FSM 기반의 자율 에이전트를 이용한 시뮬레이션 시스템의 설계 및 구현

김동준°, 김대령, 우종우
국민대학교 컴퓨터학부
{dejey°, drforgod, cwwoo}@cs.kookmin.ac.kr

A Design and Implementation of Simulation System using FSM-based Autonomous Agent

Dongjun Kim° , Daeryung Kim, Chongwoo Woo
School of Computer Science, Kookmin University

요 약

최근 시뮬레이션의 분야에 인공지능의 요소가 적용된 연구가 진행되고 있다. 본 연구에서는 다양한 인공지능의 기법들 중에서 FSM을 기반으로 한 자율적 에이전트를 설계하고 이를 핑-퐁 게임에 적용시켜 실험하였다. 이러한 FSM 기반의 에이전트를 이용하게 되면 모델 설계가 용이하며, 시스템을 구성하고 있는 모든 에이전트들의 상태를 쉽게 확인 할 수 있고, 시뮬레이션 모델설계를 단순화할 수 있는 장점이 있다.

1. 서 론

시뮬레이션이란 실 세계의 다양한 객체들의 구조와 행위에 대한 자료를 수집하여 모델화하고 이를 컴퓨터 프로그램으로 모의 실험함을 말한다. 주로, 복잡한 시스템의 내부구조 및 객체들의 상호작용을 관찰하거나 상태천이를 예측하는 경우, 또는 새로운 시스템을 설계하는 경우 등에 효과적으로 사용된다. 이러한 시뮬레이션은 연구용, 훈련용, 오락용 등으로 그 사용목적 및 적용분야가 다양한데, 국내의 연구를 보면, 대표적으로 의료 분야 [1,2], 정보보호 분야 [3, 4], 그리고 다수의 군 관련 연구분야 [5, 6]에 관한 연구가 활성화 되고 있다. 이러한 최근의 시뮬레이션 연구의 기반은 대부분의 경우 DEVS(Discrete Event Systems Specification)이론[7]을 중심으로 특정 분야에 적용을 하고 있으며, 최근에는 에이전트를 기반으로 한 연구가 시작되고 있다 [8, 9].

본 연구에서는 다양한 인공지능의 기법들 중에서 FSM(Finite State Machine)을 기반으로 기본적인 시뮬레이션 모델을 설계하였으며, 이를 기반으로 지능형 에이전트의 기능을 수행할 수 있게 구성하였다. 시스템의 성능을 테스트하기 위하여 Ping-pong 게임을 시뮬레이션 하였다. 이러한 FSM 기반의 에이전트를 이용하게 되면 모델 설계가 용이하며, 시스템을 구성하고 있는 모든 에이전트들의 상태를 쉽게 확인 할 수 있고, 시뮬레이션 설계 모델을 단순화 할 수 있는 장점이 있다.

본 연구의 시뮬레이션 엔진은 현재 라이브러리로 개발 중이며, 궁극적으로는 범용적인 에이전트기반의 시뮬레이션 엔진구축을 목표로 하여, 추론엔진, 스크립트 및 모델링 제작 툴까지 점진적으로 추가될 예정이다.

본 논문의 구성은 다음과 같다. 2장 관련 연구에서는 시뮬레이션연구에 필수적인 이론적 기반에 대하여 설명하고, 3장에서는 FSM기반의 에이전트 시스템의 설계에 관하여 기술한다. 4장에서는 에이전트 시스템의 시뮬레이션 테스트 결과를 기술하고, 5장에서는 결론을 맺는다.

2. 관련 연구

시뮬레이션을 위한 주요 기반 기술들로는 구조적 오토마타, Petri-Net, 에이전트, DEVS 형식론 등이 존재한다. 본 장에서는 이러한 이산사건 시스템의 모델링 시뮬레이션을 하는 DEVS 형식론과 에이전트 시스템에 대하여 기술한다.

