|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **작성자** | **2018180033 이세철 2018180046 허재성 2018182009 김승환** | **팀명** | XD |
| **계획 시트** | <https://docs.google.com/spreadsheets/d/1rUOOKpcVfkma18YsvTZXZz44z6UhbGfOsrimVgs-mzE/edit#gid=0> | | | | |
| **주차** | **10~11** | **기간** | **2023.04.14~2023.04.27** | **지도교수** | **정 내 훈**(서명) |
| **이번주 한일** | * **이세철: 1. 맵 충돌**   **2. UI 개선**  **3. npc idle 상태 개선**   * **허재성: 1. 스프라이트 애니메이션**   **2. 그림자 오류 수정**  **3. 3D파티클 / 총알 개선**  **4. 시점 변환 개선**  **5. 서버 충돌 관련 도움.**   * **김승환: 1. 이동과 회전의 동기화 로직 개선**   **2. 서버에 유니티 맵 정보 로드**  **3. Player-Player간, Player-NPC간 충돌 여부 판단**  **4. 서버에서 사용할 총알의 충돌계산을 위한 레이캐스팅 계산식** | | | | |

**<상세 수행내용>**

1. **이세철**
2. **씬 모델에 따른 변경 및 충돌박스 추출 & 충돌**

****

**Scene에 나오는 모델들을 최종 변경 하며, 위 사진과 같이 배치되었습니다. 배치함에 따라 모든 건물들의 충돌 범위를 설정 및 추출하여 서버로 정보를 넘겨서 해당 정보를 담길 수 있도록 하였습니다.**

**다만, 유니티에서의 바운딩 박스의 좌표는 같으나 scale 값이 다른 경우가 좀 많아 추출문제는 아닌 것 같아 클라나 서버에서 임의로 scale 값만 조정하여 충돌처리를 진행하고 있습니다.**

1. **UI 개선**



**NPC HP를 헬리콥터 UI를 띄워 어디가 위험한지 표시하게끔 좌하단에 배치하였습니다. (현재 총알 문제로 모든 객체가 총알과 충돌이 되지 않아 해당 UI는 모든 부위가 정상인 상태로 출력됩니다.)**

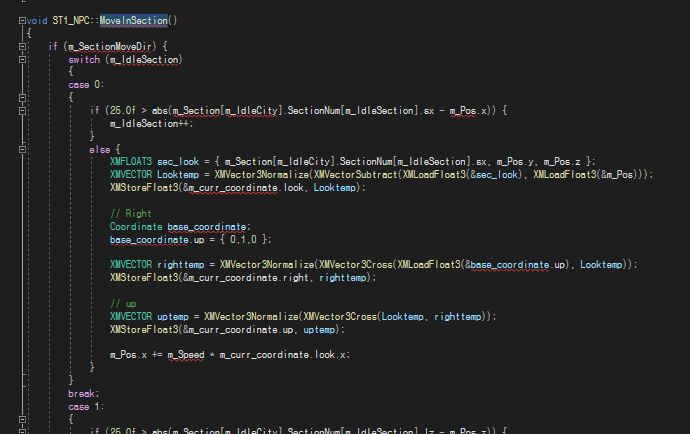
**우측 하단은 총알 개수를 알려주는 아이콘을 배치하였습니다.**

**중앙 맨 위는 현재 바라보는 방향을 배치하여 y축으로 돌아가는 값만큼 해당ui도 변경되게 수정하였습니다.**

**점령율과 점령 게이지 바는 npc 객체가 5명 이하로 남을 경우 ui가 띄워지도록 하였습니다.(현재 NPC가 총 5명인 관계로 바로 뜨게됩니다.)**

**우측상단에는 남아있는 적 NPC 객체의 수를 출력해줍니다.**

1. **NPC IDLE 개선**

****

**현재 npc를 init할 때 각 도시와 도시에서의 섹션을 나눠 해당 도시를 루트 별로 돌아다닐 수 있게 해두었습니다. 기존의 방식은 생성되지마자 원운동을 하는 것이었으나, 이 운동이 이상함을 느끼고 순찰할 수 있도록 모든 NPC 객체에게 도시 id, 섹션 id를 부여하며 웬만하면 도시 id는 바뀌지 않도록 하여 이어지는 섹션만 순찰할 수 있게 됩니다.**

