|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **주차** | 15 주차 | **기간** | 2020.04.13 ~ 2020.04.19 | **지도교수 이용희** | (서명) |
| 이번주 한일 요약 | -오브젝트 매니저에 플레이어 객체 넣기  -전투기 가속, 감속 구현  -Yaw애니메이션 및 기능 구현  -애프터 버너 효과 기초작업  -행동트리 툴 제작을 위한 GUI오픈소스 탐색  -UI매니저 클래스 구현 | | | | |

<상세 수행내용>

**이재원 :**

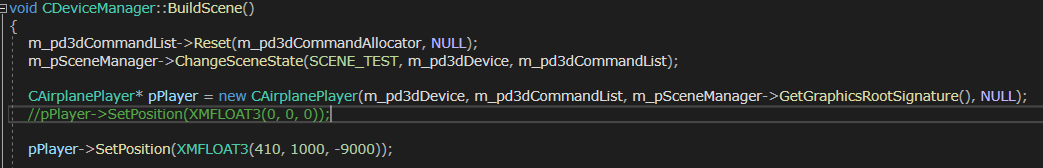
**-오브젝트 매니저에 플레이어 객체 넣기**

지금까지는 디바이스 매니저에서 플레이어를 갱신하는 것을 수업시간 코드를 그대로 사용하여

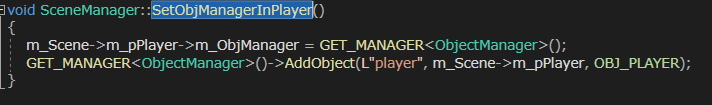
디바이스 매니저 내의 FrameAdvence함수 내에서 업데이트와 랜더를 처리하였다.

이후 프레임워크의 다른 곳에서 플레이어의 정보가 필요 할 때 넘겨주는 것이 불편하여 미루고 있던 작업으로 플레이어 객체를 오브젝트 매니저에 넣어 관리하도록 개선하였다.

플레이어 내의 함수명과 오브젝트 매니저가 담는 CGameObject의 함수 이름과 형식이 호환되지 않아 플레이어 내의 함수 형식을 CGameObject의 함수와 우선 통일시켜 준 후



디바이스 매니저의 BuildScene에서 플레이어를 생성하고 CScene클래스의 맴버변수에 플레이어를 포인터로 넘겨준다.



씬 객체로 넘겨받은 플레이어 객체를 오브젝트 매니저를 통해 컨테이너에 적재한다.

이로써 플레이어 정보도 오브젝트 매니저를 통해 접근할 수 있게 되었다.

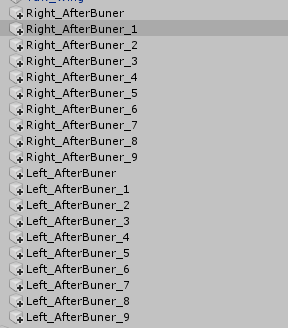
**-전투기 가속, 감속 구현, 애프터 버너 효과 기초작업**

지금까지는 전투기는 처음 생성될 때 미리 지정되었던 속도로만 움직였었다.

키보드 W키로 가속, S로 감속을 구현한다.

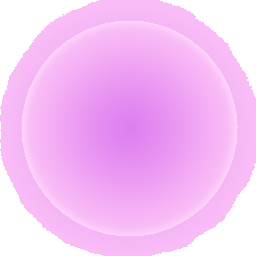
또한 애프터 버너 또한 빌보드 객체로 만들 것이고 유동적이진 못하지만 단순한 방법으로 제작한다.

우선 유니티로 사용하는 전투기 모델의 애프터 버너 위치를 추가한다.

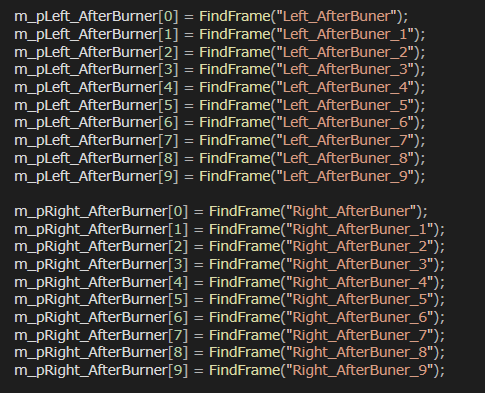


F-4 Phantom II 모델의 계층구조내에 Left, Right 애프터 버너의 위치를 빈 객체로 일정한 간격을 두어 만들어 놓고 익스포트 한다.

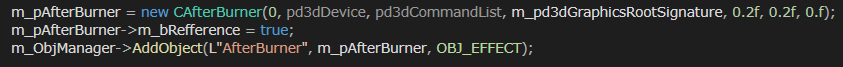
그리고 포토샵 프로그램으로 애프터 버너의 이미지를 생성한다.

 (생성된 애프터 버너 이미지)

이제 만들어진 애프터 버너 이미지를 사용하여 애프터 버너 객체를 생성하고 유니티에서 작업했던 애프터 버너 위치에 매 프레임마다 위치 시킨다.