2-1. DEVS 형식론

Zeigler에 의해 제안된 DEVS 형식론은 이산사건 시스템을 모듈 별로 나누어서 계층적으로 모델링 할 수 있는 방법을 제공한다. 기본적으로 집합이론에 기반을 두고 이산사건 시스템을 수학적 공식에 의거하여 객체지향적으로 모델링 할 수 있는 툴을 제공한다. 또한 입력사건/출력사건을 명시적으로 정의하고 전체시스템을 부 시스템들의 계층적 결합으로 모델링 한다. 전체 시스템을 부 시스템으로 나누어 가는 과정에서 부 시스템 객체를 결합모델(coupled model)로 정의하며 더 이상 나

눌 수 없는 모델 객체를 원자모델(atomic model)로 정의한다. 이러한 원자모델은 시스템의 상태천이를 집합기호를 사용한 수학적 공식에 의해 명세화 하며 원자모델들 간에는 어떤 정보도 공유되지 않으며, 필요한 정보교환은 오직 입/출력에 의해서만 이루어지게 하는 모듈(modular) 명세 기법을 사용하고 있다. DEVS 원자모델은 내부에 그 모델의 상태를 표시하는 state set을 갖고 적절한 transition function 에 따라 자신의 state 를 변화시키는 것으로 시스템을 표현한다. 또한 결합모델은 모델내부에 포함될 구성요소가 무엇인가와 이들 구성요소 사이의 입/출력 인터페이스를 명세화한다. 이때 구성요소는 원자모델 혹은 결합모델이 될 수 있다.

2-2. 에이전트 시스템

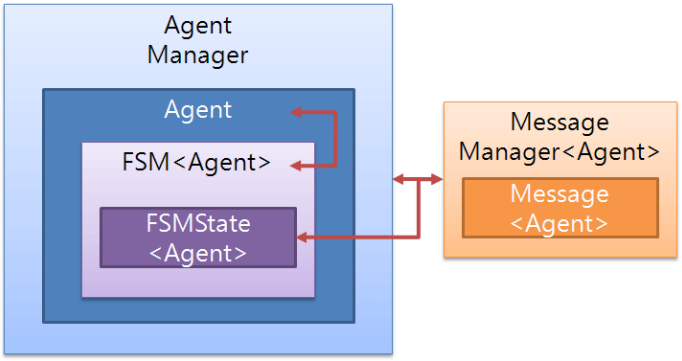
에이전트란 다양한 해석이 제공되지만 기본적으로는 사용자를 대신하여 하는 작업을 자동으로 해결해주는 소프트웨어라고 할 수 있다 [10]. 따라서 에이전트는 자신의 목적을 달성하기 위하여 자율적으로 행동하고 환경에 반응하게 된다. 또한 자율적으로 생성, 작동, 소멸하면서 다른 에이전트와의 협력을 통해 문제를 해결하고 목표를 달성해 나가며 기존의 software component와는 다르게 자동성, 목표지향성, 유연성, 자율성, 사회성, 적응성, 그리고 이동적 특성을 가질 수 있다.

이러한 에이전트는 다양한 기술들로 개발이 가능하지만, 본 논문에서는 유한한 상태들로 표현하는 FSM(Finite State Machine)을 기반으로 설계 및 개발하였다. FSM의 알고리즘 자체는 간단하지만, 거의 모든 시스템들이 상태들로 표현이 될 수 있어 많이 사용되고 있다. FSM의 기본기능은 하나의 입력에 의해 현재 상태로부터 다른 상태로 전이하게 된다. 이렇게 구성된 에이전트는 시뮬레이션의 각 객체로 모델링을 하게 되며, 각각의 에이전트가 시뮬레이션 진행 도중, 주어진 상태로 변화하게 되며 그 변화에 따른 특정 작업을 수행하게 된다.

