Journal of the KIMST, Vol. 21, No. 6, pp. 842-849, 2018
ISSN 1598-9127
DOI http://dx.doi.org/10.9766/KIMST.2018.21.6.842

Research Paper 정보 · 통신 부문

이동체 모의시험을 위한 경로 생성 및 추종 시뮬레이터 개발

한영민*,1) · 홍동호1) · 장택수1)

1) LIG넥스원㈜ 유도무기2연구소

Development of Path Generation and Following Simulator for a Simulation Test of a Moving Object

Youngmin Han*,1) · Dongho Hong1) · Taeksoo Jang1)

1) *Launcher Control System for Rockets and Missiles, LIG Nex1, Korea*

(Received 23 May 2018 / Revised 13 September 2018 / Accepted 2 November 2018)

ABSTRACT

This research suggests the development of simulator for a Simulation Test of a moving object's path generation and following. There are many kinds of moving objects in weapon systems, such as vehicles, missiles, robots and so on. So need tests of moving simulations during development process of weapon systems. To simulate a moving object, need an flexible path. So this report suggests a Bézier curve algorithm for generation of smooth curve path. And when new developments of weapon systems are started, many kinds of simulators are created. But, these simulators are not reused in other project because there are different kinds of development environment. So need to allow users to add specific features, And this report suggests using Dynamic Link Library(DLL)

Key Words : Simulator(시뮬레이터), Moving Object(이동체), Path Generation(경로생성), Bézier Curve(베지어 곡선), Explicit Link(명시적 연결)

1. 서 론

무기체계에는 다양한 종류의 이동체가 존재한다. 유도탄, 어뢰, 로켓 등 다양한 종류의 타격 무기가 이동하는 물체를 표적으로 삼고 있으며, 그 자체로도 이동체이다. 또한 자동차, 비행기, 배와 같은 전통적인 이동 수단뿐만 아니라 최근에는 드론과 같은 새로운 플랫폼도 등장하고 있으며, 다양한 종류의 자율주행 로봇도 무기체계 분야에 적용되고 있는 추세이다.

이처럼 다양한 종류의 이동체는 연구, 개발의 대상이기도 하며 때로는 무기체계의 한 축을 구성하는 중요 요소이기도 하다. 따라서 개발 시험 중에 이동체의 이용은 필수라고 할 수 있다. 하지만 시간, 비용, 장소 등의 제한으로 인하여 실제 이동체를 이용한 시험은 제한적이다. 예를 들어, 대공 레이더 개발 시 대공 표적에 대한 추적 시험을 위하여 항상 비행기를 띄우기는 어려운 일이다. 또한 함대지 유도탄을 개발한다고

* Corresponding author, E-mail: youngmin.han@lignex1.com

가정하면 함정의 이동에 따른 유도탄 영향성 검토 시험을 위하여 매번 함정을 바다에 띄우기는 어려운 실정이다.

이와 같은 제한 사항을 극복하기 위하여 프로젝트마다 개발 과정 초기에 다양한 이동체 시뮬레이터(Simulator)를 개발하여 시험에 사용하고 있다. 하지만 시뮬레이터가 해당 프로젝트의 주요 개발 대상이 아닌 경우가 많아 시뮬레이터 개발에 시간과 비용을 투자하기 어려운 상황이다. 이러한 이유로 위치정보 정도만 전송하거나 직선경로 이동만 가능한 단순한 형태의 시뮬레이터를 개발하여 사용하는 경우가 많다. 또한 한번 개발한 시뮬레이터를 타 프로젝트에서 재사용하려 해도 이동체의 물리적 특성, 통신 방식/규약 등이 다른 경우가 많아 재사용에 어려움을 겪고 있다.

본 논문에서 제안하는 시뮬레이터를 개발하게 된 계기도 개발에 참여한 여러 종류의 유도탄 개발 과정에서 기존의 위치정보 시뮬레이터를 사용하면서 발생한 불편함을 해결하기 위해서였다. 대부분의 개발/시험이 실제 유도탄이 발사될 플랫폼(Platform)에서가 아니라 실내 시험장의 고정된 장소에서 진행되기 때문에 유도탄에 플랫폼의 가상의 위치정보를 입력해주어야 한다. 하지만 기존의 무기체계 사업에서 사용하던 위치정보 시뮬레이터는 한 점(Point)의 위치 정보만 전송 가능하였다. 따라서 연속적으로 이동하는 플랫폼(배, 비행기, 차량 등)에서 유도탄을 발사하는 모의시험은 불가능하였다.