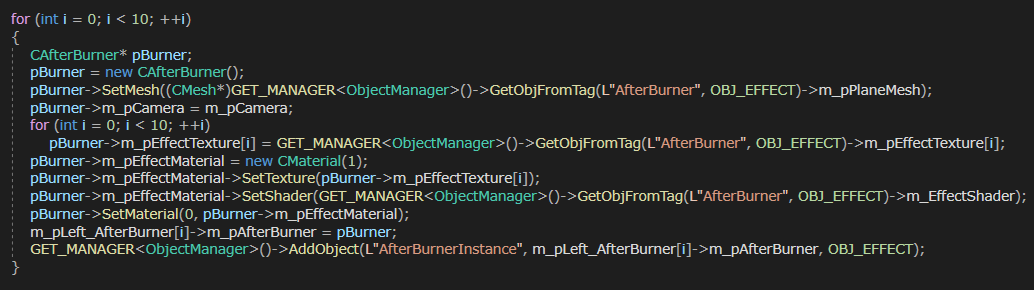


게임 오브젝트로 활용하기 위해 해당 프레임을 찾아서 연결한 후

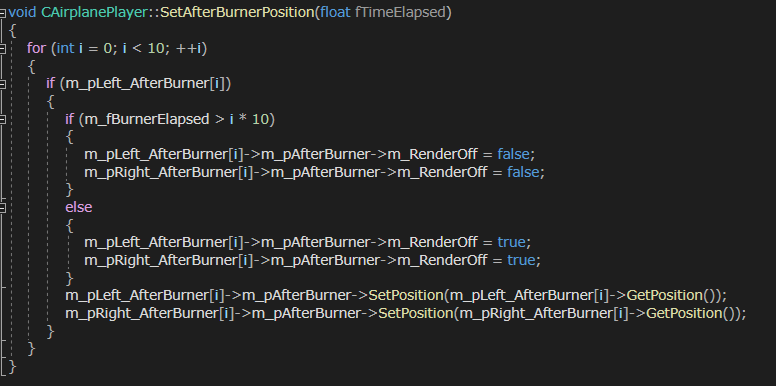


CTestScene 클래스에서 래퍼런스 오브젝트로 애프터 버너를 생성한다.

빌보드 객체임으로 CAfterBurner의 구조는 CMissleFog와 같다.



애프터 버너를 래퍼런스 오브젝트의 정보로 동적 할당하여 각 프레임에 넣어준다.(사진은 Left만 있지만 Right도 같이 하였음)



그 후 매 프레임 마다 애프터 버너 프레임의 위치를 받아와서 할당하였던 맴버변수

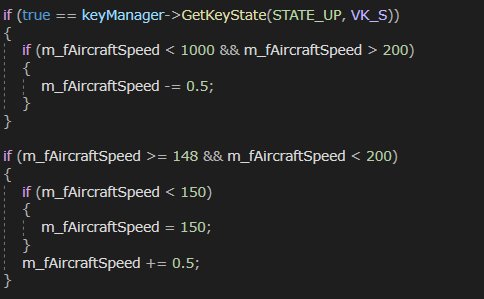
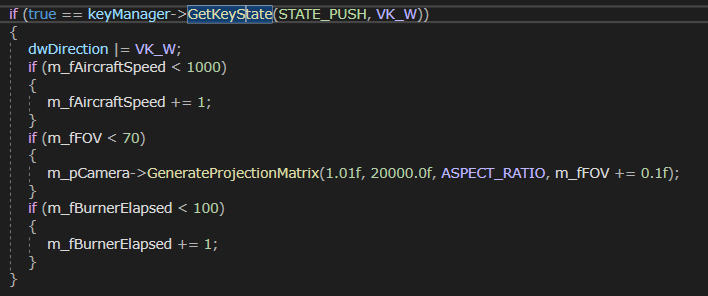
m\_pAfterBurner의 위치를 갱신해준다.

위치를 갱신할 때 m\_fBurnerElased의 수치가 10씩 증가할 때마다 하나씩 켜지게 만들어서 버너

가 엔진의 제일 가까운 곳부터 먼곳 순서대로 켜지도록 한다.

그리고 가속 감속은 m\_fAircraftSpeed 라는 CPlayer의 맴버변수를 생성하여

W 혹은 S키가 눌리는 동안 실시간으로 변화하게 만들어 속도를 조절하게끔 한다.



위의 코드에선 W가 눌렸을 경우와 S를 뗐을 경우의 행위를 정의한 경우이다. 외에도 S가

눌렸을 때, W가 떼어졌을 때, 그리고 아무 키도 안 눌렸을 때 처리하는 부분을 따로 작성하였다.

가속과 감속 시, m\_fFOV라는 변수에 수치를 조절하여 카메라의 FOV를 점진적으로 조절하여

가, 감속 시 시각적인 연출을 주었다.

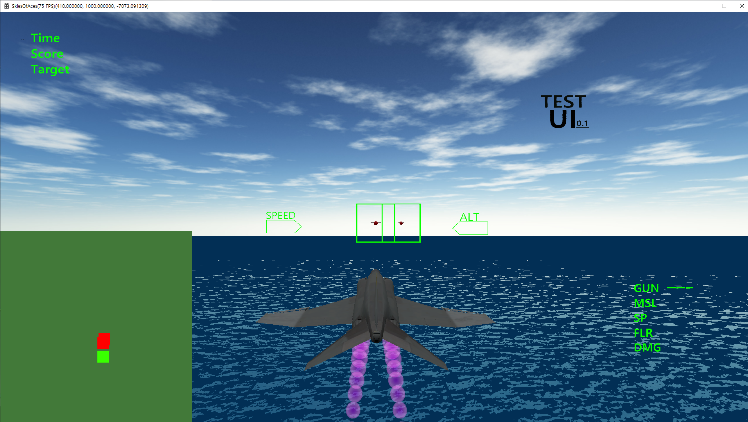
또한 m\_fBurnerElased수치 또한 조절하여 애프터 버너가 켜지도록 하였다.

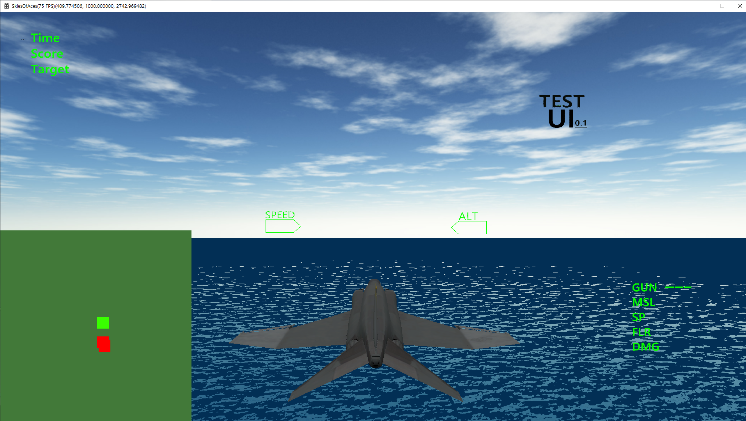
비행기의 최고 속도 수치는 1000이며 S를 누르면 크게 감속하고, W, S 모두 누르지 않은 상태면 서서히 감속하도록 만들었다.