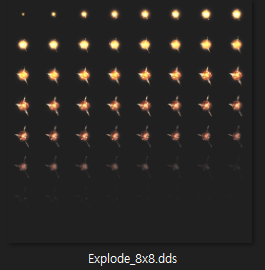
**추가적으로 NPC가 idle 상태에서 다른 상태로 변화하다가 다시 idle 상태로 진입하게 되는 경우**

**자기 자신이 먼저 가까운 도시를 검색하여 해당 도시 id를 부여받고 어떤 섹션으로 이동할지 계산하여 진행합니다. (현재, Scene 변경으로 인해 NPC 위치가 변경되지 못하여 건물들에서 튀어나오는 경향이 존재합니다. 또한 NPC와 맵 충돌은 아직 되지 않아 튀어나오고 있습니다. 총알 문제로 인해 충돌이 되지 않아 NPC 격추 모션이 보이지는 않습니다. 다만 로직상으론 존재하며 이전에 호출된 기록이 있었기에 총알 문제만 복구되면 재작동 할 것 같습니다.)**

1. **허재성**
2. **스프라이트 애니메이션**

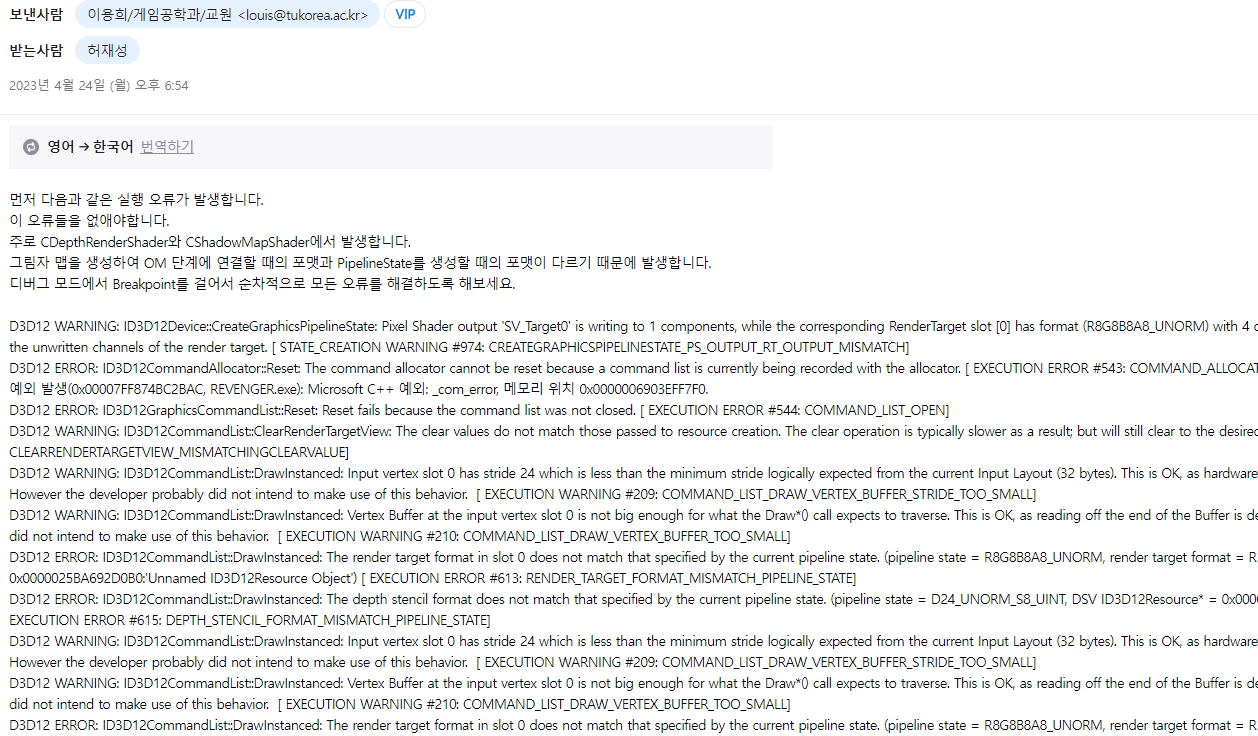
**빠른 폭발을 순간적으로 보이게 하고싶을 때, 사용하기 위해서, 폭발 모션중 하나인 요소로 스프라이트 애니메이션을 구현하였습니다.**

