| 주차/일자 | 1주차 / 2023.12.26~30 | 작성자 | 이상민 |
| --- | --- | --- | --- |
| 개요 | 방학 기간 중의 계획 내용 | | |
| 회의 내용 | 2023.12. 28일 주간회의  [2023.12.28~2024.03.02(1~10주차)] 방학 동안 할 일 계획내용   * 공통   + 매주 회의는 금요일 오후 3시 * 클라이언트/서버 초기 코드 작성 * 모델 에셋 탐색 / 확정 * 지형지물   [SCI-FI Modular Kitbash | 3D Sci-Fi | Unity Asset Store](https://assetstore.unity.com/packages/3d/environments/sci-fi/sci-fi-modular-kitbash-39522) (탈락)  [Sci-Fi Containers Props | 3D Sci-Fi | Unity Asset Store](https://assetstore.unity.com/packages/3d/environments/sci-fi/sci-fi-containers-props-64345) (맵 오브젝트, 구매 예정)  [Sci-fi box vol 3 | 3D Sci-Fi | Unity Asset Store](https://assetstore.unity.com/packages/3d/environments/sci-fi/sci-fi-box-vol-3-142353) (아이템 박스, 구매예정)   * 캐릭터   [CyborgPack | Characters | Unity Asset Store](https://assetstore.unity.com/packages/3d/characters/humanoids/sci-fi/cyborgpack-217954) (플레이 캐릭터,구매 예정, 구매 전까지 무료 에셋 대체 사용)  [Robot\_2 | 3D Characters | Unity Asset Store](https://assetstore.unity.com/packages/3d/characters/robot-2-236130)(보스 로봇, 구매 예정)  [Robot Kyle | URP | 3D Robots | Unity Asset Store](https://assetstore.unity.com/packages/3d/characters/robots/robot-kyle-urp-4696) (일반 로봇, 무료 에셋 확정)   * 초기 제작 무료 에셋 * 로봇   [Robot Kyle | URP | 3D Robots | Unity Asset Store](https://assetstore.unity.com/packages/3d/characters/robots/robot-kyle-urp-4696) (일반 로봇, 확정)   * 모션 애니메이션   <https://assetstore.unity.com/packages/3d/animations/basic-motions-free-154271> (캐릭터 이동 모션, 임시)  < 주차별 계획>   * 클라이언트 (이상민, 김진선)   + 다렉 프레임워크 제작 (~3주차)     - 모델 로드   + 애니메이션 적용 (~7주차)     - 이동, 점프, 살금살금     - 로봇이 공격하면 기절(애니메이션 적용) * 충돌처리(~8주차) * 스테미너 시스템 (8주차) * 맵 오브젝트 배치(3~10주차) * 서버   + 게임 서버 프로그래밍 텀 프로젝트 코드 분석 ( ~3주차)   + 패킷 타입 정하기 ( ~4주차)   + 맵 지형 띄우고 충돌 처리 ( 4~9주차)   + NPC의 이동과 상태 변환( 6~10주차) | | |
| [다음 주 회의 안건] (2024.01.05)   * git repository 생성 (초기 프로젝트 코드 완성해 오기) * 개인별 진척도 확인 * 문제점 파악 및 피드백 | | |
| 주간  수행 내용 | [김정훈 - 서버]  게임 서버 프로그래밍 수업 복습   * IOCP의 동작 원리   + 서버 메인 루프의 GetQueuedCompletionStatus()를 통해 I/O의 완료를 검사한다.   + 클라이언트가 접속 시 CreateIoCompletionPort()로 IOCP에 연결 후 Recv()상태.   + I/O의 결과 Recv()면 패킷 처리를 하고 다시 Recv상태로 돌아오고, Send()의 완료가 검사되면 메모리를 반환한다.   + 소켓I/O 이외의 timer같은 작업들은 PQCS()로 처리한다. * 멀티 쓰레드 환경에서 클라이언트들과 NPC들은 어떤 컨테이너로 관리할까?   + 클라이언트는 연결/종료 등을 여러번하지만 NPC는 항상 고정돼있음.   + 같은 class로 생성하면 필요 없는 멤버들로 인해 메모리 낭비가 심하고 다른 class로 생성하면 비슷한 역할을 하는 함수들을 중복해서 작성해야된다.   + 해결 방법: 하나의 컨테이너로 관리하되, 컨테이너에 포인터를 저장해서 두 개의 방법들의 장점만을 취한다. (but 구현 난이도의 단점)   + 필요한 공부: 포인터를 사용하여 컨테이너를 관리하기 위해 Shared\_ptr를 공부해야함. (날포인터를 사용하고 data race를 극복하는 방법도 있음) | | |
