| **비단길(Silk Road)** | |
| --- | --- |
| **19주차** | **2024.04.28 (일) ~ 2024.05.04 (토)** |

**■ 다음 주 세부 계획**

**▶ [권순원]**

* 플레이어 패킷 완성
* 연결된 플레이어끼리 애니메이션 동기화

**▶ [박준영]**

* 플레이어 추가 스킬 구현
* 데모용 맵 추가 작업(폭발 테스트 등 가능하게 오브젝트 배치)

**▶ [엄장헌]**

* 회전물리 적용

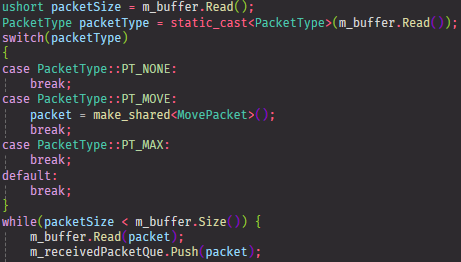
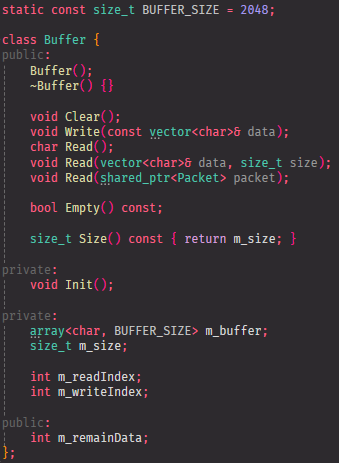
**■ 작업 일지**

**▶ [권순원]**

이전에 패킷 작업을 계속하면서 직렬화와 패킷 클래스 등을 계속 제작해보았는데 생각보다 패킷 타입을 확인하고 패킷을 캐스팅해 적용하거나, send와 recv 타이밍 문제 등이 계속 발생했다. 혼자서는 너무 많은 시간이 필요할 것 같다는 생각이 들어 작년 졸작 서버를 담당한 친구에게 요청해 조언을 구했다.

일단 현재 작업하던 패킷 직렬화는 포기하기로 했다. 대신 수신받은 패킷을 가공하고 적용하는 과정에 중점을 맞추기로 했고, 구조 개선에 대한 조언을 받았다.

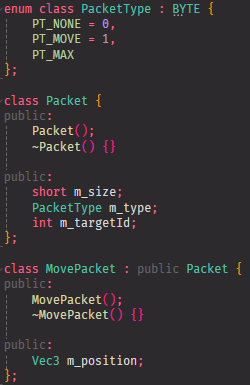
우선 recv할 때 데이터를 받을 버퍼 클래스를 제작했다. 버퍼 클래스는 데이터를 원형큐의 형태로 보관하고, 원형큐는 클래스 내에 인덱스를 선언해 직접 제작했다.



버퍼는 클래스는 Network 클래스에 하나씩 존재하고 해당하는 클라이언트로 들어오는 모든 데이터를 저장한다. recv 함수에서 임시로 vector<char>로 데이터를 받아오고 이를 Write 함수를 통해 저장하는 방식으로 하였다.

데이터를 받아온 후에는 버퍼의 맨 앞의 데이터를 읽어 현재 패킷의 사이즈를 확인하고 다음 데이터를 읽어 패킷의 타입을 확인한다. 그리고 타입에 따라 패킷의 종류를 확정짓고 버퍼에서 데이터를 읽어와 클라이언트에 적용하기 위한 LockQue에 삽입한다.

대략 이런 과정을 통해 패킷을 받는 형식으로 변환하였다. 이에 따라 패킷 클래스도 변경했다.



아직 다른 팀원들이 만든 플레이어에 적용하기 위한 패킷이 아닌 임시로 만든 패킷이지만 이런 형태로 패킷의 종류를 늘려나갈 예정이다.

