| 주차/일자 | 7-8주차 / 2.4~2.17 | 작성자 | 이상민 |
| --- | --- | --- | --- |
| 개요 | 7~8주차 진행 보고서 | | |
| 회의 내용 | 2024.02.12 주간회의  (\* 설날 연휴로 연휴 마지막 날 회의 진행)   * 구현 내용 확인 * 코드 병합 논의 (서버, 클라이언트(애니메이션 블렌딩, 충돌처리))   + 깃허브 브랜치 정리   + 작업 과정 맞춤   + 병합은 매번 회의 때 마다 정리하기 * player.cpp   + 플레이어를 framework가 아닌 Scene에서 작업하도록 수정 필요   + 이 외에도 클라이언트 코드 정리 필요 * 창업 현장 실습 관련 회의   + 세무서에 문의     - 필요 서류, 과정     - 임대차계약서 관련 문의   + 상민: 일단 제가 집주인한테 물어볼게요,,,,   2024.02.16 주간회의 (14:00) | | |
| [다음 주 회의 안건] | | |
| 주간  수행 내용 | [김정훈 - 서버]   * 이동 회전 동기화 작업 및 이동하는 NPC 생성   + 이동 회전 동기화 작업 1. 클라이언트에서 이동을 한 후의 위치를 서버에 보낸다.   2. 서버는 클라이언트에게 받은 위치를 확인 후 이동 가능한 위치인지 확인하고 모든 클라이언트에게 위치를 동기화 한다. (현재는 2048 x 2048이지만 맵이 완성된 후에는 맵의 정보를 통해 이동가능 여부를 판별해야됨)  3. 이동 불가능한 위치로 이동했을 시 서버는 클라이언트의 위치를 마지막 이동 가능했던 위치로 이동시킨다. (Move 패킷의 멤버로 bool 값을 설정하는 방법 vs MoveOK 와 MoveFail 두개의 패킷으로 나누는 방법 중 효율적인 방법을 고민중)   * + 이동하는 NPC 구현하지 못하여 차주에 해야됨.   현재상태: 이동하지 않고 위치’만’ 바뀌는 NPC (차주에 문제점을 찾고 고칠 예정) | | |
| [김진선 - 클라이언트]  **7주차**   * 기존 BB 예제 분석   + file에서 mesh정보 불러올 때(쉐이더에서 해당 객체 buildobj()) bound 정보 받아 m\_xmBoundingBox.orientation에 저장   + Framework.buildobj()에서는 다른 객제보다 바운딩 박스의 쉐이더 먼저 생성   + .orientation에 저장된 내용을 animate 할 때 업데이트 해줌 * 적용      * 수정사항   + 먼저 연산하기 위해 material의 PrepareShader 에서 생성      * + 박스를 라인으로 띄우기 위해 topology를 변경하면 쉐이더 전체의 topology가 다 변경되기 때문에 인자로 topology 어떤거 쓸건지 받아오는 CreateShader() 추가      * + 바운딩 박스의 경우 버텍스 버퍼로 띄우는거라 바운딩 박스를 위한 버텍스 버퍼 연산 추가 함.   **8주차**   * 모델 입출력 기존 코드 수정   + <mesh>에는 기존 fbx의 바운딩박스 정보가, <skinnedMesh>에는 유니티에서 수정한 바운딩 박스 정보가 들어오기 때문에 <skinnedMesh> 상의 바운딩 박스 내용만 사용하기로 결정   + <mesh>의 바운딩 박스 정보를 사용하지 않으며, 모델 export/import 과정의 순서상 <skinnedMesh>의 바운딩 박스 정보대신 입력되므로 해당과정을 삭제      * 모델별 수정&적용   + Robot 모델의 바운딩 박스가 회전되어있으나 회전 내용을 추출할 수 없으므로 직접 값을 조정하여 사용      * + <skinnedMesh>에서 바운딩 박스 정보를 가져올 때 메인 박스를 제외한 다른 객체의 바운딩 박스가 예상치 못한 값으로 들어오는 현상 발생.  