|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **주차** | 19 주차 | **기간** | 2020.5.04 ~ 2020.05.10 | **지도교수 이용희** | (서명) |
| 이번주 한일 요약 | -내장 그래픽카드로 실행 시 게임 화면 안 나오는 문제(원인불명)  -기총 발사, 기총 카메라 제작  -행동 트리 작동 테스트  -비행기 고도, 플레이 타임, 게임 점수, 미사일 탄 수 표시 폰트 애니메이션 추가  -시작 메뉴 씬 추가 | | | | |

<상세 수행내용>

깃 주소 : <https://github.com/zzcx88/DKS_Project.git>

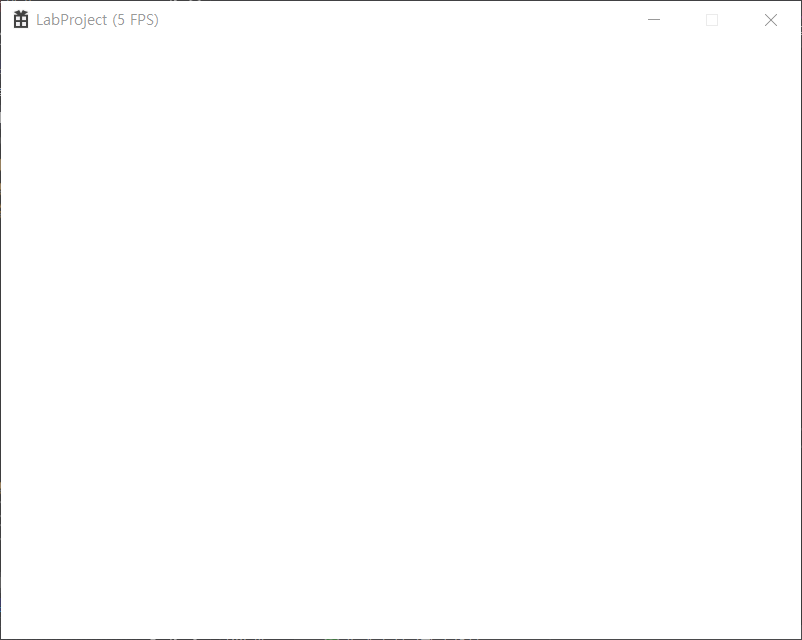
**이재원 :**

**1. 내장 그래픽카드로 실행 시 게임 화면 안 나오는 문제(원인불명)**

팀원의 노트북이 내장 그래픽 카드만 탑재되어 있는데 정식 개강함과 동시에 집이 아닌 외부

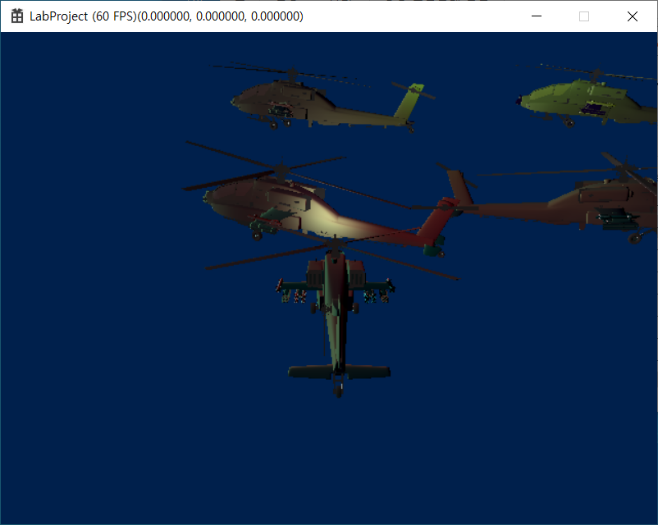
에서 프로젝트를 실행하려고 하면 프로그램에 문제가 발생한다고 하여 해결해 보기로 함

내장 그래픽카드를 사용하여 프로그램을 실행하면 프로그램은 실행되나 아무것도 렌더 되지 않는 현상이 있음



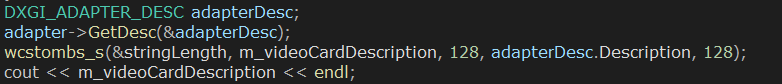
위 사진처럼 FPS가 표시되며 작동하지만 하얀 화면만 나옴

하지만 텍스처를 사용하지 않는 프로그램은 제대로 작동하는 것을 확인

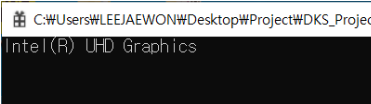


텍스처를 사용하지 않는 예제의 경우 위 사진과 같이 양호한 프레임으로 제대로 렌더 되는 것을 확인함

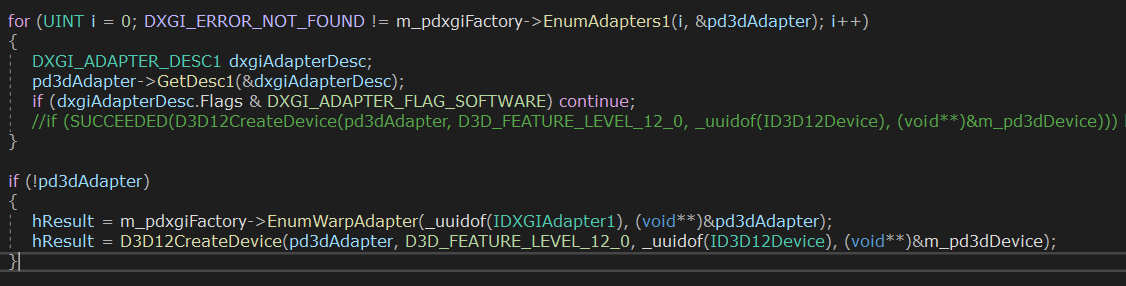
혹시 내장 그래픽 카드를 인식하지 않는 것인지 확인하기 위해 디바이스를 생성하기 전 인식하는 그래픽 카드를 출력해 보기로 함