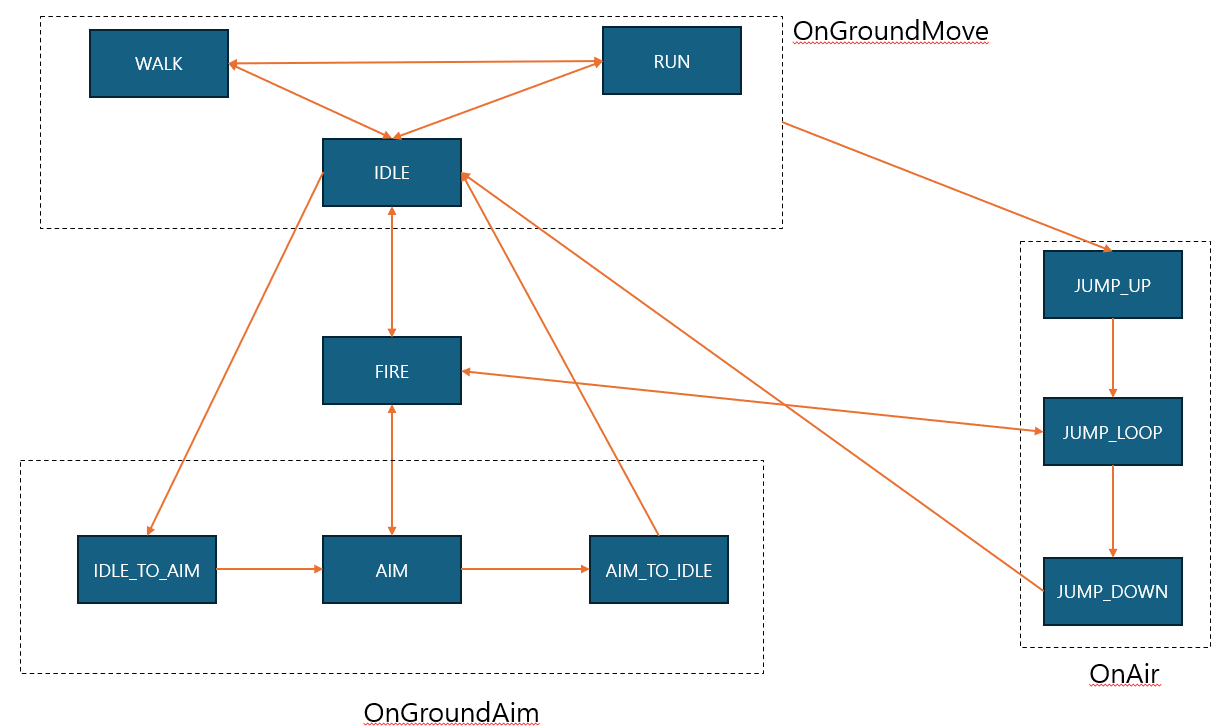
위 코드에서 문제가 될 것 같은 부분이 있는데, 버퍼에 들어오는 패킷의 순서가 잘못되는 경우이다. 현재 코드에서는 게스트는 호스트하고만 통신하고 호스트는 각 게스트와 각각의 스레드를 통해 통신을 한다. 때문에 문제가 발생할 확률이 적을 것이라고 생각한다. 하지만 호스트와 한 명의 게스트하고만 통신 과정에서 보내진 패킷이 어떤 문제로 인해 순서가 꼬인다면 현재 패킷에서 시간이나 패킷의 아이디를 삽입해 해결해야하는 문제가 생길 수도 있다고 생각하고 있다. 앞으로 이 점을 염두해두고 테스트와 코딩을 해야할 것 같다.