3. 시스템 설계

3-1. 시스템 개요

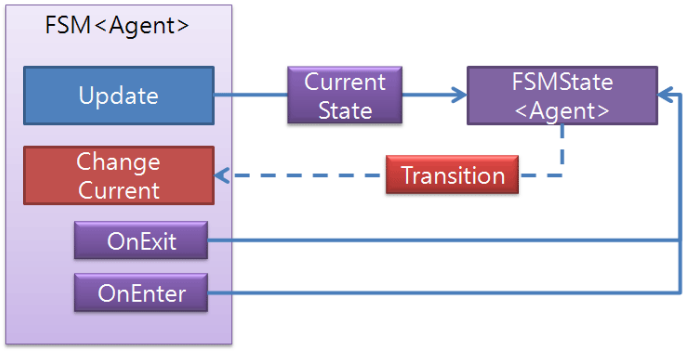
본 논문에서 제안한 FSM 기반의 에이전트 시스템은 라이브러리형식으로 개발이 되어, Client나 Server에 구애 받지 않고 적용이 가능하며, 전체적인 구조는 다음 [그림 1]과 같다. 전체 시스템의 구성은 FSM 처리 모듈과 메시지 처리 모듈, 그리고 두 모듈을 사용하는 에이전트 모듈로 구성되며, 각각의 에이전트는 각자의 FSM을 갖고 있으며 Message Manager를 통해 서로간에 메시지를 주고 받는다.



[그림 1] 시스템 구조

3-2. FSM 모듈

FSM 모듈은 다음 [그림 2]와 같은 구조를 가진다.

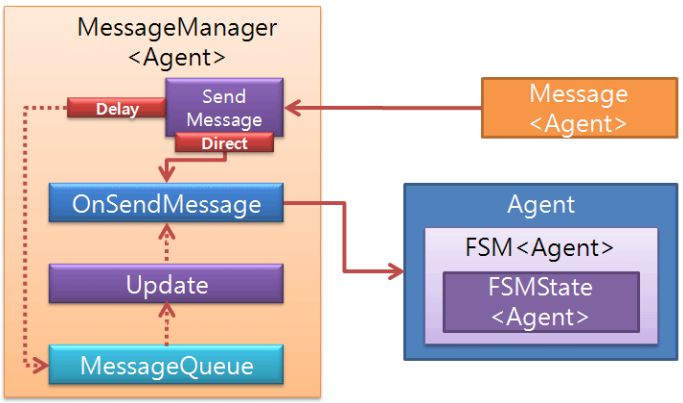


[그림 2] FSM 모듈 구조

FSM 모듈은 에이전트의 각 상태들을 관리하는 모듈로서, 각 상태에서의 Update와 전이를 처리하는 FSMState 클래스와 FSM State 객체들을 관리하는 FSM 클래스로 구성된다.

3-3. 메시지 처리 모듈

메시지 처리 모듈은 다음 [그림 3]과 같은 구조를 가진다.



[그림 3] 메시지 처리 모듈 구조

메시지 처리 모듈은 에이전트간에 주고 받는 메시지

를 처리하는 모듈로서, Message 클래스와 Message Manager 클래스로 구성된다. Send Message 함수를 통해 메시지를 전달할 수 있으며, 바로 전달할 수도 있고 지연 시간을 주어 전달할 수도 있다.

4. 시스템 개발

4-1. 개발환경

본 시스템의 구현을 위해서 다음 [표1]과 같은 개발 환경 및 도구를 사용하였다.

<표1. 개발 환경>

System	Pentium IV1.6GHz(512 MB)
OS	Windows XP
Editor	Microsoft Visual studio.Net

시스템은 라이브러리 형식으로 개발 하였으며, Microsoft Visual Studio.Net 환경에서 STL를 이용하여 Pure C++로 구현하였다.

4-2. 개발 시스템 특징

본 논문에서 구현한 시스템은 소스코드 레벨에서 소유객체의 타입에 따라 코드를 변환 시켜줘야 하는 문제점을 해결하고 코드의 재사용을 높이기 위하여 Template을 사용하여 FSM을 구현하였다. 이로 인해 라이브러리 확장이 훨씬 용이해졌으며, 응용 어플리케이션을 개발 시 소스코드 작성시 발생하는 많은 번거로움을 감소시켰다. 다음 [그림 4]는 핑-퐁 시뮬레이션을 테스트하는 소스코드의 일부분이다.