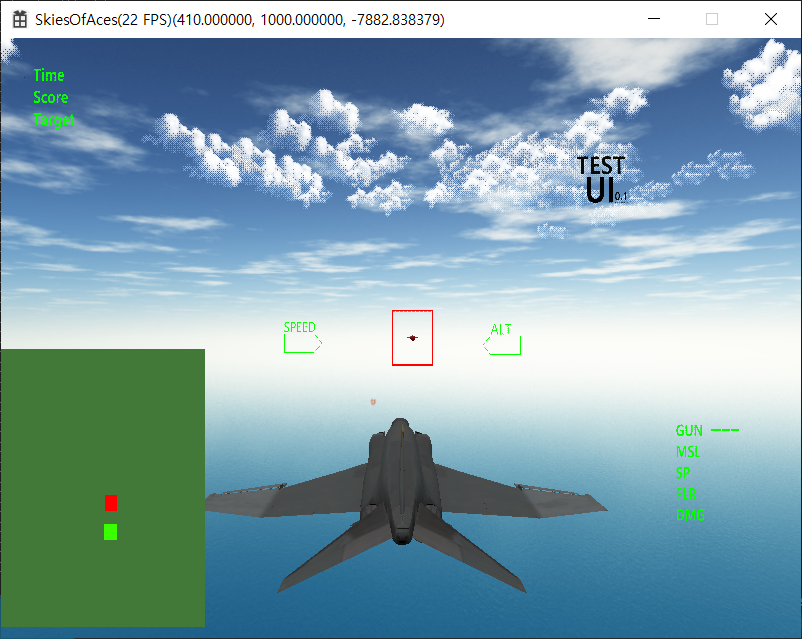
위의 코드로 어댑터의 정보를 출력한 결과 내장 그래픽 카드를 인식한 것이 확인됨



그러나 여전히 아무것도 출력되지 않는 현상이 발생하여 waprAdapter를 강제로 사용하여 디바이스를 생성해 보기로 함



사용 가능한 어댑터를 나열하고 디바이스를 생성하는 부분을 주석을 걸어 하단의 warpAdapter를 통해 디바이스를 생성한 결과 실행이 되는 것을 확인함



하지만 그래픽 카드의 도움 없이 오로지 cpu로만 렌더를 해서 그런지 바다를 모두 없애고 구름을 대폭 축소하고 오브젝트도 플레이어 포함 2개만 남겨 놓았는데도 프레임 레이트가 매우 떨어짐

결과적으로 원인을 찾지 못하여 내장 그래픽 카드를 사용하는 환경에선 프로그램을 실행하는데 제약이 생겨 버림

**2. 기총 발사, 기총 카메라 제작**

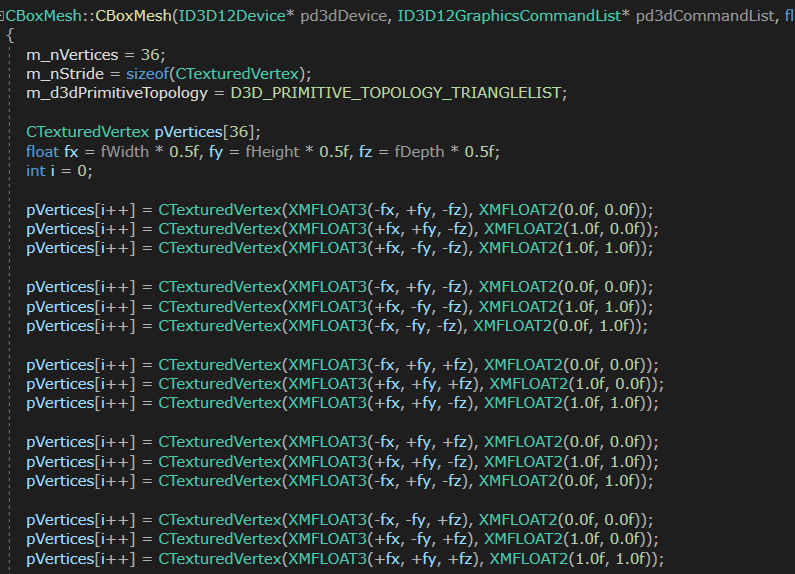
플레이어 기체에서 단거리 무기인 기총을 발사하는 기능을 구현함

기총은 미사일보다 밋밋한 느낌이 있음으로 이를 보완하기 위해 카메라 무빙을 추가함

우선 기총의 총알은 길다란 노란색의 박스로 하기로 함, 또한 기총의 총알은 발사할 때

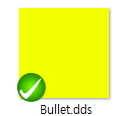
실시간으로 날아가는 것이 보여야 하기 때문에 예광탄(야간사격시 탄환이 발화하며 발사되어

어디에 착탄 되는지 시각적으로 확인할 수 있는 탄 종) 조명과 상관없이 그림자가 발생하지 않고 선명하게 보여야 함으로 FBX모델을 가져오지 않고 직접 박스 메쉬를 생성하여 조명계산없이 그리는 쉐이더를 사용해 만듦