위의 예와 같은 문제를 해결하기 위하여 다음과 같은 목표를 가지고 시뮬레이터를 개발하였다.

첫째, 이동체의 이동 경로를 생성하고 생성한 경로를 연속적으로 이동하며 위치/자세 정보를 전송하는 시뮬레이터를 개발하였다. 여기서 이동 경로는 단순직선 경로만이 아니라 사용자가 자유롭게 변경 가능한 부드러운 곡선 경로 생성을 목표로 한다. 경로가 생성되면 이동체는 경로를 추종하여 이동하며 이동 중에는 이동체의 물리적 특성을 반영하여 Roll, Pitch 값을 지속적으로 모의한다.

둘째, 다양한 이동체의 물리적 특성과 통신 기능을 본 시뮬레이터의 소스 수정 없이 확장이 가능하여 다양한 프로젝트에 사용할 수 있는 유연한 구조의 시뮬레이터 개발하였다.

2. 설 계

2.1 경로생성

이동체의 이동을 모의하기 위해서는 우선적으로 경로 생성이 필요하다. 그리고 대부분의 이동체는 단순한 직선 이동보다는 완만한 곡선 형태로 이동 한다. 따라서 단순히 점과 점 사이를 연결한 직선 경로만으로 이동체의 움직임을 모의하기에는 부족함이 따른다. 또한 사용자가 원하는 형태로 손쉽게 경로를 수정할 수 있어야 한다.

본 논문에서는 곡선 경로 생성을 위하여 내부 제어점을 이용하여 곡선을 변경할 수 있는 베지어(Bézier) 곡선 알고리즘을 이용하였다. 최소 3개 이상의 제어점(Control Point)을 이용하여 사용자가 자유롭게 곡선 경로를 수정할 수 있도록 개발하였다.

베지어 곡선은 그래픽 분야에서 주로 사용되는 매개곡선(Parametric Curve)으로, Fig. 1과 같이 시작점과 끝점 그리고 그 사이에 위치하는 제어점 사이의 이동에 의해 곡선을 추출하는 알고리즘이다[1].

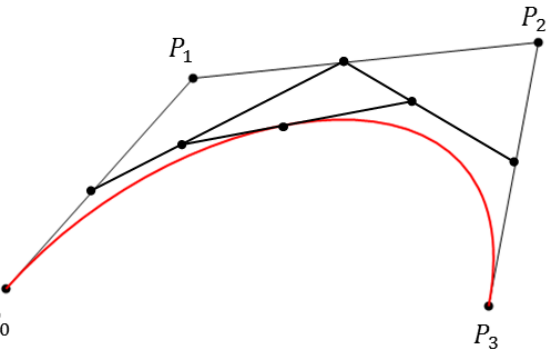


Fig. 1. Example of Bézier curve

$$B(u)=\sum_{k=0}^{N} P_k \frac{N!}{k!(N-k)!} u^k (1-u)^{N-k} \quad (1)$$

베지어 곡선의 기본 공식은 (1)과 같으며 N은 제어점의 개수, P_k는 제어점의 벡터이다[2].

제어점의 개수가 많을수록 복잡한 곡선 생성이 가능하지만 계산 양이 증가하게 되고 사용자가 많은 수의 제어점을 조절하기에도 불편함이 따른다. 따라서 제어점을 5개 이상 늘려도 실용적인 측면에서 특별한 이점이 없으므로 Fig. 2와 같이 시작점과 끝점을 포함

하여 4개의 제어점을 사용하는 3차 베지어 곡선을 일반적으로 사용한다.

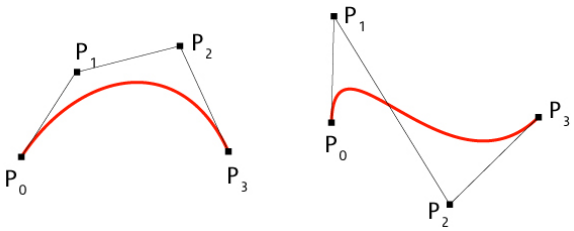


Fig. 2. Example of degree 3 Bézier curve

본 논문에서도 3차 베지어 곡선을 선택하였으며 기본 공식에서 유도한 3차 베지어 곡선 공식은 다음과 같다.

$$B(u) = P_0(1-u)^3 + 3P_1u(1-u)^2 + 3P_2u^2(1-u) + P_3u^3 \quad (2)$$

$$u = \frac{Resolution}{Distance} \quad (3)$$

u 값은 곡선을 이루는 점과 점 사이의 거리(Resolution)을 시작점과 제어점, 끝 점까지를 연결한 거리(*Distance*)를 나눈 값으로 *Resolution* 값이 작을수록 곡선의 정밀도가 높아진다. 본 논문에서는 *Resolution* 값을 10 m로 설정하였다.

