해야 할까? 불필요해 보이는 통계 자료, 디버깅 변수들을 남겨 두는 건 어떨까? 여러 상태를 모아두는 클래스를 두는 건 어떨까?
- 로그가 디버깅에 얼마나 도움이 되는가? 로그가 도움이 되려면 무엇을 준비해 두어야 하는가? 로그를 프로그램 설계 만큼 공을 들여서 하고 있는가? 미리 로그 형식이나 로그 지점을 설계하고 규약으로 만들어서 맞춘다면 더 나을까? 얼마나 나을까?

규약

- 코딩 규약(컨벤션)이 있는 것과 없는 것의 차이는 무엇인가?

- 규약은 어느 정도로 상세해야 하는가? 규약은 합의 가능한가? 합의 안 되더라도 규약을 지키는 것이 나을까?
- 코딩 스타일과 코딩 규약은 어떻게 다른가?
- 코딩 규약에 필수적인 내용은 무엇일까?
- C++ Core Guidelines의 내용을 지킬 필요가 있을까? 어디까지 지켜야 할까? GSL 라이브러리 중 쓸만한 것이 있는가? 게임 서버에 적용 가능한가?
    - https://github.com/isocpp/CppCoreGuidelines/blob/master/CppCoreGuidelines.md
        - C++ Core Guidelines 문서

핵심 기능

메모리 관리

- 일반적인 형태의 메모리 관리자가 OS의 메모리 관리기보다 빠른가? 빠르다면 얼마나 빠른가? 다양한 조건에서 항상 빠른가?
- 오브젝트 풀 형태가 더 낫지 않은가?
- 메모리 관리자가 없더라도 메모리 항당 / 해제는 추적이 필요할 때가 많다. 왜 그런가?
- shared_ptr / unique_ptr로 충분하지 않은가? new / delete가 없는 코드를 작성하면 되지 않을까?

로거

- printf 스타일 로거는 크래시를 발생시킬 수 있다. 안전한 printf 로거를 만들 수 있는가?
- spdlog와 같이 fmt 로그 형식을 사용하는 건 괜찮을까?
- 비동기 파일 로그를 제공하지 않는 로거는 어떤 문제가 있을까?
- 다국어를 로거에서 지원하기 위한 방법은 무엇이 있을까? WCHAR로 충분한가? UTF8로 하는 건 어떤가?
- 여러 파일에 로깅이 가능한가? 여러 파일에 로깅을 남길 필요가 있을까?
- 로그를 빠르게 살펴 볼 수 있는 방법은 무엇이 있을까? grep으로 충분한가? 다른 검색 도구는 없을까?
- elasticsearch의 검색 기능을 활용하면 통합 검색이 가능하다. 이런 시스템을 사전에 구축하는 게 유용할까? 얼마나 유용할까? 무엇을 준비 해 두어야 할까?

통신

구현 선택

- IOCP나 epoll / kqueue 기반으로 직접 통신 코드를 작성하는 것과 libuv나 boost.asio 같은 플래폼 호환 라이브러리를 사용하는 것의 장단점은 어떻게 될까? 직접 작성한다면 어떻게 작성할 것인가? 검증은 어떻게 할 것인가?
- 통신은 이미 해결된 문제라고 하는 이유는 무엇일까? 버그 없는 완결된 게임용 통신 라이브러리가 게임 쪽에 아직 없는 이유는 무엇일까? Raknet 같은 라이브러리를 모두 다 사용하지 않는 이유는 무엇일까? 서버 프로그

래머들이 통신 라이브러리도 직접 작성하는 이유는 무엇일까? 앞으로도 그래야 할까? 어느 정도면 회사, 업계에서 인정된 수준의 라이브러리라고 할 수 있을까?
- 락과 복사를 최소화하기 위한 방법은 무엇이 있을까? 전송 시 모아서 보내는 방법은 큐가 나을까, 아니면 버퍼가 나을까? 수신 패킷의 메모리 복사를 최소화 하려면 어떻게 해야할까? 피할 수 없는 복사 횟수는 얼마일까?

