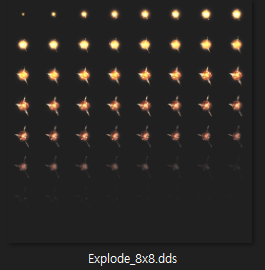
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **작성자** | **2018180033 이세철 2018180046 허재성 2018182009 김승환** | **팀명** | XD |
| **계획 시트** | <https://docs.google.com/spreadsheets/d/1rUOOKpcVfkma18YsvTZXZz44z6UhbGfOsrimVgs-mzE/edit#gid=0> | | | | |
| **주차** | **10~11** | **기간** | **2023.04.14~2023.04.27** | **지도교수** | **정 내 훈**(서명) |
| **이번주 한일** | * **이세철:** * **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. 서버에서 사용할 총알의 충돌계산을 위한 레이캐스팅 계산식** | | | | |

**<상세 수행내용>**

1. **이세철**
2. **허재성**
3. **스프라이트 애니메이션**

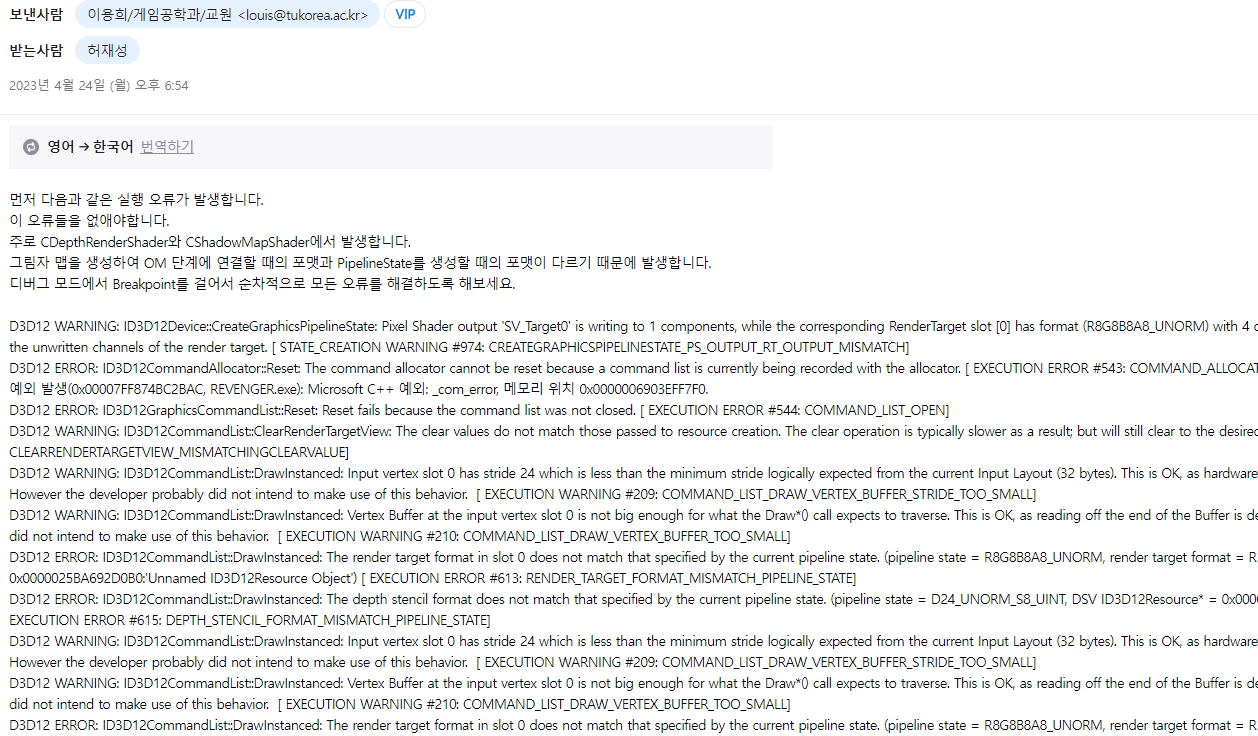
**빠른 폭발을 순간적으로 보이게 하고싶을 때, 사용하기 위해서, 폭발 모션중 하나인 요소로 스프라이트 애니메이션을 구현하였습니다.**