그리고 한 가지 고민해야할 부분이 남아있다. 아직 완전히 작업하지 않은 패킷을 적용하는 부분에 대한 문제인데, 현재 네트워크 패킷 업데이트를 씬을 업데이트하기 전에 진행한다. 만약 네트워크에서 꼭 적용되어야 하는(예를 들면 몬스터의 체력) 데이터가 적용된 다음 씬 업데이트를 하면서 덮어씌워 진다면 한 플레이어 화면에서는 몬스터가 죽었지만 다른 플레이어의 화면에서는 죽지 않고 움직이는 모습을 볼 수도 있게된다. 그렇다고 네트워크 업데이트를 씬 업데이트 뒤에 수행하게 된다면 호스트를 기준으로 움직여야 하는 게임이 게스트의 데이터로 덮어씌워지고 다시 호스트 기준으로 움직였지만 다시 덮어씌워지는 문제가 발생할 수도 있을 것이란 생각이 들었다. 때문에 패킷 적용도 일반 업데이트와 Late 업데이트로 나눠야하는 고민을 했다. 하지만 이 문제는 아마 패킷의 수가 많아지면 발생할 확률이 높아질 것 같기에, 이 문제 또한 염두해두고 작업을 진행할 예정이다.

**▶ [박준영]**

기존의 플레이어의 상태별 애니메이션 변경, 동작등은 단순하게 조건문의 나열과 중첩으로 처리했었다.

본격적인 컨텐츠 개발을 하며 이러한 방식으로는 다양한 플레이어의 동작들을 처리하기에 코드가 매우 복잡해지고 유지보수가 불가능한점에 의해 계층적 FSM을 적용하여 플레이어의 상태를 컨트롤하도록 하였다.

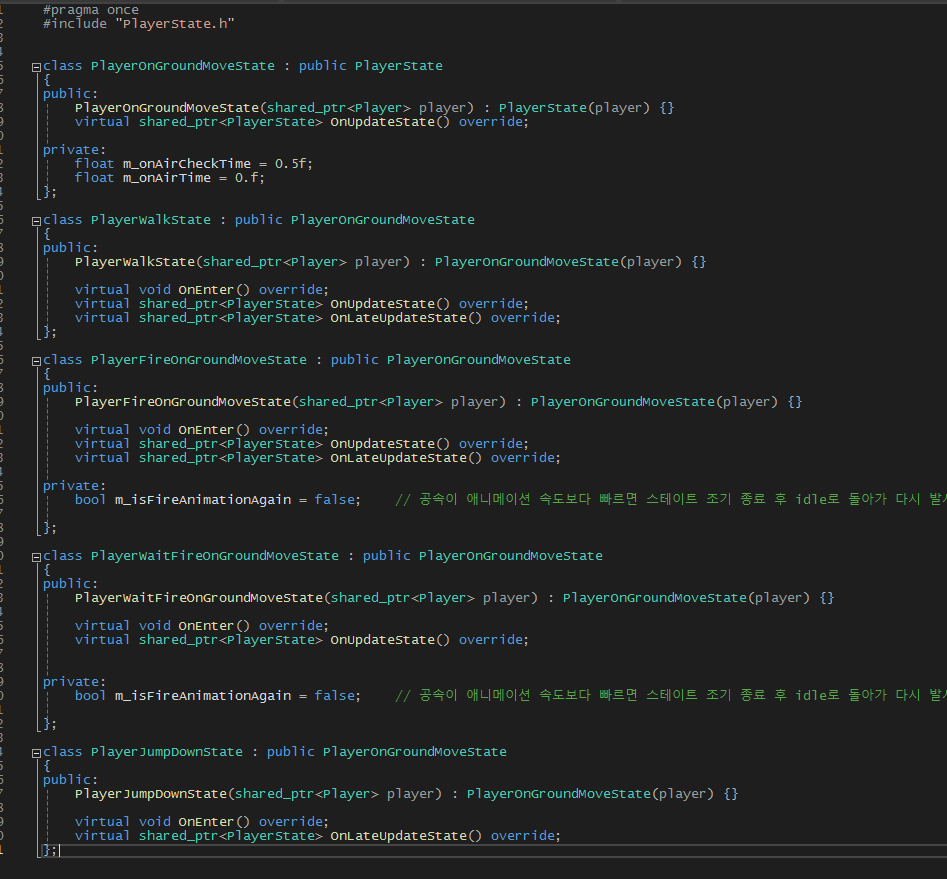
****

플레이어의 상태간 전환을 간단하게 나타낸 흐름도이다.