프로토콜과 언어 지원

- 여러 프로토콜을 지원할 필요가 있는가? 다중 프로토콜을 지원할 때 어떤 구조적인 차이가 있을까? UDP를 MMORPG 서버에서 사용하지 않는 이유는 무엇일까? JSON이나 HTTP 프로토콜 지원이 있다면 좋을까? 활용한다면 어떻게 활용할 수 있을까? boost.beast와 같은 웹 서버를 별도로 포함하는 게 나을까?
- C#이나 javascript 같은 C++ 외의 언어와 통신할 필요가 있다면 어떤 구조가 좋을까? 무엇이 필요할까? IDL과 파서로 메세지의 생성과 시리얼라이제이션 코드 생성이 필요할까? 있다면 얼마나 도움이 될까?

보안 기능

- 통신 라이브러리의 보안 기능은 어느 정도가 적합한가? 세션 단위 암호화 기능이 꼭 필요한 이유는 무엇일까? 패킷 단위 암호화를 지정하고 사용하게 해야 하는 이유는 무엇인가? 세션 단위 암호화 기능을 구현하는 적합한 방법은 무엇일까? 공개키 암호화로 세션 키를 전달할 필요가 있을까? 순서 체크나 체크섬 기능은 필요한가? 모든 패킷에 필요한가? 통신 라이브러리 단에서 패킷의 필드별 타잎과 길이 체크가 있다면 보안에 도움이 될까? 구현이 복잡해지지는 않을까? DoS (많은 연결 시도와 패킷을 보내는 공격)를 라이브러리 단에서 막을 수 있을까? 어디까지 막을 수 있을까?
- 통계 기능을 구현하면 디버깅에 도움이 될까? 이를 보안 기능에도 활용할 수 있을까? 세션에 보안 관련 특성을 설정하고 정책을 지정해서 처리하게 하면 어떨까? 얼마나 도움이 될까?

사용과 검증

- 세션을 애플리케이션에 참조하는 방식은 어떤 형태가 좋을까? 세션이 유효한 지 검증이 필요한가? 세션이 해제되었을 때 잘못된 사용을 방지하려면 어떻게 해야 할까? 세션 오브젝트 풀링을 사용할 경우 안전하게 사용하지 못 하도록 하는 방법은 무엇이 있을까? 애플리케이션에서 세션을 아이디로 참조하거나 포인터로 참조할 수 있는데 어느 방법이 나을까? 다른 방법은 없을까?
- 안정성과 성능을 검증하기 위한 테스트는 어떻게 해야 할까? 라이브 환경에서 검증된 것에 필적할 만한 테스트는 어느 정도일까? 전세계에 분산된 클라우드 환경에서 테스트를 진행하면 충분할까? 어느 정도 어떤 특성의 성능이 나오면 좋은 라이브러리라고 확신할 수 있을까?
- 코드 전체를 증명 가능한 수준으로 리뷰를 통해 검증할 수 있는가? 검증하려면 어떤 invariance(들)를 통해 증명해야 할까?

송신과 수신

- 패킷 타잎은 단일한 것이 좋은가 아니면 계층을 갖는 것이 나은가? 계층이 있을 경우 그룹별 전달이 효율적일 수 있는가? 어떤 경우에 효율적인가?
- 수신 흐름을 지정할 때 받을 오브젝트에서 등록하는 것이 좋을까 아니면 전달하는 쪽에서 결정하는 것이 좋을까? 함수 정도로 구분하고 함수에서 최종 대상을 결정하는 것이 좋을까?
- 송신은 직접 세션이 쓰는 것이 좋을까? 송신도 등록하고 트리거하면 세션(들)에 보내도록 하는 것이 좋을까?

- 패킷 타잎 외에 필드 정보를 포함하여 전달하는 구조를 만들려면 어떻게 할 수 있을까? 필드 정보를 포함해서 전달 하는 방식이 장점이 있을까?