시작점은 시뮬레이터 지도상의 현재위치이며 사용자가 원하는 위치로 이동가능하다. 사용자가 끝점을 생성하면 Fig. 3과 같이 시작점과 끝점을 3등분하여 2개의 중간 제어점이 자동 생성되며 시작점에서 끝점까지 직선 경로가 생성되도록 구현하였다.

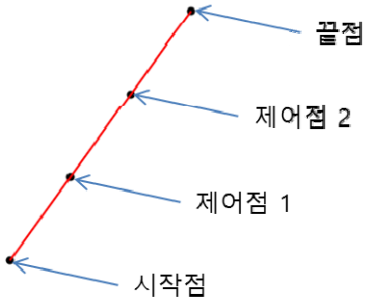


Fig. 3. Creation of end point and control points

Fig. 4와 같이 사용자는 시작점과 끝점, 2개의 제어점을 조절하여 곡선 경로를 생성, 조절할 수 있다.

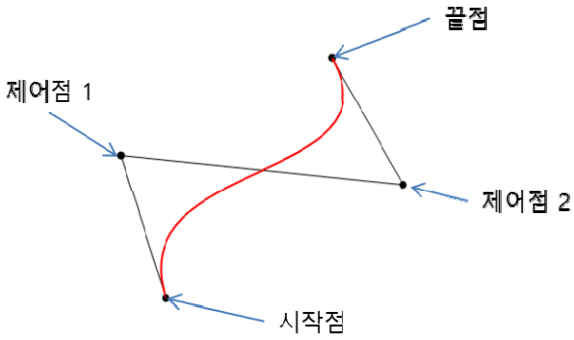


Fig. 4. Adjustment of control points

2.2 경로추종

서론에서 서술 한 바와 같이 이동체의 종류가 다양하기 때문에 경로추종을 위해서는 수행하는 프로젝트에 따라 이동체의 물리적 특성을 변경할 수 있어야 한다.

그러므로 시뮬레이터를 사용하기 위해서는 프로젝트의 환경에 따라 사용자의 수정이 불가피하다. 하지만 본 시뮬레이터의 소스 코드를 사용자들이 직접 수정한다면 소프트웨어 형상 관리에 어려움이 따르게 된다. 따라서 본 시뮬레이터의 수정 없이 기능을 추가할 수 있는 구조가 필요하다.

본 논문에서는 DLL(Dynamic Link Library)의 명시적(Explicit) 연결 기능을 이용하여 기능을 추가할 수 있는 구조의 설계를 제안한다[3].

DLL의 명시적 연결은 라이브러리를 로드하고 사용할 함수의 주소(포인터)를 직접 알아내서 호출하는 방식이다.

따라서 시뮬레이터 사용자가 DLL 파일을 직접 생성하여 미리 약속된 이름과 형식의 함수를 이용하는 방식으로 기능을 추가할 수 있다.

사용자는 DLL 프로젝트를 생성하여 Fig. 5와 같은 함수를 정의한다. 해당 함수에 시뮬레이터로부터 이동체의 최대 속도와 위/경도 좌표가 입력되므로 사용자는 이 정보를 이용하여 이동체의 자세 및 속도를 모의하는 코드를 작성하여 시뮬레이터 실행파일과 같은 폴더에 저장하면 된다.

최고속도는 이동체의 고유 특성이지만 시뮬레이터에서 해당함수로 입력해주는 이유는 시험 목적에 따

라 이동체의 실제 최고속도보다 느린 속도로 시험을 진행해야 하는 경우가 발생할 수 있어 추가하였다.

```
extern "C" __declspec(dllexport) void SimulateObjectMoving(
        double dMaxVelocity,
        double dLatitude, double dLongitude,
        double &dSimRoll, double &dSimPitch,
        double &dSimHeading,
        double &dSimVel
        );
```

Fig. 5. Function of SimulateObjectMoving()

시뮬레이터를 실행하면 같은 폴더에 있는 모든 DLL 파일을 검색한다. 검색한 파일에 Fig. 5와 같은 함수가 존재하면 해당 DLL을 저장하고 Fig. 6과 같이 사용자가 지정한 DLL에 해당 이동체의 위치/최대속도이 입력되고 출력된 모의정보(Roll, Pitch, Heading, 속도)를 이용하여 경로를 추종한다.