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

****

1. **그림자 오류 수정**

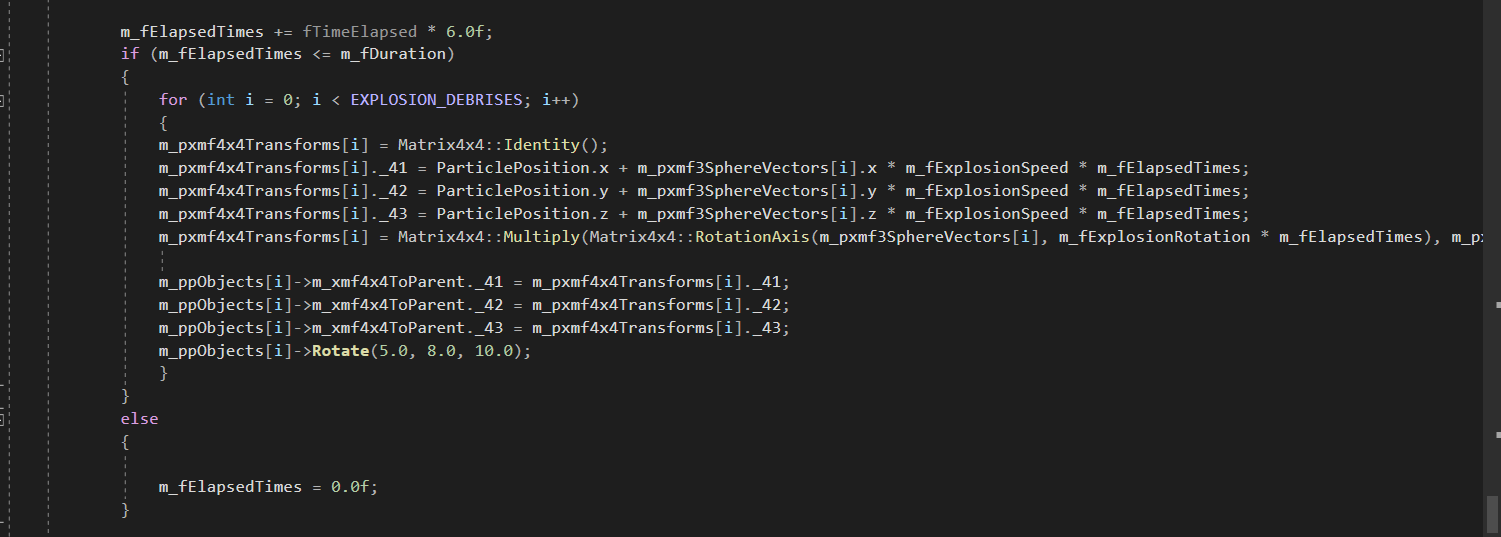
**텍스쳐가 맵핑된 객체에 그림자가 여전히 나오지 않아, 이용희 교수님에게 질문을 드렸습니다. 그 과정에서 지금까지 해결하지 못했던 출력창에 뜨는 오류들이 많았어서 교수님과 계속 연락을 주고 받으며, 차근차근 하나씩 없애 나아가는 중입니다. 이 출력 오류들을 없애는데 시간이 소요될거같아, 중간발표 전까지는 자연스럽게 못 보여드릴 확률이 클것같습니다.**

