MATLAB 프로그래밍 보고서

- 텀프로젝트 : 미사일 격추 시뮬레이션-

세종대학교

학 과
전자정보통신공학과
기 간
05.18 – 06.08
참 여 인 원
송용석 이재훈 정유종 조재형 최지훈

목 차

1장 프로젝트 계획

1.1 개요

MATLAB 프로그램을 통하여 x와 z축에 있는 미사일을 전투기에 격추 시키는 시뮬레이션을 진행하여 3D VR 애니메이션으로 구현하려고 한다.

1.2 임무분담

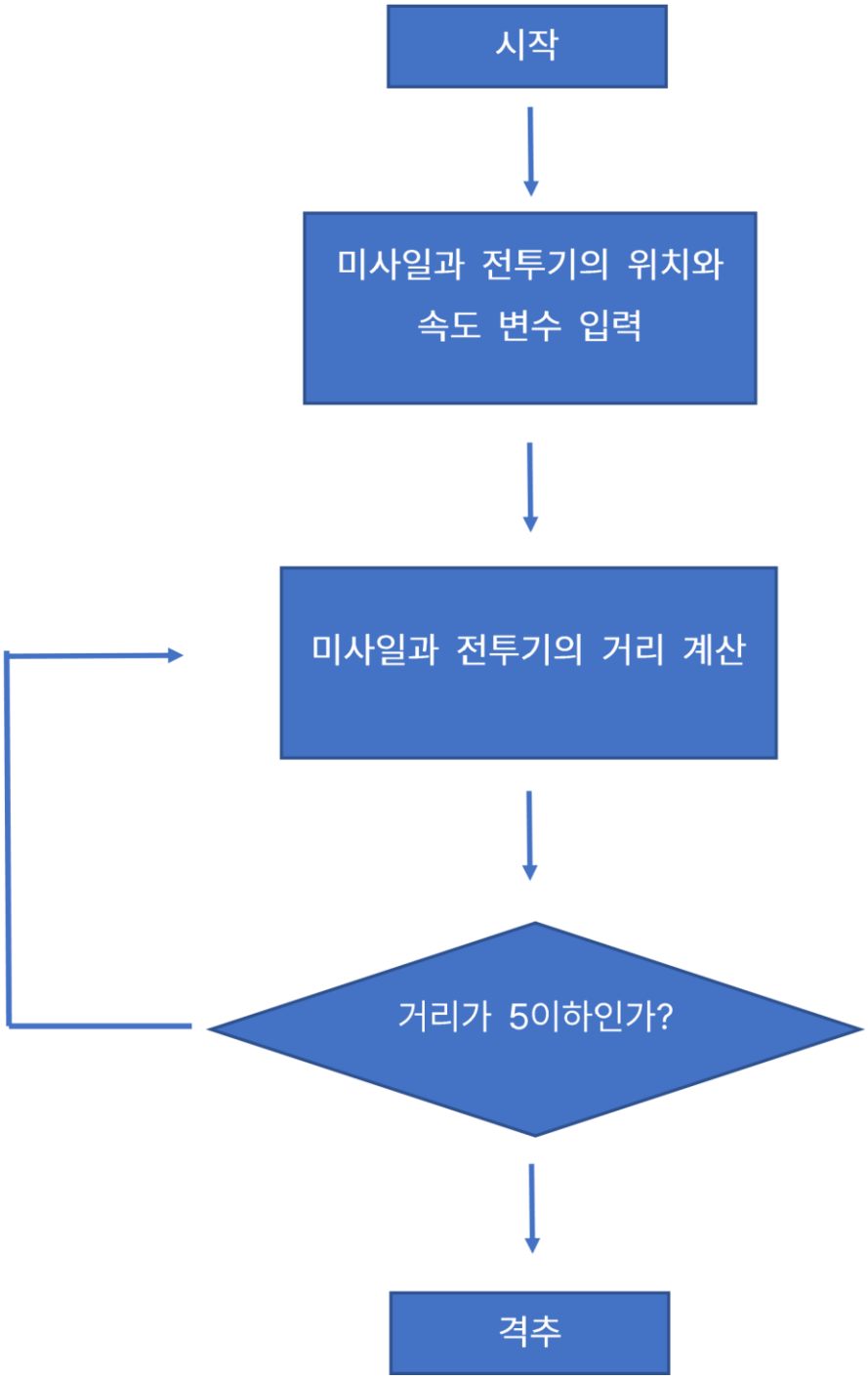
이름	임무
조재형	디자인 및 Simulink 담당
최지훈	디자인 및 Simulink 담당
정유종	스크립트 코딩 및 Simulink 담당
송용석	PPT 작성 및 발표
이재훈	PPT 작성 및 발표