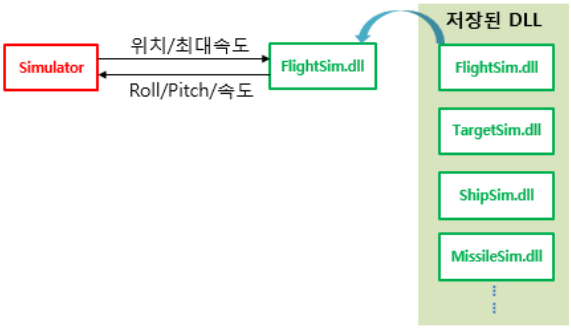


Fig. 6. Load simulation DLL

2.3 데이터 전송

시뮬레이터에서 모의한 이동체의 정보(위치, 자세, 속도 등) 처리 또한 프로젝트에 따라 달라진다. 모의정보를 이더넷 통신이나 시리얼 통신 등을 이용하여 전송할 수도 있고, 파일 저장이 필요할 수도 있다. 하지만 통신 기능과 파일 저장 기능을 시뮬레이터에 구현한다고 해도 통신 패킷 구조와 파일 구조를 모든 프로젝트에 적용할 수 있도록 개발할 수는 없다.

보안 등의 이유로 프로젝트 간의 정보 공유가 어렵고 공통된 프로토콜이 거의 없는 무기체계 개발 환경을 고려하면 위와 같은 다양한 변수를 모두 적용하여 시뮬레이터를 개발한다는 것은 불가능하다고 할 수 있다. 이 문제점을 해결하기 위하여 앞서 설명한 DLL의 명시적 연결 기능 이용을 동일하게 사용하였다.

시뮬레이터를 실행하면 같은 폴더에 있는 모든 DLL 파일을 검색한다. 검색한 DLL 파일에 Fig. 7과 같은 함수가 존재하면 해당 DLL을 저장하고 해당 DLL의 함수에 모의 정보가 입력된다. 사용자는 입력된 정보를 용도에 맞게 처리할 수 있도록 DLL의 함수를 작성하면 된다.

```
extern "C" __declspec(dllexport) void RecvSimInform(
        double dPosX, double dPosY, double dPosZ,
        double dRoll, double dPitch, double dYaw,
        double dVel
        );
```

Fig. 7. Function of RecvSimInform());

Fig. 8과 같이 DLL 파일들이 시뮬레이터에 저장되고 각각의 DLL에는 Fig. 7과 같은 함수가 명시되어 있다. 따라서 시뮬레이터가 실행되면 일정 주기로 각각 DLL의 Fig. 7 함수를 호출하게 되고 DLL에 구현된 데로 RS-422, TCP, UDP 등을 통하여 데이터를 전송하고 파일로 저장한다.

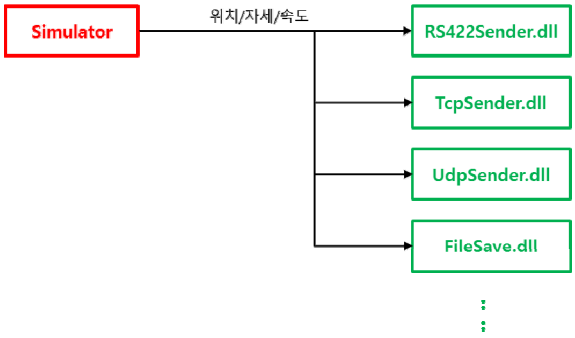


Fig. 8. Send data to DLL

3. 구 현

본 논문에서 소개하는 시뮬레이터는 사용자의 직관적인 조작을 위하여 Fig. 9와 같이 GUI(Graphic User Interface) 기반으로 개발하였다.

고해상도의 세계지도(Shape 기반)을 지원하며, 마우스를 이용하여 확대/축소/이동 기능을 지원한다.

화면 좌측에는 이동체의 위치/자세/속도 정보가 전시되며 자세정보(Roll, Pitch)를 그래픽 화하여 전시한다. 이동체 아이템은 마우스로 드래그(Drag)하여 이동할 수 있으며, 상세정보를 입력할 수 있는 창을 제공한다.

Fig. 9. GUI of simulator

3.1 경로생성

Fig. 10과 같이 이동체를 우 클릭하면 상세정보 창이 팝업(Pop-up)되고, 이동체의 위치를 지정할 수 있도록 구현하였다.

Fig. 10. Designation of item position

상세정보창의 생성 버튼을 누르면 경로생성 작업이 시작되고 마우스를 이동하여 목적지를 클릭하면 Fig. 11과 같은 직선경로와 중간 제어점 2개가 생성된다.

