논문 16-41-11-31 The Journal of Korean Institute of Communications and Information Sciences '16-11 Vol.41 No.11
http://dx.doi.org/10.7840/kics.2016.41.11.1557

자율형 UAV 에이전트 검증을 위한 DEVS/Unity3D 연동 시스템 설계

하 선 호◆, 김 정 호*, 김 현 근*, 신 석 훈**, 지 승 도°

DEVS/Unity3D Integrated System Design for the Autonomous UAV Agent Testing

Sun-ho Ha◆, Jeong-ho Kim*, Hyun-geun Kim*, Suk-hoon Shin**, Sung-do Chi°

요 약

인간이 개입하지 않는 UAV 시스템은 다양한 행동을 자율적으로 수행하여 임무를 달성해야하는 어려움이 있다. 그러나 인간이 개입하지 않는 환경에서의 실제 실험은 많은 위험과 비용이 발생한다. 따라서 실제 환경 실험에 앞서 시뮬레이션 실험을 통한 검증이 필요함은 자명하다. 본 논문에서는 에이전트 기반 자율 UAV의 실험을 위한 3차원 가상환경과 이산사건 시뮬레이션 환경 구축하고 연동한 시스템을 제안한다. 재난 상황에서의 UAV 3기를 이용한 구조임무 시뮬레이션을 통해 그 유효성을 검증 하였다.

Key Words : DEVS, Visualization, Simulation environment, Unity3D

ABSTRACT

The UAV systems working in difficult environment should be able to performs various actions autonomously required to achieve the given mission without the human interventions. However, the actual tests for such UAV system will take heavy cost. Thus, the simulation test in advance before the actual test is important. This paper proposes a 3D visual simulation environment for autonomous agent-based UAV systems. The several simulation tests performed on the rescue scenarios will demonstrate our techniques.

Ⅰ. 서 론

UAV(Unmanned Air Vehicle)는 조종사를 탑승하지 않고 지정된 임무를 수행할 수 있도록 제작한 비행체로써 군사적 목적뿐만 아니라 실생활에서도 널리 활용되고 있다. 특히, 인간이 어려운 재난상황에 대한 정찰용 무인기의 필요성이 급격히 대두되고 있다.[1]

인간이 접근 할 수 없는 지역에서의 무인기의 임무수행을 위해서는 무인기가 인간의 개입 없이 자율적으로 임무 수행할 수 있어야 한다. 무인기가 자율적으로 상황을 판단하고 행동하기 위해서는 자율 에이전트의 기능이 중요하다.[2]

하지만 에이전트를 완벽히 설계하더라도 검증을 하지 않은 채로 실제 무인기로 임무를 수행을 하게 된다

※ 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업의 연구결과로 수행되었음. (IITP-2016-H8601-16-1010)
◆ First Author : Korea Aerospace University Department of Computer Engineering, hsh205204@gmail.com, 학생회원
° Corresponding Author : Korea Aerospace University Department of Software, sdchi@kau.ac.kr, 정회원
* Korea Aerospace University Department of Computer Engineering, gspmkjh@gmail.com; rgdkdlel4@naver.com
** Korea Aerospace University Intelligent System Research Lab, ev4shin@kau.ac.kr, 학생회원
논문번호 : KICS2016-08-176, Received August 1, 2016; Revised October 7, 2016; Accepted November 18, 2016

면 미처 예상치 못한 결함으로 인해 무인기가 파손되는 등의 수많은 비용이 발생할 수 있다.[3] 이러한 문제점을 해결하기 위해 실비행 전에 시뮬레이션을 통해 에이전트의 오류를 수정하여 에이전트의 성능을 충분히 검증하여야 한다.

본 논문에서는 Unity3D를 활용하여 에이전트를 검증할 수 있는 DEVS 기반의 시뮬레이션 환경을 제안한다. 또한 재난 상황에서의 사례연구를 통해 자율형 UAV 에이전트의 검증 및 제안하는 시뮬레이션 환경이 적합한지 확인한다.

Ⅱ. 관련연구

2.1 자율형 UAV 에이전트

자율형 에이전트란 구축된 지식베이스를 기반으로 정보 융합, 추론, 학습 등을 통해 인간의 개입 없이 스스로 의사결정을 지원하기 위한 지능을 가진 요소이다.[4]

에이전트 기술은 국방 M&S 시스템에 도입되어 널리 연구되어 지고 있는데, 특히 정찬호 등 (2009)[5]은 에이전트의 의사결정 수준을 높여 복잡한 상황 묘사를 하기 위해 정보장교, 함장 및 함대사령관 등 역할에 따라 지능적 기능을 수행할 수 있는 계층구조적 다중 에이전트 시스템을 성공적으로 제안하였다. 또한 유용준, 지승도 등(2010) 은 LVC 연동 시뮬레이션을 경제적이고 효과적으로 수행할 수 있도록, 실제 사람을 대체할 수 있는 컴퓨터생성 가상전투객체를 효과적으로 모델링하고, 모델링된 가상전투객체가 자율성을 갖추고 목표지향적 행위를 할 수 있는 방안으로 HEAP 기반의 지능 에이전트 기법을 제안하였다.[6]