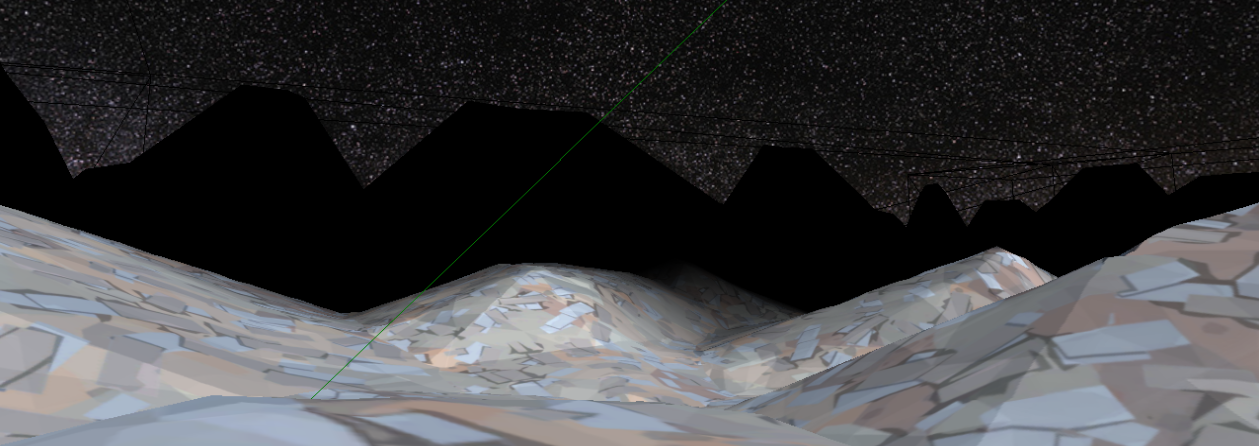
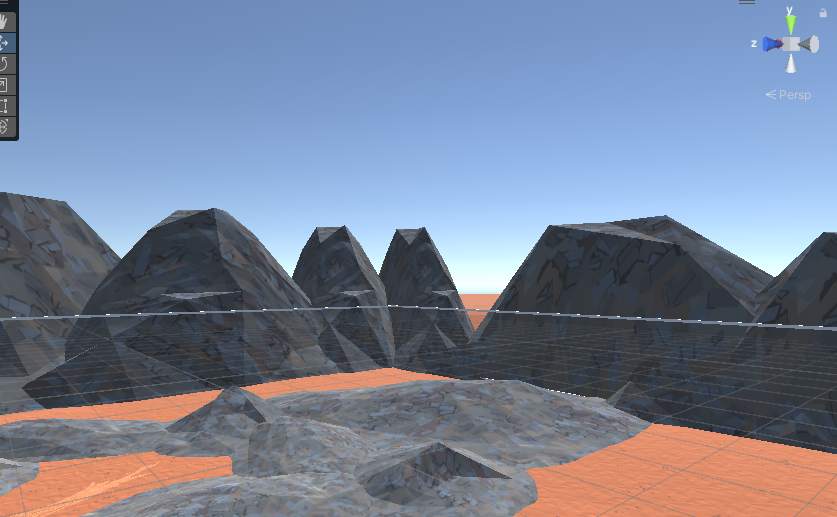
IDLE\_TO\_AIM이나 JUMP\_UP 등의 특정 상태로 전환되기 위한 애니메이션을 상태로 구분하였다.