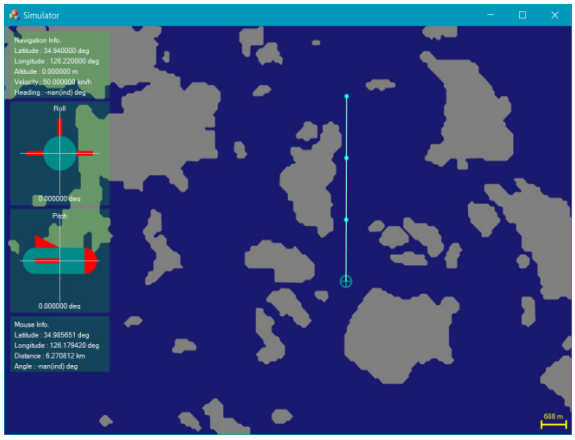


Fig. 11. Creation of a straight path

Fig. 12와 같이 제어점을 마우스로 드래그하여 경로의 형태를 사용자가 바꿀 수 있다.

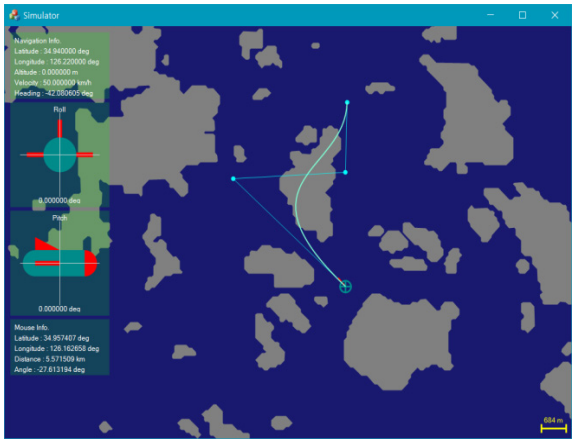


Fig. 12. Adjustment of control points

3.2 경로추종

생성된 경로를 확대하면 Fig. 13과 같다. 수식 (3)에서 u값의 *Resolution*을 10 m로 설정하였으므로 Fig. 13의 빨간 점 사이는 10 m 간격으로 나누어진다.

경로추종을 시작하면 이동체는 입력된 속도에 맞춰 빨간 점을 따라 이동하며, 변화된 이동체의 위치 값은 사용자가 선택한 DLL에 해당 함수를 호출하여 입력된다.(DLL이 없을 경우, Roll, Pitch 모의 없이 경로를 따라 Heading만 변화시키며 단순 이동 한다.)

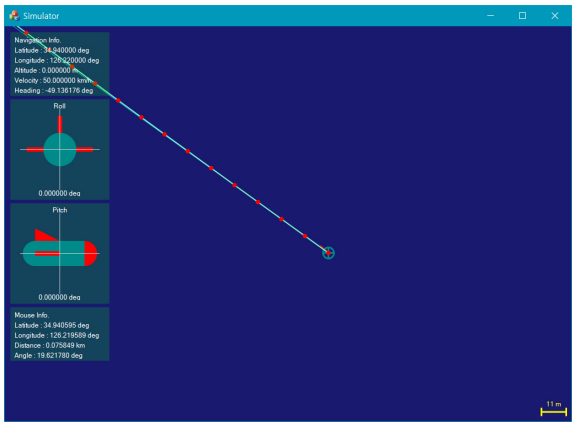


Fig. 13. Magnification of the path

이동체의 물리적 특성을 반영하여 경로를 추종할 수 있도록, DLL 파일을 생성하고 Fig. 5 함수에 Roll, Pitch를 모의할 수 있도록 수식 (4), 수식 (5)을 적용하여 코드를 작성하였다.

본 논문에서는 이동체를 비행체로 가정하고 오일러각(Euler Angle)을 이용하여 Roll(φ)과 Pitch(θ)를 모의하였다.[4] 비행 중의 고도 변화는 생략하여 Z축에 대한 가속도(A_Z)은 0으로 가정하였다.

$$\varphi = \mathrm{atan}\left(\frac{A_Y}{\sqrt{A_X^2 + A_Z^2}}\right) \tag{4}$$

$$\theta = \mathrm{atan}\left(\frac{A_X}{\sqrt{A_Y^2 + A_Z^2}}\right) \tag{5}$$

X, Y축에 대한 가속도(A_X, A_Y)는 비행체의 현재 위치와 이전 위치를 이용하여 계산하였다. Heading 또한 같은 방법으로 현재위치와 이전 위치를 이용하여 계산하였다.

