| **비단길(Silk Road)** | |
| --- | --- |
| **15주차** | **2024.03.31 (일) ~ 2024.04.06 (토)** |

**■ 다음 주 세부 계획**

**▶ [권순원]**

* 서버 방식 변경 계속 작업

**▶ [박준영]**

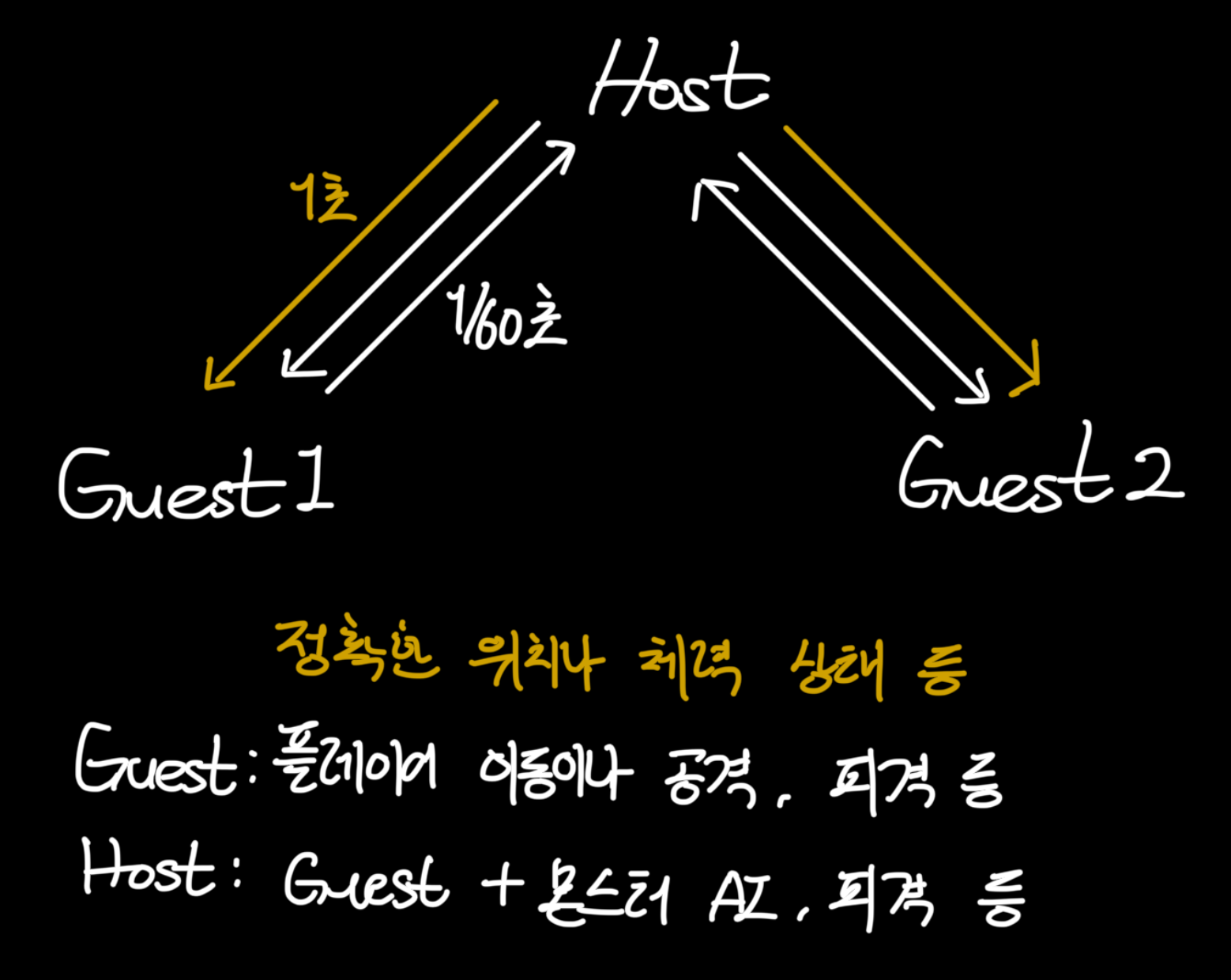
* 일부 오브젝트 렌더 구조 변경
* ai 개발

**▶ [엄장헌]**

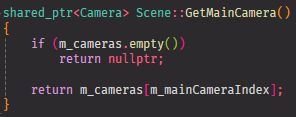
* 물리엔진에 전체적인 구조에서 결함이 있어서 수정 예정

**■ 작업 일지**

**▶ [권순원]**

원래 클라이언트를 할 생각으로 시작해서 서버를 겸해서 하다보니 모르는 부분이 많아 다시 수정할 작업이 생겼다.  
 현재 서버의 큰 틀은 호스트의 서버 쓰레드에서 모은 패킷을 통해 게임을 업데이트하고 업데이트된 데이터를 게스트와 호스트의 클라이언트를 담당하는 쓰레드에게 패킷형태로 넘겨주는 것으로 제작했다.  
 현재는 아직 제대로된 테스트 패킷으로 플레이어의 움직임만 주고 받아 서버 쓰레드가 별도의 오브젝트들의 정보를 가지고 있지 않아도 되었기 때문에 큰 문제가 없다고 생각하고 작업을 하고 있었다. 이제 본격적으로 패킷을 구체화하고 처리를 하려고 하다보니 서버도 오브젝트의 정보들을 간단하게라도 가지고 있거나, 동기화 방법을 만들어 호스트 클라이언트의 씬의 오브젝트들에게 접근해야 하는 문제가 있었다. 