또한 플레이어의 기본적인 지형상에서의 이동을 총괄하는 OnGroundMove, 지형상에서의 조준 상태를 총괄하는 OnGroundAim, 점프나 추락등의 상황에서 공중에 떠 있는 상태를 총괄하는 OnAir라는 부모 클래스들을 만들고, 세부적인 상태들이 이를 상속받도록 하였다.

이에 기반하여 idle상태, 플레이어를 감지하고 따라오는 walk 상태, 플레이어가 일정 범위 내에 들어왔을때 공격을 수행한는 attack 상태를 통해 적 객체또한 개발중이다.

****

추가적으로 기존에 맵의 끝을 제한하는 것을 단순히 코드로 포지션 이동을 막기만 했었는데, 자연스러운 비주얼과 안개 적용시 산맥이 보이도록 하기 위하여 맵 가장자리에 돌산을 이용한 경계용 메시를 배치하고, 거대한 바운딩박스를 4개 배치하여 맵의 가장자리를 막았다.

****

또한 플레이어의 발사 동작 시 sphere 형태의 콜라이더를 갖는 bullet을 생성하여 발사한다. bullet은 Player 스크립트 내에서 오브젝트 풀링 방식으로 관리한다.

30개 가량의(이는 플레이어의 공격속도에 따라 늘릴 수 있다) bullet 오브젝트를 미리 생성하여 vector에 담아두고, 발사 동작 시 bullet 스스로 플레이어의 위치로 이동 후 메인카메라의 Look 벡터 방향으로 Addforce를 수행하여 날아가게 된다.

이때, bullet pool 내에서 아직 발사되지 않은 bullet들은 Active 상태를 false로 하여 Render, Update, LateUpdate, FinalUpdate를 수행하지 않게 한다.

Bullet은 발사되자마자 자신의 상태를 Active하여 이동과 Render등을 수행하고

발사된 Bullet은 일정 거리를 이동하거나 다른 객체와 충돌하였을때 Active 상태를 false로 하여 다시 Bullet Pool 내에서 자신이 발사되기를 대기한다.

이를 구현하기 위해 기존의 GameObject 구조를 수정하여 Active를 On/Off 할 수 있도록 하였다.

****

**▶ [엄장헌]**

기존 계획이었던 1인칭 시점을 위해 마우스를 숨기고, 프레임마다 마우스 이동량을 계산한다. 카메라를 player의 상대적 위치를 따라가도록 SetParent설정하고 마우스 움직임 만큼 y축 회전을 적용한다. 이때 x축 회전의 경우 플레이어가 비정상적으로 회전되기 때문에 카메라의 localRotation만 조정한다.

움직임에서 플레이어의 look벡터, right벡터를 이용해서 움직임을 구현하는데

y의 요소를 제거하고 addForce를 이용해 구현

****

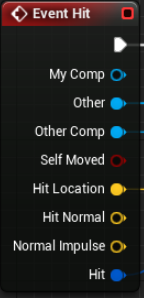
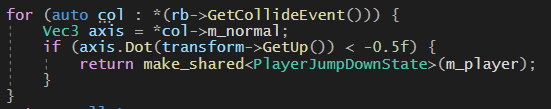
물리엔진 관련해서는 마찰력을 추가하였다.

m\_frictionCoef를 조절하여 0의 경우 마찰력이 없는 경우이고, 커질수록 마찰이 올라간다. 또한 중력이 적용되는지 여부와 중력의 수치를 조절할 수 있게하여

객체별로 다르게 적용가능하다.

다음으로는 충돌 이벤트를 각 객체가 받아 올 수 있게 만들었다.

모든 RigidBody를 갖는 객체는 GetCollideEvent함수를 통해 해당 프레임에 자신과 충돌한 대상의 RigidBody와 충돌 방향, 정확한 충돌지점을 받아온다.  
넘어오는 RigidBody는 shared\_ptr로 넘어오며 대상에게 직접 AddForce를 해주거나  
public함수등을 호출 할 수 있다.