생성한 DLL파일을 시뮬레이터와 같은 폴더에 복사한 후, 시뮬레이터의 환경설정 창을 열면 Fig. 14와 같이 로드한 DLL 파일 정보를 확인할 수 있다.

3.3 모의데이터 처리

모의 데이터 처리 시험을 하기 위하여 TCP와 UDP 통신을 이용하여 서버로 데이터를 전송하는 DLL (TcpSender.dll, UdpSender.dll)과 데이터를 파일로 저장하는 DLL(FileSave.dll)을 생성하였다.

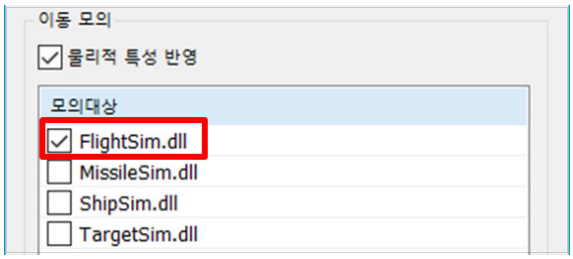


Fig. 14. Load FilghtSim.dll

각각의 DLL에는 Fig. 7과 동일한 형태의 함수가 작성되어 있으며 각각의 함수에 이동체의 모의정보가 입력되어 TCP, UDP 소켓으로 데이터가 전송되고, CSV파일로 저장하는 코드를 작성하였다.

생성한 DLL파일을 시뮬레이터와 같은 폴더에 복사한 후, 시뮬레이터의 환경설정 창을 열면 Fig. 15와 같이 로드한 DLL 파일 정보를 확인할 수 있다. 그리고 여기서 설정한 주기에 맞춰 이동체의 물리적 특성을 모의하고 이와 동시에 데이터를 전송/저장하게 된다.

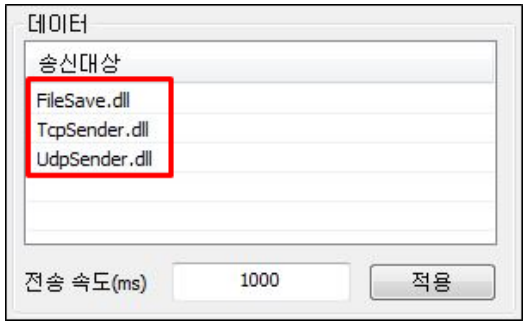


Fig. 15. Load data DLL files

4. 시 험

이동체를 임의의 위치(위도 : 34.94도, 경도 : 126.22도)로 이동시켜 섬을 회피하는 경로를 생성하고 생성된 경로를 50 km/h의 속도로 추종하는 시험을 실시하였다.

4.1 경로생성 결과

Fig. 16과 같이 제어점을 조절하여 정상적인 경로가 생성됨을 확인하였다.

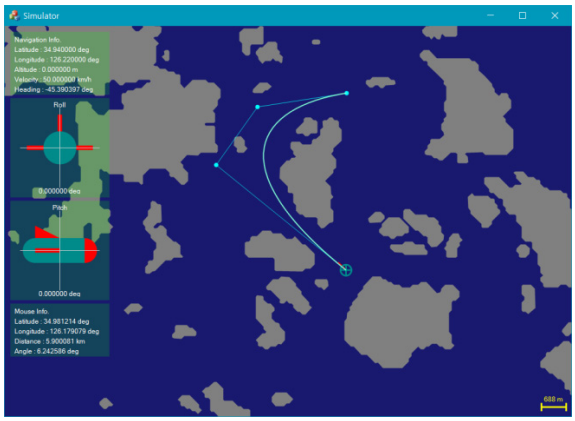


Fig. 16. Completion of the path

4.2 경로추종 결과

경로추종을 시작하면 1000 ms 주기로 생성된 경로를 따라 이동을 하며 현재 위치(위/경도)와 헤딩 값을 기록한다.

또한 이동체의 물리적 특성을 모의하는 DLL(본 시험에서는 FlightSim.dll 이용)을 정상적으로 호출하여 Roll, Pitch 값이 모의됨을 확인하였다.

이동체의 경로추종 결과는 Table 1과 같다.

4.3 데이터 전송 결과

TcpSender.dll과 UdpSender.dll 파일을 로드하여 모의 데이터가 Fig. 17, Fig. 18과 같이 TCP, UDP 서버로 정상적으로 전송되고 Table 1 경로추종 결과와 동일함을 확인하였다.

또한 FileSave.dll 파일을 로드하여 Fig. 19와 같이 모의 데이터가 CSV파일로 정상적으로 저장되고 그 값이 Table 1 경로추종 결과와 동일함을 확인하였다.

