**2주차 결과 보고서**

|  |  |  |
| --- | --- | --- |
| **1주차 계획 진척도** | | |
| **구현 내용** | **진척도** | **비고** |
| 게임 로비 구현 | 80% | 로비 맵 배치만 남은 상태 |
| IOCP Echo Test 구현 및 테스트 | 100% |  |
| 데이터 패킷 정리 | 60% -> 90% | 단순 패킷 생성 작업만 남은 상태 |

|  |  |  |
| --- | --- | --- |
| **2 ~ 3주차 계획 진척도** | | |
| **구현 내용** | **진척도** | **비고** |
| UI 제작 | 95% | 꾸미기 작업 남음 |
| 객체 충돌 처리 구상, Skill 종류 확장 | 0% |  |
| 몬스터 구현 | 30% | 기본 몬스터만 있음 |
| 클라이언트 스탯 구현 | 90% | 충돌 처리의 스탯 활용만 남음 |
| 클라이언트 움직임 동기화 | 100% |  |
| 클라이언트 프로토타입 구현 | 30% | 아직 플레이어, 몬스터가 동기화 되지 않음 |

작업 내용 커밋

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

**UI 제작**

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

UI class와 UIMGR class로 UI를 생성만 해주면 자동으로 관리하게끔 구현했다.

텍스트, 폰트, 스크린샷, 그래픽이(가) 표시된 사진

자동 생성된 설명 UI 생성 예시

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

AddComponent와 마찬가지로 AddUI를 통해 UI를 추가할 수 있다.

텍스트, 스크린샷, 블랙, 폰트이(가) 표시된 사진

자동 생성된 설명 실행 결과

UI 디자인이 마음에 안들어 추후에 수정할 예정이다.

**클라이언트 스탯 구현**



텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

객체의 Status 값을 갖는 Component 추가했다.

객체의 피격처리는 이 값들을 조정해서 처리하면 된다.

**클라이언트 움직임 동기화**

클라이언트

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

지금까지 만든 서버에서 들어오는 데이터 종류와 처리하는 부분이다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

단, 서버에서 데이터가 들어오는 족족 처리를 해버리면 미리 프레임워크로 작성한 Update – Render 순서가 꼬일 수 있다. 그래서 서버에서 들어오는 데이터 중 Scene Update 순서에 영향이 갈 경우 Queue에 저장하여 Scene의 Update에서 처리하도록 했다.

서버

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

클라이언트에서 보내는 데이터를 처리하는 부분이다.

클라이언트에서 패킷이 하나 들어오면 그 패킷에 맞는 적절한 처리를 한 뒤, 바로 해당 클라이언트 혹은 모든 클라이언트에 데이터를 전송한다.

**코드의 직관성과 가독성을 위한 정리**

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명 패킷을 사용하는 용도를 쉽게 알기 위한 정리

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명 Monster class Start

Object를 생성하고 Component를 붙일 때, 빈 Object class에 직접 하나하나 붙이는 것 보다

자식 class를 만들어 Start 부분에서 처리하면 편하게 사용할 수 있을 것이라 생각했다.

그 외 중복되는 코드를 줄이기 위한 정리를 했다.



예시: 현재 Scene에 사용하는 전역 변수 값 변경할 수 있도록 변수 생성

자기 반성

사소한 부분에서 시간 소모가 크다.

효율적인 서버 제작에 대한 지식이 만족스럽지 않다.

리뷰

우리 게임의 클라이언트 퀄리티가 다른 조에 비해 떨어지는 느낌을 받았다.