1.3 주차별 진행도

기간	진행도
05.18 - 05.22	개인 시뮬링크 및 VR 사용법 숙달
05.23 – 05.28	프로젝트 분석 및 스크립트 파일 코딩
05.29 – 06.05	시뮬링크 및 VR 애니메이션
06.06	문제점 개선 및 분석
06.07-06.08	프로젝트 발표 준비

2장 알고리즘 분석

2.1 순서도



2.2 분석

시뮬링크로 실행하기전 매트랩 스크립트 파일로 확인 후 옮겨서 확인. 시뮬링크로 바로 실행할 경우 시간이 너무 오래 걸릴 수 있음.

미사일과 전투기의 위치와 속도를 입력변수로 받는다. 그 후 두 위치값을 이용해 전투기와 미사일 사이의 초기거리를 계산한다. While 구문을 이용해 위치벡터와 미사일과 전투기 사이의 거리의 반복되는 계산을 처리한다. 만약 거리가 5이하가 나온다면 미사일은 비행기에 격추 되었다고 판단, While문을 종료하고 프로그램은 종료된다.

3장 프로그램 작성

3.1 스크립트 파일

먼저 스크립트 파일로 프로토타입을 만들어 실험을 해본다.

```
%입력부
fighter_x = input('fighter_x : ');
height_fighter = input('height_fighter : ');
Speed_fighter = input('Speed_fighter : ');
location_missile = input('location_missile : ');
missile_z = input('missile_z : ');
Speed_missile = input('Speed_missile : ');

%distance = 전투기와 미사일 사이의 초기 거리 계산
distance = sqrt((fighter_x - location_missile)^2 + (height_fighter - missile_z)^2);

time = 0;   %시간 초기화
```

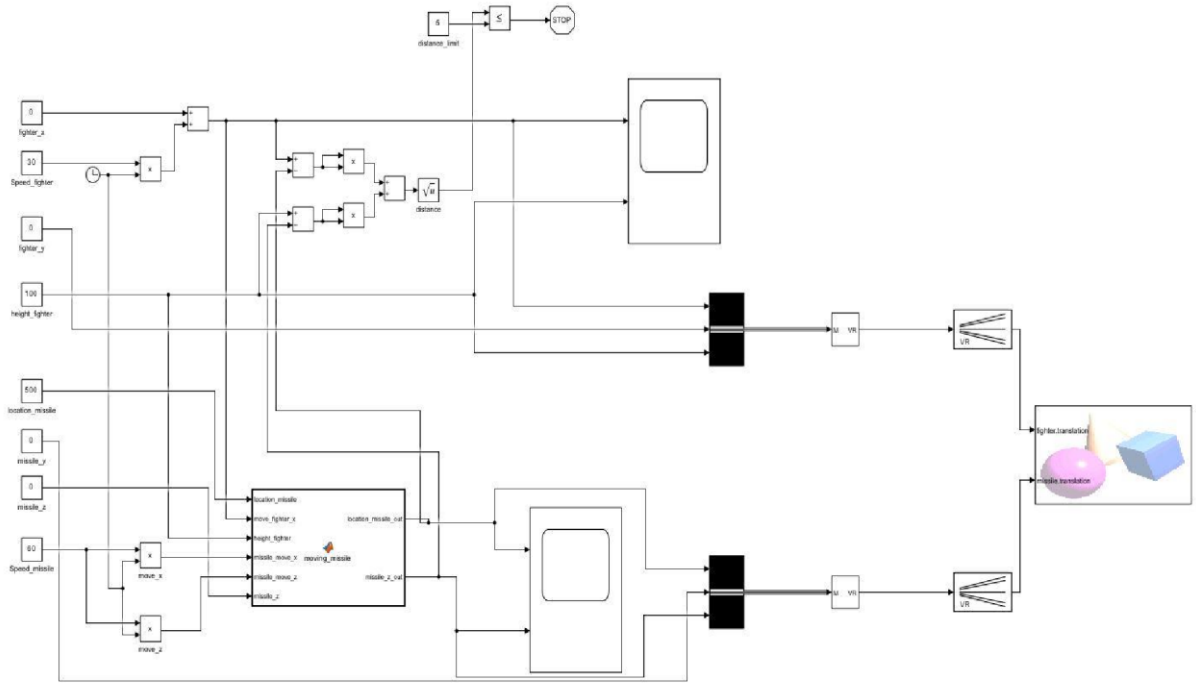
입력부에서는 전투기와 미사일의 초기 위치와 속도를 받아 저장한다. 점과 점사이의 거리를 이용하여 전투기와 미사일 사이의 거리를 계산하고 시간은 0초로 초기화 시킨다.

```
%계산부
while true
    vec_x = fighter_x - location_missile;
    vec_z = height_fighter - missile_z;

    vec_x = vec_x / distance;
    vec_z = vec_z / distance;

    location_missile = location_missile + (Speed_missile * time * vec_x);
    missile_z = missile_z + (Speed_missile * time * vec_z);

    fighter_x = fighter_x + (Speed_fighter * time);

    distance = sqrt((fighter_x - location_missile)^2 + (height_fighter - missile_z)^2);

    figure(1)
    plot(fighter_x, height_fighter, 'ko', location_missile, missile_z, 'ro')
    grid on, xlabel('x-axis'), ylabel('z-axis')
    hold on

    if distance <= 5
        break
    end

    time = time + 0.001;
end
```