| [김진선 - 클라이언트]  <Direct3D 방학특강>   * **복습 정리 / 본 오프셋 이론 강의 (ppt ->2학기 animation)**   + 각 스킨 메쉬들이 뼈의 좌표계로 따라가기 위해 각 정점들의 정보를  to root 변환 각 뼈의 정점 좌표계로 변환하고 그것의 역행렬을 통해 얼마나 이동했는지에 대한 정점 값을 구함   >> (각 뼈마다)본 오프셋 행렬을 만들고 이의 역행렬을 이용해 애니메이션 구현   * 본 오프셋 행렬 정보는 어디서? 메쉬? 애니메이션?   >> 만약에 스킨 메쉬가 여러개인 경우 각 메쉬의 개수만큼 본 오프셋 행렬이 필요하기 때문에 (뼈대의 계층구조는 1개) 본 오프셋 행렬 정보는 메쉬에 있어야 한다.   * 애니메이션은 뼈의 트랜스폼을 바꾸는 것이며, 각 메쉬는 이 뼈를 따라가도록 되어있음. * 애니메이션 == 시간에 대한 각 뼈들의 움직임을 나타내는 변환행렬 \* 꼭 뼈 개수만큼 변환 행렬이 있는 것은 아님 * 보통은 이를 위한 행렬 계산 과정을 Vertex shader에서 해줌 * **샘플 프로젝트 분석&수정 / SampleProject 7-8-5**   + 해당 프로젝트의 문제점       (위)모델 export 형식과 (아래)모델을 읽어오는 부분에 차이가 있음  >> BoneIndeces 와 BoneWeights  (아래) 모델을 읽어오는 함수인 LoadSkinInfoFromFile() 에서는 BoneWeightes에서 인덱스 처리도 같이 해줌    >> 모델 export 형식에 맞게 Mesh.cpp의 LoadSkinInfoFromFile() 함수 수정    >> 졸업작품에 사용할 모델로 애니메이션 출력 | | |
| [이상민 - 클라이언트]  <3D 게임프로그래밍2 복습>  32비트의 루트파라미터는 64개의 32비트 배열로 구성됨  -> 즉 루트 파라미터의 값은 최대 64개까지 줄 수 있음, 잊고 있었던 내용을 다시 한 번 상기해보았다.  이 샘플 프로젝트(SampleProject 7-8-5)를 완벽히 이해하기 위해서 우선 FrameWork를 살펴볼 필요가 있었다.    프레임 워크의 흐름도 인데 사실 어렵다고 생각했던 Direct3D 프로젝트 자체는 가운데 파란 박스 안의 내용만 제외한다면 생각보다는 단순한 흐름으로 진행된다는 것을 알았다.  이제부터 내가 집중적으로 공부할 연구 분야인 효율적인 메모리 관리는 OnCreate()부분의 게임 객체 생성과 그에 따른 렌더 타겟 뷰 생성/설정 그리고 OnDestroy()부분이다.  그렇다면 다시 샘플 프로젝트로 돌아가서 문서와 동일하게 게임 객체를 관리하는지에 대해 분석해 보자면     1. 이 샘플프로젝트는 Scene의 BuildObject()에서 지금부터 사용할 리소스의 모든 크기를 더한 만큼의 DescriptorHeap을 할당 받는다.    1. 이 부분은 유지보수적인 부분과 가독성면에서 심하게 효율성이 떨어진다. 상수값부분을 Define으로 정의 하거나 혹은 DescriptorHeap을 Object안에서 할당 받는 등의 형식으로 수정이 필요해 보인다.      1. 이 프로젝트에서 모든 객체는 클래스 CGameObject를 상속받아 사용된다. 그래서 1번 작업 후에 CGameObject\*\*자료형인 멤버변수를 만들고 객체 갯수만큼 할당받아서 사용한다.(Shader 내부에서 CGameObject를 할당받는 객체도 존재)    1. 이 부분도 다른 자료구조와 객체의 갯수를 통으로 작업하는 방식이 아닌 다른 방식으로 연구해서 작업해야 겠다.      1. 그 후 모델의 정보를 담을 수 있는 클래스 CLoadedModelInfo의 \*자료형을 만들어서 그 안에 모델의 정보를 읽어서 가르킨다.    1. 이 부분은 단순한 작업이어서 이 방법이 가장 직관적이고 효율적일 것 같다.      1. 그 후 m\_ppHierarchicalGameObjects의 상수 번째 포인트 배열에 상속받은 클래스인 CAngrybotObject를 할당받아 각각 필요한 정보를 넘겨 준다.   전체적으로 보았을 때 오브젝트 혹은 리소스 관리 부분에서 다양한 버전을 사용하는 사람들이 알아보기 쉽게 예전부터 사용한 방법을 쓰거나 알아보기 쉽게 상수로 표현을 많이 하였지만 우리는 이 코드를 계속 수정하고 추가할 것이기 때문에 기반을 잡을 때 자료구조 선택과 상수로 표현된 부분을 수정할 때 신경을 쓰고 이어서 다음 주에는 그 부분 수정과 함께 수정된 방식으로 우리가 사용할 에셋을 불러와볼 예정이다. | | |
| 특이사항 | 없음 | | |
| 다음 주 수행 계획 | [김정훈 - 서버]   * 게임 서버 프로그래밍 텀 프로젝트 코드 분석 * 서버 초기 코드 작성(기존의 텀 프로젝트 코드에서 변경되는 부분 수정)   [김진선 - 클라이언트]   * 다렉 개발 기반 환경 제작(평지+오브젝트 1개) * 애니메이션 로드 공부   [이상민 - 클라이언트]   * 프레임워크 수정(모델 로드, 3D 리소스 관리) * 애니메이션 로드 공부 | | |