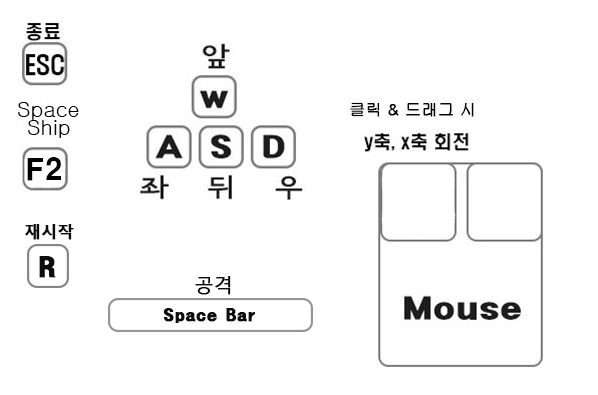
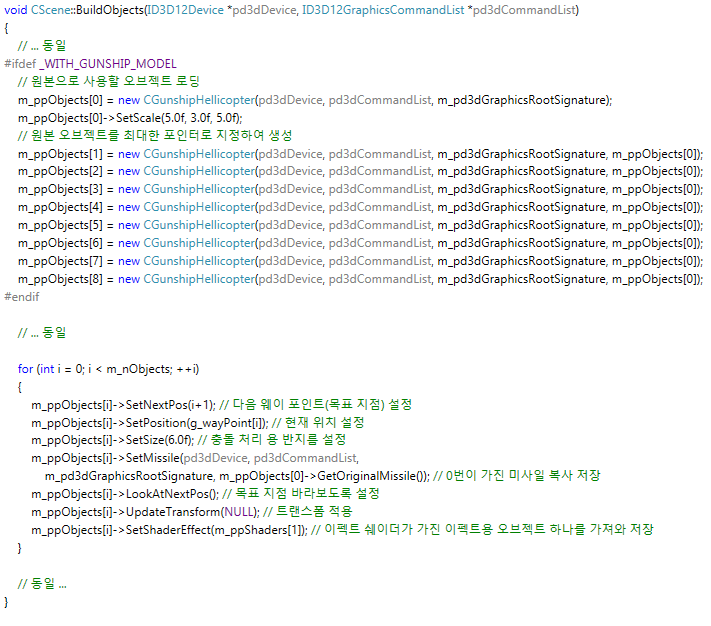
1. 이번 과제의 목표
2. 제공된 모델을 띄워 지형을 따라 움직이도록 한다.
3. 지형의 디테일 맵에 노말 매핑을 적용한다.
4. 스카이 박스를 그린다.
5. 빌보드를 사용하여 풀과 나무를 그린다.
6. 오브젝트가 사라지는 경우 폭발 이펙트가 발생하도록 한다.
7. 미니맵을 그린다.
8. 프로그램에 대한 가정
9. 이번 과제의 목표 5번의 미니맵을 그리는 부분은 구현하지 못하였습니다.
10. 4번 과제에서 구현할 기하 쉐이더를 이용한 빌보드를 구현했습니다.
11. 지형은 기존 지형을 편집하여 높이가 조금 더 부드럽게 변하도록 하였습니다.
12. 플레이어는 비행체이므로 ‘Space Ship’ 시점만 사용할 수 있도록 하였습니다.
13. 적의 경우 시작 지점에서부터 9개의 웨이 포인트마다 배치하였으며 업데이트 시 자신의 다음 웨이 포인트를 향해 이동하도록 하였습니다.
14. 적은 마지막 웨이 포인트에 도달한 경우 다시 처음 위치로 이동하여 계속해서 웨이 포인트를 따라 이동하도록 하였습니다.
15. 적이 사망한 경우 적의 상태를 false로 하여 그리지 않도록 하였습니다.
16. 적이 나타나는 시간을 일정하게 하기 위하여 적 오브젝트가 마지막 웨이 포인트에 들어가 다시 처음 위치로 돌아갈 때 상태를 true로 바꿔줘 항상 동일한 간격으로 적이 등장하도록 하였습니다.
17. 지형 정점의 탄젠트 값은 그 점을 공유하는 삼각형의 탄젠트 값의 평균으로 적용하였습니다.
18. 모델은 최초 하나의 경우 그냥 로드 한 뒤 이후 동일한 모델을 사용하는 경우 다시 만들지 않아도 되는 부분은 최대한 포인트로 참조하였습니다. 이 부분을 적용하여 최초 모델 9개를 띄울 때 로딩 시간이 16~17초에서 10초~11초로 감소하는 것을 확인하였습니다.
19. 미사일도 마찬가지로 하나를 불러와 최대한 참조하여 사용하였으며 적 오브젝트는 2발, 재장전 시간 2초, 날아가는 지속시간 2초로 하였으며 플레이어의 경우 8발을 가지며 재장전 시간 0.7초, 날아가는 지속시간 2초로 하였습니다.
20. 미사일의 처리는 평소 False 상태이다가 플레이어의 경우 발사 버튼, 적의 경우 플레이어가 공격 가능 거리 안에 들어온 경우 미사일을 발사하며 미사일의 주소를 받아와 저장한 뒤 미사일의 Look 방향으로 특정 속도로 이동하고 충돌 검사를 수행하는 오브젝트 클래스를 만들어 리스트에 저장하여 발사 중인 미사일만 업데이트 및 렌더링이 수행 되도록 하였습니다.
21. 폭발 이펙트는 사각형 메쉬를 가지는 오브젝트를 인스턴싱으로 처리하였으며 폭발 이펙트의 숫자는 미사일의 숫자(적 2 \* 9개 + 플레이어 8개) + 오브젝트의 개수(적9기 + 플레이어1기)로 적용하였으며 나열한 오브젝트들이 각기 하나씩 포인터로 보유하도록 하였습니다.
22. 폭발 이펙트의 적용은 폭발이 일어나는 상황(미사일 시간 만료, 미사일 적중, 오브젝트 파괴)에서 폭발 이펙트의 포인터를 받아 플레이어를 바라보도록 하고 시간에 따라 UV값이 바뀔 수 있도록 해 주는 오브젝트를 만들어 리스트에 저장하여 현재 보여줘야 할 폭발 이펙트가 적용이 되도록 하였습니다.
23. 조작 방법