**빌보드를의 텍스쳐 UV 좌표를 상수버퍼를 이용하여, N x N 의 크기로 자르고, 그 크기만큼 다음 좌표로 넘어가면서 순차적인 애니메이션이 직행되도록했습니다.**

****

1. **그림자 오류 수정**

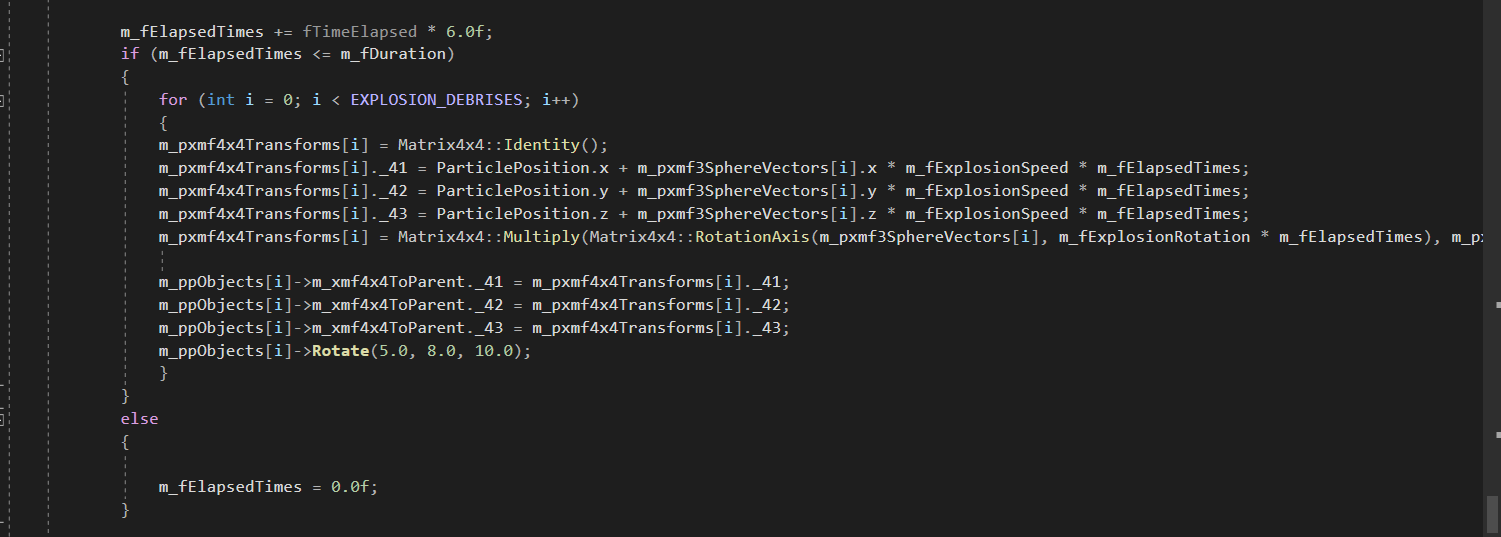
**텍스쳐가 맵핑된 객체에 그림자가 여전히 나오지 않아, 이용희 교수님에게 질문을 드렸습니다. 그 과정에서 지금까지 해결하지 못했던 출력창에 뜨는 오류들이 많았어서 교수님과 계속 연락을 주고 받으며, 차근차근 하나씩 없애 나아가는 중입니다. 이 출력 오류들을 없애는데 시간이 소요될거같아, 중간발표 전까지는 자연스럽게 못 보여드릴 확률이 클것같습니다.**

