|  |  |  |
| --- | --- | --- |
| EmberFall 개발 일지 | | |
| **6주차** | **2025.02.02 ~ 2025.02.08** | 회의 내용 |
| 금주 회의는 E 동에 모여 주간 작업물을 공유하고, 함께 작업하는 시간을 가졌다. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| EmberFall 개발 일지 | | 작성자 : 김승범 | |
| **6주차** | **2025.02.02 ~ 2025.02.08** | | 이번 주 진행 사항 |
| **2월 2일**  ShaderManager 에서 관리할 Shader 객체를 완성하였다.  샘플 셰이더 파일을 생성하여 동작을 테스트하는 과정에 동작에 문제가 있어 그 문제를 수정하였다.  **2월 3일**  간단한 3차원 도형을 띄우기 위해 뼈대가 없는 PlainMesh 클래스를 작성하였다.  **2월 3일 – 2월 6일**  PlainMesh 와 Material ( Shader ) 를 연결하여 렌더링 엔진을 완성하기 위한 고민을 하였다.  먼저 모델을 렌더링하기 위한 독립된 System 을 설계하였으나 ( ECS 중 System )  이 방법은 컴포넌트-시스템 차원에서 컴포넌트의 역할이 크게 떨어진다는 문제가 있고,  셰이더를 기준으로 하는 배치 렌더링이 복잡해진다는 문제가 발생하였다  또한 DirectX12 와 게임 로직 처리를 분리하는 것이 프로젝트의 주 뼈대였으나, 이 방법은 게임 로직 파트에서  과하게 DirectX12 를 참조하는 문제도 발생하였다.  따라서 일반적인 System 을 사용하되, System 사이의 업데이트 우선 순위 설정을 가능하게 하고,  DirectX12 를 참조하는 파트는 Renderer 에서 별도로 만들어 의존성 주입 방식을 통해 구현하기로 결정하였다.  ( 만들고 지우는 과정을 반복하다 보니 상당히 많은 시간이 소요되었다 )  **2월 7일**  해당없음  **2월 8일**  6일 결정된 구조를 구현하기 위해 Renderer 에 MeshRenderManager 을 만들었다.  이 과정에서 DefaultBuffer 가 누적 버퍼 작성이 불편한 부분이 있어서 버퍼에 값을 누적해서 쌓을 수 있도록  변경하였다. | | | |
| EmberFall 개발 일지 | | 작성자 : 김성준 | |
| **6주차** | **2025.02.02 ~ 2025.02.08** | | 이번 주 진행 사항 |
| 02-02  해당없음(본가에서 기숙사 이동)  02-03  서버에서 클라이언트에게 플레이어들의 위치, 회전, Scale을 모두 묶어서 행렬로 만든 후에 송/수신하는 기능을 만듦 위치를 송/수신하는 데에는 문제가 없었으나 행렬을 받아 업데이트 하는 과정에서 불편함이 있어 그냥 다시 위치, 회전, Scale을 따로 나누어 보내는 방식으로 바꾸었다.  02-04 서버에 간단한 타이머를 만들고 Delta Time을 이용해서 업데이트 할 수 있도록 했다. 위치 송/수신 플레이어 입장/퇴장 처리를 만들었으므로 여러 클라이언트가 접속하여 플레이하는 상황을 테스트하던 중, 데드락이 발생하여 코드를 다시 살펴보고 있다.  02-05 데드락 해결. PacketHandler부분에서 여러 쓰레드가 버퍼에 쓰기 연산을 하고 하나의 쓰레드가 읽기 연산을 하는 상황을 가정해서 두개의 atomic\_bool 변수로 이를 제어하고 있었는데, 이 두개의 atomic\_bool 변수로 동기화를 해결하려는 발상이 문제였다…  atomic\_bool 변수에서 wait하는 코드와 store 코드 사이에서 Interrupt가 발생하면 제대로 동작하지 않는다는 것을 깨닫고 Locking으로 문제를 해결하기로 했다. 이미 하나의 버퍼에 쓰는 작업은 여러 쓰레드가 동시에 접근해도 안전하게 만들었고, 읽기와 쓰기가 동시에 일어나지 않아야 하기에 SRWLock을 사용하여 문제를 해결했다. 다만, SRWLock에서 쓰기 연산에 SharedLock, 읽기 연산에 ExclusiveLock을 사용했다.  이후에는 실외지형의 높이맵을 읽고 위치가 float일 경우 적절히 보간 하여 높이를 반환하는 기능을 만들었다. 테스트용 클라이언트에서 GL로 실외지형을 그리는데 문제가 생겨 테스트는 내일 하기로 했다. | | | |
| EmberFall 개발 일지 | | 작성자 : 김성준 | |
| **6주차** | **2025.02.02 ~ 2025.02.08** | | 이번 주 진행 사항 |