계산부 에서는 While 루프를 이용해 반복적으로 위치벡터를 계산하여 미사일의 위치를 그래프 상으로 확인을 한다. 계산된 위치벡터 vex_x와 vec_z를 distance로 나눠 정규화를 진행시켜준다. 그리고 미사일과 비행기의 위치를 업데이트 시켜주고 새로운 두 물체 사이의 거리 distance를 계산한다. 물체 사이의 거리가 5이상일 경우 미사일이 위치벡터에 맞게 지속적으로 이동이 일어난다. 그 후 만약 distance의 값이 5이하가 된다면 전투기는 격추가 되며 While 루프는 종료가 된다.

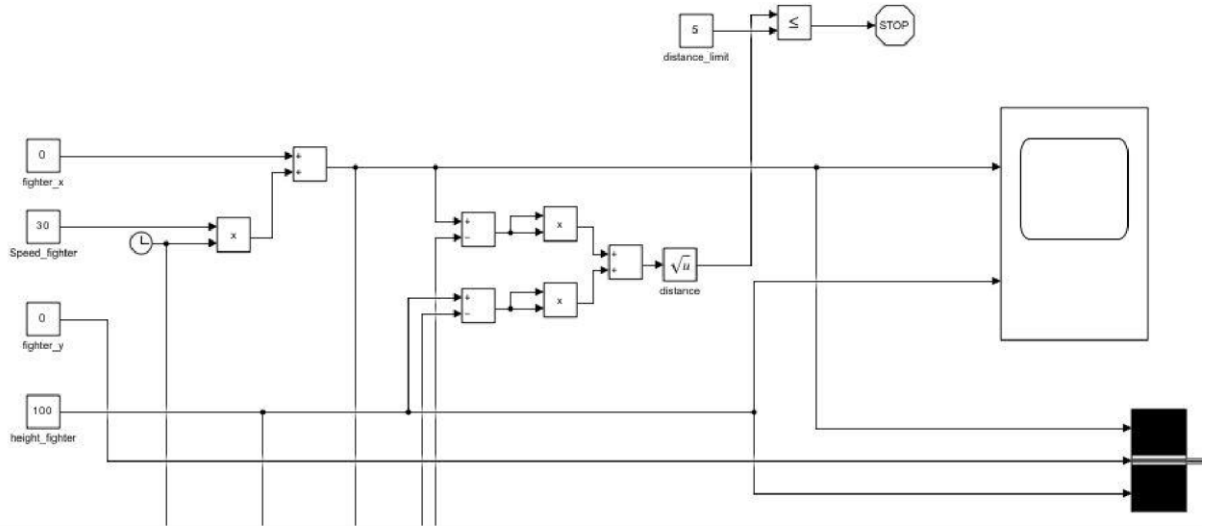
3.2 시뮬링크

시뮬링크 프로그램을 통해 위에서 코딩했던 프로그램을 시뮬레이션 시켜본다.