****

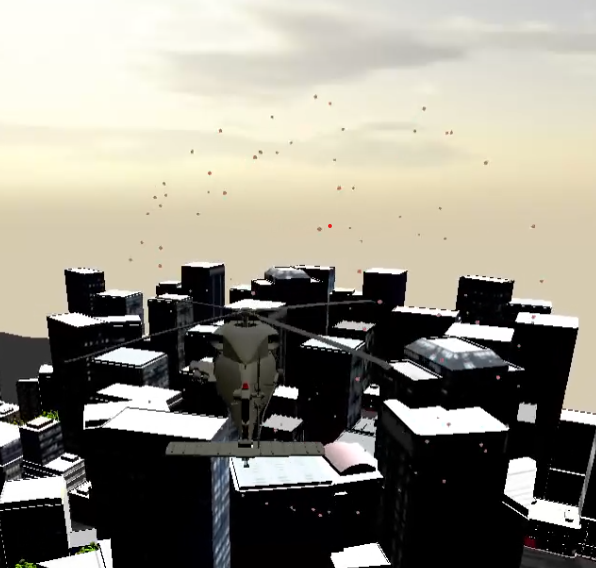
1. **3D파티클 / 총알 개선**

**파티클이 이상하게 나오던 부분을 수정해서, 불꽃놀이 처럼 터지게 바꾸었습니다.**

**또다른 변환 행렬변수에 , 파티클의 위치와 속도, 랜덤한 방향값들을 업데이트 하면서, 변환 행렬 변수에 저장된 값들을 객체들의 정보에 넣어 주었습니다.**

****

**서버를 최적화 하기 위해 패킷 전송방식을 변경하느라, 위치는 랜덤으로 설정해 두었습니다. 면담이 끝나는 대로, 위치와 개별 파티클 피격에 대한 충돌을 서버와 연결시켜서, 빠른 시간내로 완성하겠습니다.**

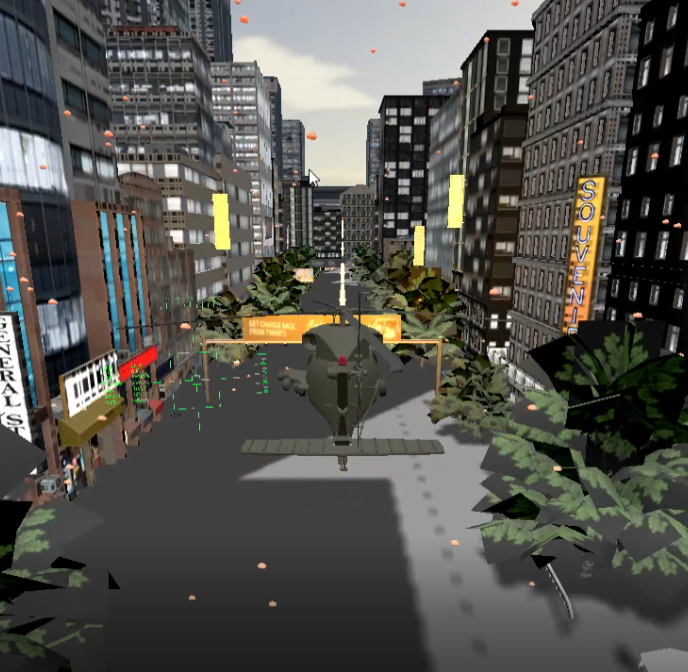
****

**다음은, 어색하게 계속 끊기는 느낌이 생겼던, 발칸총알 발사의 개선입니다.**

**서버와 클라이언트간의 패킷 전송방식을 바꾸었기 때문에, 기존에서 서버에서 계산하고 있던 발사체 로직이 클라이언트 로직 기반으로 나온것이기에, 객체의 소멸과 할당 부문에서 자연스럽지 못했습니다.**

**클라이언트에서 총알을 발사 하고, 레이캐스트 연산으로 총알의 위치를 파악하기에, 클라이언트에서도 자연스러운 발사처리를 할 수 있었습니다.**

**때문에 끊기지 않고, 연속으로 발사처리가 가능합니다.**

****

1. **카메라 시점 자유변환**

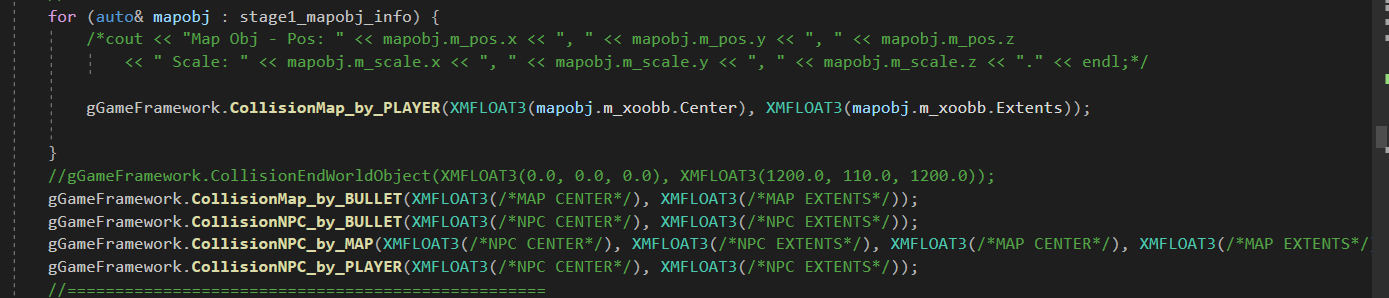
****

**F2 키를 게임중에 누르게 되면 카메라가 내부 시점모드로 바뀌어 적을 좀더 가까이 볼 수 있습니다.**

**다시 F3키를 누르면 3인칭 모드로 돌아오게 됩니다. 내부 시점모드일때는, 조종이 살짝 어려워서 이동 스칼라 값을 소폭 감소 시켰습니다.**

1. **서버 – 클라이언트 충돌관련 도움**

