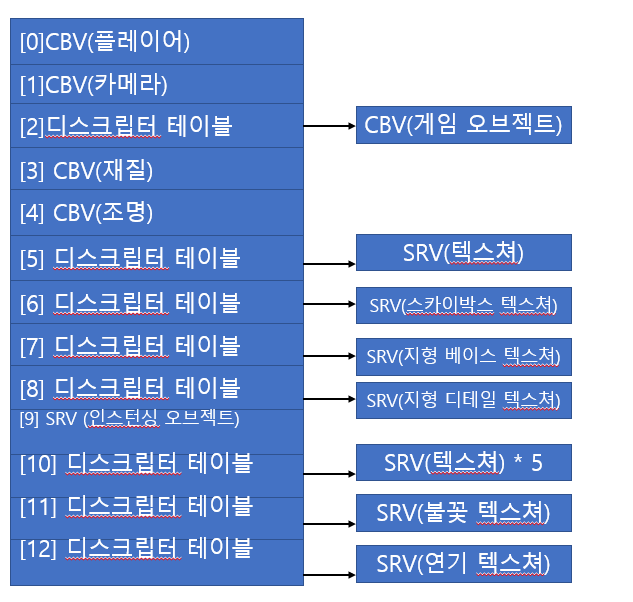
**목표 : 실제 오브젝트 메쉬(비행기)를 불러와서 비행 게임 만들기**

목차---------------------------------------------------------------------------------------------------------

1. **루트 파라미터**
2. **조작법**
3. **텍스쳐 애니메이션**
4. **미사일**
5. **기타**
6. **실행 화면**

--------------------------------------------------------------------------------------------------------------

1. **루트 파라미터**



**Scene.cpp 273줄**

3D게임2 01, 3D게임2 02, 동안 루트파라미터에 서술자, 테이블, 루트 상수 등 메모리를 설명 할 수 있는 뷰 (루트상수는 제외)를 넣어놓고, 테이블은 테이블의 주소를 따라가 보면 또 다른 테이블이 나오고 원하는 데로 테이블을 설정하는 작업들을 하여 그 부분은 무리없이 진행 하였습니다

다만 텍스쳐는 루트디스크립터로 넘겨 줄 수 없고 따라서 테이블로 설정하는 부분 등 기존에 했던 작업들과 동일합니다.

CBV는 b레지스터, SRV는 t레지스터에 정확히 이어주는 작업이 필요하였고, 만들어진 루트 파라미터 인덱스에 제대로 셋하는 과정이 필요했습니다. 루트파라미터가 많아 질수록 코드 어느 구석진 부분에 루트파라미터가 제대로 셋 되지 않아 초기에 작업이 늦어졌습니다.

1. **조작법**

방향키 : W키, A키, S키 , D키

Q키 : 플레이어 상승이동

Z키 : 플레이어 하강이동

마우스 좌클릭 + 마우스 이동 : 플레이어 회전

F3 : 3인칭 시점 변경

F1 : 1인칭 시점 변경

SPACE : 미사일 발사

1. **텍스쳐 애니메이션**

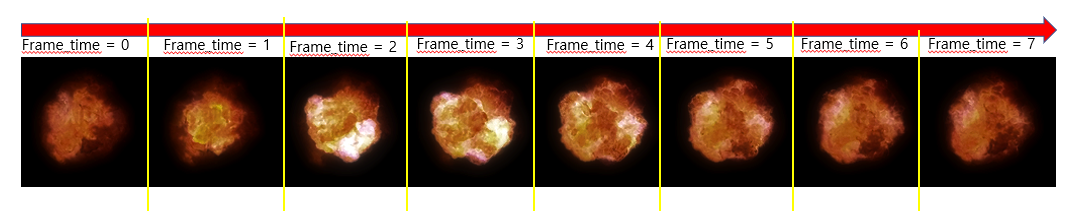
**처음 가정**

애니메이션할 텍스쳐가 6개라 하면 빌보드 사각형 6개를 준비해서 응용프로그램에서 시간이 지날 때 마다 하나하나 바꿔주기로 합니다.