최종 시뮬링크 도면이다. 도면 윗 부분은 전투기 변수들과 구문 아랫부분은 미사일 변수들과 구문으로 구성 되어있다.

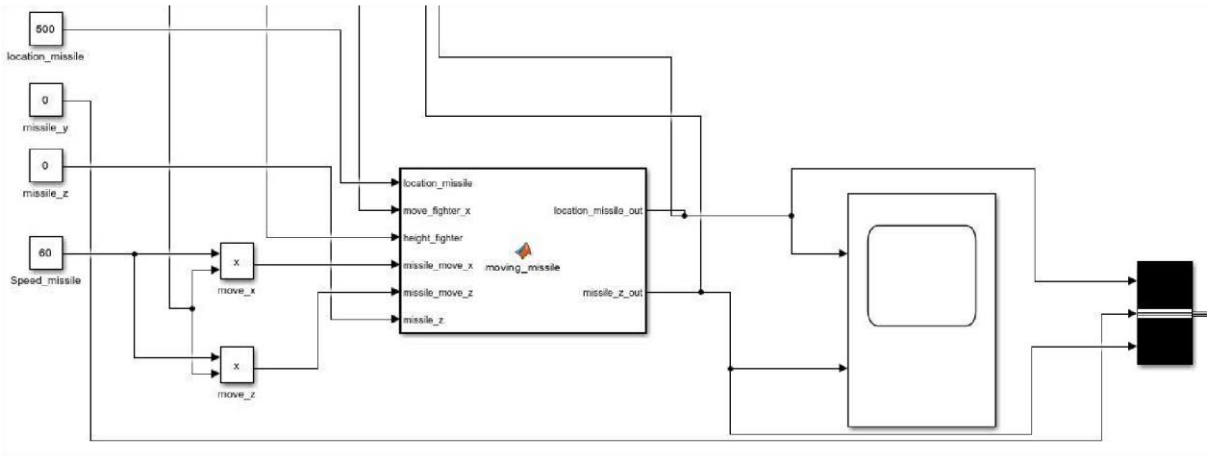
*전투기



시뮬링크의 전투기 부분이다. 전투기는 정속도 비행을 하도록 설계했다. 그 후 아랫부분 미사일 변수들과 함계 점사이 거리를 이용해 distance값을 만들어 5이하가되면 시뮬레이션이

종료되게 설계하였다.