Table 1. Result of path following

번호	위도	경도	Heading	Roll	Pitch
1	126.220000	34.940000	-46.356	0.000	0.000
2	126.219881	34.940093	-46.356	0.000	10.000
3	126.219762	34.940186	-46.304	0.172	9.885
4	126.219644	34.940279	-46.253	0.256	9.724
5	126.219526	34.940371	-46.200	0.312	9.537
6	126.219409	34.940464	-46.148	0.346	9.207
7	126.219291	34.940557	-46.096	0.371	8.791
8	126.219174	34.940649	-46.043	0.391	8.359
9	126.219057	34.940742	-45.990	0.406	7.931
10	126.218941	34.940834	-45.937	0.419	7.515
11	126.218825	34.940926	-45.886	0.427	7.117
12	126.218709	34.941019	-45.834	0.435	6.736
13	126.218593	34.941111	-45.779	0.444	6.370
14	126.218478	34.941203	-45.726	0.450	6.020
15	126.218363	34.941295	-45.674	0.455	5.691
16	126.218248	34.941387	-45.620	0.460	5.373
17	126.218134	34.941479	-45.565	0.465	5.067
18	126.218020	34.941571	-45.513	0.468	4.782
19	126.217906	34.941663	-45.457	0.473	4.509
20	126.217793	34.941754	-45.404	0.476	4.248
21	126.217679	34.941846	-45.350	0.503	3.937
22	126.217567	34.941938	-45.296	0.530	3.574
23	126.217454	34.942029	-45.242	0.531	3.230
:	:	:	:	:	:

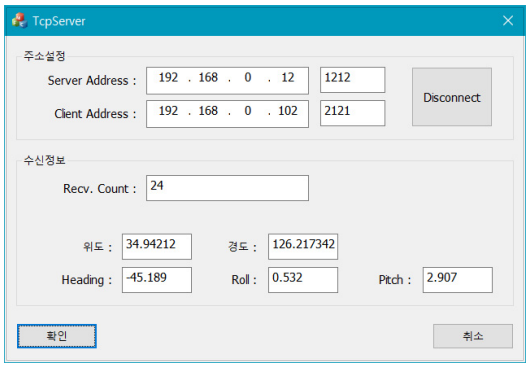


Fig. 17. TCP server

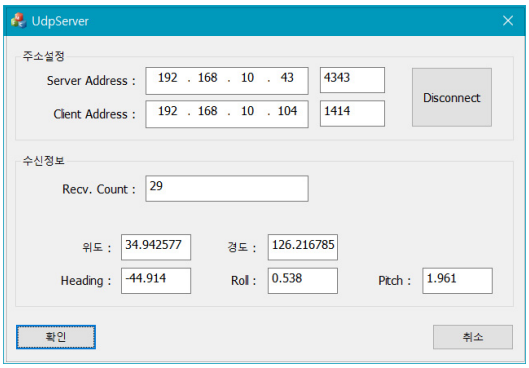
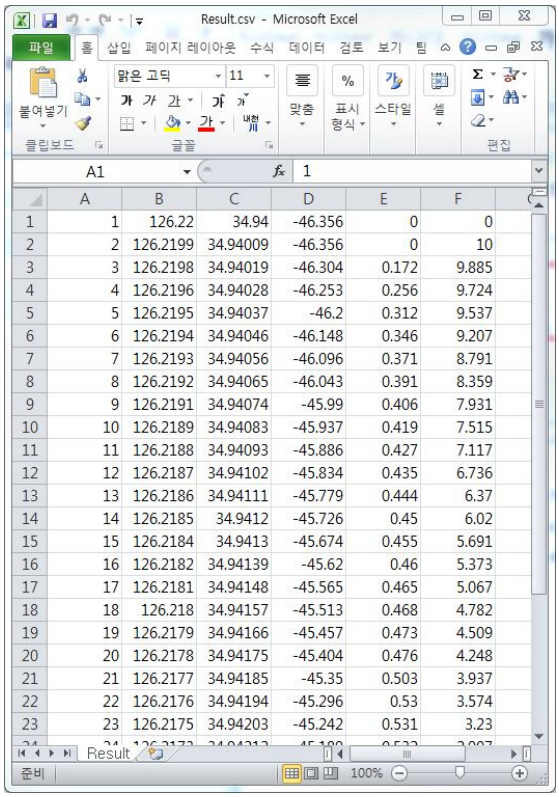