만약 전투기의 속도가 200 이하로 떨어질 경우엔 150 이하로는 떨어지지 않고

아무것도 누르지 않을 경우 200까지 자동으로 속도를 회복하도록 하였다.

150 이하로는 실속에 빠지는 것을 추가할 예정이다.





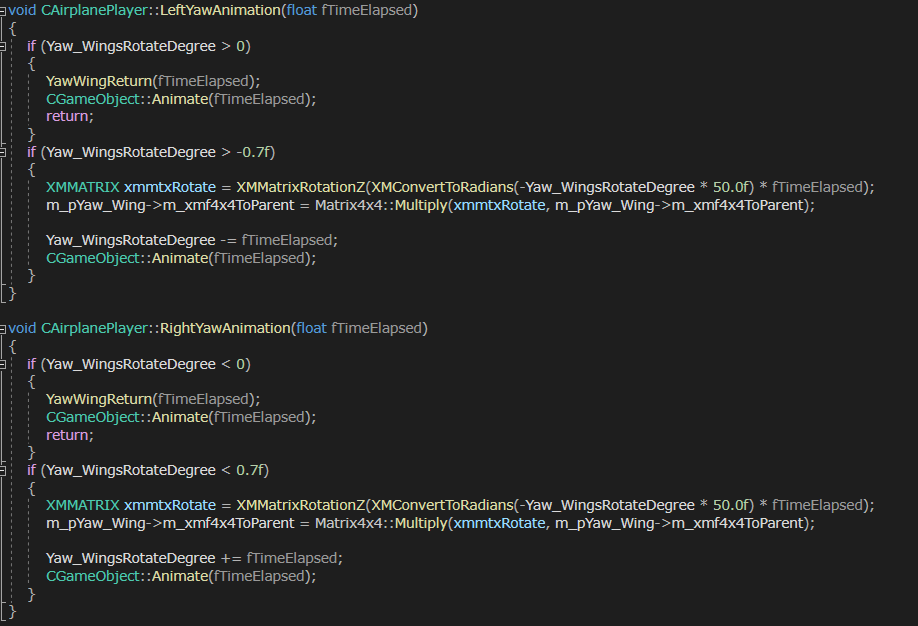
(일정 시간동안 가속하여 모든 애프터 버너가 켜지고 FOV 수치가 증가한 상태와 그 반대 상태)

**-Yaw 애니메이션 및 기능 구현**

이미 Roll과 Pitch 애니메이션을 이전에 구현하였음으로 이전 코드를 활용하여 Yaw태그를 가진

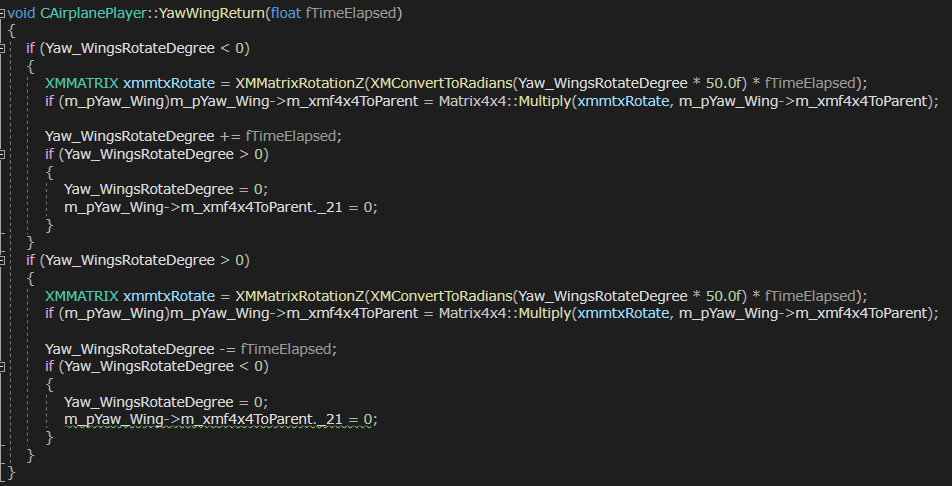
날개 프레임의 행렬을 Q, E키가 눌렸을 때 애니메이션과 전투기의 Y축 회전이 점진적으로 이루어

지도록 만들었다.



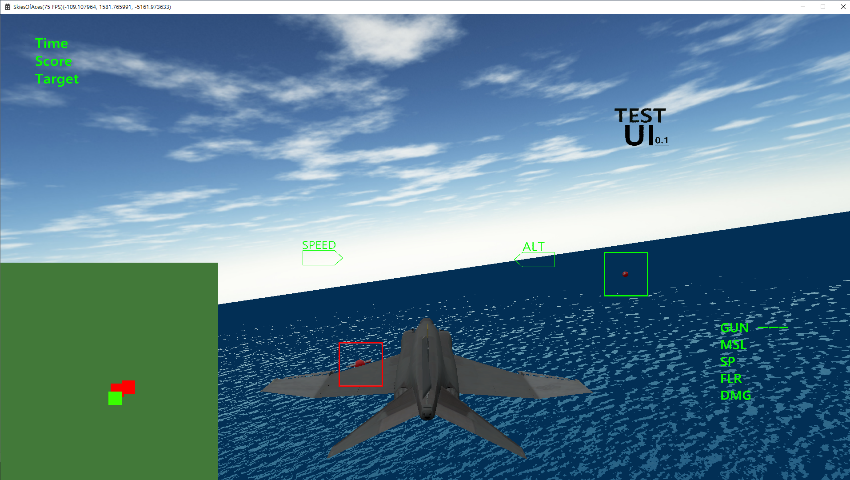
(요 애니메이션을 실행하는 코드, RotateDegree 만큼 해당 부위 날개를 회전시키고 아무것도