*미사일



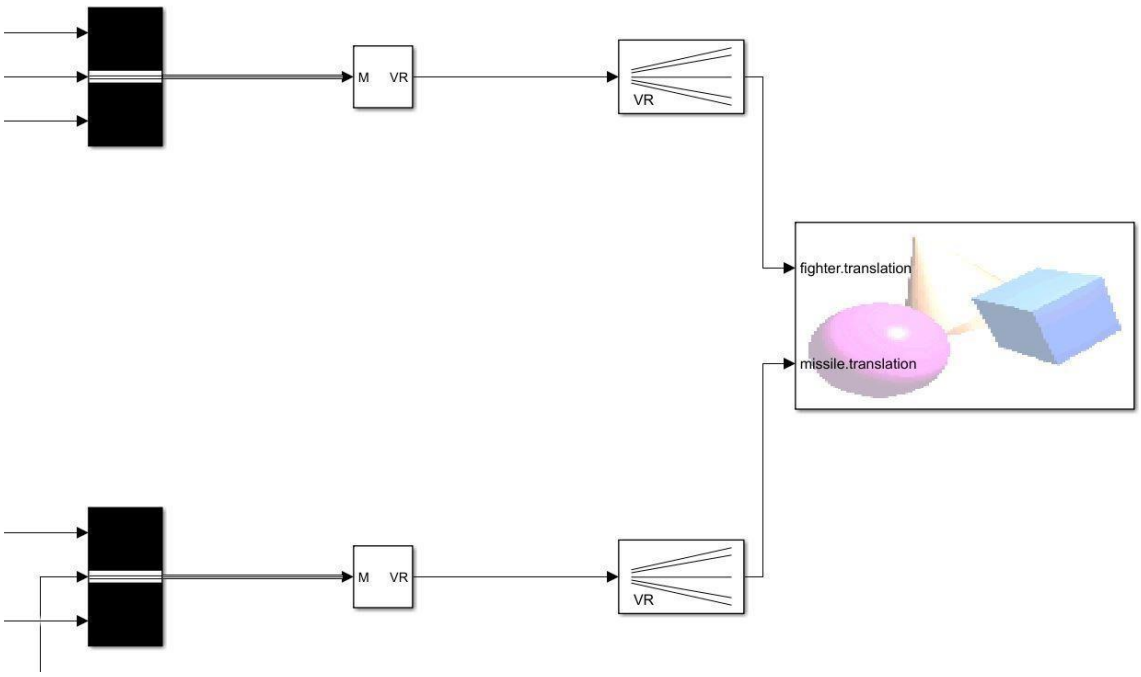
미사일 부분 도면이다. 시간과 속도를 곱하는 블록을 만들어 미사일의 움직임을 위한 move_x, move_z 추가로 생성하고 function 박스로 대입한다.

```
function [location_missile_out, missile_z_out] = moving_missile(location_missile, move_fighter_x, height_fighter, missile_move_x, missile_move_z, missile_z)
    distance = sqrt((move_fighter_x-location_missile)^2+(height_fighter-missile_z)^2);
    location_missile = location_missile + missile_move_x * ((move_fighter_x-location_missile)/distance);
    missile_z = missile_z + missile_move_z * ((height_fighter-missile_z)/distance);
    location_missile_out = location_missile;
    missile_z_out = missile_z;
end
```

미사일의 움직임을 위한 function박스의 while구문이다. 미사일의 위치는 기존 위치 x축과 y축 위치를 두 물체를 이어주는 벡터를 곱한 만큼 움직인다. 그리고 output 식을 만들어 계속해서 움직임이 업데이트가 되도록 한다. 그 결과로 미사일은 전투기를 추적하는 움직임을 예상 할 수있다.

3.3 3D VR

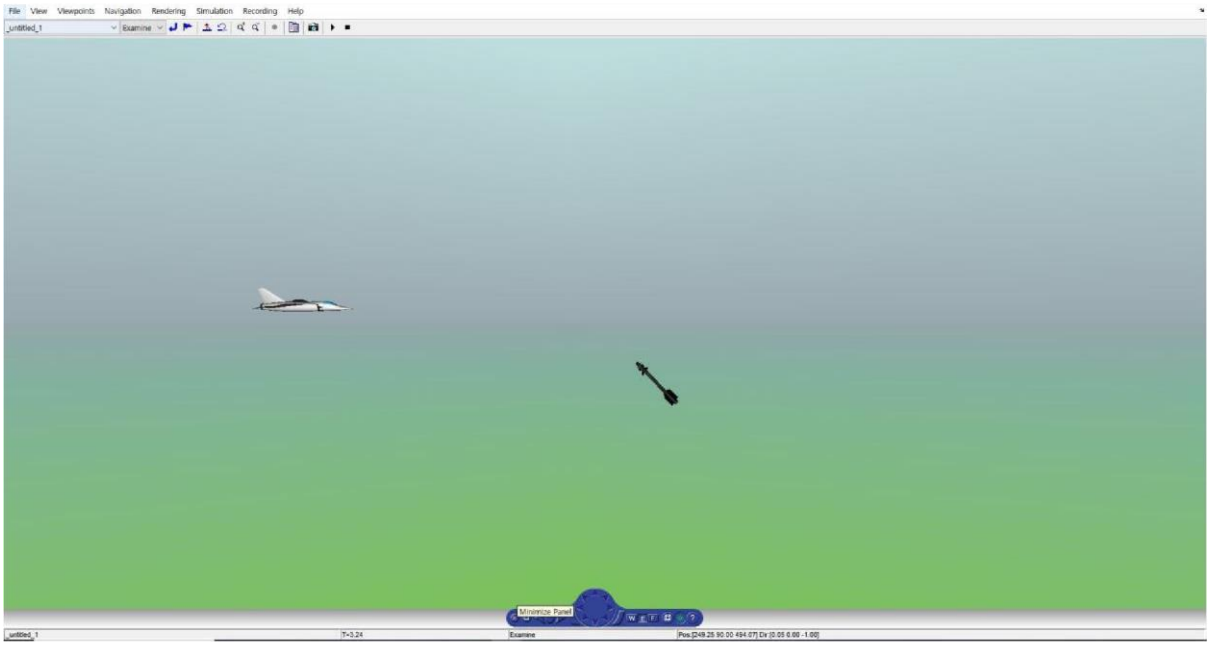
시뮬링크를 눈으로 확이하기 위해 3D 애니메이션을 만들어본다.