2.2 시뮬레이션 실험 환경 연구

공간상에서 벌어지는 시뮬레이션 검증을 위한 연구들은 오래전부터 있어왔다. 시뮬레이션 기법 중 하나인 이산사건 시뮬레이션(Discrete Event Simulation)이란 시간이 지나면서 발생하는 사건에 따라 모델의 상태가 변화되면서 진행되는 시뮬레이션이다. 시뮬레이션 할 대상이 활동하는 공간 모델(Space Model)로서 Zeigler 등(1999)과 Sarjoughian 등(2001)은 그림과 같이 Propagator, Logger, Spatial Encounter Predictor(SEP) Geographic Information System(GIS)의 4가지 모델로 구성된 공간 모델을 제안하였다.[2]

GIS 모델은 전장의 지형정보를 관리하며, Propagator 모델은 각 플랫폼 모델의 행위에 대한 결과를 다른 플랫폼 모델에게 전달해주는 모델이다.

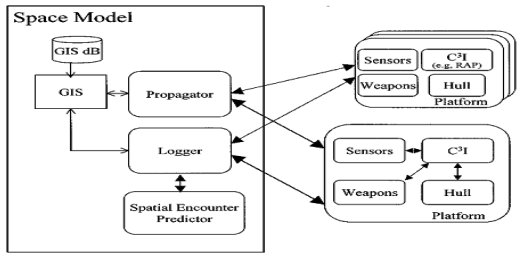


그림 1. 공간 모델의 구성 (Zeigler 등, 2000)
Fig. 1. Composition of Space Model (Zeigler et.al, 2000)

Logger 모델은 각 플랫폼 모델의 위치정보를 관리한다. SEP 모델은 다음 이벤트까지의 시간을 예측하여 건너뛰는 역할을 함으로써 시뮬레이션 시간을 단축시킨다.

시뮬레이션 할 대상에 대해 그에 적합한 공간모델의 구축은 중요한 요소이다. 하솔(2013) 등은 DEVS(Discrete Event System Specification) 기반 공간 모델을 적용하여 수중운동체 교전 시뮬레이션을 수행하고 이를 통해 공간 모델의 기능을 검증하였다.[7] 또한 기존의 전장 환경 요소가 플랫폼 모델에게 미치는 영향을 플랫폼모델에서 계산하는 대신에 독립적인 공간모델이 계산함으로서 보다 현실성 있는 시뮬레이션을 수행할 수 있다.

한편 시뮬레이션 가시화 분야에서는 범용의 게임엔진 Unity3D를 활용한 연구가 진행된바 있다. 데이터 가시화의 도구로써 Unity3D를 활용한 연구를 한 사례로 Gajananan, Kugamoorthy(2013)는 Unity3D를 이용하여 3D 가상 환경에서 AI 자동차들와 다른 차들을 시뮬레이션 할 수 있는 환경을 구축하여 실험 대상자들에게 운전을 할 수 있는 3D 환경을 제공하여 실험 대상자들이 운전 시뮬레이션을 하여 얻게 된 운전습관 데이터를 분석할 수 있었다.[8] 또한 Nader Hasan Khalifa 등(2015) 는 Unity3D를 이용하여 방대하고 다차원적인 생물 의학 및 유전 데이터를 분석하는 시스템의 사용자를 도울 수 있는 시각화 기법을 제시하였다.[9]

Ⅲ. 자율형 UAV 에이전트 검증을 위한 DEVS/Unity 연동 시뮬레이션 환경 및 가시화 시스템

그림 2는 실세계의 객체에 대응되는 시뮬레이션 모델을 개념적으로 나타낸 그림이다. 시뮬레이션 모델은 실세계의 공간자체를 모델링한 공간모델, 비행기를 모

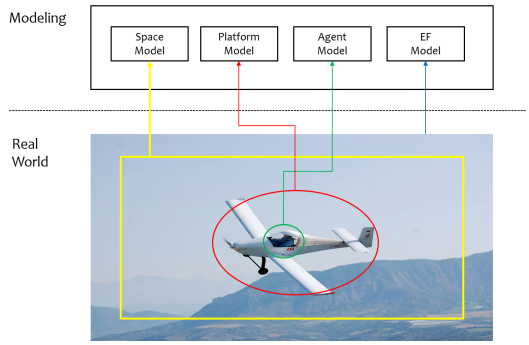


그림 2. 시뮬레이션 개념도
Fig. 2. Concept of Simulation

델링한 Platform 모델, 스스로 상황을 판단하여 결정을 내리는 비행기의 파일럿을 모델링한 Agent 모델, 실세계의 조건을 변경해 가며 실험 할 수 있는 EF(실험틀, Experimental Frame)[4]모델로 구성된다.

본 논문에서는 그림 3과 같이 기존 Zeigler 등(1999)이 제안하는 공간 모델의 구조를 기반으로 Unity3D게임엔진을 가시화 및 물리계산에 활용할 수 있도록 Unity 모델을 추가한 공간모델을 구축하였다. 또한, 시뮬레이션의 가시화를 통하여 사용자에게 시각적인 데이터를 제공하였다.