그래서 다시 생각해보니 어차피 P2P 방식으로 서버를 기획했고 모든 것은 호스트 기준으로 게임이 진행된다는 것을 감안하고, 인원도 최대 3명으로 제한하고 작업을 했기 때문에 이런식의 처리보다 다른 방식이 더 좋을거 같다는 생각을 했다.  
 때문에 약간의 방향을 바꾸어 작업을 다시 해보기로 했다. 일단 1/60초에 한 번씩 패킷을 주고 받는다. 이때 호스트는 단순히 다른 게스트에게 패킷을 전달하는 역할을 한다. 게스트에서 발생한 플레이어의 움직임은 호스트를 통해 다른 게스트에게 전달되는 방식으로 모든 플레이어가 같은 패킷을 공유하고 각자 적용하는 방식으로 한다. 그리고 몬스터의 움직임이나 공격 같은 정보는 호스트가 처리한 결과를 게스트들에게 전달한다.  
 그리고 이 과정에서 위치나 몬스터의 체력 등이 어긋나면 1초(테스트를 통해 가장 적절한 시간을 찾아봐야할 거 같다.)에 한 번씩 보정 데이터를 호스트가 게스트들에게 전송하는 방식으로 작업을 하기로 했다.****

일단 호스트가 각 게스트의 패킷을 받아 다른 게스트에게 전달해주는 기능과 호스트의 패킷을 각 게스트에게 전달하는 기능까지 구현했다.  
 호스트에서 보정을 위한 패킷을 보내는 부분까지 다음주 안으로 구현할 것이다.

디버그를 하기 용의하도록 카메라 변환기능을 추가했다.

메인 카메라로 사용할 카메라를 인덱스로 지정하기 위한 변수와 메인 카메라는 구하는 함수를 수정하여 런타임 중에 카메라를 바꾸어 1인칭 플레이어 시점과 3인칭 관찰자 시점을 번갈아 가면서 볼 수 있게 변경하였다.