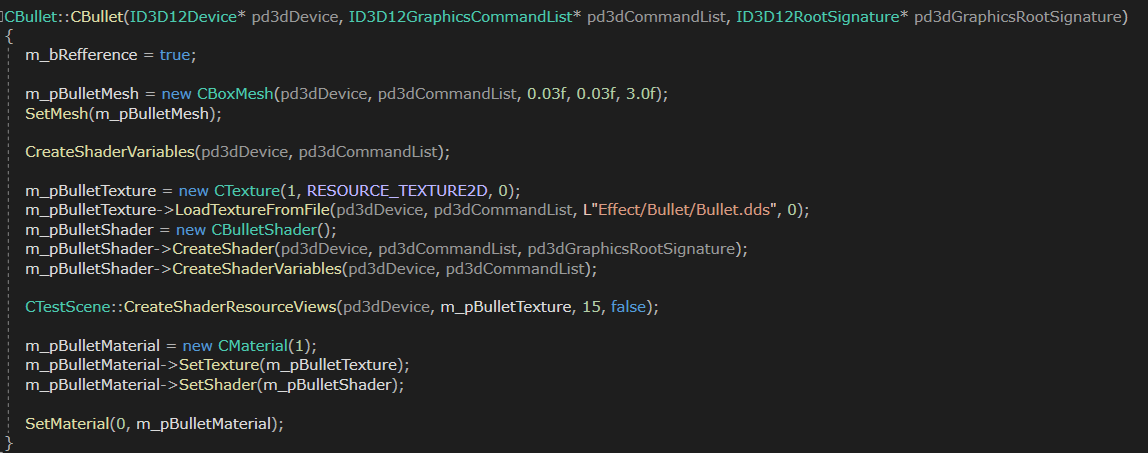


36개의 정점을 사용하는 박스 메쉬를 기존에 UI나 빌보드에 사용했던 평면 메쉬를 참고하여 정의했다.

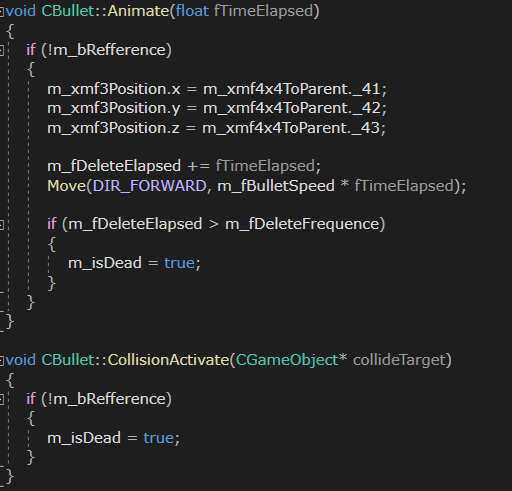
그리고 만들어진 박스 메쉬를 사용하여 Bullet 클래스를 정의했다. 텍스처는 노란색 단일 텍스처를 사용했다.



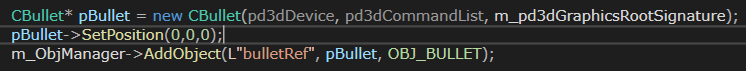
Bullet 클래스는 기존의 평면 오브젝트 클래스를 정의하는것과 같은 방식으로 정의한다.



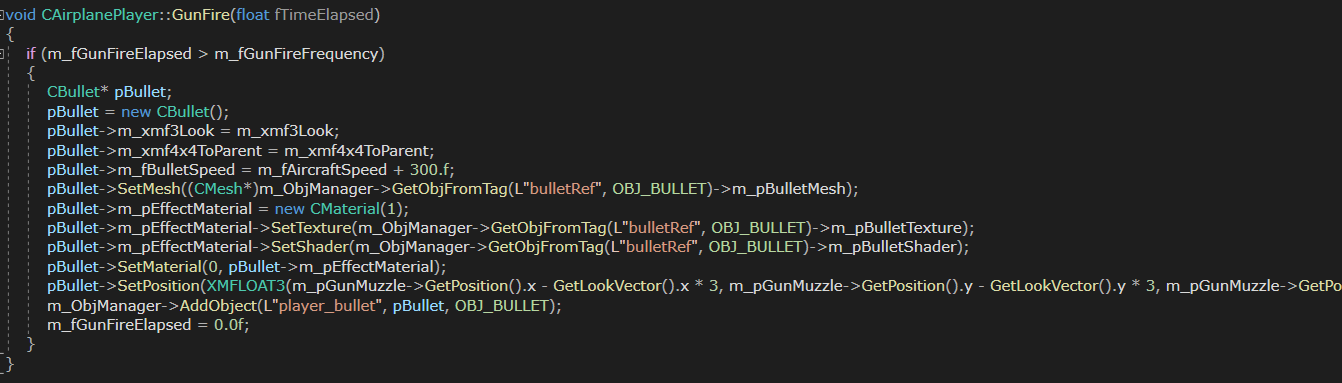
또한 기총이 날아가는 동작과 시간에 이 지나면 자동으로 삭제되는 동작을 정의한다.



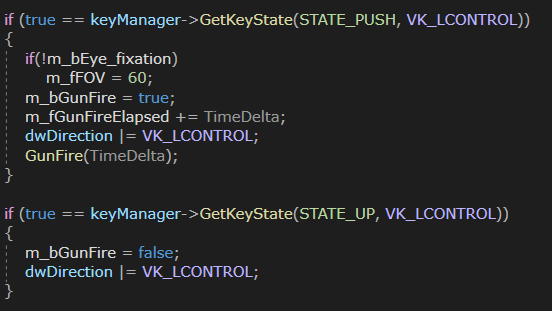
추가로 추후 충돌처리가 되었을 때 CollisionActive함수가 실행되면 래퍼런스 오브젝트가 아닐 경우 삭제되게끔 한다.