| 02-06 충돌감지와 처리를 위한 Collider 클래스를 만들었다. Collider 클래스에서는 DirectX 라이브러리의 Bounding~~~ 들을 이용해서 서로 교차하는지를 이미 만들어진 Intersects함수로 판단하도록 했다. 충돌처리 코드를 적다 보니 GameScene에 너무 의존하는 경향이 있어 충돌처리를 담당할 새로운 클래스 CollisionWorld를 만들기로 했다. 옵저버 패턴에서 영감을 받아서 만들었으며 다음과 같이 작동하도록 했다. 1. ColisionWorld 에서는 그룹을 단위로 충돌처리를 진행한다. 2. 그룹은 두 가지 타입의 오브젝트를 위한 리스트로 구성된다 (std::pair<std::vector, std::vector>) 3. 각 오브젝트는 CollisionWorld의 특정 그룹에 들어갈 수 있으며 충돌이 일어나면 GameObject의 OnCollision함수 호출을 통해 이를 통지한다.  CollisionWorld를 만들고 충돌 감지가 제대로 되는지 까지 확인했다.   실외지형 부분에서 테스트용 클라이언트에서 실외지형을 그리는 것에 성공했으나 서버에서 송신하는 y좌표와 실제 그려지는 실외지형이 맞지 않는 버그가 있었다. 이미지를 읽을 때에 상하를 반전해서 읽어야 하나 서버에서는 그대로 읽고 있었기에 이를 수정하고 제대로 동작하는 것을 확인했다.  FlatBuffer 적용은 나중에 다시 생각해보기로 했다. 아직 패킷의 크기, 종류도 많지 않으며 게임 오브젝트의 종류도 하나로만 진행하고 있기에 이르다고 느껴졌고, 아직 FlatBuffer에 대해서 잘 안다고 하지 못하기에 보류하기로 했다.  02-07 기본적인 물리연산을 위해 중력을 적용하고 속도를 기반으로 여러 속도를 누적시켜 최종 움직임에 반영하는  물리연산을 만들었다. 최대한 간단하게 만들었으나 여러가지 문제점이 있어 내일 고치기로 했다.  1. 두 물체 A, B가 겹쳐 있을 때 A에서 B로가는 방향, B애서 A로 가는 방향으로 적절하게 서로 밀어내게 코드를 작성했으나 충돌 후 완전히 밀어내는데 시간이 걸리거나, 너무 많이 밀어내어 위치가 순간적으로 너무 멀어지는 문제들이 있었다. 2. 점프 후 상대방의 위에 올라갔으나 파고 드는 현상. | | | |
| EmberFall 개발 일지 | | 작성자 : 김성준 | |
| **6주차** | **2025.02.02 ~ 2025.02.08** | | 이번 주 진행 사항 |
| 02-08 OBB를 이용해 충돌 처리를 하는 과정에서 결국은 겹친 정도를 계산해서 가장 짧은 이동거리로 충돌상태를 해소할 수 있는 벡터를 찾아야 한다. AABB에서는 간단한 연산으로 가능하나 OBB에서는 분리축 이론을 이용해서 이를 구현하기로 했다.  먼저 OBB의 8개 꼭지점을 월드 좌표계로 옮긴 후에 평행하지 않은 세 모서리 벡터, 면의 법선 벡터를 얻고, 두 OBB에 대해서 법선 벡터를 하나씩 선택하면서 8개 꼭짓점을 사영한 후 겹치는 영역을 계산하고,  두 OBB의 모서리 벡터를 각각 하나씩 선택해서 외적 한 후 나온 벡터에 8개 꼭짓점을 사영한 후 겹치는 영역을 계산한 후 겹치는 영역이 가장 작은 축을 찾아 그 축을 방향 벡터로, 겹치는 영역을 길이로 가지는 벡터를 사용하여 충돌처리를 진행했다.  연산량은 많은 것 같지만 결과적으로 어제 생긴 문제점을 모두 해결했기에 꽤 만족스러운 결과가 나온 것 같다.  충돌처리를 어느정도 완성하고 테스트하는 과정에서 데이터를 송신하는 쪽에서 오류가 생겨 문제를 파악하고 있다. | | | |
| EmberFall 개발 일지 | | 작성자 : 정영기 | |
| **6주차** | **2025.02.02 ~ 2025.02.08** | | 이번 주 진행 사항 |
| **02-02**  그림자 구현을 위한 프로젝트를 수정하는 작업을 진행하였다. 조명 회전, 오브젝트와 평면을 구현하였다.  **02-03**  깊이맵 텍스쳐를 생성 후, 깊이값을 저장, 깊이맵 출력을 위한 렌더링 패스를 구현하였다.  **02-04**  깊이맵에 깊이값 저장에는 성공했지만, 디버깅을 위한 뷰포트가 생성되지 않는다. 문제를 해결 못했다.  **02-05**  뷰포트 미생성 해결에 대부분의 시간을 할애했다. z값 오류를 발견 후, 수정하였다.  **02-06**  깊이맵을 디버깅의 용도로 활용하기위해 렌더링 했으나, 노이즈가 발생하였다.  노이즈를 해결하였고, 디버깅 레이어를 구현 후, 활성화 시켰다.  **02-07**  그림자 매핑을 시작하기 앞서, 텍셀와 픽셀의 매핑을 시각적으로 확인하기위해 이미지 텍스쳐를 활용한 텍스쳐 투영 매핑을 구현하였다. 투영을 성공적이지만 깊이값 판별을 적용시키면 음영구현에 문제가 생겼다.  **02-08**  음영구현의 문제점을 해결하였다.  기존의 스팟조명을 태양빛 구현을 위한 방향성 조명으로 전환하였고, 그림자 구현 환경을 평면과 다양한 오브젝트로 변경하였다. 조명에서의 문제점이 발견되어 보완을 진행하고 있다. | | | |

|  |  |  |
| --- | --- | --- |
| EmberFall 개발 일지 | | |
| **6주차** | **2025.02.02 ~ 2025.02.08** | 다음 주 목표 |
| 1. 김승범 ( 클라이언트 )   일요일까지 Render 시스템 완료하기  늦어도 화요일부터 애니메이션 파싱 시작하기.  애니메이션 파싱 및 애니메이션 렌더링 준비   1. 김성준 ( 서버 )   1. 플레이어가 아닌 다른 오브젝트 정보를 송/수신 2. 데이터 송신 문제 해결   1. 정영기 ( 클라이언트 )   그림자 매핑 완료, 계산셰이더, 블러링, 테셀레이터 작업 | | |
| **특이사항** | | |
| 김승범 ( 2월 7일 ) – 이날 두통이 심해 쉬었다. | | |