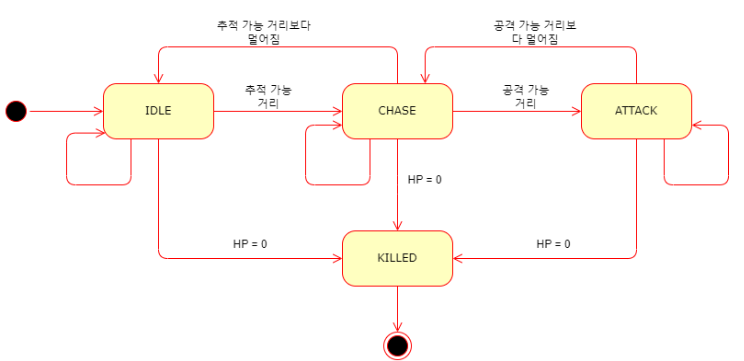
추가로 파일들의 인코딩을 모두 utf-8로 바꾸는 작업을 진행했다. 비주얼 스튜디오의 인코딩과 git 클라이언트와 인코딩이 달라서 git에서 한글이 깨져보이는 문제가 계속 발생하고 어떤 경우에는 아예 비주얼 스튜디오에서도 한글이 깨져 TODO 주석이나 참고용으로 달아놓은 주석이 알 수 없는 문자로 바뀌어 파일을 모두 utf-8로 통일하여 그런 문제가 발생하지 않도록 하였다.

**▶ [박준영]**

전 주 부터 진행하던 assimp를 통한 모델 import 작업을 진행하였다. 확실히 기존 fbxsdk는 control point 기반으로 오브젝트의 메시들을 읽어와 런타임시 메모리 사용량이 비교적 크고 최초 실행 시 로딩 시간도 비교적 오래 걸렸으나 assimp를 통해 불러올 시 모델에 따라 최대 3~4배까지 메모리 사용량을 줄일 수 있었다. 아쉽게도 처음 개발 당시 fbxsdk를 사용하다 보니 장헌이가 담당중인 애니메이션 등에 기존의 모델 로드 구조가 연관되어 있는 구조가 많이 남아있어 모든 모델을 assimp로 불러오긴 힘들었고, 맵을 구성하는 다양한 맵 오브젝트등에 대한 메시는 assimp를 통해 불러와 간혹 발생하던 텍스쳐 로드 오류를 해결함과 동시에 위에서 말한 퍼포먼스 향상이 이뤄졌다.

AI개발의 경우 현재 우리 프로젝트 상에서 게임 오브젝트에 적용하여 매 update 시 에 함께 update 될 수 있도록 하는 AI컴포넌트를 개발하여 적 개체의 행동 판단에 필요한 기본적인 어트리뷰트들을 넣어 둔 상태이다.

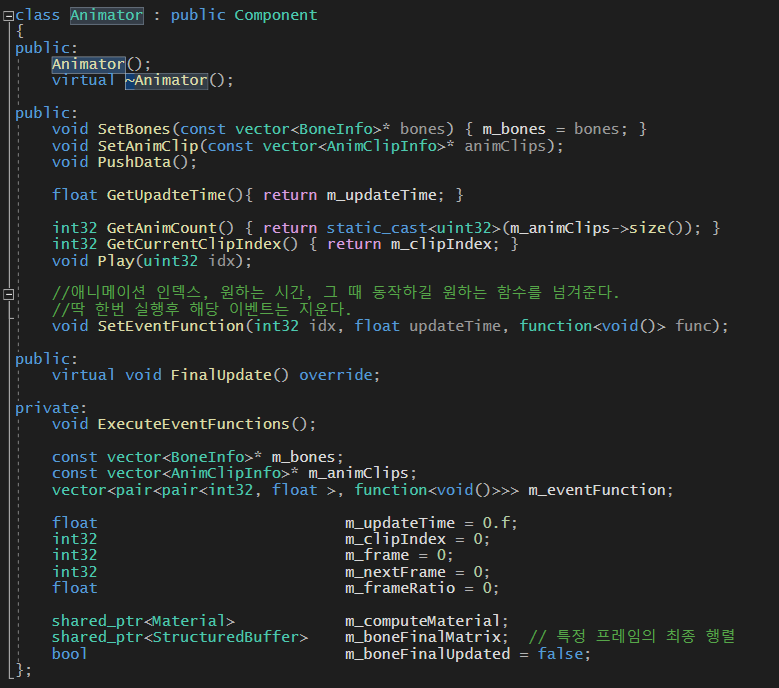
현재 중간발표때 사용할 맵을 유니티 에디터로 구현한 상태여서 바로 우리 프로젝트에서 AI를 완성하기 이전에 유니티 상에서 플레이어 위치에 따라 동작하는 상태머신을 간단하게 만들어서 테스트 중이다.