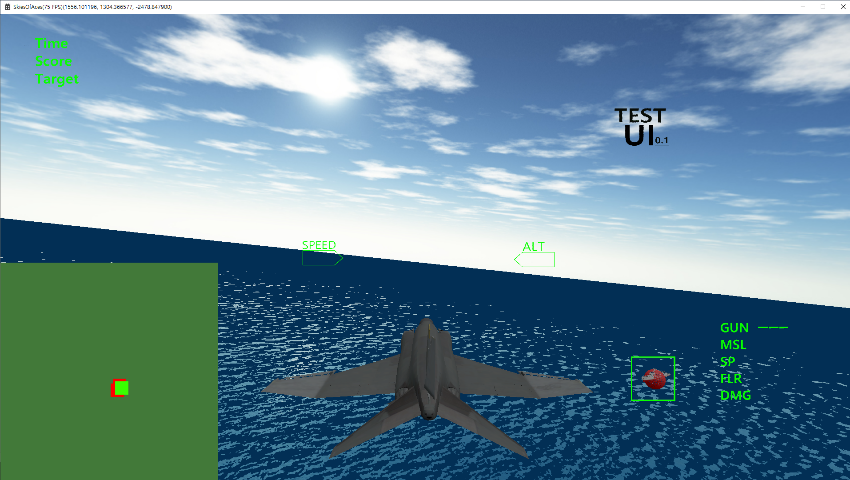
누르지 않으면 원래 자리로 복구시킨다.)



(원래 자리로 날개를 복구시키는 WingReturn 함수 틀어졌던 Rotate만큼 반대방향으로 다시

돌려준다.)





(수직 꼬리 날개가 틀어진다.)

**-행동트리 툴 제작을 위한 GUI오픈소스 탐색**

현재 imGUI라는 오픈소스 라이브러리를 알아냈고 사용법을 알아가는 중

**[문제점]**

행동트리 툴 제작의 진행이 더뎌 지고 있는 와중에 다른 잡다한 작업들이 많이 남아있어서 촉박함.

**[해결방안]**

하루에 잡다한 작업 진행을 하나씩 끝내고 툴제작에 시간을 더 투자하는 방향으로 진행해야 함.

**김령운 :**

**UI 관련 여러 객체들을 관리하기 위해서 UI 매니저 클래스를 작성하여 수정, 추가 할 수 있도록 했다. 본래 의도는 미니맵에 여러 객체들을 동시에 그려내고 관리하기 위한 미니맵 매니저 클래스 구현이었으나 적 오브젝트와 아군 오브젝트들 관련 UI를 통합적으로 관리할 수 있는 UI 매니저(미니맵, 락 온 등)로 수정하였다.**

1. **UI 매니저 (미니맵)**

앞으로 추가되는 적 오브젝트, 아군 오브젝트 다수의 위치를 미니맵에 표시하고 관리가 편하게 하기 위해서 CUI.cpp의 MoveMinimapPoint() 함수를 오브젝트의 리스트를 순회하며 오브젝트마다 호출해주는 작업이 필요했다.

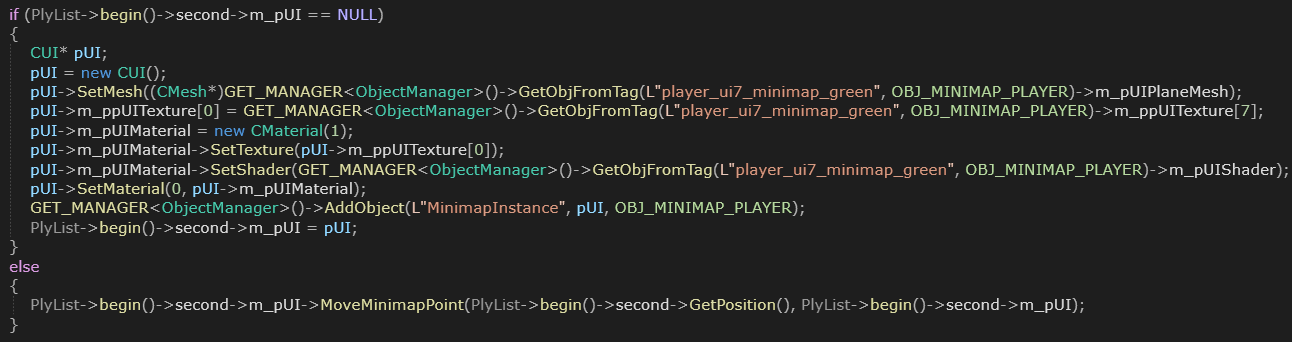
이를 위해서 모든 오브젝트들을 관리하고 있는 오브젝트 매니저의 Update 함수에 오브젝트의 태그네임으로 오브젝트를 전달하는 UI 매니저의 MoveMinimapPoint() 함수를 호출하도록 했다. 오브젝트 매니저의 Update() 함수는 테스트 씬의 AnimateObjects 에서 호출되는 함수다.