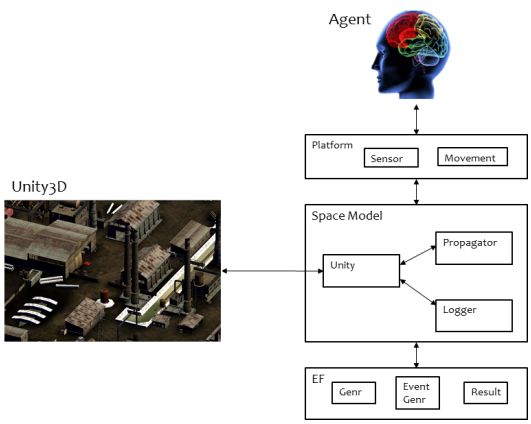


그림 3. Unity3D와 연동 개념
Fig. 3. Concept of Inter-operation with Unity3D

3.1 시뮬레이션 환경의 구성

본 연구에서 제안하는 시뮬레이션 환경은 공간모델, EF 모델, Platform 모델로 구성된다.

3.1.1 공간모델(Space Model)의 구축

공간모델은 Propagator, Logger, Unity 모델로 구성되며, 전체 시뮬레이션에서 건물, 장애물 등을 포함한 시뮬레이션 할 대상이 움직이는 공간을 표현한다.

(1) Propagator 모델

각 무인기들이 센서들을 통해 탐지 할 수 있는 범위 내에 존재하는 물체들의 정보를 플랫폼 모델로 전달해주는 모델이다.

그림 4와 같이 Unity 모델로부터 계산 된 '공간모델에서의 탐지 된 정보'를 SensorInfo_in 포트로 부터 받게 되며, 이를 SensedInfo 1,2,3 포트를 통해 3개의 Platform 모델에게 전달하며 Agent 모델로부터 계산된 이동명령에 대한 정보를 Command_in 1,2,3 포트로 부터 전달 받는다. 이를 CommandInfo_out 포트를 통해 Unity 모델에게 전달한다. 한편, 그림 4는 3개의 에이전트 모델 검증에 대한 Space 모델이다.

(2) Logger 모델

시뮬레이션이 일어나는 동안 각 플랫폼 모델의 위치 및 시나리오 상에 있는 객체 (건물, 구조자 등)의 위치의 데이터를 가지는 모델로써, 각 무인기들의 이동 후의 새로운 좌표를 갱신하여 저장하며, 이 데이터를 EF모델의 Result 모델에게 전달한다. 그림 4와 같이 PosInfo_out 포트를 통해 객체들의 위치정보를 받아 저장하고, 이를 Log_out 포트를 통해 EF모델에게 전달한다.

(3) Unity 모델

Propagator에서 요구되는 연산을 Unity 3D의 API를 이용할 수 있고, 시뮬레이션의 가시화를 위해 만들어진 모델이다. 또한, Unity 3D 프로그램과 통신을 하여 Unity3D 프로그램에서 무인기의 움직임을 가시화하는 동안 시뮬레이션의 진행을 정지시키며 시뮬레이션의 Logical Time을 증가시키는 역할을 한다. 또한 돌발상황을 시뮬레이션에 반영하는 역할을 한다.

Unity 3D 프로그램과의 연동은 그림 4와 같이 CommandInfo_in 포트를 통해 에이전트로부터 받은 명령을 전달 받고 Unity게임엔진에서 0.1 단위 시간이

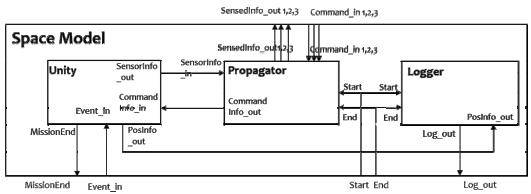


그림 4. Space 모델의 구조
Fig. 4. Structure of Space Model

지났을 때의 위치, 센서 값 등을 계산하며 SensorInfo_out 포트를 통해 Command가 적용된 후의 센서값을 Propagator 모델에게 전달한다. 또한, PosInfo_out 포트를 통해 Command가 적용된 후의 시뮬레이션 내의 모든 객체들의 위치를 Logger 모델에게 전달한다. 돌발상황에 대한 적용은 Event_in 포트를 통해 EF 모델로부터 돌발상황에 대한 정보를 전달 받아 Unity게임엔진에 적용한 후 계산한다. 마지막으로 MissionEnd 포트를 통해 시뮬레이션이 종료되었음을 알린다.

3.1.2 Platform모델

Platform 모델이란 공간 모델 내에서 독립적으로 동작하고 임무를 수행할 수 있는 모델로써 각 무인기별로 표현된다. 즉, 공간모델에서 움직이는 무인기의 몸체를 표현 한 것이며 Sensor 모델과 Movement 모델로 구성된다.

