다중작업 운영체제하에서 화이트-박스 시뮬레이션 게임의 구현

White-Box Simulation-Based Game in a Multi-Tasking Operating System

김동환*
Dong-Hwan Kim

Abstract

Traditionally, simulation-based learning games which are known as flight-simulators have been constructed as a black-box game. Within a black-box game, game-players can view and modify only a part of model parameters. Game-players cannot change the structure of a simulation model. In a black-box game, game-players cannot understand and learn the system structure which is responsible for the system behavior. In this paper, the multi-tasking at the level of operating systems is exploited to enhance the transparency of simulation-based learning game. The white-box game or transparent-box game allows game-players to view and modify the model strucuture. The multi-tasking solution for white-box learning game is implemented with Smalltalk language on MS-Windows operating system.

1. 문제의 제기: 인공적 시간을 통한 일정계획의 한계

컴퓨터 시뮬레이션은 전통적으로 수많은 개체(entity)들 간의 반독립적인 상호작용을 다루어 왔다. 따라서 컴퓨터 시뮬레이션에서는 개념적으로 다양한 개체들의 병렬적인 활동을 상정하여 왔다. 컴퓨터 운영체제의 입장에서 볼 때, 개체들간의 독립적인 활동은 다중작업을 의미한다. 최근 유닉스, OS/2, 윈도우즈 및 윈도우즈NT 등의 보편화에 따라 운영체제의 다중작업기능은 워크스테이션급 컴퓨터에서 뿐만 아니라 개인용 컴퓨터에서도 기본적인 기능으로서 지원되고 있다.

그러나 일반적으로 독립적인 개체들의 활동을 시뮬레이션하기 위하여 운영체제의 다중작업기능이 동원되지는 않는다. 이산형 시뮬레이션의 경우 개체들의 독립적인 활동은 사건-리스트를 통한 사건일정계획(event-scheduling)방식에 의해 이루어진다[10]. 연속형 시뮬레이션의 경우에는 시스템 요소들의 값이 동시에 변화되는 것으로 간주된다. 사건일정계획방식이나 시스템 요소들의 동시적 변화의 가정은 다중작업을 수행하지 않고도 개체들의 독립성과 병렬성을 구현할 수 있도록 하였다. 이와같이 개체들의 독립성.병렬성이 다중작업을 통하지 않고 수행될 수 있었던 것은 인공적인 시간개념을 통하여 개체들의 독립성과 병렬성을 쉽게 시뮬레이션할 수 있었기 때문이다.

* 전자통신연구소 초빙연구원.

최근 시뮬레이션 기법의 확산에 따라 시뮬레이션 기법은 시스템 설계의 도구로서 뿐만 아니라 일반인에게 시스템을 학습시키는 도구로서도 각광을 받고 있다[4][14]. 더우기 시뮬레이션-기반 학습게임은 일반인 또는 기업경영자들의 잘못된 사고과정을 밝히는데도 커다란 기여를 하고 있다[18]. 시뮬레이션-기반 학습게임에 있어서 시뮬레이션 모델과 시뮬레이터 및 게임수행자(학습하는 사람)들은 실시간으로 상호작용하게 된다. 여기에서 시뮬레이션 모델과 시뮬레이터 및 게임수행자들은 각기 하나의 독립적이고 병렬적인 개체들로 간주될 수 있다. 이때 인공적인 시간을 사용한 시뮬레이터의 일정계획메카니즘은 그 한계점을 노출하게 된다. 왜냐하면 비록 시뮬레이션 모델 속의 개체들은 인공적인 시간에 의해 통제될 수 있지만, 게임수행자인 인간은 인공적인 시간에 의해 통제될 수 없기 때문이다.

본 논문에서는 전통적인 시뮬레이션 수행방식이 시뮬레이션-기반 학습게임에 있어서 어떠한 문제점을 가져오는지를 살펴보고, 운영체제의 다중작업기능을 통하여 그 문제점을 해결할 수 있음을 보이고자 한다. 본 논문에서 제안한 해결책은 윈도우즈 환경하에서 Smalltalk를 사용하여 구현되었다. 본 논문의 주요 연구대상은 연속형 시뮬레이션 파라다임의 하나인 시스템 다이나믹스(system dynamics)의 시뮬레이션-기반 학습게임이다.

2. 시뮬레이션-기반 학습게임의 투과성

최근 시스템 다이나믹스 계열의 시뮬레이션 분야에 있어서 시뮬레이션-기반 학습게임이 보편화되고 있다[5][17]. 시뮬레이션-기반 학습게임이란 시뮬레이션 모델에 기반을 두어 게임수행자가 스크린을 통해 게임을 수행하는 것을 의미한다. 게임수행자는 모델의 시뮬레이션 과정에 개입하여 수단적인 변수들의 값을 변화시켜 목적변수들의 값을 통제하기 위하여 노력한다. 게임수행자는 이러한 게임을 통해 시스템의 동태적인 전개과정에 대하여 스스로 실험해볼 수 있으며 실험을 통한 학습을 수행한다. 시뮬레이션-기반 학습게임은 특히 기업경영진에 대하여 이루어져 왔다.