1. 코드 설명
2. Scene
   1. 스카이 박스와 지형을 그리기 위한 오브젝트를 추가하였습니다.
   2. 미사일과 폭발 이펙트 처리를 위한 리스트를 추가하였습니다.
   3. 각 웨이 포인트에 파란색 포인트 라이트를 추가하였습니다.
   4. 적 오브젝트의 모델 로딩 시간을 최소화 하기 위한 생성자를 사용하여 적 모델을 생성하였습니다.



* 1. 지형 노멀 맵, 스카이 박스, (풀, 나무), 이펙트를 위해 DescriptorRange와 Descriptor Table을 추가하였습니다.
  2. 이펙트 인스턴싱을 위한 Descriptor를 추가하였습니다.
  3. 미사일과 이펙트의 업데이트를 추가했습니다.

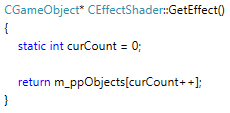


* 1. 스카이 박스, 지형, 미사일을 그리도록 Render를 추가했습니다.
  2. 오브젝트간 충돌을 담당하는 CollisionTest함수를 추가했습니다.
  3. 플레이어에게 미사일을 전달하는 SetPlayerMissile함수를 추가했습니다.
  4. 플레이어가 미사일을 발사하도록 하는 LaunchPlayerMissile함수를 추가했습니다.
  5. 적이 플레이어에게 미사일을 발사하도록 하는 LaunchEnemyMissileToPlayer 함수를 추가했습니다.