(1) Sensor 모델

각 무인기가 가지고 있는 센서를 표현하는 모델이다. 그림 5와 같이 FromSensor_in 포트를 통해 공간모델 내의 Propagator 모델로 부터 현 위치와 자세에서 센서로부터 탐지되는 객체의 정보를 전달 받으며, 이를 무인기 자신과 관련된 정보인 SelfInfo와 주변상황과 관련된 정보인 SensInfo로 나누어 SelfInfo는 SelfInfo_out 포트를 통해, SensInfo는 SensInfo_out 포트를 통해 각각 에이전트 모델로 전달한다.

(2) Movement 모델

각 무인기, 즉 플랫폼 모델이 공간모델 상 에서 이동할 수 있게 해주는 역할을 한다. 에이전트 모델로부터 의사결정 된 정보(이동 할 위치)를 Command_in 포트를 통해 받고 Command_out 포트를 통해 Propagator 모델에게 전달한다.

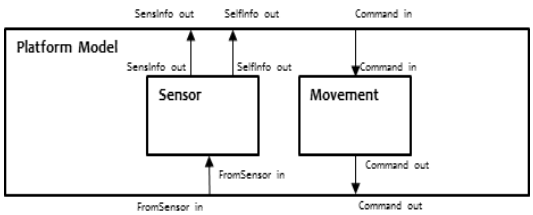


그림 5. Platform 모델의 구조
Fig. 5. Structure of Platform Model

3.1.3 EF모델

EF는 시뮬레이션 대상모델을 실험하기 위한 모델로써 본 논문에서는 Generator 모델, EventGenerator 모델, 그리고 Result 모델로 구성된다.

(1) Generator 모델

전체 시뮬레이션을 시작, 종료시키며 무인기의 초기 위치, 자세 값을 공간모델로 전달한다.

(2) EventGenerator 모델

시뮬레이션이 진행되는 동안 새 떼 등 돌발상황을 발생 시키고 돌발상황의 움직임을 업데이트하여 공간모델로 전달한다.

(3) Result 모델

시뮬레이션 각 step의 마다 저장되는 결과를 분석하여 시뮬레이션 종료조건이 충족된다면 전체 시뮬레이션 종료시점을 Generator 모델에게 전달하여 전체 시뮬레이션을 종료시킨다.

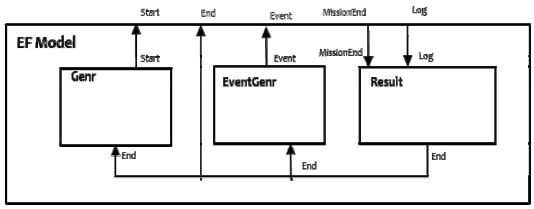


그림 6. EF 모델의 구조
Fig. 6. Structure of EF Model

3.2 Unity3D의 활용

Propagator 모델에서 요구되는 연산을 Unity3D의 물리엔진을 이용하여 계산하였다. 각 무인기들은 가지고 있는 소나센서와 카메라의 탐지범위 내에 물체가 있는지 Logical time이 지날 때 마다 연산한다. 만약 물체가 탐지범위 내에 있다면 Unity3D의 물리엔진을 이용해 물체의 절대좌표, 크기, 자세 정보, 인식여부에 대한 정보를 종합하여 DEVS 기반 공간모델의 Unity 모델에 전송한다. 공간모델과 Unity3D의 사이의 연동은 TCP 통신을 이용하였다.

또한 그림 1의 GIS 모델의 기능을 Unity3D로 대체할 수 있었다. 뿐만 아니라, GIS 모델을 변경할 때에도 시각적으로 확인하며 쉽게 할 수 있었다.

시뮬레이션 가시화는 Unity3D에서 3D모델 에디터에서 모델링한 객체들이 가지고 있는 C# 기반 스크립트를 이용하여 구현하였다. 연동 된 공간모델과 Unity3D 사이에서 무인기의 자세 값, 절대위치 값을 통신하여 Unity3D의 객체의 자세 값, 절대위치 값을

입력하여 가시화를 하였다.

Ⅳ. 사례연구

4장에서는 3장에서 제시한 시뮬레이션 환경의 유효성의 검증을 위해 재난상황에서 자율적으로 임무수행 할 수 있는 에이전트를 검증하는 실험을 협업, 돌발상황 대처, 임무재계획을 검증할 수 있는 시나리오로 나누어 실시하였다.

4.1 에이전트 모델 구성

에이전트는 그림 7과 같이 5개의 단위 에이전트인 Self Estimation, Identification, Situation Awareness, Planning, Path Planning으로 다중에이전트를 구성하였다. 각각의 에이전트는 SES 온톨로지로서 정보를 구성하였다.

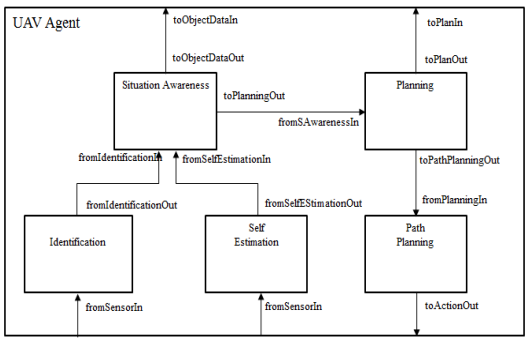


그림 7. 다중에이전트 구조
Fig. 7. Structure of Multi-agent

4.1.1 Self Estimation

자기 상태를 추정하는 에이전트로 배터리 상태와 운영시간에 따라 UAV의 상태가 결정된다. Platform 모델의 SelfInfo 포트로부터 무인기 자신의 상태에 관한 정보를 입력으로 받아 자기 상태를 추정한다.

