다음 발표는 제가 맡아 이어가도록 하겠습니다. 먼저 제가 맡은 세부 기술 중에 충돌, 층별 레이어 로직에 대해 설명 드리겠습니다.

그림에서처럼 기본적인 오브젝트와의 충돌 로직에 더하여 각 층별로 서로 다른 레이어가 적용되도록 하였습니다.

이 로직을 구현하기 위해 다음 사진과 같이 붉은색 블록과 푸른색 블록을 그려 서로 다르게 배치하고, 1층에 있을 때는 붉은 색 레이어와 충돌하게, 2층에 있을 때는 푸른색 레이어와 충돌하도록 하였습니다.

간단하게 실제 쓰여진 코드를 통해 보여드리자면 다음과 같이 7번 블록에는 붉은색 블록을, 8번 블록에는 푸른색 블록을 넣어 객체화 했으며, 1층에 있을 때는 7번 블록과, 2층에 있을 때는 8번 블록과 충돌을 일으키도록 하였습니다.

오브젝트와의 상호작용은 기존 충돌로직을 활용하였습니다. 실제 충돌이 일어나지 않더라도 활성화된 오브젝트를 표시하기 위해 기존 충돌로직값에서 1픽셀씩 값을 더 주었으며,

캐릭터 위치인 X, Y 값을 불러와 타일 사이즈만큼 나누어주는 식으로 해당 오브젝트의 열과 행 값을 구했습니다.

또한 어떤 오브젝트와 상호작용할 지를 지정하기 위해 Interactive 라는 배열에 각각의 오브젝트 번호를 넣어 활성화하도록 하였습니다.

다음으로 반응형 카메라 기능에 대해 설명 드리겠습니다. 앞서 본두씨도 카메라 기능에 대해 말했었는데요. 저는 캔버스 외의 컨텐츠와 창 크기에 따른 반응형 카메라 기능을 맡아 구현했습니다.

카메라 객체를 생성할 때 window.innerWidth 값과 height 값을 사용하여, 새로고침 하였을 때도 창 크기에 맞게 크기와 위치가 조정되도록 하였으며, 전체 맵 끝에 도착하였을 때 카메라 크기를 조절하여

캔버스가 추가로 그려지지 않도록 하였습니다. 간단한 시연영상 보시고 이어가도록 하겠습니다.

반응형 카메라를 구현하기 위해 주로 쓰인 변수값은 보시는 바와 같이 remainX,Y 값과 Chat.width 값이 쓰였으며, 간단히 도식화하였을 때

반응형 카메라의 넓이를 구하기 위해 전체맵의 가로길이를 먼저 구하고 여기에 innerWidth값과 remain값을 반영하여 값을 대입시켰습니다.

Chat.width의 값은 창 크기가 일정 이하로 줄어들 때 사라지므로 IF문으로 분기문을 만들어 값을 적용하였습니다.

캐릭터 이동 로직은 간단한 영상을 보시고 설명을 이어가도록 하겠습니다. 보시는 바와 같이 키보드로 이동이 가능하도록 구현하였으며, 마우스로도 이동이 되도록 구현하였습니다.

캐릭터 이동 로직 중 키보드 이동을 간단히 설명드리자면, 각 방향을 입력하였을 떄 dirX,Y 값을 주었으며, 이와 동시에 tempX,Y 값에도 동일한 값을 주도록 하였습니다. 그 이유는 다음에 나올 locatX 변수와 마찬가지로 칸 이동을 구현시키기 위해서였으며, 자세한 로직은 시간 관계상 추후 질의응답 시간에 물어보시면 잘 대답해드리겠습니다.

캐릭터 스피드에서 실제로는 저 수치를 그대로 사용하지 않았다. 타일 이동을 위한 로직에 맞춰 타일 크기의 약수만큼 이동해야 했기에 기존 스피드 계산 값에서 가장 가까운 약수를 구하는 공식을 만들어 사용했다.

마지막으로 저희 프로젝트의 수행결과를 간단한 영상으로 소개시켜드리겠습니다.

자신만의 아바타 커스터마이즈, 자유로운 이동과 교류, 전시된 웹페이지 감상이 가능하며

무엇보다 가장 멋진 점은 다른 사람과 재미있게 상호작용 가능하다는 것입니다.

저희들을 자체 평가하자면 이 프로젝트의 완성도는 90%라 보는데요

그 이유는 목표로 했던 기능을 대부분 구현했다는 점에서 저희들 스스로 높은 점수를 주었으며

짧은 기간이라는 점과 두명이라는 적은 팀원이 이루어낸 결과물로는 충분한 볼륨과 완성도를 보여주었다는 점에 있었습니다.

조금 아쉬었 던 점은 많은 인원이 동시 접속시 통신 장애가 있었다는 점과 기존에 영상채팅을 구현하려 하였으나 시간상 추후에 구현하기로 하였다는 점입니다.