씬의 BuildObject함수에 마찬가지로 래퍼런스 오브젝트를 생성한 후(OBJ\_BULLET의 enum값을 새로 추가했다.)



총알을 생성하는 함수로 GunFire를 정의하는데 방식은 미사일 발사와 유사한 방식으로 총알이 플레이어의 총구 위치로부터 동적 생성되게끔 레퍼런스 오브젝트의 정보를 사용하여 생성한후 오브젝트 매니저에 적재를 하게끔 하고

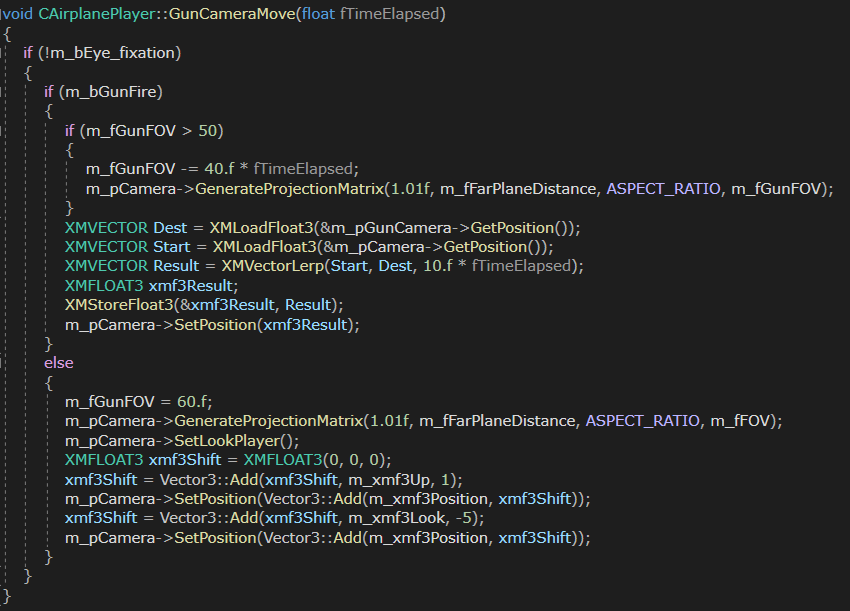


오른쪽 컨트롤 키를 누르는 동안 위에서 정의했던 GunFire함수가 호출되어 총알이 생성되어 발사되게끔 하고 기총 발사동안 카메라 무빙과 가 감속 FOV조절에 예외처리를 두기 위해 m\_bGunFire라는 불 값 변수를 껐다 키도록 한다.



기총이 발사 되는 동안은 W, S 키를 눌러도 가, 감속만 될 뿐 FOV는 조절되지 않도록 예외처리를 한다.

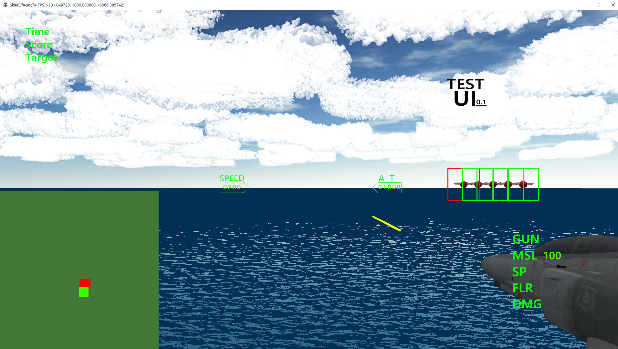
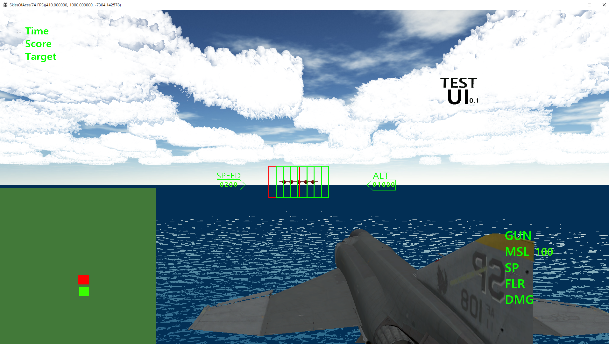
이제 기총을 발사하는 순간부터 중단할 때까지 카메라를 전투기 왼쪽 주익상단에 위치시켜 기총기 발사되는 것을 역동적으로 감상할 수 있도록 카메라 무빙 함수를 작성한다.



플레이어가 TAB을 눌러 적에게 시선 고정을 하지 않고 기총을 발사 중이라면 FOV를 서서히 줄여 줌인 하는 효과를 줌과 동시에 Lerp함수로 현재 카메라 위치와 왼쪽 주익 상단의 위치를 보간 하여 카메라를 서서히 이동시킨다.

만약 기총 발사중이 아니라면 FOV와 카메라 위치를 원래의 디폴트(게임 시작시 카메라 위치)로 되돌린다.

이렇게 정의한 이 함수는 매프레임 호출된다.