4.1.2 Identification

감지된 주변 물체를 식별하는 에이전트로 Platform 모델의 SensInfo 포트로부터 무인기의 센서가 감지한 정보를 입력으로 받아 무인기 주변 물체를 식별한다.

4.1.3 Situation Awareness

무인기의 자기 상황을 추정하는 에이전트로 무인기 자신의 상태와 식별된 주변물체들 간의 관계성을 토대로 상황을 추정하고 전체 상황을 추정하여 계획에 반영 가능하도록 기능하는 에이전트이다.

4.1.4 Planning

현재 수행 중이거나 계획 중인 임무를 추정한 상황을 반영하여 임무를 재계획하는 에이전트이다. 현재 수행중인 임무와 회피, 조치, 식별, 접근 순으로 개별 상황의 우선순위를 결정하고 그 후 대상 물체와 자신 간의 거리 혹은 임무가 생성된 순서에 따라 우선도가 결정된다.

4.1.5 Path Planning

현재 임무와 임무 진행 진도 그리고 상황에 따라 수행할 행동을 결정하는 에이전트이다. 현재 장애물이 5단위 거리 내 접근 시 회피를 위해 무인기와 장애물의 상대방향으로 더 빠르게 이동하여 회피한다.

4.2 시뮬레이션 설정

그림 5와 같이 정육면체 모양의 빌딩에 화재가 일어난 상황이며 생존자들이 흩어져 있는 상황(그림 5의 ◯)에서 무인기 3대 (그림 5의 □)가 역할을 나누어 구조하는 상황을 구성하였다. 무인기의 센서는 거리 내의 물체를 탐지하는 소나센서와 카메라센서로 구성된다. 소나센서의 탐지범위는 60 으로 설정하고, 카메라는 거리 30 이하에 렌더링 되는 물체를 인식할 수 있다고 가정한다.

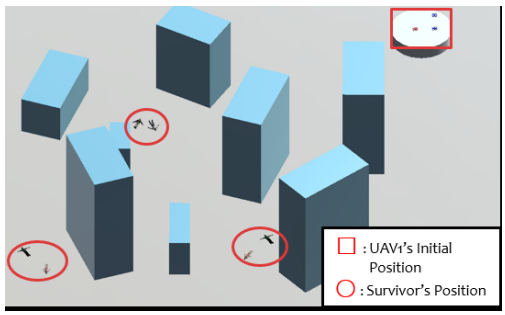


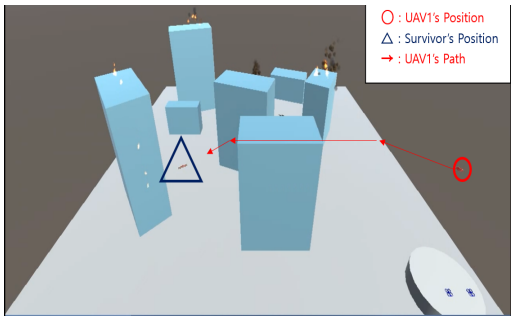
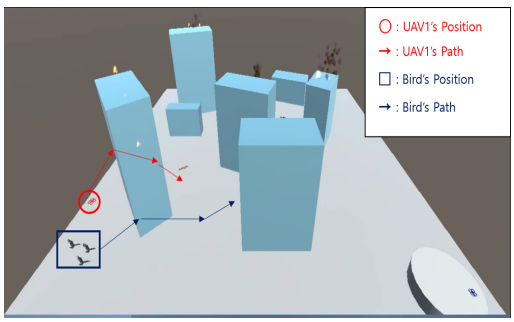
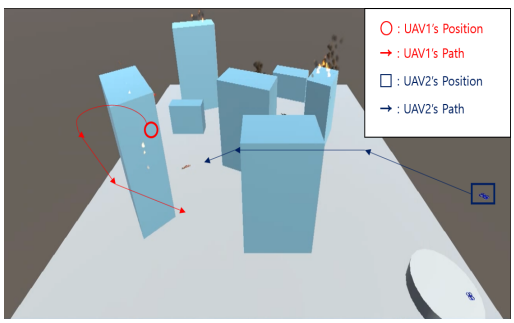
그림 8. 시나리오 전체상황
Fig. 8. General View of Simulation Scenario

4.3 결과

표 1은 각 시간대에서의 공간모델의 상태와 Unity3d의 진행상황을 나타낸다. 상황1 에서는 첫 번째 무인기(그림에서 ◯ 표시)가 정찰을 하다가 첫 번째 생존자를 발견하여 두 번째 무인기에게 출동명령을 내린 후 다시 정찰을 하는 상황이다.