데이터 관리

기본 흐름

- 엑셀이나 Access를 사용하여 데이터를 편집하고 게임에서 CSV나 MDB, SQL DB에서 로딩하는 현재 흐름이 가장 효율적인가? 쓸만한가? 더 나은 방법이 있는가?
- 데이터의 공동 작업이 가능하려면 어떤 기능이 필요한가? 공동 작업이 필요한가? 얼마나 자주 하나의 파일에 접근해야 하는가?

현재 작업 흐름은 관계형 자료 구조를 기반으로 csv나 DB (SQL, MDB)를 일반적으로 사용한다. 엑셀에서 편집하고 export 하는 형태가 일반적이다. 관리 코드의 생성을 별도 스키마, 엑셀 셀 정의, DB 테이블로 하는 경우들이 있다. 이 흐름을 기본으로 하고 추가 질문을 한다.

코드 생성

- 테이블 스키마 기반으로 관리 코드를 생성하려면 어떤 구현이 필요한가? 파서 기반이 좋을까 아니면 json과 같은 이미 파서가 있는 형식으로 충분한가?
- 생성된 코드에 포함될 내용은 무엇인가? 타잎 검증 코드는 어디까지 가능한가? 주 키 (Primary key)외에 인덱스를 추가해야 할 필요가 있는가? enum 타잎은 어떻게 정의할 수 있을까? 외부 키 (Foreign key) 검증은 필요한가?

재로딩과 관리

- 데이터를 자동으로 재로딩 하려면 어떻게 해야 할까? 데이터 필드에 대한 참조를 어떻게 구현하면 재로딩이 가능할까? 이미 데이터 기반으로 생성된 오브젝트들을 재로딩에 영향이 없는데 이럴 경우 어떻게 해야 재로딩이 가능할까? 재로딩을 하는 인터페이스는 치트 명령이 가장 효율적일까? 다른 방법이 있을까?
- 최종 데이터를 DB에 넣거나 작업 시부터 DB를 기반으로 하는 건 어떨까? 효율적일까? 엑셀 없이 기획 작업이 가능할까? 엑셀로 작업을 주로 진행한다면 최종 단계에 DB에 넣는 게 여전히 장점이 있는가?

처리

처리 모델은 결정하고 진행해야 한다. 액터 모델, 메세지 패싱 두 가지 모델에서 가져온 기능 / 데이터 단위 쓰레드 모델을 사용한다. 실행 단위를 task로 한다. 서버에서 실행 단위는 task를 상속 받은 service가 된다. 이런 모델을 선택한 주된 근거는 다음과 같다.

- AKKA (Scala)와 다양한 응용
    - Spark
- Erlang과 Couchbase

    - Couchbase는 Erlang으로 만들어짐
    - Erlang의 액터 지원 함수형 언어
- Node.js와 비동기 처리 모델
- Go 언어의 채널
    - CSP 모델로 액터와 다르나 근간의 공통점이 있음 (메세지 패싱)

개인적으로 이 방향으로만 가기로 결정했다.

스케줄링

- task 실행 스케줄링은 어떻게 하는 것이 좋을까? task 별 실행 특성을 반영해야 하는가? task 개수면 충분한가? 특정 쓰레드에서 특정 task를 실행하는 기능(thread affinity)이 필요한가?
- 타이머는 timer wheel이 효율적이다. 가장 효율적인 타이머 구현 알고리즘은 무엇인가? 수 만개의 타이머가 있다고 할 때 적절한 스케줄링은 무엇인가? 타이머 콜백 인터페이스는 무엇이 좋을까? 아이디 기반으로 콜백 처리 코드를 작성할 때 switch 문 안에서 처리하는 경우가 많다. 람다로 하면 이점이 있을까? 성능은 어떨까?
- 태스크가 없을 때 다른 쓰레드에 양보하는 방법은 sleep으로 충분한가? sleep 시간은 얼마로 해야 하는가? 전체 쓰레드가 task 큐를 공유하고 꺼내 오는 방식이 나을까?