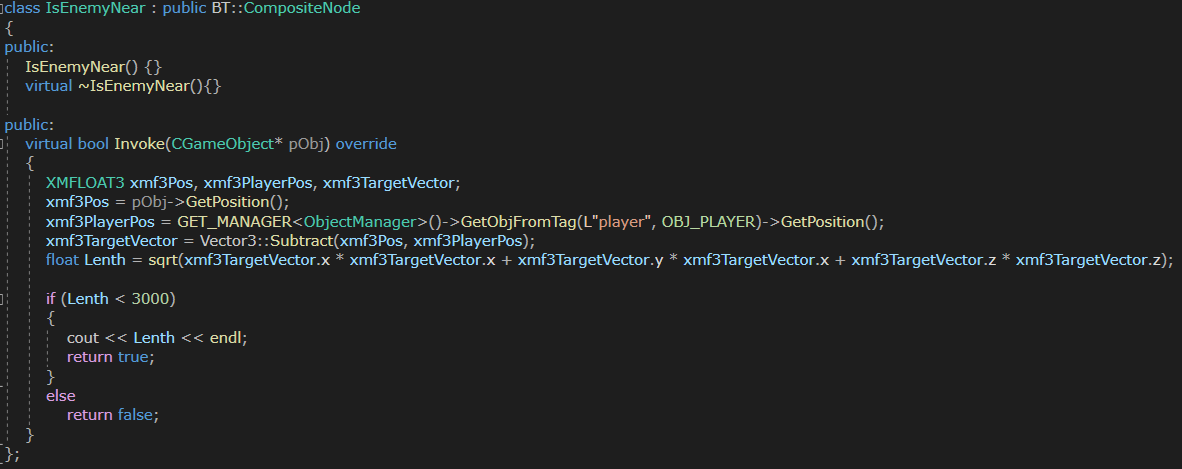


결과적으로 기총 발사시 위 사진과 같이 카메라가 이동한 후 기총이 발사가 된다.

**3. 행동 트리 작동 테스트**

저번주에 만들었던 행동 트리의 노드를 가지고 행동이 작동하는지 테스트 해보았다.

우선 시퀀스 노드로서 실행될 행위를 하나 정하여 노드 클래스를 생성한다.



IsEnemyNear클래스는 이 클래스를 호출하는 오브젝트 주위에 적이 있는지 판별하는 클래스이다.

이 클래스를 호출하는 주체는 적 오브젝트가 될 것임으로 우선 적 오브젝트가 플레이어의 위치를 감지하도록 작성한다.

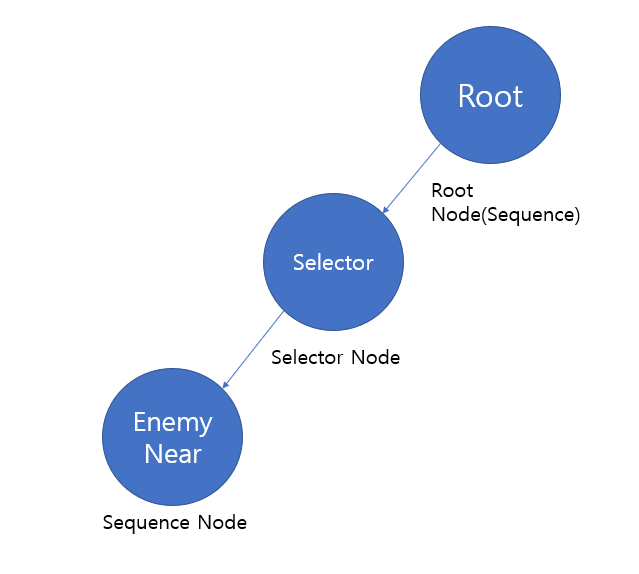
거리가3000유닛 이하라면 콘솔창에 오브젝트와 플레이어 간의 거리를 출력하도록 한다.

완성된 IsEnemyNear 클래스는 노드임으로 AIManager에서 Tree를 생성할 때 사용한다.



MakeAction 함수에서 여러 노드를 만들었지만 실제로 트리를 구성하는데 사용한 노드는 Root, Selector 그리고 시퀀스 노드인 IsEnemyNear 뿐이다.

그래프화 하면 다음과 같은 단순한 사향 트리가 된다.



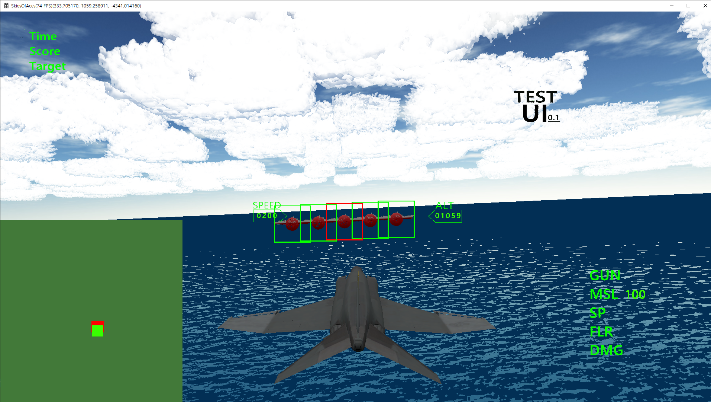
그리고 해당 트리의 행위를 사용할 오브젝트의 정보를 매개변수로 입력받아 실행할 수 있도록 함수의 매개변수를 수정하고 오브젝트가 생성되고 DoAction 함수를 최초 호출할 때 트리를 구성하여 생성할 수 있도록 AI를 가지고 있는지 판별하는 bool변수를 오브젝트가 가지고 있게 하여

AI가 생성되면 다음 프레임 부터는 트리가 재 생성되게 하지 않게 하여 최초 한 번만 실행되도록 장치를 해둔다.

만약 해당 오브젝트가 트리를 가지고 있다면 매프레임마다 루트 노드부터 순회하여 행위를 실행할 것이다.



이제 AIManager를 통하여 오브젝트가 DoAction을 호출하도록 한다. DoAction을 통하여 행위를 할 주체는 바로 자신이기 때문에 this를 매개변수로 넘겨준다.