**그림1. 오브젝트 매니저 클래스의 Update()에서 호출되는 MoveMinimappoint()**

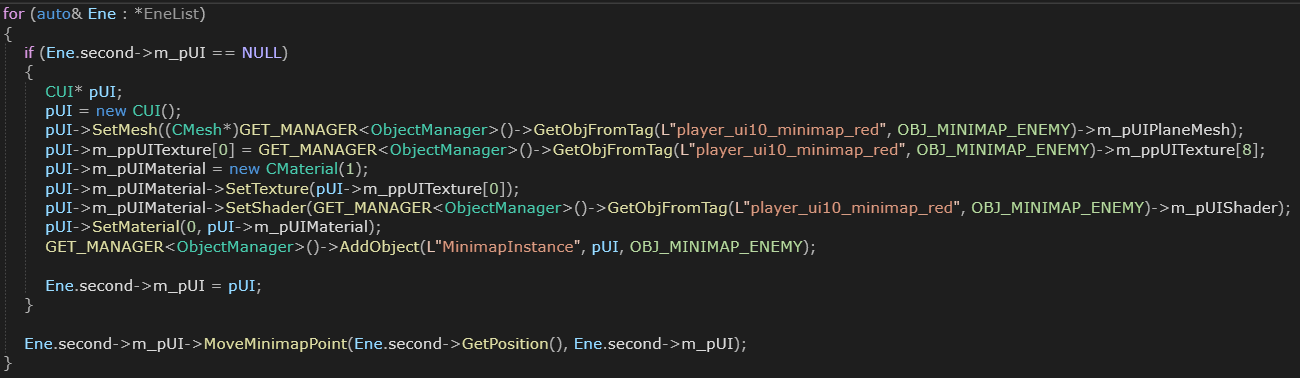


처음 UI 매니저에서 MoveMinimapPoint() 함수를 작성할 때는 오브젝트의 수가 늘어날 때마다 UI를 동적으로 생성하고 Vector 자료구조에 동적 생성된 UI를 추가했다. 오브젝트마다 Vector의 배열 내에 들어있는 UI객체들에게 월드 좌표를 전달해주어 Vector에 오브젝트 수만큼 들어가 있는 UI가 미니맵에 표시되게끔 구현했다. 하지만 이 방법은 함수 내부에서 오브젝트들의 리스트를 순회하고 벡터도 순회해야 하는 단점이 있었다. 이 단점은 또 다른 개발원인 이재원 팀장이 오브젝트들에게 UI 객체를 가지고 있게 하고 오브젝트들의 리스트를 순회할 때 UI객체가 NULL 값인 오브젝트만 동적 생성하게 하여 오브젝트들의 리스트를 한번 순회하는 것만으로 수정하여 해결했다.