비동기 처리

- 블럭킹 되거나 오래 걸리는 코드는 어떤 영향을 미치는가? 쓰레드를 대기 상태로 만드는 시스템 호출은 어떤 것들이 있는가? 이들은 왜 비동기로 처리해야 하는가? 이들은 항상 비동기로 처리해야 하는가?
- 오래 걸리는 코드는 어떻게 찾을 수 있을까? 프로파일러는 어떻게 구현하는 게 좋을까? 프로파일링 결과를 로그로 남기면 충분할까?

메세지 전달 (수신)

통신.송신과 수신과 일부 중복된다. 서비스와 컨텐츠 개발자의 입장에서 바라본다.

- 수천개 정도의 메세지 (패킷) 타입의 디스패칭 등록이 불편할 수 있다. 자동화가 바람직한가? 필드 데이터를 사용하여 디스패칭해야 한다면 자동화가 가능한가? 계층화된 패킷 타입을 사용하면 좀 더 편리한가?
- 수신하면 안 되는 메세지(패킷)에 대한 검증과 방어 코드가 필요한가? 로그로 충분한가?

분산

서버 간 분산 처리는 일부 또는 전체에 대해 항상 필요하다.

기본 문제들

- 동적인 오브젝트 (서비스 / 객체)는 분산 관리하기가 어렵다. 정적인 구조로 분산하고 정적 구조 안에서 동적인 처리가 나을가? 동적인 대상도 구현만 잘하면 괜찮을까?

- 게임 서버로 동작하는 노드가 연결이 끊어졌다고 하면 어떤 처리를 해야 할까? 일시적인 단선일 경우는 어떻게 처리할 수 있을까? 일시적인 단선일 경우 서비스를 복구해 줄 수 있을까?
- DB 처리 서버와 같이 매우 중요한 서버가 연결이 끊어졌다고 하면 어떤 처리를 해야 할까? 일시적인 단선을 복구할 수 있을까? 어떤 처리로 어디까지 구현 가능할까?
- 분산하면 요청 후 응답을 처리해야 할 경우가 많고 요청 정보를 참조해야 하는 경우도 많다. 이와 같은 비동기 요청과 응답의 연속된 처리를 하는 효율적이고 프로그래머 입장에서 쉬운 구조는 무엇일까? CPS (Continuation Passing Style)라고 하는 함수형 언어의 호출 체계 중 하나를 이 쪽에 적용할 수 있을까? 어떤 식으로 메세지에 CPS를 포함해야 할까? 메세지 전달은 어떻게 해야 할까?

DB

DB 처리를 담당하는 전용 서비스를 갖는 서버를 말하며 다음의 과정으로 이루어진다.

- 트랜잭션의 비동기 처리
    - 게임 쓰레드 상의 요청 준비
    - 트랜잭션 쓰레드와 DB 상의 트랜잭션 처리
    - 게임 쓰레드 상의 결과 처리

- 트랜잭션을 처리하는 쓰레드 풀의 크기는 어느 정도가 적절한가?
- DB 연결을 사용 못 하게 되는 예외가 있는가? 예외가 발생하면 DB 연결을 다시 맺는 것이 나은가?
- DB 연결이 끊어지거나 DB를 사용 못 하게 될 경우 어떻게 해야 하는가?
- 트랜잭션의 비동기 처리를 위한 구조는 어떻게 하는 게 좋은가? Transaction 클래스를 만들고 게임 상의 준비, DB 처리, 결과 처리를 구분하여 만드는 것이 좋은가?
- 대부분의 DB API로 ODBC면 충분한가? 다른 API와 성능 차이는 없는가?
- 서비스 중에 스토어드 프로시져 변경이 즉시 반영되려면 어떻게 해야 하는가? 모든 쿼리를 스토어드 프로시저로 하는 것이 나은가?
- DB 응답 성능을 프로파일링 하는 기능을 서버에도 포함하는 것이 좋은가? 포함한다면 어떤 형태로 구현하는 것이 적절한가?
- DB Stored Procedure 파싱을 통해 DB 실행을 처리하는 코드의 자동 생성이 가능한가? 이는 세 단계로 분할되는 비동기 처리 특성상 어려울 것으로 보인다. 매크로를 포함한 DSL을 개발하여 빠르게 C++로 작성할 수 있도록 하는 게 나을 수도 있다. 그래도 가능할까? 얼마나 도움이 될까? 다른 접근이 있을까?