**문제점**

뭔가 없어보이고 에니메이션이 100개, 1000개, 10000개짜리 텍스쳐라면, 상당히 비효율적입니다

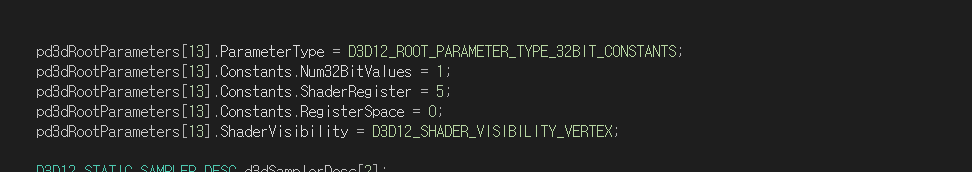
차라리 픽셀쉐이더에서 u,v좌표를 시간(frame\_time)에 따라 변경시켜서 sample 하는것이 좋다고 판단하였습니다.



위의 사진 같은 경우는 총 8등분 할 수 있습니다. Frame\_time마다 텍스쳐의 x좌표를 1/8 \* frame\_time 만큼 텍스쳐 매핑 하도록 하면 처음 가정한 것 보다 효율적입니다. (프레임 타임만 바꿔주면 되기 때문에) **관련된 코드 shader.hlsl 405줄**

연기 텍스쳐도 위와 동일한 방식으로 구현 하였습니다.

그렇다면 frame\_time을 응용프로그램에서 받아오는 작업이 필요합니다. 루트상수를 사용하면 따로 버퍼나 힙을 생성할 필요 없어 간편하고, frame\_time은 64비트로 충분히 표현 가능 함으로 루트상수를 사용하려 하였습니다.



여기서 ShaderVisibility의 뜻을 제대로 알게되었습니다. 수업 주에 설명을 들었을 때는 버텍스쉐이더에서만 보이게 할지, 전체에서 보이게 할지 등, 당시 렌더링 파이프라인의 전체적인 구조도 이해 못하였던 상태라 그땐 그렇구나 하고 넘어갔었습니다.

하지만 저기서 ShaderVisibility 를 SHADER\_VISIBILITY\_VERTEX로 해버리면 정작 넘겨줄 루트상수의 값(frame\_time)은 픽셀쉐이더에서 사용하지 못하여 계속 0이라 똑같은 텍스쳐만 나오게 되었습니다. 이 부분을 생각하지 못해 작업시간이 많이 늦어지게 되었습니다. **(SHADER\_VISIBILITY\_PIXEL)이나 (SHADER\_VISIBILITY\_ALL)로 바꿔줘야 함**

**그 후 다음 가정**

어차피 게임오브젝트의 정보는 shader.hlsl 의 cbGameObjectInfo (상수버퍼)를 보면 응용프로그램에있는 월드변환행렬과 재질의 정보를 넘겨받습니다.

그렇다면 차라리 이 부분에 넘겨주는 과정에서 frame\_time도 같이 넘겨주는 방법으로 구현하였습니다.

**관련코드**

CB\_GAMEOBJECT\_INFO에 m\_iFrameTime을 추가하고.( **Object.h의 25줄**) frame\_time을 업데이트 해주고,(**Shader.cpp1362줄**) 응용프로그램의frame\_time 을 hlsl에 넘겨주는 작업을 합니다. (**Shader.cpp 2409줄, Shader.cpp 1183줄 )**