상황 2에서는 상황 1에서 출동명령을 받고 출동 위치로 이동하는 장면이다. 이 때 공간모델은 상황 1에서 내린 호출명령을 받으면 에이전트로 두 개의 탐지정보(SensInfo1, SensInfo2)가 두 개의 다른 에이전트로 보내지고 그 결과로 두 개의 명령정보

표 1. 시간대에 따른 공간모델의 상태 및 Unity3D 가시화 현황
Table. 1. State of the Space Model and Visualization in Unity3D According to Time Zone

Time	State of Space Model		Visualization State of Unity3D
0.0~10.9 Situation 1: Take off~Survivor discover	Generator	Creation of mission, UAV initial position and Start simulation	
	Propagator	Calculation of angle of view Detection of straight line distance to surrounding objects	
	Platform	Move command delivery	
	Propagator	Transfer command received from agent	
	Unity	Apply Command	
	Logger	Update Location	
...	...		...
11.0~18.5 Situation 2: Cooperation of 2 UAVs	Propagator	Calculate angle of view for 2 UAVs Detection of straight line distance to surrounding objects	
	Platform	Deliver command information to two UAVs	
	Propagator	Transfer command received from agent	
	Unity	Apply Command	
	Logger	Update Location	
...	...		...
15.0~18.0 Situation 3: Unexpected Occurrence and Avoidance	Propagator	Calculate angle of view for 2 UAVs Detection of straight line distance to surrounding objects	
	Platform	Deliver command information to two UAVs	
	Propagator	Transfer command received from agent	
	Unity	Apply Command	
	EventGenerator	Creation of initial position and movement path of unexpected situation	
	Logger	Update Location	
...	...		...

(CommandInfo1, CommandInfo2)를 받아 Unity3D에서 두 대의 무인기(그림에서○, □)가 움직이게 된다.

상황 3은 정찰 중인 무인기가 돌발상황(새 떼)을 만날 때의 공간모델의 상태와 Unity3D의 진행상황이다. 돌발상황은 EventGenerator로부터 생성되며, Logger로부터 적용된다.

그림 9~10은 표 1의 상황1에서 Logical time 8.7과 10.9의 Unity3D의 무인기 카메라센서가 비추는 화면 이고 표 2~6은 그 때의 Logger 모델에서 저장된 값이다.

표 2는 그림 9의 위치에 무인기가 있을 때 Logger 모델에서 저장되는 무인기가 탐지한 물체의 데이터 값이다. 표의 첫 행은 현재 Logical time의 값이고 첫

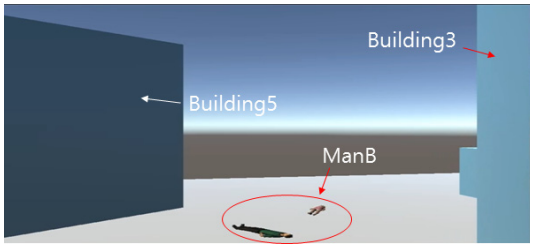


그림 9. Logical time 8.7의 무인기 카메라센서의 화면
Fig. 9. Screen Capture of Camera Sensor in UAV Model at Logical time 8.7

표 2. Logical time 8.7의 무인기에 탐지 된 데이터
Table. 2. Sensed Data of UAV Model at Logical time 8.7

Logical Time										
8.7										
Object Name	Pos X	Pos Y	Pos Z	Scale X	Scale Y	Scale Z	Slope X	Slope Y	Slope Z	isClassified
UAV1	58.2	14.9	49.1	0.3	0.3	0.3	-4.8	0	1.2	TRUE
Building3	56.3	12.2	59.1	20	25	10	0	0	0	TRUE

번째 열부터 마지막 열까지는 각각물체의 이름, 절대 좌표 x,y,z 값 , 물체의 크기 x,y,z 값, 물체의 x,y,z축으로부터 기울기(°), 인식여부 값이다. 그림 9처럼 Building3은 무인기로부터 30 이하의 거리에 있고 Building5는 60 이상의 거리에 있기 때문에 표 2와 같이 UAV1(무인기 자신)에 대한 정보와 Building3에 대한 정보와 무인기가 인식 하였다고 기록된다. 그림 10]은 무인기가 생존자 근처로 더 접근한 상황이고 표 5는 이 때의 무인기가 탐지한 물체의 데이터 값이다. 표 5를 보면 UAV1, Building3, ManB, Building5가 기록되어 공간모델의 Propagator 모델의 연산이 무인기의 현재 위치와 자세에 따라 제대로 계산되었다는 것을 알 수 있다.