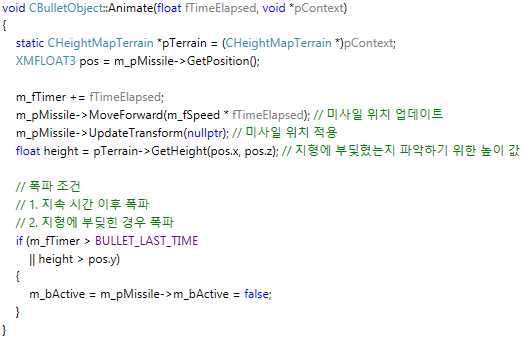
1. Shader
   1. 지형을 그리기 위한 CTerrainShader를 추가했습니다.
   2. 스카이 박스를 그리기 위한 CSkyBoxShader를 추가했습니다
   3. 풀, 나무를 그리기 위한 CBillBoardShader를 추가했습니다.
   4. 이펙트를 그리기 위한 CEffectShader를 추가했습니다.
   5. CBillBoardShader의 경우 BuildObject에서 각 빌보드의 위치, 크기, 어떤 오브젝트를 나타내는지를 가리키는 정보를 가진 정점을 2000개 만들어 가지게 하였으며 쉐이더가 작동하면 이 정점을 파이프 라인에 넘겨 기하 쉐이더 단계에서 빌보드가 만들어지도록 하였습니다.

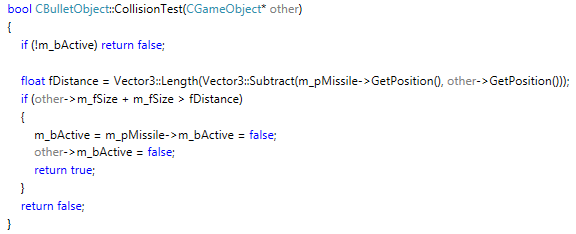


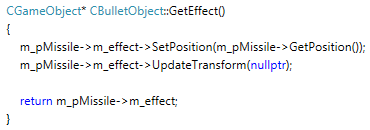
* 1. CEffectShader은 이전 과제의 빌보드 쉐이더와 동일하나 외부에서 업데이트를 처리 하기 때문에 AnimateObject에서 아무런 작업을 하지 않도록 했습니다.
  2. 이펙트 오브젝트를 폭발하는 오브젝트가 가지고 있도록 하기 위해 이펙트 오브젝트를 리턴 해 주는 GetEffect함수를 추가했습니다.



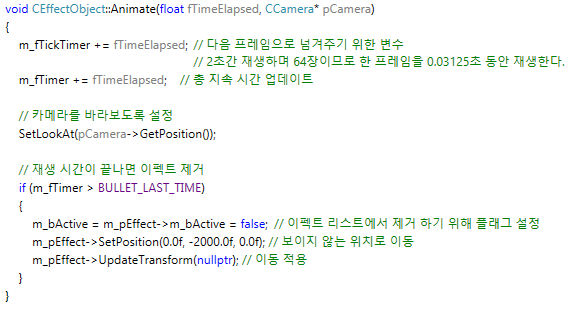
1. GameObject
   1. 스카이 박스를 그리기 위한 CSkyBox를 추가하였습니다.
   2. 미사일을 처리해 주기 위한 CBulletObject를 추가했습니다.
   3. 이펙트를 처리해 주기 위한 CEffectObject를 추가했습니다.
   4. CBulletObject는 생성 시 미사일 오브젝트와 해당 미사일을 발사한 게 누구인지 판단하기 위한 Owner를 받습니다. Owner는 Player와 Enemy 두 종류가 있으며 이 종류에 따라 미사일의 속력, 충돌 시 반지름을 적용하였습니다.
   5. CBulletObject의 업데이트는 미사일의 이동과 지형과의 충돌을 처리하였습니다.