**그림2. UI 매니저 클래스의 MoveMinimapPoint() 함수 내 플레이어 미니맵 UI 생성, 호출**

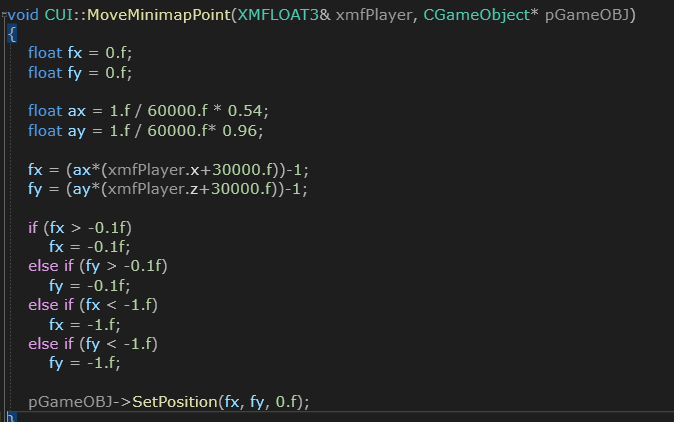
****

**그림3. UI 매니저 클래스의 MoveMinimapPoint() 함수 내 적 오브젝트 미니맵 UI 생성, 호출**

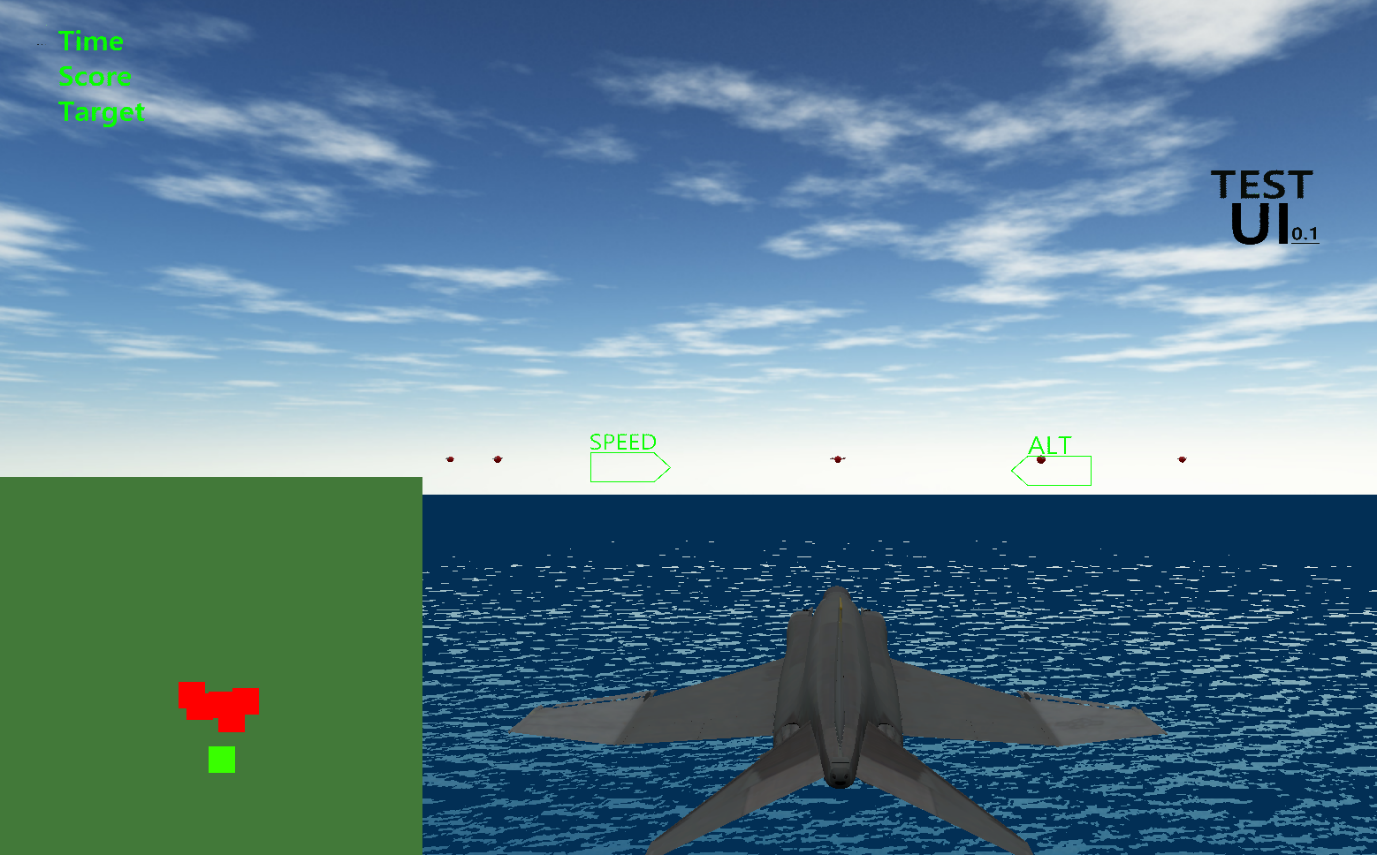
****

동적생성에 쓰이는 자원들은 오브젝트 매니저에 저장된 오브젝트로부터 받아와서 사용했다.

**그림4. CUI 클래스의 MoveMinimapPoint() 15주차와 함수 로직은 동일하다.**

****

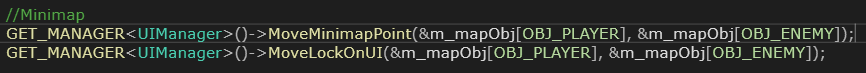
**그림5. 실행화면**

****

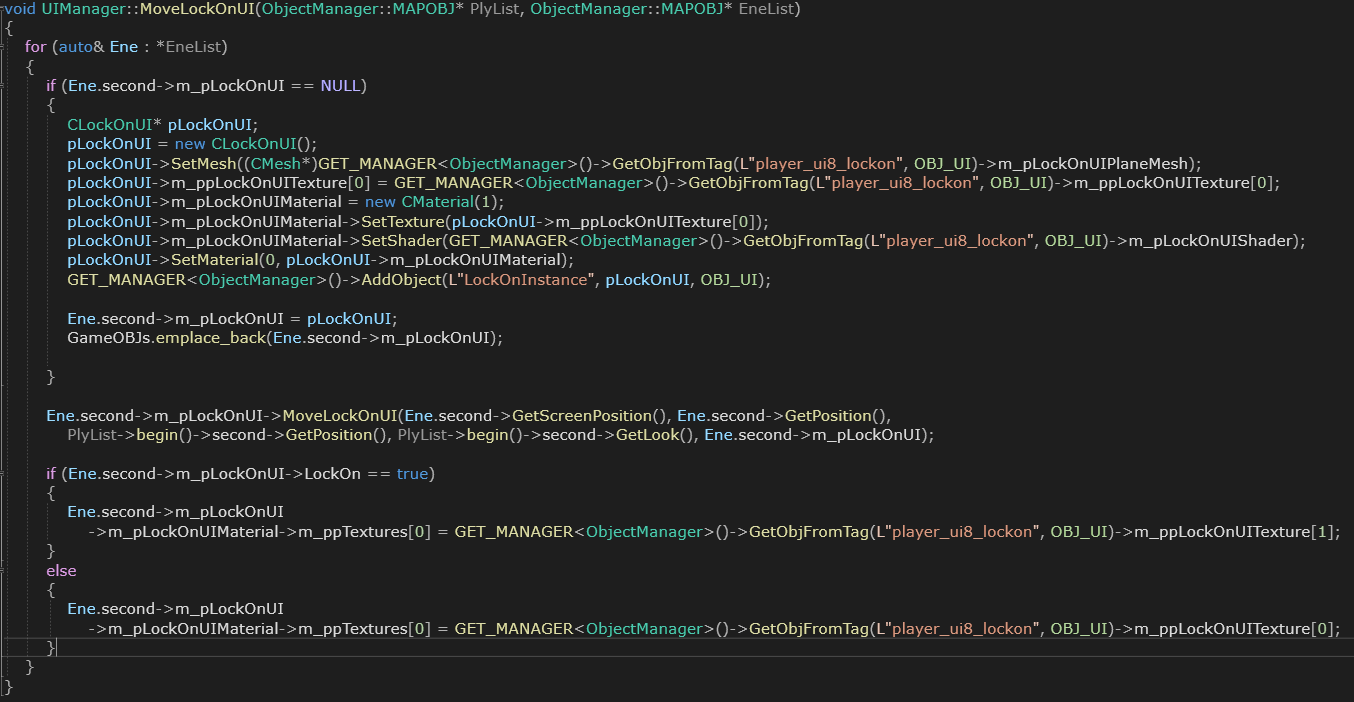
1. **UI 매니저 (락 온)**

앞으로 추가되는 적 오브젝트, 아군 오브젝트 다수의 조준UI를 화면에 표시하고 관리가 편하게 하기 위해서 CLockOnUI.cpp에서 작성된 MoveLockOnUI() 함수를 객체마다 호출해주는 작업이 필요했다.