**이번에 바뀐 서버구동 방식에서 충돌을 판단할 때, 클라이언트의 충돌이 적용될 대상에 해당되는 객체들의 현재 위치와 방향을 업데이트를 해주어야합니다.**

****

**서버에 필요한 해당함수들을 만들어주고, 해당 모든 객체의 바운딩 박스 정보를 넘겨주었습니다. NPC의 경우 서버에서의 바운딩박스 Center, 와 Extents 값을 넘겨받습니다.**

**3. 김승환**

**1. 동기화 방식 개선**

1) 이동, 회전 동기화

기존의 이동과 회전 동기화 방식은 클라이언트에서 키입력을 받아 어떤 키가 눌렸는 지를 서버로 보내주고, 서버는 받은 패킷의 키 종류에 따른 패킷 처리를 하고 그 결과를 클라이언트로 브로드캐스트하는 방식이었습니다.

도표이(가) 표시된 사진

자동 생성된 설명

이러한 구조로는 클라이언트의 즉각적인 반응을 기대하기 어렵고, 특히 발표 현장에서는 Loopback이 아닌 Remote 통신을 하게 될 텐데 Remote 환경에서는 네트워크 딜레이로 인한 클라이언트의 랙 현상이 일어날 수 있기 때문에 통신 방식의 개선이 필요하다고 생각하였습니다.

도표이(가) 표시된 사진

자동 생성된 설명

따라서 동기화 방식을 위 그림과 같이, 클라이언트에서 이동 키를 입력받는 즉시 그 클라이언트의 Player 객체를 이동 또는 회전시킵니다. 그러고나서 클라이언트가 서버로 이동/회전의 결과를 보내줍니다. 그러면 서버는 클라이언트로부터 받은 그 결과가 제대로 된 결과인지 검사합니다. (이를 테면 맵을 뚫고 지나가는 이동이나 텔레포트를 하지는 않는지 등을 검사합니다) 검사가 끝나고, 만약 잘못된 이동이라고 판단되면 클라이언트에게 RollBack 패킷을 주고, 올바른 이동이면 이동/회전을 한 클라이언트를 제외한 나머지 클라이언트들에게 결과를 전송하는 방식으로 동기화를 하도록 수정하였습니다.