****

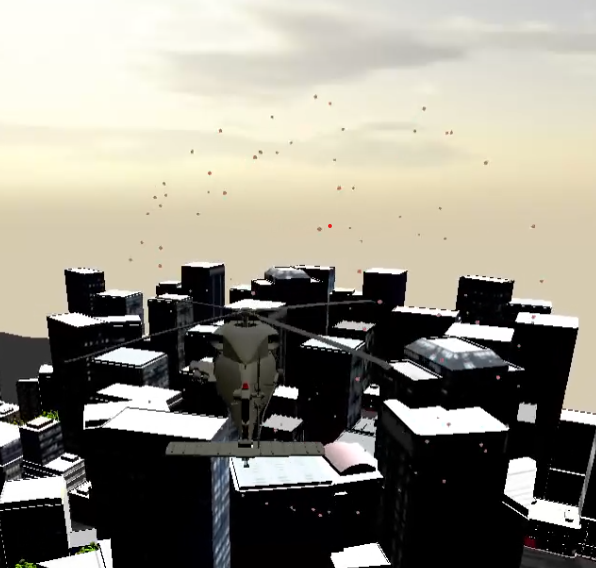
1. **3D파티클 / 총알 개선**

**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.**

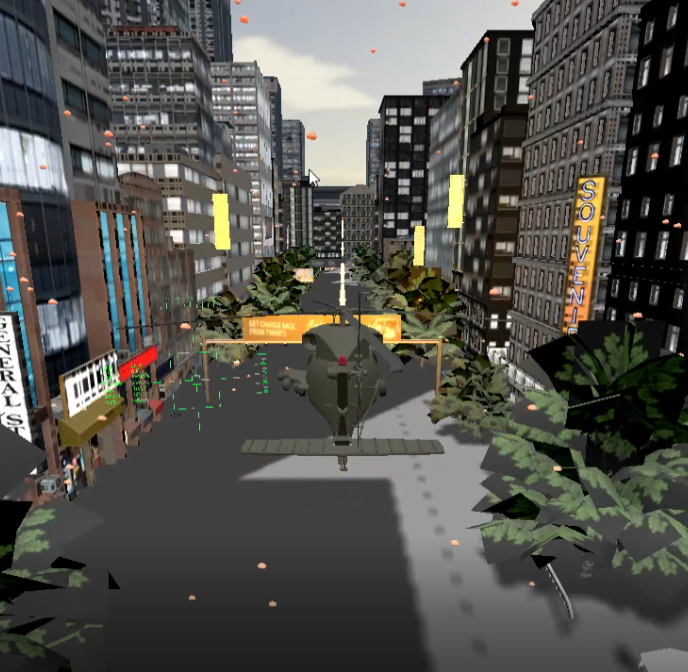
****

**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**

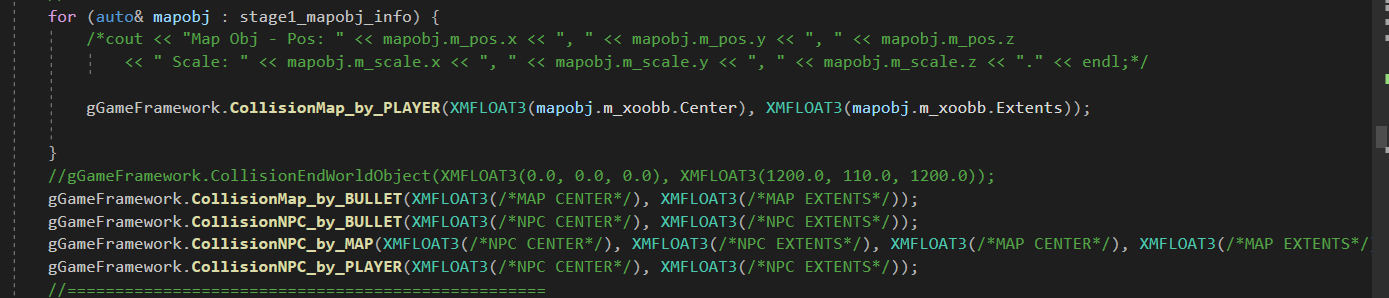
****

**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. 이동과 회전의 동기화 로직 개선**

**2. 서버에 유니티 맵 정보 로드**

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

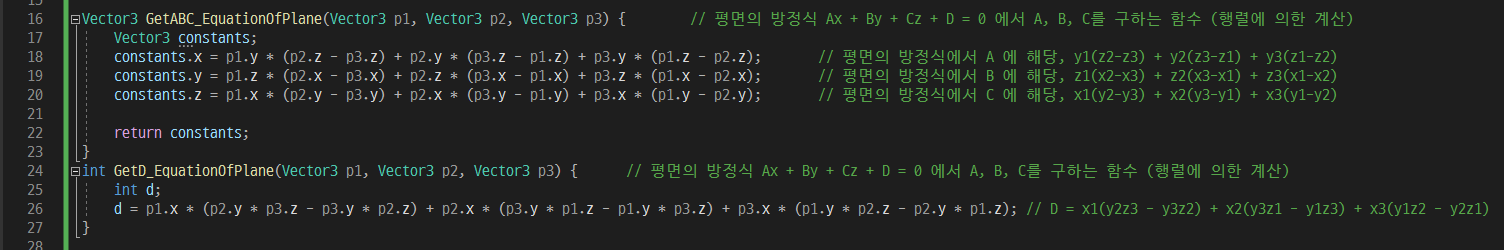
DirectX의 RayCast 함수를 사용하고 싶었으나 이를 사용하기 위해서는 Ray Tracing API를 설치하고 Windows SDK에 추가해야하는 파일도 생기는 등 여러 과정을 거쳐야 하는데, 졸업작품 발표는 저희의 컴퓨터가 아닌 과 실습실 컴퓨터에서 실행될 것으로 예상되기 때문에 이러한 추가설치 과정은 발표할 때 예상치 못한 사고를 불러올 것이라 우려되어 레이캐스팅과 유사한 기능을 하는 함수를 직접 만들어 사용해야겠다는 생각이 들었습니다.

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, 반직선의 시작점(객체의 Position)과의 거리를 계산하여 가장 가까운 점이 최종적인 교차점이 됩니다. 이 경우에는 교차점이 존재하므로 “충돌했음”을 결과로 리턴하고, 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 테스트까지 완료한 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** |  | | |