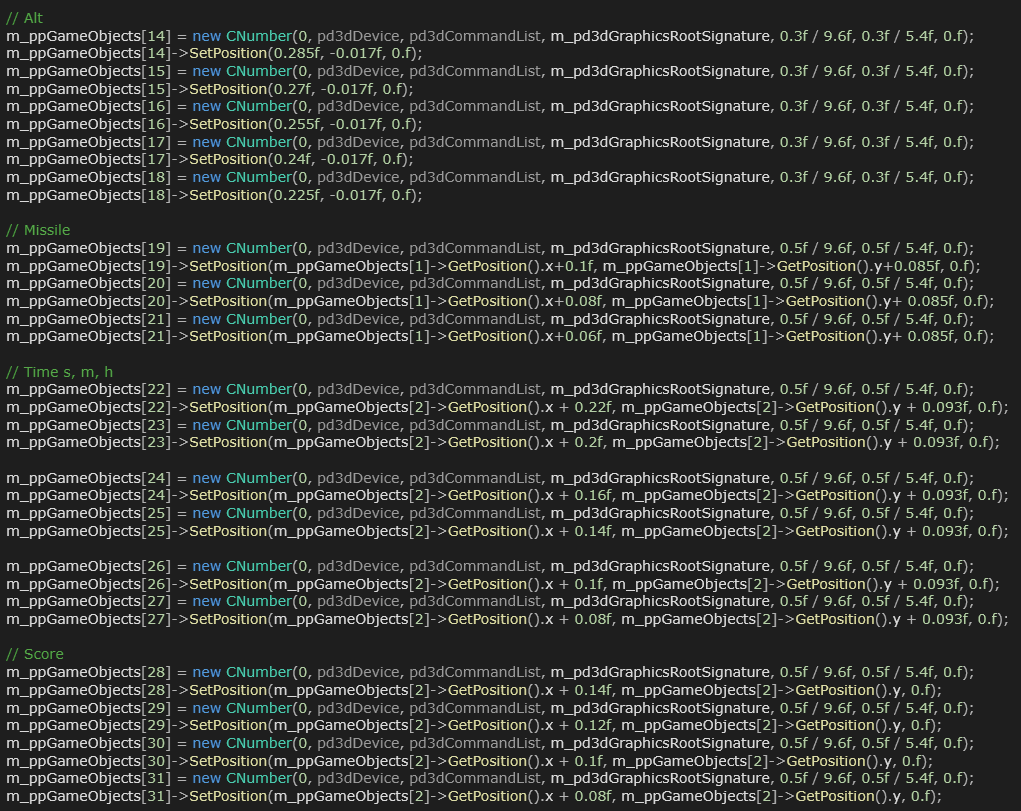
실행 시 플레이어와 적과의 거리가 충분히 가까워지면 플레이어와 적과의 거리가 콘솔창에 매프레임 출력되는 것을 확인하였다.

**김령운:**

1. **비행기 고도, 플레이 타임, 게임 점수 폰트 애니메이션 추가**

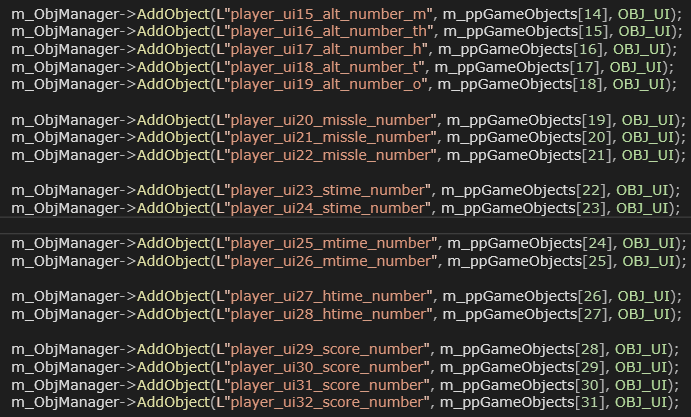
플레이어 기체의 고도를 실시간으로 알려주는 폰트 애니메이션과 게임 진행 시간을 알 수 있는 플레이 타임, 플레이어의 게임 점수를 확인할 수 있는 폰트 애니메이션을 추가했다.

**그림1. CTestScene Class의 BuildObjects 함수 부분(폰트 텍스처 빌드)**

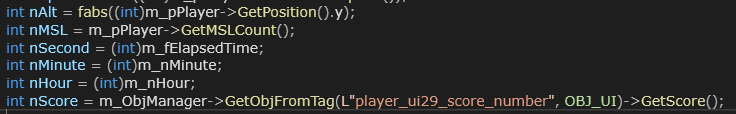
****

플레이어 고도의 최대 자리 수는 1’0000, 미사일 잔탄 개수의 최대 자리 수는 100, 플레이 타임의 최대 자리 수는 초 분 시 마다 10, 플레이어 점수의 최대 자리 수는 1000이다. 이 이상의 자리 수의 데이터는 표현하지 않는다(예를 들어 고도가 10’0000 이 될 경우 1’0000의 자리 수 까지만 표현하기 때문에 00000 이 표시됨). 각 최대 자리 수는 플레이어가 게임 플레이를 하면서 저 수치를 넘지 않을 것이라는 가정하에 설정하였다.