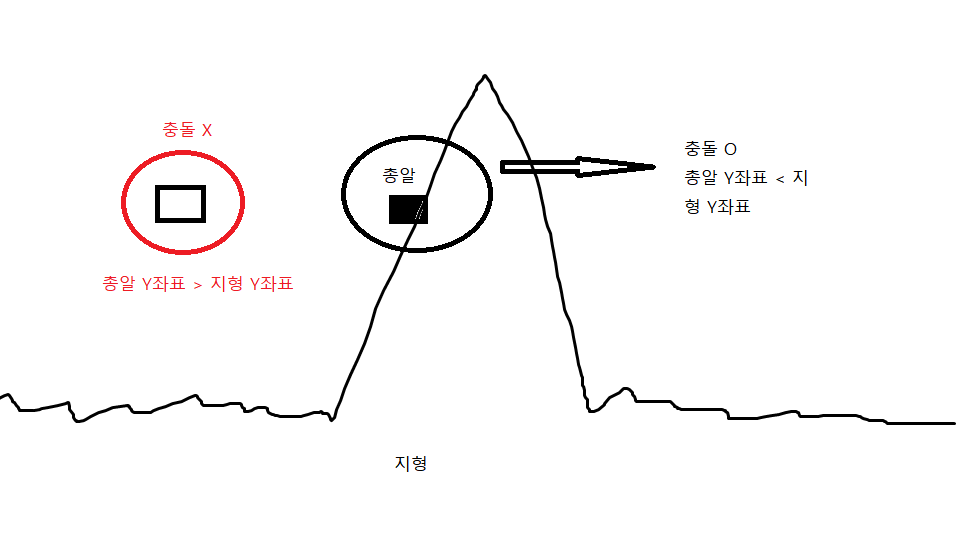
1. **미사일**

미사일 자료구조는 배열을 사용했고 총 100발까지 발사 가능합니다.

**충돌처리**

지형 **(관련코드 shader.cpp 1012줄)**

* 총알의 x,y,z 좌표는 알고있습니다.
* 총알의 x,z를 Getheight(x,z)하면 총알이 x,z위치일때 지형의 y좌표를 알 수 있습니다.
* 총알의 y좌표가 GetHeight(x,z)로 구해진 y좌표보다 작으면 충돌한 것입니다.



적 오브젝트 **(관련코드 Shader.cpp 976)**

DirectX에서 제공하는 충돌 함수를 사용 하였습니다.

미사일 발사 해서 충돌한 위치에 파티클을 만들어야 합니다.(**shader.h 18줄 m\_f3CrushPosition)** 이 변수는 SmokeShader(연기), ParticleShader(불꽃), 에서 공유하여 사용하기 위해 static으로 사용했습니다.

그 후 충돌한 위치로 파티클 들을 이동 시켜줍니다. **(shader.cpp 1376줄), (shader.cpp 2597줄)**

일정 시간이 지나면 파티클들을 사라지게 하기위해 시간을 새는 타이머가 필요 하였습니다.

충돌하면 그 순간 타이머를 작동시키고**(shader.cpp 1032줄)**, 타이머의 시간이 5초 이하일 때 까지만 그리게 되도록 하였습니다.**(shader.cpp 1405줄, 2634줄)**

1. **기타**

적 비행기 오브젝트와 플레이어 오브젝트의 거리가 일정거리가 되면 적 비행기가 플레이어 위치로 이동합니다.**(shader.cpp 938줄)** 이는 플레이어의 위치벡터에서 적의 위치벡터를 빼 이동할 방향벡터를 적 오브젝트의 방향벡터로 하여 업데이트 해주었습니다.

풀,나무,잔디 인스턴싱은 + 빌보드는 3D게임2 02 내용과 작업이 동일합니다.

스카이박스 , 지형 구현은 3D게임2 02 내용과 동일합니다.

1. **실행화면**



나무 2500개, 풀 2500개, 잔디 2500개 인스턴싱으로 구현하였습니다.



총알이 충돌하면 총알이 충돌한 위치에 연기와 불꽃이 생기고 , 연기텍스쳐와 불꽃 텍스쳐를 블랜딩 해주었습니다.



오브젝트와 총알의 충돌 모습