표 3은 무인기가 이동해야 할 위치를 나타내는 데이터로써, 에이전트 모델의 Path Planning 모델로부터 얻은 에이전트모델에서 계산한 최종 값이다. 첫 행은 Logical time을 나타내고 두 번째 행은 UAV1의 명령

표 3. Logical time 8.7의 에이전트의 명령 데이터
Table. 3. Command Data from Agent Model at Logical time 8.7

Logical Time								
8.7								
Object Name	Pos X	Pos Y	Pos Z	Slope X	Slope Y	Slope Z	Speed	Comm Type
UAV1	23	15	58	166.4	90	76.4	5	1
UAV2	105	5	16	0	0	0	0	0
UAV3	110	5	16	0	0	0	0	0

표 4. Logical time 8.7의 공간모델내의 물체들의 위치
Table. 4. Location of Objects in Space Model at Logical time 8.7

Logical Time									
8.7									
Object Name	Pos X	Pos Y	Pos Z	Scale X	Scale Y	Scale Z	Slope X	Slope Y	Slope Z
UAV1	58.2	14.9	49.1	0.3	0.3	0.3	0	0	0
UAV2	105	5	16	0.3	0.3	0.3	0	0	0
UAV3	110	5	16	0.3	0.3	0.3	0	0	0
Building1	66.1	12.7	33.1	20	25	10	0	0	0
Building2	79.8	5.2	107.8	20	10	10	0	0	0
Building3	56.3	12.2	59.1	20	25	10	0	0	0
Building4	27	15.4	96.8	20	30	10	0	0	0
Building5	22.1	14.6	42.6	20	30	10	0	0	0
Building6	88.8	10.2	74.2	10	20	15	0	0	0
Building7	25.8	5.2	77.5	15	10	5	0	0	0
Building8	58.5	5.2	92	10	10	5	0	0	0
ManA	24.3	0.9	113.9	2	2	2	0	0	0
ManB	29.2	0.9	55.1	2	2	2	0	0	0
ManC	75	0.8	84.6	2	2	2	0	0	0

그림 10. Logical time 10.9의 무인기 카메라 센서의 화면
Fig. 10. Screen Capture of Camera Sensor in UAV Model at Logical time 10.9

표 5. Logical time 10.9의 무인기에 탐지 된 데이터
Table. 5. Sensed Data of UAV Model at Logical time 10.9

Logical Time										
10.9										
Object Name	Pos X	Pos Y	Pos Z	Scale X	Scale Y	Scale Z	Slope X	Slope Y	Slope Z	isClassified
UAV1	48.3	11.6	51.2	0.3	0.3	0.3	-4.2	-2.4	0.8	TRUE
Building3	56.3	12.2	59.1	20	25	10	0	0	0	TRUE
ManB	29.2	0.9	55.1	2	2	2	0	0	0	TRUE
Building5	22.1	14.6	42.6	20	30	10	0	0	0	TRUE

표 6. Logical time 10.9의 에이전트의 명령 데이터
Table. 6. Command Data from Agent Model at Logical time 10.9

Logical Time								
10.9								
Object Name	Pos X	Pos Y	Pos Z	Slope X	Slope Y	Slope Z	Speed	Comm Type
UAV1	23	15	58	166.4	90	76.4	5	1
UAV2	105	5	16	0	0	0	0	0
UAV3	110	5	16	0	0	0	0	0

표 7. Logical time 10.9의 공간모델내의 물체들의 위치
Table. 7. Location of Objects in Space Model at Logical time 10.9

Logical Time									
10.9									
Object Name	Pos X	Pos Y	Pos Z	Scale X	Scale Y	Scale Z	Slope X	Slope Y	Slope Z
UAV1	48.3	11.6	51.2	0.3	0.3	0.3	0	0	0
UAV2	105	5	16	0.3	0.3	0.3	0	0	0
UAV3	110	5	16	0.3	0.3	0.3	0	0	0
Building1	66.1	12.7	33.1	20	25	10	0	0	0
Building2	79.8	5.2	107.8	20	10	10	0	0	0
Building3	56.3	12.2	59.1	20	25	10	0	0	0
Building4	27	15.4	96.8	20	30	10	0	0	0
Building5	22.1	14.6	42.6	20	30	10	0	0	0
Building6	88.8	10.2	74.2	10	20	15	0	0	0
Building7	25.8	5.2	77.5	15	10	5	0	0	0
Building8	58.5	5.2	92	10	10	5	0	0	0
ManA	24.3	0.97	113.9	2	2	2	0	0	0
ManB	29.2	0.9	55.1	2	2	2	0	0	0
ManC	75	0.8	84.6	2	2	2	0	0	0

유형(이동)을 나타내며, 첫 번째 열부터 마지막 열까지는 명령대상의 이름, 이동해야할 위치의 x,y,z 좌표값, 움직일 때 무인기의 x,y,z축으로부터 기울기(°), 이동할 속도를 나타낸다.

표 4는 Logger모델에서 저장하는 공간모델 내의 모든 물체의 정보를 나타낸다. 첫 행은 Logical time을 나타내며, 첫 번째 열부터 마지막 열까지는 표 2의 구성과 같으나, 마지막 열의 인식여부는 제외한다.

표 1,3,4,6,7을 통해 무인기의 명령 데이터와 위치 데이터가 적절히 기록 되고 있고 그 때의 가시화가 일치함을 알 수 있다.