공간

RPG에서 공간 정보의 생성과 사용은 기본이다. 근래는 네비게이션 매시와 충돌 정보 일부를 사용하고 2D 판정을 하는 게임이 대다수이다. 복셀 기반으로 충돌을 확장하거나 트리 기반 (스피어 트리, AABB 트리) 으로 판정에 사용하는 경우도 있다. 처리 비용과 효과를 고려하여 선택해야 하며 지속적인 연구가 필요하다.

공간 정보

- 네비게이션 매시와 2D 판정으로 지원 가능한 MMORPG는 어느 정도까지일까? 날아다니거나 차폐가 중요한 게임은 어떻게 처리해야 할까?

- 복셀과 2D 판정으로 지원 가능한 MMORPG는 어떤 형태일까? 어디까지 가능할까? 복셀 처리의 메모리와 CPU 부하는 얼마나 될까? 구현하려면 어떤 알고리즘에 대한 이해가 필요한가?
- 빠르면서 게임에 적합한 2D 판정 알고리즘은 어떤 것이 있을까? GJK로 충분한가? 대상의 모양과 판정 모양을 모두 고려해야만 하는 이유는 무엇일까?
- 위 두 가지 외에 직접적인 충돌 판정을 사용하는 게임이 가능할까? 다른 방식의 공간 정보 생성은 가능할까?

계산 기하학에 대한 꾸준한 공부와 리서치가 필요한 영역이다. 실내 / 외 모두 처리가 가능해야 하며 적절한 CPU와 메모리 사용으로 구현 가능해야 한다.

공간 분할

- 옥트리 (Octree), 쿼드 트리, 그리드 방식의 장단점은 무엇일까? 구현 난이도는 어떨까? 판정과 연결하는 것이 나을까, 아니면 별도로 판정만 구현하는 것이 나을까?
- Box2D에는 AABB 기반의 충돌 판정 기능이 있다. 이를 사용해서 공간 분할과 별개로 사용할 경우 장단점은 어떻게 될까? 문제가 없을까? Box2D의 GJK 알고리즘 성능은 어떤가? 파이, Arc, 볼록 다각형 처리는 얼마나 정확하게 구현할 수 있을까? 초당 처리 가능한 판정 횟수와 CPU 사용량은 개체와 판정 모양에 따라 어떻게 변하는가?

컨텐츠 기반

복잡도

게임, 특히, MMORPG는 복잡도가 매우 높은 시스템이다. 복잡도를 높이는 동력은 다양한 시스템과 이들 간의 관계로 게임이 만들어지기 때문이다. 이 부분이 시스템으로서 매력이 가장 높은 부분이기도 하지만 가장 만들기 어렵게 만들기도 한다. 다른 측면은 기획과 사용자의 변경 요구가 많기 때문이다. 이 또한 매력인 동시에 어려움인 부분이다.

이러한 복잡도는 근본적으로 피하기 어렵기 때문에 수단을 확보해야 한다. 다음과 같은 방법이 주로 많이 쓰이고 효과를 보이는 부분이다.

- 데이터 주도 (Data Driven)
- 이벤트 주도 (Event Driven)
- 컴포넌트 분할 (Component Based Development)

데이터 주도

기획 데이터의 관리 부분에서 일부 살펴보았지만 데이터로 분리하고 개별 오브젝트를 풍부하게 만드는 과정이 기본이다. 스크립트나 판단 / 행동 지정도 데이터로 본다면 데이터로 만들 수 있는 부분은 증가한다.