**그림6. 오브젝트 매니저 클래스의 Update()에서 호출되는 UI 매니저 함수들**

****

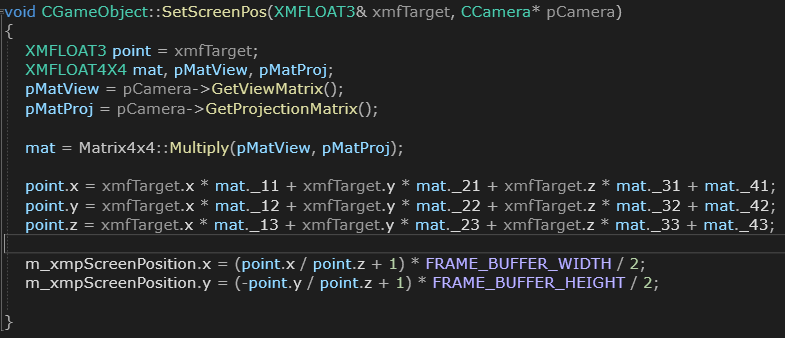
**그림7. UI매니저 클래스의 MoveLockUI()**

****

락 온UI의 동적 생성은 미니맵 UI에서 했던 것과 동일한 방식이고 락 온 UI가 조준 가능인지 불가능인지 알려주기 위해서 LockOnUI의 LockOn bool 값을 통해 텍스쳐를 교체했다.

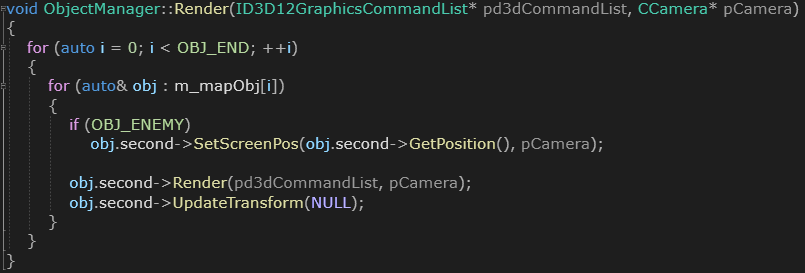
여기서 CLockOnUI 클래스의 MoveLockOnUI() 함수를 호출할 때 인자로 넘길 오브젝트들의 화면 좌표가 필요했다. 이를 위해 CGameObject 클래스에 오브젝트의 스크린 좌표를 저장할 변수를 추가하고 오브젝트의 화면 좌표를 계산해줄 SetScreenPos() 함수와 리턴 값으로 저장된 화면 좌표를 반환하는 GetScreenPosition() 함수를 추가했다.

**그림8. CGameObject 클래스의 SetScreenPos()**

****

CGameObject 클래스의 SetScreenPos() 함수는 오브젝트 매니저 클래스의 Render() 함수에서 호출되게 했다.