* 1. 미사일과 타 오브젝트의 충돌 체크를 위한 CollisionTest함수를 추가했습니다. 충돌 체크는 처음 지정한 오브젝트들의 반지름과 두 오브젝트 사이의 거리를 이용하여 계산 하였습니다.
  2. 미사일이 가진 이펙트를 업데이트용 오브젝트에 넘기기 위한 GetEffect함수를 추가하였으며 Effect를 넘겨줄 때 미사일이 폭발한 위치를 적용하여 넘겨주었습니다.



* 1. CEffectObject는 생성 시 Effect를 받아 저장하여 내부적으로 해당 Effect를 업데이트 시켜주는 역할을 합니다.

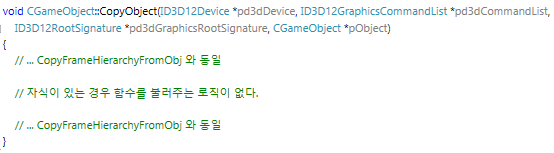


* 1. 모델 구조를 다른 오브젝트에서 복사해 오는 CopyFrameHierarchyFromObj함수를 추가했습니다. 이것은 기본적으로 LoadFromHierarchyFromFile함수와 동일하지만 불러오는 대상이 오브젝트 인 것이 차이이며, 모델 데이터 같이 공유가 가능한 것은 복사 생성하는 것이 아니라 모델을 포인터로 가리키도록 하였습니다.

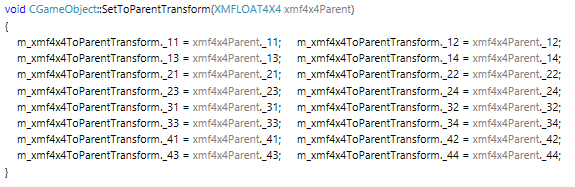
또한, 생성자에서 이 함수를 부르는 형태로 작성하였습니다.



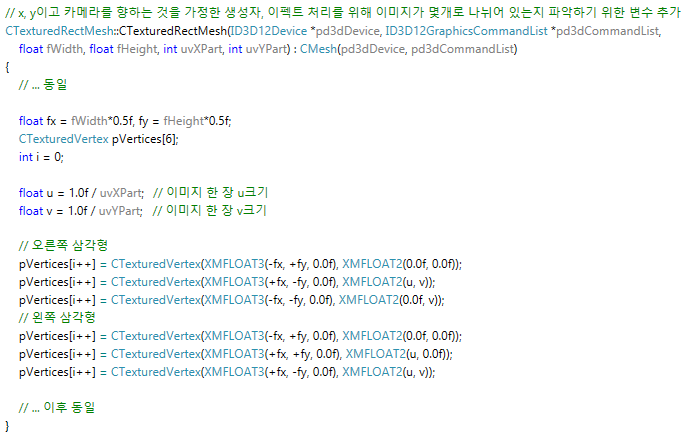
* 1. 위 함수와 동일한 목적 및 작동 방식이 동일한, 그러나 자식이나 형제 노드는 신경 쓰지 않고 해당 오브젝트마 복사해오는 CopyObject도 정의하였으며 이 함수는 미사일을 다른 오브젝트에 넘겨줄 때 사용합니다.



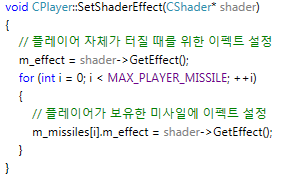
* 1. 보유하고 있는 이펙트를 넘겨주는 GetEffect함수를 추가했습니다.
  2. 미사일을 작동시키기 위해 현재 오브젝트의 위치 및 바라보고 있는 방향을 맞춰 주기 위한 SetToParentTransform함수를 추가했습니다.



1. Mesh
   1. 풀, 나무를 그리기 위해 위치(XMFLOAT3), 크기(XMFLOAT2), 텍스처 종류(UINT)를 가지는 CBillBoardVertex를 추가했습니다.
   2. 텍스처를 편하게 생성하기 위해 CTexturedRecMesh에 생성자를 추가했습니다.