기획은 데이터와 규칙의 정의로 봐야 한다. 기획을 재미를 만드는 것으로 생각할 수 있지만 기술적으로는 데이터와 규칙에 집중해야 한다. 데이터의 설계가 잘못되면 게임 개발이 난관에 부딛히고 더 이상 성장 못 하는 경우가 생기며 사전에 검증하기도 어려워진다. 따라서, 이 부분의 역량을 올리는 노력을 멈추지 않아야 한다.

서버 구현에서 데이터 테이블은 Element, Dic, Table 등 몇 가지 용어로 불린다. 관계형 구조라면 Table이라 하고 각 항목을 Row라고 부를 수도 있겠다. SkillTable, SkillTable::Row, SkillTable::Row::Ref 와 같이 용어를 선택할 수도 있고, 뭔가 있어 보이는 Element를 줄여 Elem이라고 할 수도 있겠다.

데이터 관리에서 언급한 대로 타잎 점검, 참조 점검, 주 키와 인덱스를 사용한 검색 등이 필요하다. 해당 코드를 자동으로 만들면 부담이 크게 줄어든다. 코드 생성은 자유롭지 못 하다는 한계를 포함하므로 C++ 코딩으로 편의 함수를 추가할 수 있는 구조도 함께 고려해야 한다.

이벤트 주도

이벤트를 구성에 사용하면 클래스 간 인터페이스가 매우 유연해지며, 함수 호출 방식과 달리 등록 기반의 전파가 가능해진다. 등록 기반의 전파는 멀티캐스팅이 가능하다는 장점이 있고 작성하는 쪽에서 누가 사용할 지 몰라도 되기 때문에 접합 부위를 변경하기 용이하다.

디자인 패턴의 Observer / Observable로 바도 되고, 네트워크 메세지라고 생각해도 되며, Go 언어의 채널이라고 생각해도 된다. Subscribe / Dispatch / Unsubscribe를 기본 용어로 하는데 Dispatch를 Emit으로 부르는 경우도 있다. Qt의 경우 Slot 에 connect / disconnect 하는 용어로 부르고 있다.

오브젝트와 컴포넌트 분할

Unity / Unreal 4 모두 GameObject와 Component, Actor와 Component로 클라이언트를 구조화 한다. 게임은 클라이언트 중심으로 발전해 왔고 서버 쪽은 반응형 처리와 데이터 보관 중심으로 발전했기 때문에 클라이언트의 개념을 서버로 확장하는 것이 게임의 발전 방향에 좀 더 맞는 것으로 보인다.

오브젝트와 오브젝트가 갖는 컴포넌트로 구성하면 몇 가지 이득이 있다.

- 복잡도를 오브젝트와 컴포넌트 단위로 분산
- 컴포넌트 단위의 재사용 가능
- 컴포넌트 교체로 기능 변경

컴포넌트를 만든다고 해서 전체 복잡도가 줄지는 않는다. 단지, 단위 기능의 복잡도를 컴포넌트 내부로 제한하는 효과만 있다. 게임이 복잡한 이유는 관계의 복잡성 때문이라 코드 구현에 반드시 나타나게 되어 있다. 그렇다고 하더라고 컴포넌트 내부로 기능을 숨기면 컴포넌트의 제한된 인터페이스를 통해서만 관계가 표현되므로 관계 표현에 필요한 최소한의 인터페이스만 공개된다. 작지 않은 이득이고 꼭 필요하다 할 수 있다.

오브젝트와 컴포넌트

리플렉션

- 오브젝트에 리플렉션 기능이 있다면 어떤 장점이 있을까? 없는 경우와 비교해서 어떨까?
- 리플렉션으로 동적인 타잎 시스템을 구축할 수 있는가?
- C++에서 리플렉션을 제공하는 라이브러리는 어떤 것들이 있는가? 타잎 시스템도 제공하는가?
- 언리얼 4는 자체 파서로 리플렉션과 타잎 시스템을 제공하고 있다. 이런 방법이 라이브러리를 사용하는 방법보다 나은가? 어떤 장단점이 있을까?