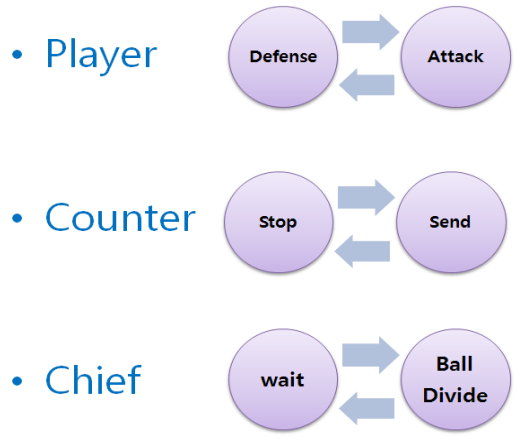
```
// 에이전트들을 메니저에 등록하고 시뮬레이션을 실행
// Add Player
LxPlayer* pPlayer0 = new LxPlayer;
pPlayer0->SetID( 0 );
pPlayer0->SetName( "Player0" );
LxAgentManager::Instance()->AddAgent( pPlayer0 );
...
// Run
LxAgentManager::Instance()->Run();
```

```
// 에이전트의 생성시 상태들을 등록하고 상태들간에 전이를 설정
LxPlayer::LxPlayer(void)
{
    // Make State
    LxDefenseState* pDefenseState = new LxDefenseState( 0 );
    m_FSM.AddState( pDefenseState );
    LxAttackState* pAttackState = new LxAttackState( 1 );
    m_FSM.AddState( pAttackState );
    // Set Transition
    m_FSM.AddTransition( 0, 1, 1 );
    m_FSM.AddTransition( 1, 0, 0 );
    //  Set Current State
    m_FSM.SetCurrentState( 0 );
}
```

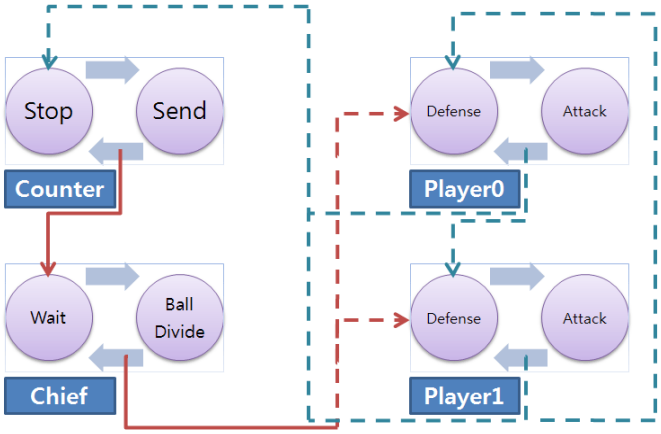
[그림 4] 핑-퐁 시뮬레이션 소스코드

4-3. 시뮬레이션 테스트

본 논문에서 구현한 FSM 기반의 에이전트 시스템을 테스트하기 위하여 핑-퐁 게임을 시뮬레이션 하였다. 핑-퐁 모델은 선수, 카운터, 심판으로 구성이 되며, [그림 5]는 각각의 상태도를 나타낸다. [그림 6]은 핑-퐁 게임의 메시지 흐름도이다. 카운터가 시작되면 심판이 기다렸다가 볼을 분배를 하게 된다. 볼은 선수0 또는 선수1에게 임의 전달이 되어 공격을 하게 된다. 상대편 선수는 수비를 성공 또는 실패를 하게 되며, 만약 수비에 실패하면 점수를 업데이트하기 위하여, 카운터에게 메시지가 전달된 후, 재 시작되는 일련의 과정을 한 선수의 점수가 21점이 될 때까지 반복하게 된다.



[그림 5] Ping-Pong State Diagram



[그림 6] Ping-Pong 메시지 흐름도

본 연구에서는 DEVS 기반 시뮬레이션과의 비교를 위해 DEVS에서 자주 사용되는 핑-퐁 모델을 사용하였으며, 모델의 구조 역시 기존의 모델 구조에 큰 변화를 주지 않았다. 모델링 관점에서 두 방법을 비교해보면, DEVS 기반 시뮬레이션은 메시지의 흐름을 우선하여 도메인을 모델링하고, 계층적인 구조(Coupled model)를 고려하여야 하는 반면, FSM 기반의 에이전트 시스템은 상태 변화를 우선하여 모델링을 하며 객체 모델들이 상위와 하위의 계층적 구조로 이루어지지 않기 때문에 이러한 구조적인 문제를 고려하지 않아도 모델링이 가능하게 된다. 즉 단순히 OOP(Object-Oriented-Programming)의 관점에서 모델링이 가능하다는 것이다.