  
언리얼의 EventHit와 비슷하게 만들어 보았고,

충돌 대상이 있다면 other, hit Location, hit Normal등이 넘어오는 형식이다.

오른쪽 코드의 경우 플레이어의 착지를 감지하는 부분인데,

충돌 방향이 아래 방향으로 +-30도의 경우 착지를 하며, 점프가 가능한 상태가된다.

다음으로 나중에 스킬에 사용될 폭발관련 컴포넌트를 만들었다

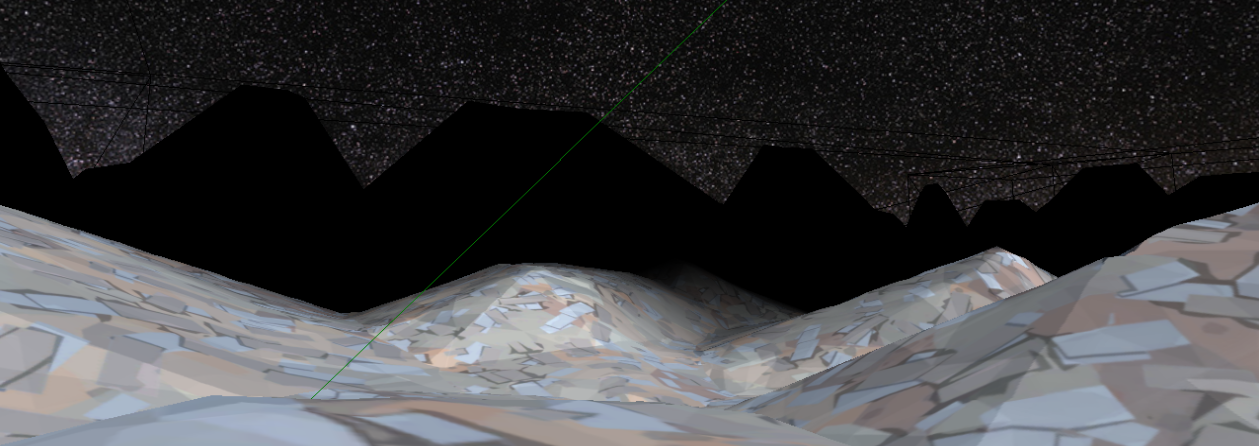


우선 콜라이더만 적용하여 구현해보았고, 해당 위치에서 범위가 점점 커지면서  
닿은 객체들을 충돌방향으로 힘을가한다.

안개효과를 구현했다. 준영이가 넣어준 가장자리 산맥 지형과, 전체 지형이 모두 보이는 점은 게임에 대한 긴장감을 낮춘다.

따라서 일정 거리부터 안개 효과를 적용해 선형적으로 안개의 수치를 높이며

최대거리부터는 쉐이더에서 clip된다. 이 때 clip되는 점은 빛 계산을 하지 않기 때문에 성능향상과 안개 효과를 통한 자연스러운 적의 등장, 긴장감도 줄 수 있다.

****

텍스쳐를 통합하였다. 우리가 구매한 에셋은 하나의 텍스쳐를 모든 모델에 적용이 가능하도록 설계되어 있었다. 하지만 지금까지 모델별로 같은 텍스쳐 파일임에도 폴더를 따로 적용해왔다. 이를 해결하고자 경로 관련 설정을 변경하였다.

최고 속도가 추락, 점프 속도에 영향을 주는점 수정하였다.

xz의 이동속도 최대값과 y의 이동속도 최대값을 분리하여 평면상의 이동속도가 점프량의 영향을 주지 않도록 수정하였다.

마지막으로 아이템 컴포넌트를 추가하였다.

수치들을 수정하여 위아래로 움직이는 주기, 회전하는 주기, 움직이는 진폭등을 설정가능하게 하였다.