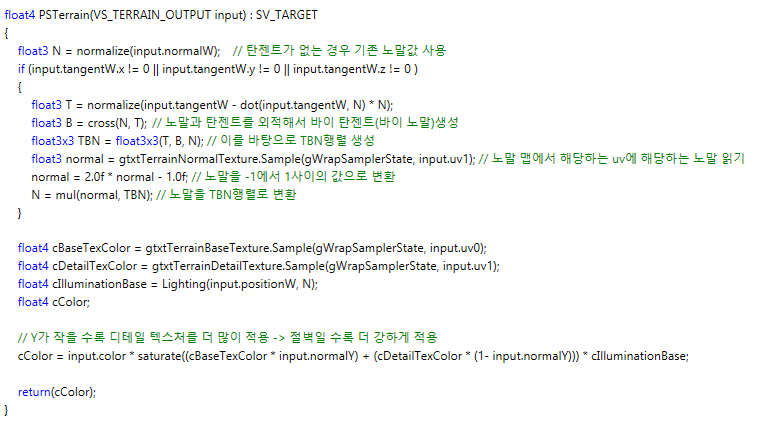


1. GameFramework
   1. 카메라 시점을 제한하기 위하여 OnProcessingKeyboardMessage에서 VK\_F1과 VK\_F3을 제거하였으며, 미사일 발사를 위해 VK\_SPACE를 추가하여 m\_pScene의 LaunchPlayerMissile함수를 호출하도록 하였습니다. 또, 플레이어 사망 시 재시작 하기 위해 ‘R’키를 누르면 플레이어가 처음 위치로 돌아가서 상태가 True가 되도록 하는 Reset함수를 실행하도록 하였습니다.
   2. 충돌 체크를 한 곳에서 관리하기 위한 CollsionTests함수를 생성하였으며 현재는 m\_pScene의 CollisionTest를 호출하게 하였습니다.
2. Player
   1. 플레이어와 지형의 충돌을 처리해 주기 위해 CollisionTestWithTerrain함수를 추가했습니다.
   2. 플레이어가 가진 미사일을 리스트에 넣어 주기 위한 GetMissile함수를 추가했습니다.
   3. 플레이어에게 미사일을 넘겨주기 위한 SetMissile함수를 추가했습니다.
   4. CEffectShader에서 이펙트를 얻어 설정해 주기 위한 SetShaderEffect함수를 추가했습니다.



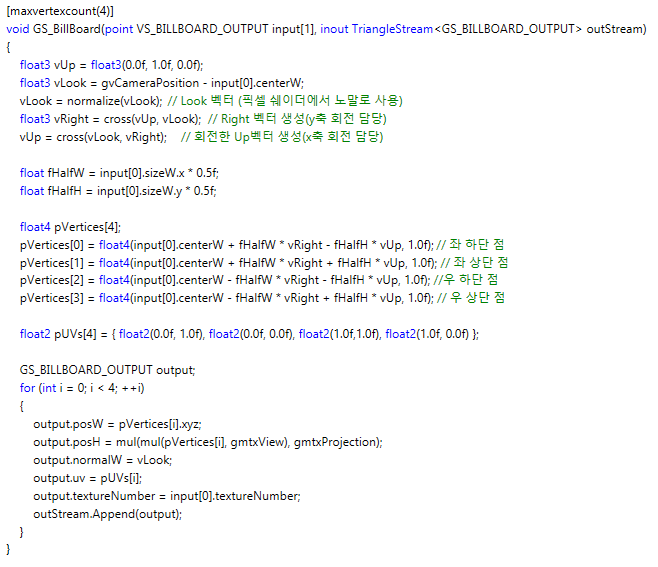
* 1. 플레이어를 처음 위치와 상태로 돌려주는 Reset함수를 추가했습니다.