다음으로 메시지 전달 관점에서 두 방법을 비교해보면, DEVS 기반 시뮬레이션은 메시지가 계층적으로 전달되고, 그 메시지를 받아야 하는 객체만 전달을 받게 된다. 이에 반해 FSM 기반의 에이전트 시스템은 메시지의 흐름이 수평적(직접 연결)으로 전달이 되고, 모든 객체들에게 메시지 전달을 하게 된다.

5. 결론

본 논문에서는 FSM 기반의 에이전트 시스템을 구현하였으며, 이를 이용하여 핑-퐁 모델을 시뮬레이션 하였다. 이러한 FSM 기반의 에이전트를 이용하게 되면 다음과 같은 몇 가지 장점을 가지게 된다.

첫째, 도메인에 대한 모델 작성이 용이하다. 기존 DEVS 형식론은 모델링 시 도메인에 대한 이해뿐만 아니라, 형식론에 대한 기반지식을 필요로 하기 때문에, 모델작성시 많은 부분이 고려가 되어야 한다. 하지만 본 연구에서 설계한 시스템은 도메인에 대한 이해만을 필요로 하기 때문에, 모델 작성이 용이하게 된다.

둘째, 모든 에이전트들의 상태 확인이 가능하다. 메시지의 흐름이 계층적인 구조가 아닌 수평적 구조로써, 직접 전달이 이루어지게 되어 모든 에이전트들의 상태가 확인이 가능하게 된다.

셋째, 모델의 단순화가 가능하다. DEVS형식론에서는 기본적으로 Atomic 모델과 Coupled 모델이 존재하여야 한다. 그러나 본 연구에서 설계한 시스템은 Coupled 모델이 필요가 없다. 각 모델들이 에이전트가 되기 때문에 계층적 구조에서 상위 개념인 Coupled 모델이 필요가 없게 된다.

향후 연구로는 추론엔진, 그래프엔진, 스크립트 및 모델링 제작 툴까지 포함된 범용적인 에이전트 시뮬레이션 엔진을 점진적으로 추가해갈 계획이다.

참고문헌

[1] 김광년 외 5인, "DEVS 모델을 사용한 심근 활성화 과정의 시뮬레이션", 한국 시뮬레이션 학회 논문지 제13권 제4호, pp1-16 2004.12

[2] 이규원 외 4인, "비례제어밸브와 혼합 제어기를 이용한 혈압 시뮬레이터의 구현", 한국 시뮬레이션 학회 춘계학술대회 논문집 pp149-153, 2005.3

[3] 유용준, 이장세, 지승도, "SIMVA를 이용한 시뮬레이션 기반의 네트워크 취약성 분석", 한국 시뮬레이션 학회 춘계학술대회 논문집, pp13-19, 2004.

[4] 정정례 외 3인, "사이버 공격 시뮬레이션을 위한 공격자 및 호스트 모델링", 한국 시뮬레이션학회 논문지 제12권 제2호, pp63-73, 2003.6

[5] 최상영, "기갑 전투그룹 교전 시뮬레이션 모델", 한국 시뮬레이션학회 논문지 제 12권 제1호, pp73-83 2003.3

[6] 강정호 외 7인, "DEVS 기반 모델링을 적용한 잠수함의 어뢰회피 성능 분석 시뮬레이션", 한국 시뮬레이션 학회 논문지 제14권 제2호 2005.6

[7] B.P. Zeigler and Tag G. Kim, and H. Praehofer, "Theory of Modelling and Simulation", Academic Press, 2000.

[8]M. Le Bars, J.M. Attonaty, N. Ferrand and S. Pinson, "An Agent-based Simulation Testing the Impact of Water Allocation on Farmers' Collective Behaviors", Simulation, Vol.81, No.3, pp223-235, 2005

[9]T. Cioppa and T. Lucas, "Military Applications of Agent-based Simulations", Proceedings of the 2004 Winter Simulation Conference, pp994-1000

[10]N. Jennings, K. Sycara, and M. Wooldridge, "A roadmap of agent research and development", Autonomous Agents and Multi-Agent Systems. Vol.1, Issue 5, pp 7- 38, 1998