**그림2. CTestScene Class의 BuildObjects 함수 부분(오브젝트 매니저에 폰트들을 저장)**

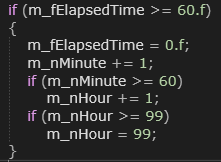
****

**그림3. CTestScene Class의 AnimateObjects 함수 부분(고도, 미사일 개수, 플레이 타임, 점수를 지역 변수에 저장)**

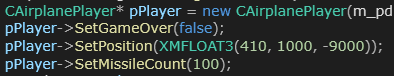
****

고도는 마이너스 값이 대입될 경우 프로그램이 오류가 떠서 절댓값으로 대입되게끔 했다. 미사일 개수를 가져오는 함수로 플레이어 클래스의 GetMSLCount()를 사용했다. 플레이 타임은 초에 ElapsedTime이 더해지도록 했으며 60초가 되거나 보다 커질 경우 분이 1 증가하고 60분이 될 경우 시간이 1 증가하도록 했다.

**그림4. CTestScene Class의 AnimateObjects 함수 부분(시간 증가)**



**그림5. CDeviceManager Class (플레이어 빌드)**

****

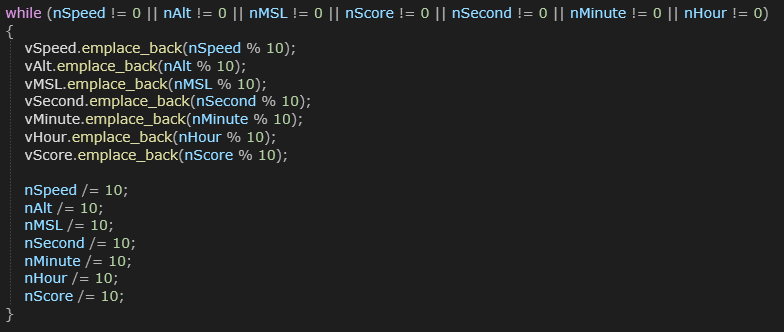
디바이스 매니저에서 플레이어를 빌드할 때 미사일 개수를 SetMissileCount(int) 함수를 이용해 초기값을 설정해줬다.

**그림6. Missile Class의 CollisionActivate 함수 부분(점수 추가)**

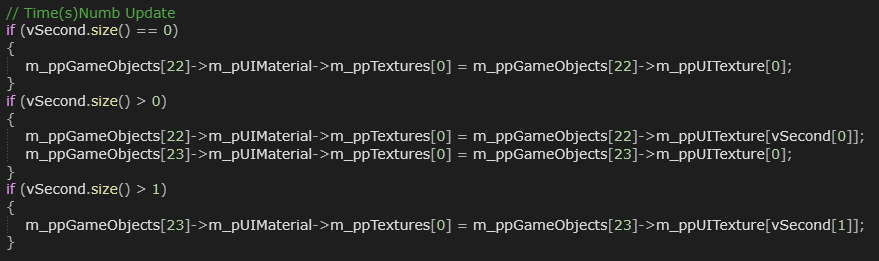


미사일과 적 오브젝트가 충돌해서 적 오브젝트가 소멸하면 플레이어의 점수가 50 점씩 추가되도록 했다. 점수는 플레이어 점수 폰트 객체가 가진 멤버 변수로 설정했다.

그림7. **CTestScene Class의 AnimateObjects 함수 부분**(벡터에 폰트를 출력할 값들을 추가)

****

**그림8. CTestScene Class의 AnimateObjects 함수 부분(폰트 값 업데이트)**

****

다음과 같은 방법으로 다른 값들도 폰트가 출력되도록 했다.

**[문제점]**

테스트 씬 클래스의 애니메이트 오브젝트 함수에 직접 작성하여 관리가 어렵다.

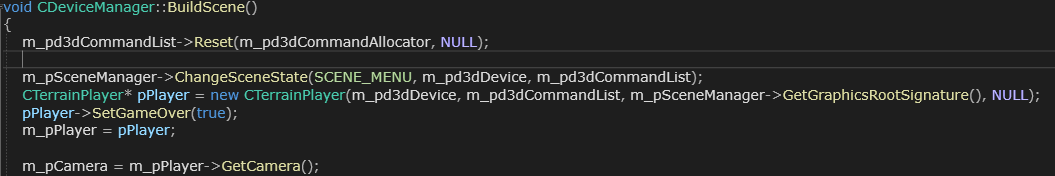
**[해결방안]**

UI 매니저에 빌드 폰트 함수와 애니메이트폰트 함수를 추가하여 오브젝트 함수의 업데이트에서 함수 호출하여 해결, UI 매니저에 폰트를 빌드하기 위해서 각 폰트마다 태그를 달리하여 각 특정 값에 접근할 수 있도록 변경한다.

1. **시작 메뉴 씬 추가**

**게임 시작 시 진입하는 시작 메뉴 씬을 추가하고 F5키를 이용하여 씬 전환이 가능하도록 구현했다.**

**그림9. CDeviceManager Class 의 BuildScene 함수 부분(처음 시작 메뉴 씬 빌드)**

****

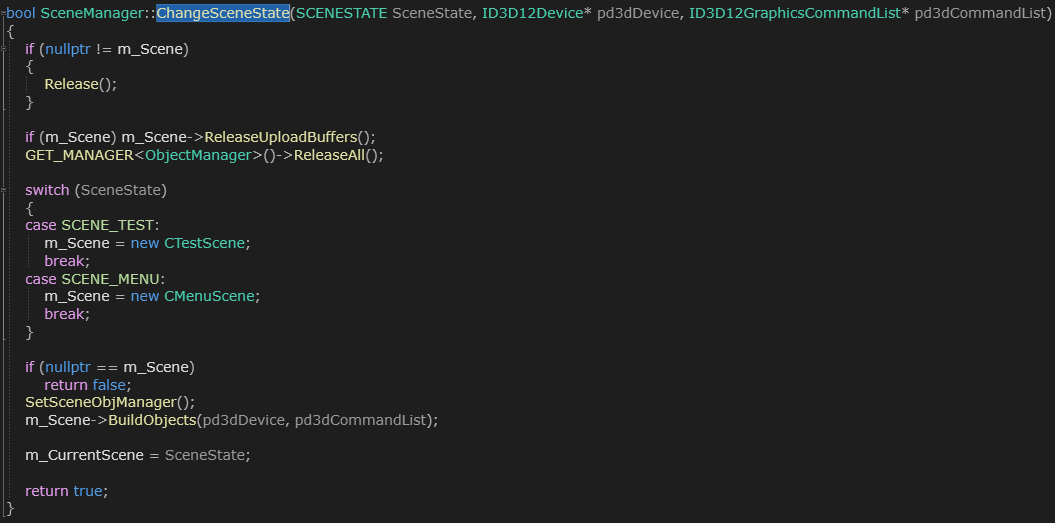
처음 씬을 빌드할 때 시작 메뉴를 빌드하도록 했다.