1. Shader.hlsl
   1. 이펙트를 인스턴싱으로 처리해 주기 위해 월드 변환 행렬과 UV의 변화 값을 가지는 TB\_EFFECTOBJECT\_INFO를 SturcturedBuffer로 t레지스터 7번을 할당해 추가했습니다.
   2. 지형을 그리기 위해 지형 기본 텍스처, 지형 디테일 텍스처, 지형 디테일 노말 텍스처를 순서대로 t레지스터 1, 2, 3번을 할당해 추가했습니다.
   3. 지형의 정점 쉐이더는 이전 프로젝트와 동일하며 픽셀 쉐이더에 현재 정점이 가지고 있는 노말과 탄젠트를 이용해 TBN행렬을 만들어 디테일 텍스처에 노말 맵을 적용하도록 하였습니다.

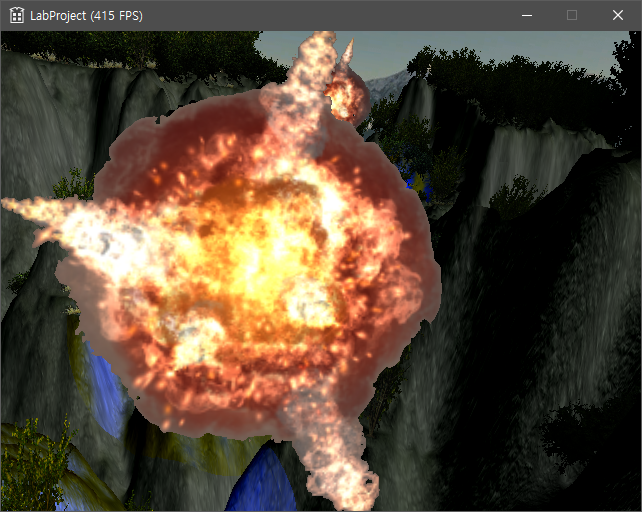
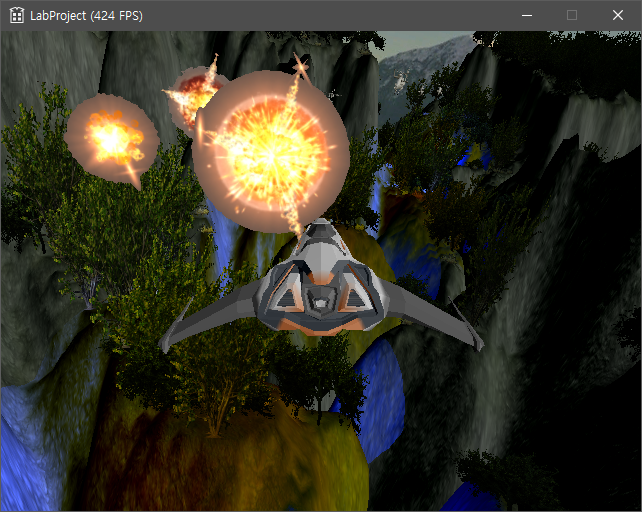


* 1. 스카이 박스를 그리기 위한 PSSkyBox를 추가했습니다.
  2. 풀, 나무를 그리기 위한 VS\_BillBoard, GS\_BillBoard, PS\_BillBoard를 추가했습니다. 저점 쉐이더는 입력 받은 값을 그대로 기하 쉐이더에 넘겨주며 픽셀 쉐이더는 기존 빌보드와 동일합니다. 기하 쉐이더에서는 카메라 위치와 입력 된 정점 위치를 이용해 Look벡터를 만들고 (0, 1, 0)을 Up으로 가정해 이 둘을 외적해 Right를 생성한 뒤 다시 Look과 Right를 외적 하여 Up을 생성하였습니다.

이 정보와 사이즈를 이용해 정점 4개를 만들어 픽셀 쉐이더에 넘겨 주도록 하였습니다.



1. 프로그램 실행 화면
   1. 폭발 이펙트 적용



* 1. 지형과 적 이동