Fig. 18. UDP server



	A	B	C	D	E	F
1	1	126.22	34.94	-46.356	0	0
2	2	126.2199	34.94009	-46.356	0	10
3	3	126.2198	34.94019	-46.304	0.172	9.885
4	4	126.2196	34.94028	-46.253	0.256	9.724
5	5	126.2195	34.94037	-46.2	0.312	9.537
6	6	126.2194	34.94046	-46.148	0.346	9.207
7	7	126.2193	34.94056	-46.096	0.371	8.791
8	8	126.2192	34.94065	-46.043	0.391	8.359
9	9	126.2191	34.94074	-45.99	0.406	7.931
10	10	126.2189	34.94083	-45.937	0.419	7.515
11	11	126.2188	34.94093	-45.886	0.427	7.117
12	12	126.2187	34.94102	-45.834	0.435	6.736
13	13	126.2186	34.94111	-45.779	0.444	6.37
14	14	126.2185	34.9412	-45.726	0.45	6.02
15	15	126.2184	34.9413	-45.674	0.455	5.691
16	16	126.2182	34.94139	-45.62	0.46	5.373
17	17	126.2181	34.94148	-45.565	0.465	5.067
18	18	126.218	34.94157	-45.513	0.468	4.782
19	19	126.2179	34.94166	-45.457	0.473	4.509
20	20	126.2178	34.94175	-45.404	0.476	4.248
21	21	126.2177	34.94185	-45.35	0.503	3.937
22	22	126.2176	34.94194	-45.296	0.53	3.574
23	23	126.2175	34.94203	-45.242	0.531	3.23

Fig. 19. Saved CSV file

5. 결 론

본 시뮬레이터는 유도탄, 배, 표적 등과 같은 이동체를 모의하여 무기체계 개발에 필요한 다양한 시험을 지원할 수 있도록 개발하였다. 이와 같은 목적을 달성하기 위해서 사용자가 자유롭게 변경할 수 있는 곡선형태의 경로를 생성할 수 있어야한다. 이를 위하여 베지어 곡선 알고리즘을 적용하여 유연한 형태의 경로를 생성할 수 있었다. 또한 사용자가 이동체의 물리적 특성 모의 및 결과 데이터 처리 기능을 시뮬레이터에 확장, 추가할 수 있는 방안으로 DLL을 이용하는 방법을 제안하였다.

본 시뮬레이터를 이용한 시험에서 사용자가 자유롭게 변경할 수 있는 곡선형태의 경로를 생성하고 추종할 수 있음을 확인하였다. 또한 사용자가 이동체의 물리적 특성 및 모의 데이터 처리 기능을 DLL을 이용하여 기능의 추가, 변경이 가능함을 확인하였다.

지금까지 새로운 무기체계 개발이 시작되면 항상 해당 프로젝트에 필요한 이동체 시뮬레이터(자함모의기, 이동표적 모의기 등)를 새로 개발하거나 유사사업에서 사용한 시뮬레이터를 수정해서 사용하는 불편함이 있었다. 하지만 본 논문에서 제안한 시뮬레이터는 사용자가 경로를 자유롭게 수정할 수 있고, 이동체의 종류와 상관없이 특성과 데이터 처리 방법을 사용자가 추가 확장할 수 있기 때문에 다양한 무기체계 개발, 시험 과정에 사용할 수 있을 것이다. 따라서 무기체계 개발 초기 과정마다 새롭게 개발하던 이동체 시뮬레이터를 대체하여 시간을 절약할 수 있을 것이다.

후 기

논문 작성에 도움을 주신 모든 분께 감사의 말씀을 드립니다.

References

[1] Sanghoon Lee, Changmook Chun, Tae-Bum Kwon, Sungchul Kang, “Bezier Curve-Based Path Planning for Robust Waypoint Navigation of Unmanned Ground Vehicle,” Journal of Institute of Control, Robotics and Systems, 17.5, pp. 429-435, 2011. 5.

[2] Hyeock Jin Kim, Ha-Jine Kimn, Yonghoon Kwon. “Construction of Triplicated Piecewise Bezier Cubic - Curve as PC - Graphics Tool,” Journal of the Korea Information Science Society, 20.2, pp. 225-232, 1993. 1.

[3] Junyong Shim, Yongheon Lee, Kyutae Cho, Saehwan Kim, “An Interface Design Method of the Message Object for a Dynamic Plug-in Dynamic Linked Library,” KIISE Annual Conference Proceedings, 37.2A, pp. 38-39, 2010. 11.

[4] Jin-Seok Kim, Young-Do Lim, Jea-Young Heo, “The Simulator for Control of Quadcopter using Sensor Combination,” The Journal of Korean Institute of Information Technology, 10.7, pp. 1-11, 2012. 7.