위 방식으로 바꾸고 이동/회전 동기화가 여전히 문제없이 잘 되는 것을 확인하였고, 추가로 서버에서 클라이언트로 넘겨준 플레이어 객체의 좌표 및 벡터 계산결과와 클라이언트에서 계산한 카메라의 값과의 불일치로 인해 발생하던 카메라가 덜덜 떨리는 현상이 해결되었습니다.

패킷 검사결과는 이루어지고 있으나, 아직 검증 결과가 잘못된 이동이라 판단되었을 때 Rollback패킷을 주는 과정과, 클라이언트에서 그 패킷을 받았을 때 어떻게 처리하는 지를 구현하지 않았기 때문에 시연때는 평범한 이동/회전 동기화만 보여드릴 수 있어서 아쉽지만, 이번 주말 내로 빠르게 해결하도록 하겠습니다.

**+ 이동 조작 개선**

기존의 플레이어의 이동 로직은 다음과 같았습니다.

1) 클라이언트에서 마우스 이동을 감지하고 마우스가 이동한 거리를 서버로 보내줍니다.

2) 서버에선 패킷을 받아 패킷 처리 과정에서 마우스를 움직인 정도에 따라서 객체를 마우스의 deltaY 값이 더 크면 x축 기준 회전(pitch 회전)을, deltaX 값이 더 크면 z축 기준 회전(roll 회전)을 하고 클라이언트로 회전된 벡터 값들을 전송합니다.

3) 타이머 스레드에서 객체의 기울어진 정도를 바탕으로 객체를 기울어진 방향으로 이동시킵니다.

4) 이동된 좌표를 클라이언트에게 전송합니다.

이러한 로직은 두 가지 문제점이 있었습니다.

첫째는 이동 조작의 난이도가 급 상승하는 문제점입니다. 이는 지난 면담 때 지나치게 어려운 조작은 게임이 아닌 시뮬레이터라는 지적을 받은 부분이었습니다.

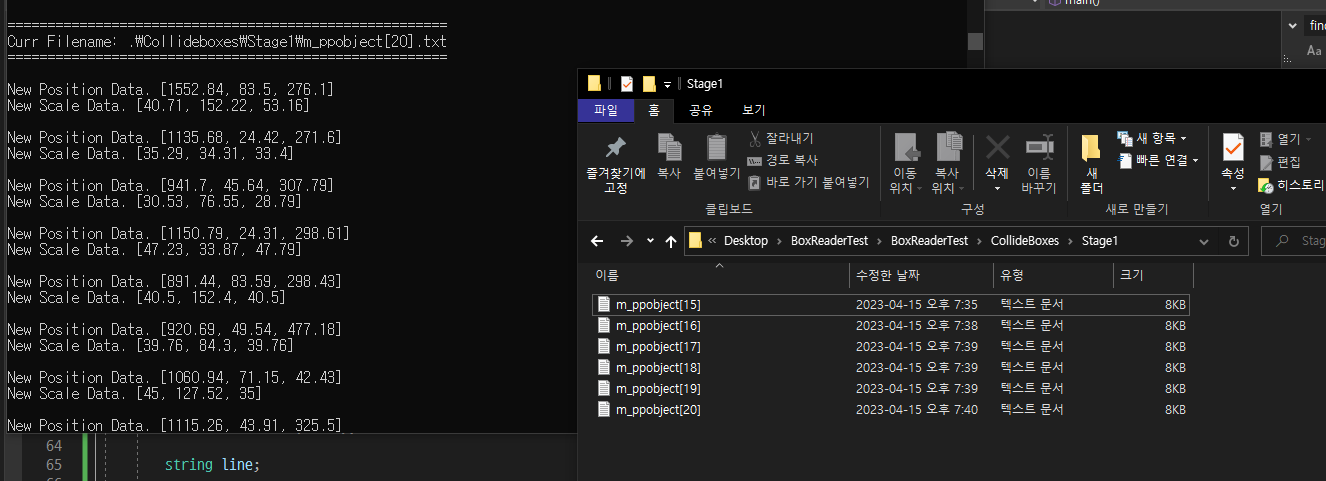
둘째는 타이머스레드의 과부하입니다. 가뜩이나 NPC 로직 처리로 인해 바쁜 타이머스레드를 더 바쁘게 하는 요인이 되었기에 서버 성능 하락의 원인 중 하나가 이것이었다고 생각됩니다.