V. 결 론

본 연구에서는 사례연구에서 고자율성 에이전트의 실험을 통해 Unity3D를 이용한 고자율성 에이전트를 검증할 수 있는 시뮬레이션 환경을 구축하였고, 가시화하였다. Unity3D를 이용함으로써 Propagator 모델에 필요한 연산인 특정위치에서의 화각계산 등과 같은 복잡한 연산을 Unity3D의 API를 통해 비교적 간단히 해결하였고 고자율성 에이전트의 무인기를 재난상황에서 시뮬레이션을 진행하여 시뮬레이션이 진행되는 시간대 별로 DEVS기반 공간모델의 상태변화와 가시화가 되는 모습을 확인할 수 있었다. 또한 특정시간에서의 Result 모델에서 저장된 로그 값을 확인하여 무인기의 현재위치에서의 올바른 센서 값이 계산되는지를 확인하였고 에이전트로부터 얻은 명령 값을 확인하였다. 이를 통해 에이전트 설계자는 실제 드론을 사용하지 않고도 시뮬레이션을 통해 그 기능을 현실감 있게 검증할 수 있을 것이라 예상한다. 또한 구축한 시뮬레이션 환경이 자율성 에이전트를 검증하는 데에 유효함을 알 수 있었다.

향후에는 DEVS/Unity 연동 시스템과 실세계의 비행데이터를 연동함으로써 L-V-C 기반의 시뮬레이션을 구축하여 다양한 분야에 응용할 예정이다.

References

[1] S. Y. Im, "Utilizing drones from the disaster site safe," *Sci. Technol. Policy*, vol. 25, no. 6, pp. 16-19. Jun. 2015.

[2] H. S. Sarjoughian, B. P. Zeigler, and S. B. Hall, "A layered modeling and simulation architecture for agent-based system development," in *Proc. IEEE,* vol. 89, no. 2, pp. 201-213, 2001.

[3] S. H. Shin, E. B. Lee, and S. D. Chi, "LVC indoor environmental studies for the verification of multiple autonomous UAV," *J. Ind. Eng. Joint Conf. Proc.*, pp. 2155-2159, Jeju Island, Korea, Apr. 2015.

[4] B. P. Zeigler, H. Praehofer, and T. Kim, *Theory of modeling and simulation*, 2nd Ed., Academic Press, 2000.

[5] Y. J. You, S. D. Chi, and J. I. Kim, "A study of HEAP-based intelligent agent applied to warship combat simulation," *J. Korea Soc. for Simulation*, vol. 19, no. 4, pp. 281-289. Dec. 2010.

[6] C. H. Jung, Y. J. You, H. E. Ryu, J. S. Lee, J. I. Kim, and S. D. Chi, "Multi-platform warship M&S system using the hierarchical multi-agent system," *J. Korea Soc. for Simulation*, vol. 18, no. 4, pp. 117-125, Dec. 2009.

[7] S. Ha, N. K. Ku, K. Y. Lee, and M. I. Roh, "Development of battle space model based on combined discrete event and discrete time simulation model architecture for underwater warfare simulation," *J. Korea Soc. for Simulation*, vol. 22, no. 2, pp. 11-19. Jun. 2013.

[8] K. Gajananan, et al., "An experimental space

for conducting controlled driving behavior studies based on a multiuser networked 3D virtual environment and the scenario markup language," *IEEE Trans. Human-Machine Syst.*, vol. 43, no. 4, pp. 345-358, 2013.

[9] N. H. Khalifa, et al., "A visualization system for analyzing biomedical and genomic data sets using unity3D platform," in *Proc. HIKM 2015*, pp. 47-53, Sydney, Australia, Jan. 2015.

하 선 호 (Sun-ho Ha)

2016년 2월 : 한국항공대학교 소프트웨어학과 졸업
2016년 3월~현재 : 한국항공대학교 컴퓨터공학과 석사과정
<관심분야> 인공지능, 무인시스템, VR, LVC

김 정 호 (Jeong-ho Kim)

2016년 2월 : 한국항공대학교 소프트웨어학과 졸업
2016년 3월~현재 : 한국항공대학교 컴퓨터공학과 석사과정
<관심분야> 지능 에이전트, DEVS, 무인시스템

김 현 근 (Hyun-geun Kim)

2016년 8월 : 한국항공대학교 소프트웨어학과 졸업
2016년 9월~현재 : 한국항공대학교 컴퓨터공학과 석사과정
<관심분야> 머신러닝, GPGPU, 분산 시뮬레이션

신 석 훈 (Suk-hoon Shin)

2008년 8월 : 한국항공대학교 컴퓨터공학과 졸업
2011년 2월 : 한국항공대학교 컴퓨터공학과 석사
2011년 3월~현재 : 한국항공대학교 컴퓨터공학과 박사과정
<관심분야> 에이전트, 국방 M&S, 무인시스템

지 승 도 (Sung-do Chi)

1982년 2월 : 연세대학교 전기공학과 졸업
1984년 2월 : 연세대학교 전기공학과 석사
1991년 : 아리조나대학교 전기전산공학 박사
1992년~현재 : 한국항공대학교 소프트웨어학과 교수
<관심분야> 무인자율시스템, 지능시뮬레이션, 자율에이전트