시그널 (이벤트 주도)

- C++의 observer / observable 패턴을 구현한 라이브러리들은 어떤 것들이 있는가? 각각의 장단점은 무엇인가?
- 람다로 콜백을 사용할 경우 오브젝트 레퍼런스는 어떻게 포함되고 유지되는가? 오브젝트가 참조 되면서 발생할 수 있는 문제들은 무엇인가? 이를 보완하거나 제거할 방법은 무엇일까?
- sub / unsub / pub는 각각 Subscribe / Unsubscribe / Publish (Emit)에 해당한다. sub / unsub를 자동으로 하는 기능을 오브젝트에 어떻게 포함할 수 있을까? 대상은 어디로 해야 하는가?

컴포넌트

- 캐릭터 (Pc와 Npc)의 구현에 필요한 컴포넌트들은 어떤 것들이 있을까? 어떻게 나누고 관계를 맺어야 구조가 깔끔할까?
- 업적을 처리하는 CompAchievement가 있다면 어떻게 구현하는 것이 좋을까? 개별 이벤트는 어떻게 통지 받는 것이 좋을까? 퀘스트는? 미션은?
- 어떤 컴포넌트는 캐릭터 상태에 따라 실행되면 안 된다. 이런 처리는 어떻게 하면 일관성 있게 한번에 될까? Activate / Deactivate 같은 기능을 넣을 경우 문제는 무엇일까? 다른 더 나은 방법은 없을까?
- 컴포넌트 단위로 처리를 위한 타잎 설계 등이 필요하다. 예를 들어, CompInven은 조합과 메세지 처리를 담당하고 내부에 InvenBase, InvenBag, InvenRebuy, InvenGuild 등이 필요하다. 이런 내부 타잎들의 설계는 개별 컨텐츠 단위로 정리되어야 한다. 사전에 패턴화 할 수 있는 범위는 어디까지일까? 게임별로 항상 유지되는 부분은 어떤 것들일까? 프로토콜, DB, 타잎 (멤버와 함수)을 사전에 정리하면 도움이 될까?

구성 (조합)

- 특정 아이템을 갖고 있어야 특정 맵에 들어갈 수 있는 경우의 처리를 한다고 할 때, CompInven를 참조해서 물어봐야 한다. 어떻게 구성하면 좋을까? CompInven에 해당 기능을 구현해야 하는가?

공간 처리

공간 정보와 판정은 미리 살펴 보았다.

인스턴스 관리

요즘은 사용자 분산을 위해 MMORPG에서 지역 채널을 모든 게임이 지원하므로 동적인 인스턴스 관리가 필요하다. 단일 서버일 경우는 낫지만 분산된 게임 서버들을 다수 가질 경우 인스턴스 관리 프로토콜이 잘 설계 되어야 한다.

- 단일 책임의 원칙이 여기서도 중요하다. 게임 인스턴스의 생셩과 소멸, 진입과 진출을 단일 책임의 원칙하에 구현한다고 할 때 인스턴스 관리 서비스와 인스턴스 실행 서비스 간의 적절한 프로토콜을 무엇일까?
- 인스턴스를 여러 서버에 배당할 때, 서버 선택의 기준은 무엇이 되어야 하는가? 현재 2명이 있는 500명을 수용 가능한 채널 인스턴스는 얼마의 비중으로 판단해야 하는가? 최적의 인스턴스 배정 알고리즘은 무엇일까?
- 인스턴스 진출입 시 여러 가지 조건 체크를 하게 된다. 이들 조건을 별도의 조건 클래스들로 만들고 판단할 수 있도록 구성하는 방법은 어떨까? 인스턴스 종류별로 클래스를 만들고 개별 클래스에서 체크하는 방법이 더 나을까?

- 파티에 귀속된 인스턴스를 관리하는 구조는 어떤 게 좋을까? 길드는? 파티를 관리하는 서비스가 분리되어 있다고 할 때, 인스턴스 관리 서비스와 교환해야 할 메세지는 어떤 것들이 있을까? 로비 서비스와 교환할 정보는?