현재는 이동을 클라이언트에서 우선 이동시키고 서버에선 올바른 이동인지 검증한 후에 나머지 클라이언트에게 뿌려주는 방식을 채택하여 서버의 부하를 줄였습니다.

**2. 서버에 유니티 맵 정보 로드**

텍스트이(가) 표시된 사진

자동 생성된 설명



우선 가장 먼저 유니티에서 추출된 맵 충돌체 데이터를 서버로 읽어오는 코드를 작성하였습니다. CollideBoxes 디렉토리와 그 하위 디렉토리에 있는 모든 .txt 파일을 읽어오기 때문에, 중간에 저 디렉토리에 있는 데이터 파일이 추가, 생성, 삭제되어도 서버 동작에 전혀 지장이 없도록 구현하였습니다.

(다만 이 과정에서 C++17의 Filesystem을 사용하였기 때문에 이제부터 비주얼스튜디오 설정에서 언어의 버전이 C++17 이상이어야만 합니다.)

텍스트, 실내, 컴퓨터, 스크린샷이(가) 표시된 사진

자동 생성된 설명

최종적으로는 유니티에서 추출한 좌표들이 서버에서 로드 되어지고, 서버에서 새로운 클라이언트가 접속할 때 맵의 충돌체 정보를 전송해줌으로써 그 결과 클라이언트에서 추출된 데이터의 값이 제대로 들어가는 것을 확인할 수 있었습니다. 이를 클라이언트 담당 팀원에게 전달하여 클라이언트에서의 맵 충돌을 할 수 있게 되었습니다.

**3. Player-NPC간 충돌 여부 판단**

도표이(가) 표시된 사진

자동 생성된 설명

NPC와 플레이어의 충돌을 검사하여 충돌되었을 때 NPC의 HP가 0이 되어 클라이언트에게 그 객체가 사망하였음을 알려주는 패킷을 보내주어 사망연출 및 후처리가 이루어지도록 하였습니다.

**4. 서버에서 사용할 총알의 충돌계산을 위한 레이캐스팅 계산식**

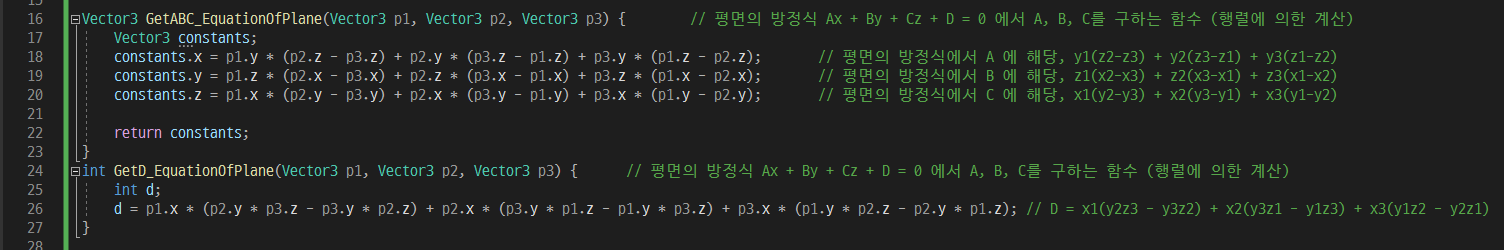
DirectX의 RayCast 함수를 사용하고 싶었으나 이를 사용하기 위해서는 Ray Tracing API를 설치하고 Windows SDK에 추가해야하는 파일도 생기는 등 여러 과정을 거쳐야 하는데, 졸업작품 발표는 저희의 컴퓨터가 아닌 과 실습실 컴퓨터에서 실행될 것으로 예상되기 때문에 이러한 추가설치 과정은 발표할 때 예상치 못한 사고를 불러올 것이라 우려되어 레이캐스팅과 유사한 기능을 하는 함수를 직접 만들어 사용해야겠다는 생각이 들었습니다.