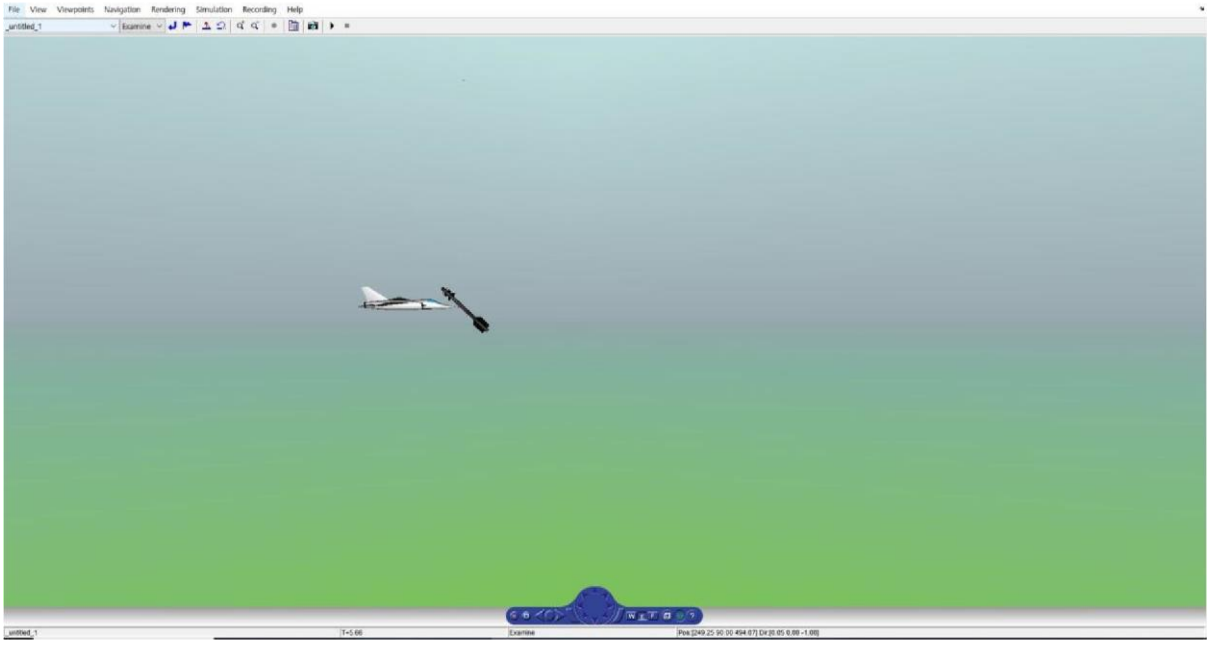
시뮬링크에 전투기와 미사일의 움직임을 표현한 3D애니메이션 장치를 설계했다.

시뮬링크가 시작되면 미사일은 오른쪽 아래 위치하고 전투기는 오른쪽 위에서부터 정속도로

움직인다.



그리고 미사일은 전투기와 미사일을 이어주는 벡터의 위치로 계속해서 이동한다.



거리가 5이내로 들어오게 되면 시뮬레이션은 종료한다. 프로젝트에서 예상한 결과를 시뮬링크 애니메이션으로 확인이 되었다.

