1. Text UI 생성 및 배치

UILayer를 이용하여 Chatting이 아닌 2D Text를 배치하려고 한다.

미리 TextBlock을 배치하고 Text를 적어 둔 다음, Scene마다 활성화돼야 하는 시점이 올 때 Render 하여 구현하려 했다.



우선 UILayer class에 정적 객체로 TextBlock 컨테이너를 만들었다. 그리고 이 컨테이너 안에 모든 고정된 TextBlock을 만들어 두고(NPC와의 대화, Room 정보 등), 필요할 때 출력할 것이다.

이제 TextBlock을 컨테이너에 넣을 것이다.

텍스트, 폰트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명

해당 TextBlock의 고유 ID를 만들어주기 위해 enum E\_UI\_ID을 만들어 어떤 TextBlock인지 구분할 수 있게 했다. 참고로 0번부터 10000번 까지는 Room Text 정보로, Room의 번호를 사용하게 된다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

LineDraw()는 m\_pUITextBlocks을 출력하기 위해 만든 함수이다.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

TextBlock 컨테이너에서 출력할 때는 현재 Scene의 type으로 분기하고, 그 안에서도 현재 Scene의 상황을 추정할 수 있는 방식으로 어떤 Text가 출력해야 하는 지 구분했다.

위의 그림 처럼 현재 Scene이 Game\_Scene인 상태에서 Start NPC와 대화하는 것으로 추정되면 그에 맞는 ID의 TextBlock을 찾아 출력한다.



텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

Text가 전부 한가지 Format과 색상으로만 이루어져 있으면 재미가 없으니 다른 Brush와 Format을 만드는 방식도 만들어 뒀다.



Text를 Render하는 방법을 완성했으니 Text를 추가해 테스트 해야 한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

TextBlock을 생성하기 전 컨테이너를 비워준다.

Text를 추가할 땐 Format과 Brush는 미리 만든 객체의 포인터를 사용하고, Rect로 출력해야 할 부분을 배치한 뒤, m\_pstrText에 WCHR[] 형태의 문자열을 넣어 저장한다.

텍스트, 스크린샷, PC 게임, 야외이(가) 표시된 사진

자동 생성된 설명

위 방식으로 NPC의 대화 상자를 완성한 모습이다. 글씨체나 글씨의 크기, 색상을 변경할 수 있다.

다음은 LobbyScene에서 Room의 정보를 나타내기 위해 Text로 띄울 것인데, 이 TextBlock은 고정된 Text만 출력하는 것이 아니라 게임을 하는 플레이어에 따라 Text가 다를 수 있기 때문에 (Room의 개수, Room 이름 등) 미리 틀만 만들어 두고 Text 내용은 Room\_UI class의 내용을 가져와 저장할 것이다.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

이 부분이 TextBlock의 틀 역할을 할 것이다. 한 페이지당 보이는 Room의 수가 6개 이기 때문에 6개의 TextBlock만 만들었다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

그리고 Page를 넘길 때 Text 내용이 남으면 안되기 때문에 Text 내용을 매 번 0으로 초기화 시켜주고,

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Room\_UI를 Render할 때 해당하는 위치의 TextBlock을 갱신한다.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

이제 Room 정보를 화면에서 확인할 수 있다

1. Boss Monster 동기화

클라이언트에 Boss Monster가 만들어졌기 때문에 이를 서버와 통신하여 움직이도록 할 것이다.

기본적인 움직임과 기본 공격, 애니메이션은 다른 Monster와 별반 차이 없이 진행했다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명







Animation Type, Position, Target, Look 등 동기화 추가

Skill을 동기화 하는 과정에서 문제가 생겼다.

클라이언트에선 Boss의 Skill Animation이 끝나는 순간 Skill 효과가 적용되는데, 서버와 연동하는 과정에서 서버에서 작동하는 Boss가 Skill 사용 중 다른 State로 넘어가 Animation이 끝나지 않게 되어 Skill이 제대로 작동하지 않는 문제가 있다.

이를 해결하기 위해 Boss Monster에 Skill Timer를 둬서 서버에서 트리거를 발생시키면 그 Skill의 시간동안 Animaition이 진행되도록 하려 했다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명 SkillState에 Timer을 추가한 모습이다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Skill을 사용할 때 m\_Timer에 초 단위의 값을 넣어 해당 시간만큼 Skill을 사용하는 동안 State가 변경하지 않도록 한다.

텍스트, 폰트, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Skill을 사용하는 동안 Timer을 작동시키고 m\_Timer의 값이 0이 되면 다른 State로 변경한다.

이론대로라면 제대로 작동할 것 같았지만 Timer의 값이 문제였는지 Boss 객체가 부자연스럽게 움직였다. 이 버그는 다음 주차에 같이 해결하려 한다.