****

그렇게 특별한 사항은 없고 위와 같은 4가지의 상태가 플레이어의 위치에 기반하여 변동되며, 플레이 시작 시, 맵 상에 각 몬스터 게임 오브젝트의 spawner 객체가 존재한다. 해당 spawner 객체는 매 update시 플레이중인 플레이어들의 위치 정보를 읽어 (현재 1플레이어로 테스트중, 중간발표땐 3명의 플레이어 각각의 위치에 대해 동작)

각 spawner의 일정 범위 내에 플레이어가 위치 할 경우 오브젝트 풀링을 통해 미리 만들어두었던 적 객체의 활성 상태를 ON 하고 땅속에서 적이 솓아나는 형식의 연출로 적 객체를 등장시킬 예정이다.

**▶ [엄장헌]**



애니메이션 정보가 있는 객체에 추가되는 애니메이터이다.

뼈에 대한 정보와 애니메이션 클립들을 벡터에 담고있다.

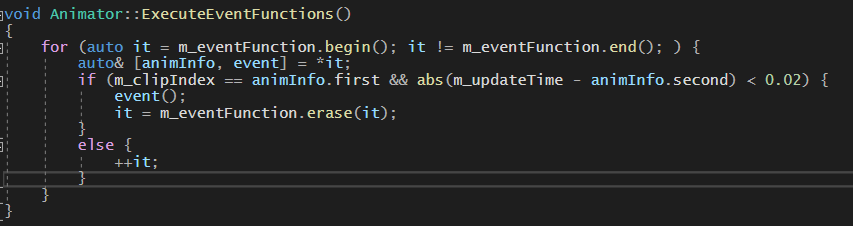
Play를 통해 해당 인덱스의 애니메이션을 시작한다.

애니메이션을 진행할 때 필요한 기능이 하나 있었다.

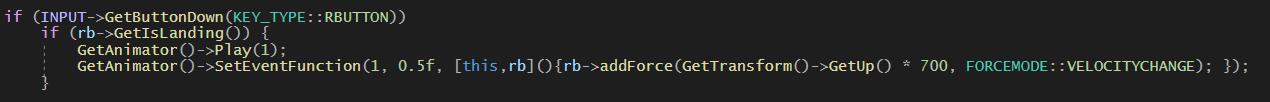
특정 시간대에 원하는 함수가 호출되는 기능이 필요했다. 예를 들어 점프 애니메이션을 플레이하면서 동시에 위쪽 방향으로 힘을 가하면 무릎을 굽히고 있는데 위로 이동하는 부자연스러운 움직임이 발생한다.

따라서 애니메이션 동작중 0.5초에 addforce를 하고싶다라는 것을 예약해야한다.

그를 위해 인덱스, 원하는시간, 원하는 동작을 담는 m\_eventFunction을 만들고, SetEventFunction으로 저장을 하면, ExecuteEventFunction에서 확인 후 처리한다.



게임의 FPS에 따라 해당 시간대에 함수가 진행되지 않을 수 도 있어서 오차는 넉넉하게 0.02초로 잡아두었다. 그리고 한 번 함수를 진행하면 이벤트 벡터에서 제외시킨다. 필요하다면 특정 애니메이션의 특정 시간대에 호출되어야하는 함수가 있다면 따로 저장할 수 있는 기능을 만들 예정이다.

다음의 경우에는 우클릭시 1번에 들어있는 점프 애니메이션을 재생하고, 해당 애니메이션의 0.5초에 함수를 실행하라는 예시 코드이다.

애니메이션은 일단 어느정도 되었기에 잠시 멈출 예정이다. 물리엔진 쪽에서 프레임에 따라 속도가 다르다는 피드백이 들어왔고, 확인해보니 fixedUpdate가 아니라는점과, 전체적으로 미흡한 부분이 발견되어 꽤 큰 수정을 할 예정이다.