원인을 찾기 위해 모델의 bin 파일과 같은 내용의 txt 파일에서 검색해봤지만 해당 값을 찾을 수 없음.   (위-디버깅 화면, 아래-.txt 파일)     * + 섬세한 조준을 필요로 하지 않으며, 위와 같은 문제가 발생하기 때문에 플레이어의 어깨선을 기준으로 한 큰 메인 바운딩 박스만 사용하도록 결정 (살금살금과 점프시의 바운딩 박스의 경우 애니메이션 블랜딩 완료후 수정하기로 함)   + 수정한 모델은 prefeb 형식으로 공유 드라이브에 업로드     \* 피봇 위치 : 모델 정수리   * Scene 변환을 위한 분리   + 주간 회의에서 의논한 바와 같이 Player의 정보를 scene에서 연산하도록 수정을 해야 추가 작업이 가능하여 기초 작업만 함. | | |
| [이상민 - 클라이언트]  7주차   * 플레이어를 3개를 렌더하는 것을 진행하였다. 각 플레이어는 다른 모델을 적용시켜서 우리가 사용할 3가지 캐릭터들을 움직일 수 있도록 하였다. 3가지 캐릭터들 마다 이동을 다르게 적용시키려 했지만 키보드 입력값을 받는 것 보다는 서버에서 직접 값을 넘기며 테스트 하는 방식이 효율적일 것 같아서 기능을 추가하지는 않았다. * 추가적으로 플레이어의 캐릭터들을 다른 자료구조 형식으로 사용하려 했는데 3가지의 플레이어만 있기 때문에 굳이 다른 방법은 사용하지 않고 나중에 객체의 개수가 많은 적 객체를 만들 때 vector를 사용해서 구현 하기로 결정하였다. * 플레이어 캐릭터의 AAD\_OBJ 기능과 REMOVE\_OBJ 기능을 구현했다. 서버에서 연결이 되면 캐릭터가 렌더되는 방식을 확인하기 위해서 필요한 기능이라고 전달받아서 구현하였다. 그 방법은 각 플레이어의 실제 메모리 값이 아닌 렌더를 하는지 마는지를 선택하는 방법으로 구현하였다. 그리고 혹시 클라이언트 쪽에서도 이 기능을 사용할 수도 있을 것 같아서 키보드 기능도 구현하였다.        * 모두 렌더를 키고 끌 수 있으며 위의 사진들이 기능을 동작했을 때와 동작하지 않았을 때의 상태이다. * 키보드 입력값을 받을 때 this = !this 식으로 사용했다가 키보드의 입력이 빨라서 거의 랜덤으로 적용되길래 여러 키보드를 사용했다.   8주차   * 애니메이션 블렌딩을 성공한 주차이다. 하지만 이 기능을 모든 애니메이션에 적용시켜야 하기 때문에 이 기능을 이제 다음 주차에 적용시킬 예정이다. * 우선 구현하기 위해서 사용했던 것은 졸업한 선배의 블렌딩 소스를 받아서 그것을 바탕으로 제작했었다. 프로젝트의 이름은 DreamGame이라는 프로젝트를 받았다. 추가적으로 프로젝트를 보면서 확인하기 편하게 소스 파일들을 정리할 필요가 있다고 느꼈다.      * Run + Idle 한 것이다. 사용한 알고리즘으로는 Matrix4x4 안에 있는 Interpolate()함수를 사용해서 두 애니메이션을 선형 보간 하였다. 아직 수정이 필요한 단계이며 키프레임이 하나만 출력되는 문제가 있다. 하지만 여태까지의 과정 중에서 가장 결과에 가까워졌으며 이 정도의 문제는 금방 해결할 수 있을 것 같은 문제이다. | | |
| 특이사항 | * 설날 연휴로 7-8주차 보고서를 통합하여 작성 | | |
| 다음 주 수행 계획 | [김정훈 - 서버]   * NPC(로봇) 이동   [김진선 - 클라이언트]   * 플레이어 프레임워크 수정 * 각 씬 구성   [이상민 - 클라이언트]   * 애니메이션 블렌딩(모든 모델 및 애니메이션) | | |