이동과 동기화

- 오토데스크 네비게이션의 Update 호출은 비용이 매우 크다. 특히, 서로 피해 가도록 할 경우 내부 알고리즘의 실행 비용이 크다. 이를 게임에서 직접 느슨하게 처리하여 비용을 줄일 수 있다. 이동을 게임에서 직접 시킬 때 상호 회피 (Avoidance)와 겹치기 방지를 포함한 최적의 알고리즘은 무엇일까?
- 어뷰징이나 공격의 포인트로 위치 보다는 시간이 많이 사용된다. 이동 패킷이 자주 올 때 이를 방어할 방법은 무엇일까?
- 캐릭터의 위치를 얻는 함수에서 항상 현재 틱 시간을 반영하여 이동 시켜 주는 코드를 본 적이 있다. 이런 접근의 장단점은 무엇일까? 왜 이런 함수를 구현했을까? 이런 고민을 수용하면서 적절하게 처리할 수 있는 방법은 무엇일까?

오브젝트 관리

- 한 공간 내 오브젝트 관리 키는 어떻게 생성하면 좋을까? UUID (GUID), 동적인 시퀀스? 여러 가지 방법을 생각해 보고 장단점과 정확한 구현이 가능할 지 살펴야 한다.
- 캐릭터 아이디와 같은 DB 정보를 키로 사용할 경우 잠재적인 위험성은 무엇인가? 동적인 키를 별도로 발급하면 해당 문제가 없어지는가?
- 오브젝트 소멸 시 바로 지우면 참조하는 곳들에서 문제가 생길 수 있다. Unreal 4는 PENDING_KILL 상태를 두고 다음 처리 시작할 때 지운다. shared_ptr 참조는 괜찮은데 오브젝트가 오래 남을 수 있는 잠재적인 버그가 있을 수 있다. 오브젝트 소멸을 고려한 참조와 사용 방법 중 최적은 어떤 것일까?
- 스킬 시전 효과가 시전 오브젝트와 분리되어 나중에 효과를 적용하는 경우가 있다. 효과 적용 시 시전자의 능력치 등이 반영되어야 한다. 이를 처리할 적절한 방법은 무엇일까?

판단과 행동

판단을 통한 행동의 선택은 HFSM과 Behavior Tree(이하, BT)가 가장 많이 사용된다. 둘은 논리적으로는 같다. 단지, Behavior Tree가 태스크 단위 재사용과 데이터 기반 구성이 더 쉬운 구조로 되어 있다. HFSM은 상태 단위 분할로 실행 코드 작성과 통합하기 수월한 측면이 있다.

- AI를 포함하여 전체 서버 오브젝트 중 액터 (공간에 직접 배치되고 능동적으로 처리되는 단위. 캐릭터가 대표적)의 실행에 대한 판단과 행동을 구조화할 가장 좋은 방법은 무엇일까? HFSM과 BT를 통합할 방법은 없을까?
- 서버 내부의 C++ 코드 상의 행동은 HFSM로 결정하고, NPC의 AI와 같이 외부에서 구성해야 하는 경우에 BT를 사용한다면 어떨까? 다른 구조화 방법은 없을까?

수치와 재화

수치화 재화는 개별 컨텐츠로 구현된다. 여기서는 일반적인 과제들을 살핀다.

- 수치와 재화의 밸런스를 검증할 수 있는 수단은 무엇일까? 로그 기반으로 가능할까? 별도의 시뮬레이터로 가능할까? 플레이 테스트는 MMORPG 밸런스 테스트에 적합한가? 어떻게 MMORPG의 밸런스를 맞출까?
- 서비스 중 데이터 수집을 통해 플레이를 분석하고 그에 맞춰 컨텐츠를 제공하는 과정이 가능할까? 어떻게 하면 가능할까? 어떤 어려운 과제들이 있는가?