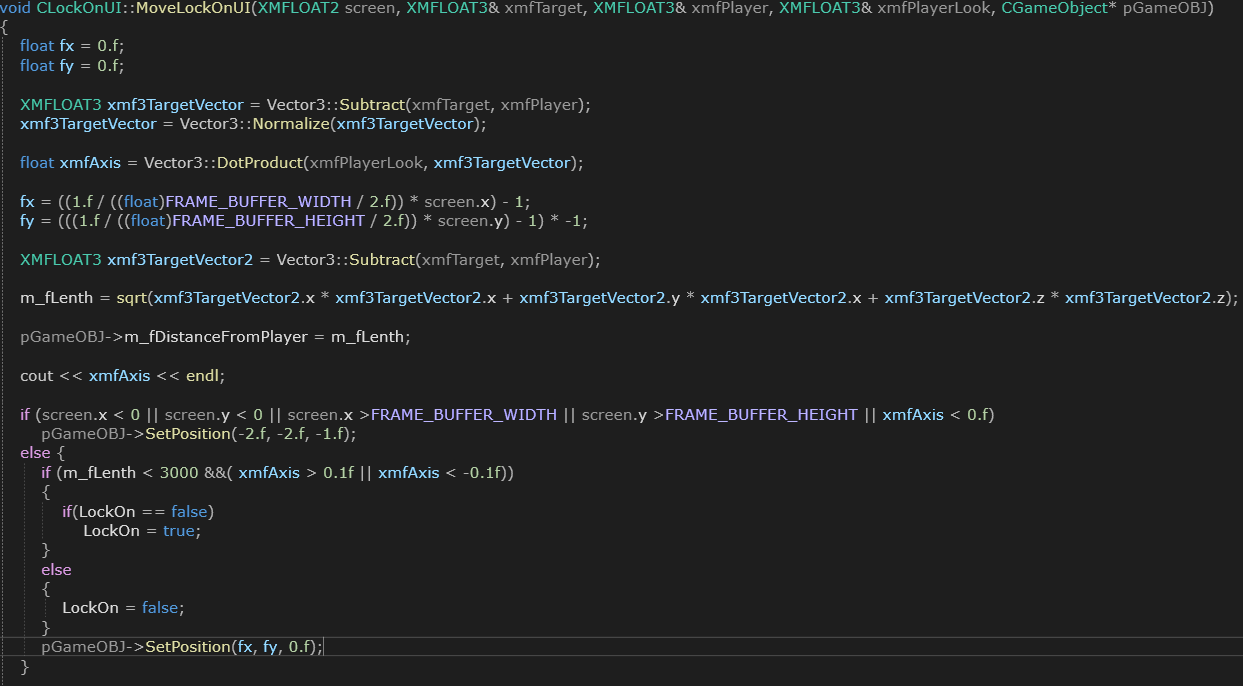
**그림9. 오브젝트 매니저 클래스의 Render()**

****

오브젝트 매니저 클래스의 unordered\_map 자료구조에 저장된 오브젝트들을 Render 하기 위해서

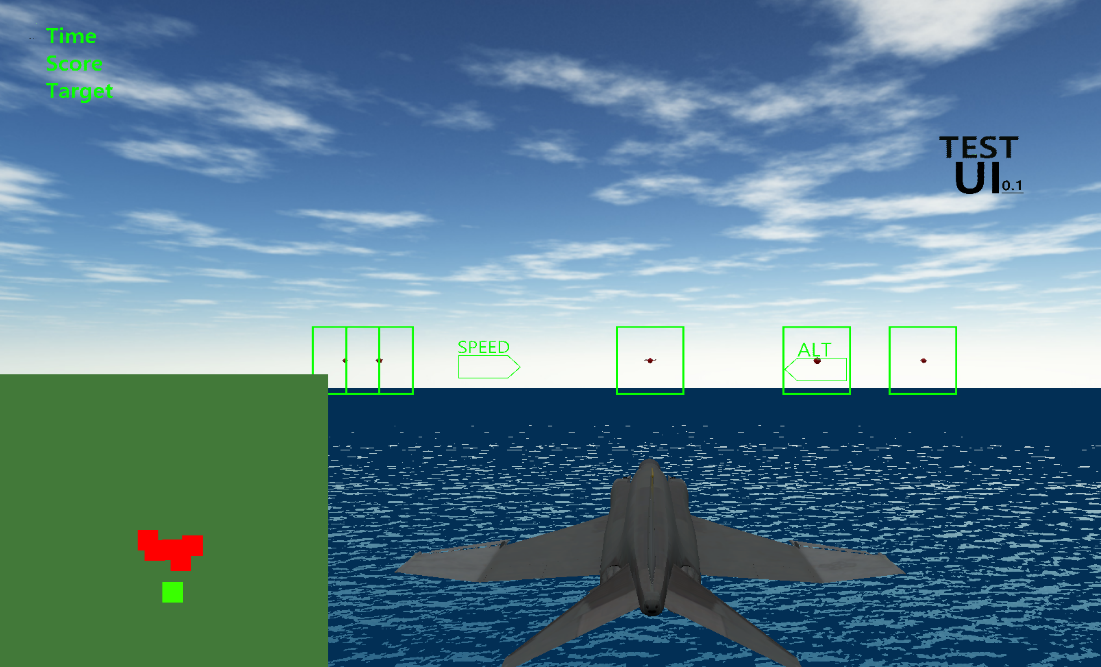
순회할 때 OBJ\_ENEMY일 경우 오브젝트가 자신의 화면 좌표를 저장하도록 했다.

**그림10. CLockOnUI 클래스의 MoveLockOnUI()**

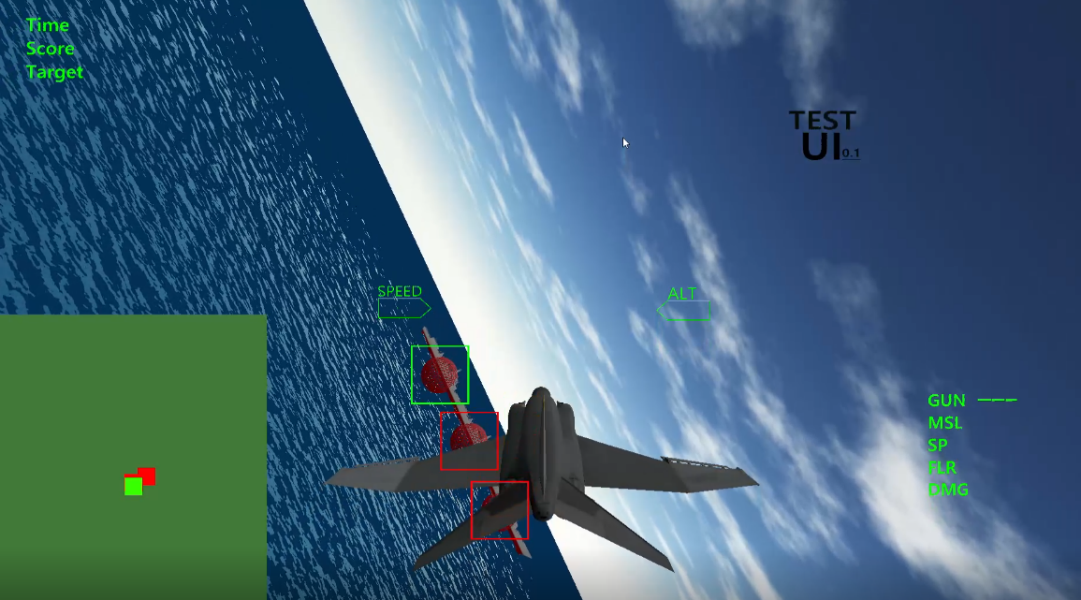


15주차에 작성했던 내용에서 플레이어와 적 오브젝트의 거리를 멤버 변수로 저장하고 락 온 UI가 표시되는 영역(플레이어와 적 오브젝트 간의 거리, 각도)을 설정하는 내용을 추가했다.

**그림11. 실행화면**

****

**그림12. [문제점] 플레이어가 조준하고 싶은 타겟을 지정할 수 없다.**

****

**[해결방안] 조준 가능한 객체들을 플레이어와의 거리에 따라서 순차적으로 관리할 수 있는 자료구조를 만들고 단축키로 타겟 변경이 가능하도록 구현한다.**

|  |  |  |  |
| --- | --- | --- | --- |
| **문제점 정리** | 행동트리 툴 제작의 진행이 더뎌 지고 있는 와중에 다른 잡다한 작업들이 많이 남아있어서 촉박함.  플레이어가 원하는 타겟을 지정하여 조준할 수 없음 | | |
| **해결방안** | 하루에 잡다한 작업 진행을 하나씩 끝내고 툴제작에 시간을 더 투자하는  방향으로 진행해야 함.  락 온 거리 내에 있는 모든 오브젝트들 중 타겟에게만 반응하도록 구현,  락 온 거리 내에 있는 모든 오브젝트들 중 타겟 변경이 가능하도록 구현 | | |
| **다음주차** | 17 주차 | **다음기간** | 2020.04.20 ~ 2020.04.26 |
| **다음주 할일** | 이재원 : Behavior 툴 제작, 기타 자잘한 작업 해결  김령운 : 단축키로 락 온 타겟 변경, 락 온 상태에서 미사일 유도  , 타겟 애니메이션, Behavior 툴 제작 | | |
| **지도 교수**  **Comment** |  | | |