**그림10. CDeviceManager Class의 OnProcessingKeyboardMessage 함수 부분(씬 전환)**

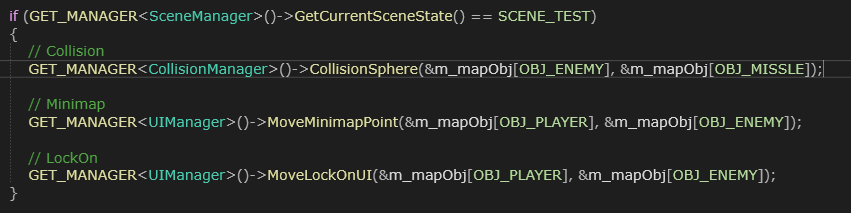


F5 키를 누를 때마다 SceneManager Class의 ChagerSceneState 함수를 이용해 기존 씬을 삭제하고 새로운 씬을 빌드한다. 그리고 커맨드 리스트를 리셋하고 플레이어도 새롭게 빌드해준다. 게임 타이머도 리셋 하여 시작 메뉴로 갔다가 플레이 씬으로 전환할 경우 항상 새롭게 게임이 시작되도록 했다. 시작 메뉴에서는 플레이어 클래스의 GameOver bool 값을 true로 설정하여 플레이어 클래스의 Update와 Render 함수가 호출되지 않도록 했다.

**그림11. SceneManager의 ChageSceneState 함수(이전 씬 삭제, 새로운 씬 빌드)**



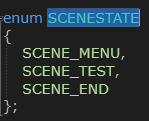
**그림12. ObjectManager Class의 Update 함수 부분**



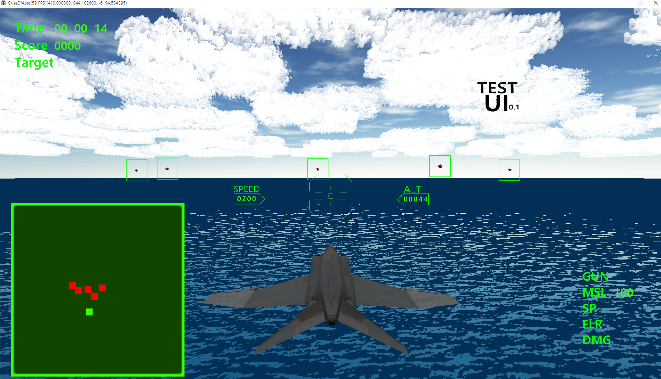
SceneManager Class의 GetCurrentSceneState 함수를 이용해 플레이 씬일 경우에만 미니맵, 락 온 충돌처리가 Update, Render 되도록 했다.

SCENESTATE GetCurrentSceneState() { return m\_CurrentScene }

**그림13. SCENESTATE enum값 설정**



**그림14, 15. 실행 화면**

****

**[문제점]**

**씬 전환 속도가 5초 정도로 좀 느린 상태인데 씬 전환 시 빌드하는 순간에는 게임이 멈춰있다. 플레이어가 차분히 빌드 시간을 기다릴 수 있도록 해야함**

**[해결방안]**

**플레이 씬을 빌드할 때 빌드 진행상황을 보여주고 빌드가 모두 끝났을 때 키 추가 입력으로 바로 게임에 진입할 수 있도록 한다.**

|  |  |  |  |
| --- | --- | --- | --- |
| **문제점 정리** | 이재원 :  -행동 트리의 작동을 확인 했지만 여전히 툴 사용까지는 시간이 필요  -기총 충돌 미 작성  김령운 :  -씬 전환 속도가 느림  -여전히 폰트 애니메이션이 유동적이지 못함 | | |
| **해결방안** | 이재원 :  -우선 기본적인 행위를 가지는 노드를 미리 작성하고 툴에서 만들어진 노드를 가지고 트리를 구성할 수 있게 유동적인 구조를 생각해야함  -기총은 박스 메쉬 임으로 구 충돌체를 쓰는 오브젝트와 박스메쉬간의 충돌 검사를 고려하여 충돌 처리를 작성해야함  김령운:  -로딩 화면을 만들어 로딩 상태를 플레이어에게 알려줌  -폰트 애니메이션과 텍스처 빌드를 UI 매니저에서 수행하도록 수정 | | |
| **다음주차** | 20 주차 | **다음기간** | 2020.05.11 ~ 2020.05.17 |
| **다음주 할일** | 이재원 : Behavior 툴 제작  김령운 : Start Scene에서 Play Scene으로 전환 시 빌드 진행 상황을 알려주는 Scene 제작, 폰트 애니메이션 UI 매니저에 추가, 모션 블러 | | |
| **지도 교수**  **Comment** |  | | |