충돌 검사 방법은 다음과 같습니다.

1) 한 점(주로 객체의 position이 될 것입니다.)을 지나고 방향(주로 객체의 Look Vector가 될 것입니다.)을 갖는 반직선 방정식을 구합니다.

2) OBB의 Center와 Extents 값을 사용하여 중심이 Center이고 변의 길이가 Extents의 각 원소 값(x, y, z)인 육면체를 만들어 냅니다.

3) 육면체의 여섯 면에 대한 평면 방정식을 구합니다.



이 때 행렬 계산을 통해

A = y1(z2-z3) + y2(z3-z1) + y3(z1-z2)

B = z1(x2-x3) + z2(x3-x1) + z3(x1-x2)

C = x1(y2-y3) + x2(y3-y1) + x3(y1-y2)

D = x1(y2z3 - y3z2) + x2(y3z1 - y1z3) + x3(y1z2 - y2z1)

평면 방정식 Ax + By + Cz + D = 0의 A, B, C, D 값을 구할 수 있습니다.

4) 반직선 방정식과 여섯 개의 평면 방정식의 연립 계산을 하기 전에, 우선 반직선의 방향벡터와 평면 방정식의 법선벡터를 내적하여 두 벡터가 서로 평행한지를 먼저 확인합니다. 이때 두 벡터의 내적이 0이면(평행하면) 다음 계산을 진행하지 않고 “충돌하지 않음”을 결과로 리턴합니다. (반직선과 평면이 평행하면 교점이 존재하지 않거나 무수히 많기 때문입니다.)

5) 내적이 0이 아니라면, 반직선 방정식과 평면 방정식을 연립 계산하여(여섯 개의 면이 있으므로 총 여섯 번의 연립 계산이 이루어집니다.) 반직선과 평면의 교차점들을 구합니다.

6) 교차점이 한 개 이상 나왔다면, 우선 그 교차점이 육면체 안에 있는지를 검사합니다. 그리고 육면체 내에 있는점들에 한해서 반직선의 시작점(객체의 Position)과의 거리를 계산하여 가장 가까운 점을 구해내면, 그 점이 최종적인 교차점이 됩니다. 이 경우에는 교차점이 존재하므로 “충돌했음”을 결과로 리턴합니다. 만약 5)의 결과로 교차점이 하나도 나오지 않았다면 “충돌하지 않음”을 결과로 리턴합니다.

위와 같은 방법을 사용하면 총알과 같이 시작점에서부터 앞을 향해 빠르게 나아가는 종류의 객체 충돌 여부를 판단할 수 있음을 이론적으로 확인하였고 따로 새 프로젝트를 만들어서 간단하게 로직 검증에 대한 테스트를 진행 중인 상태입니다. 아직 이러한 로직을 서버에서 사용하지는 않았고, 이에 대한 프로그래밍 구현은 이번 주말에 진행할 예정입니다.

또한 서버에서 이 로직으로 계산했을 때 혹시 모를 성능 저하가 있을 수도 있기 때문에 서버에서의 구현을 먼저 해보고 성능이 떨어진다면 로직 최적화도 함께 진행할 예정입니다.

|  |  |  |  |
| --- | --- | --- | --- |
| **문제점 정리** |  | **해결 방안** |  |
| **다음 주차** | **12** | **다음 기간** | **2023.04.28~2023.05.11** |
| **다음주 할 일** | 이세철:  허재성:  김승환: | | |
| **지도교수**  **Comment** |  | | |