4장 결론

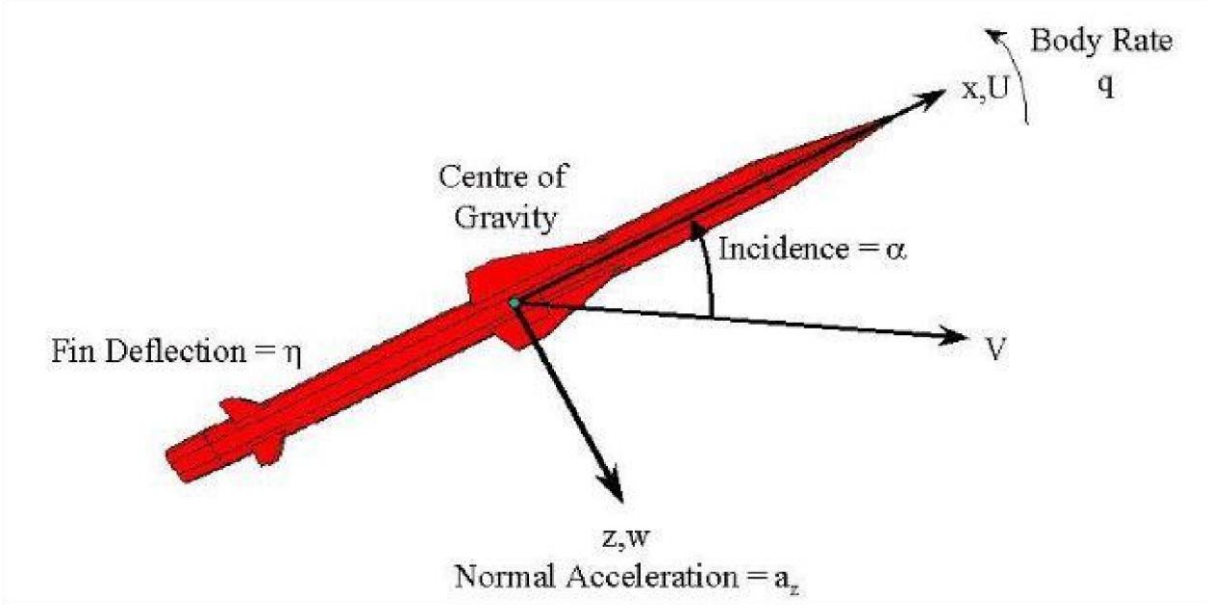
4.1 문제점

프로젝트의 핵심인 미사일의 움직임을 여러가지 함수로 표현하여 전투기를 격추시키는 시뮬레이션을 성공시켰다. 그러나 이 프로젝트를 마치고 문제점이 몇가지 있다고 생각했다. 이번 MATLAB 수업에서하는 프로젝트는 현업에서 하는 어려운 프로젝트는 아니였지만, 처음 접해보는 시뮬링크를 구현하려고 하다보니 시간이 많이 부족해 스크립트 파일에서 코딩을 하고 시뮬링크로 구현하는 부분이 많이 어려움이 있었다. 그로 인해 스크립트 파일과 시뮬링크가 부드럽게 연걸 되지 못했고 시간이 더 오래걸렸다.

시뮬링크에서 3d 애니메이션 또한 숙련도가 많이 떨어져 다양한 각도(전투기 시점, 미사일 시점, 위 아래 등)로 구현하지 못했다. 또한 전투기와 미사일의 움직임이 2차원에서 구현했기 때문에 움직임이 많이 자연스럽지 않았다.

4.2 개선방안

시뮬링크의 사용 숙련도를 더 올리고 스크립트 파일에서 코드를 생성할 때 시뮬링크에서 어떻게 사용을 할지 생각하며 프로그래밍을 한다면 시간이 더 단축될 수 있을 것 같다.



[그림 1. 미사일 각도]

Mathworks example. <Design a Guidance System>

그리고 3D 시뮬레이션을 2차원이 아닌 3차원까지 확장을 하고 여러 시점에서 볼 수있게 만들어 더 정확하게 시뮬레이션 하는 목표를 가지는게 좋을 것 같다. 추가적으로 3D 애니메이션으로 만든다면, 더 심화시켜 유도미사일 물체 자체에 대한 각도를 더 세분화 하여 만든다면 [그림 1], 더 정확한 시뮬레이션으로 개선이 가능할 할 것 같다.