시뮬레이션-기반 학습게임은 개념적으로 시뮬레이션 모델에 관한 모듈(작업)과 시뮬레이터 및 게임상호작용에 관한 모듈(작업)로 구성된다. 이들은 다중작업기능하에서 각각 독립적으로 작동될 수도 있으며, 하나의 프로그램 모듈로 통합되어 작동될 수도 있다.

기존의 시뮬레이션-기반 학습게임은 운영체제의 다중작업능력을 활용하지 않았다. 기존에는 시뮬레이션 모델을 시뮬레이터로 통합시키는 방식이 일반적으로 사용되어, 시스템 다이나믹스의 DYNAMO를 비롯하여 이산형 시뮬레이션 언어들인 GPSS, SLAM II, SIMAN 등은 모델을 컴파일하여 수행시켰다. 이러한 방식하에서 시뮬레이션-기반 학습게임을 구축하기 위하여, 추가적으로 게임상호작용 모듈을 컴파일시켜 새로운 시뮬레이터를 구축한다[15]. 시뮬레이션 모델의 모듈과 게임상호작용 모듈 및 시뮬레이터는 하나의 프로그램으로 통합된다. 따라서 학습게임의 수행에는 운영체제의 다중작업기능이 필요하지 않았다.

이러한 방식은 시뮬레이션-기반 학습게임의 효과적 수행에 매우 심각한 제한을 가져왔다. 즉, 게임을 수행하는 도중에, 사용자는 시뮬레이션 모델을 탐색하거나 수정할 수 없다는 점이다. 게임수행자가 시뮬레이션 모델의 구조에 대해 실험할 수 없는 경우, 게임수행자는 시스템의 구조에 관하여 효과적으로 학습할 수 없게 된다[6].

게임수행자가 게임의 기반이 되는 모델 구조에 접근할 수 없는 경우, 시뮬레이션-기반 학습게임은 단순한 게임으로 전락한다. 게임수행자는 게임을 수행하면서 시스템의 구조를 이해하려고 노력하지 않으며, 단순히 게임화면상의 표면적인 상관관계에 주목하게 된다. 그리고 게임수행자는 자신이 얻은 점수에 집착하게 되며, 어떠한 구조적 메카니즘에 의해 그러한 결과가 도출되었는지를 이해할 수 없게 된다. 결과적으로 모델의 구조에 접근할 수 없는 시뮬레이션-기반 학습게임은 게임수행자에게 효과적인 학습기회를 제공하지 못하게 된다.

이와같은 시뮬레이션-기반 학습게임의 한계를 제거하여 보다 효과적인 학습을 수행하기 위하여, 시뮬레이션-기반 학습게임은 '투과적(transparent)'이어야 한다고 Machuca는 주장하였다[12][13]. 여기에서 투과적이라는 용어는 게임수행자가 시뮬레이션 모델의 구조에도 접근할 수 있어야 한다는 점을 의미한다. Machuca의 논의를 종합하여 시뮬레이션-기반 학습게임의 투과성의 단계를 정리하여 보면 〈표 1〉과 같다.

일반적인 시뮬레이션-기반 학습게임은 1단계를 충족시킨다. 즉, 게임수행자가 게임수행 화면을 통하여 시뮬레이

〈표 1〉 시뮬레이션-기반 학습게임의 투과성의 단계

투과성의 단계	특 성
제 1 단계	게임수행자는 언제든지 시뮬레이션 모델의 파라미터를 변화시킬 수 있다.
제 2 단계	게임수행자는 언제든지 시뮬레이션 모델의 구조를 탐색할 수 있다.
제 3 단계	게임수행자는 언제든지 시뮬레이션 모델의 구조를 수정할 수 있다.

션 모델의 파라미터를 수정하면, 그 수정된 파라미터에 의해 시뮬레이션이 전개된다. 게임수행자는 이후 전개되는 시뮬레이션을 관찰하면서, 자신이 변화시킨 파라미터가 시스템 전반에 대해서 어떠한 영향을 가져오는지를 이해한다. 이러한 학습게임은 게임수행자가 모델의 구조를 탐색할 수 없다는 점에서 '블랙박스(black-box) 게임'이라고 불린다.

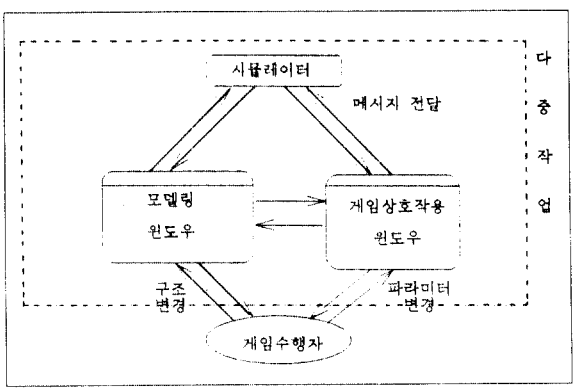
Machuca가 제기한 두번째 단계와 세번째 단계는 게임수행자로 하여금 시뮬레이션 모델의 파라미터 뿐만 아니라 시뮬레이션 모델의 구조에도 접근할 수 있도록 허용해야 한다는 점을 의미한다. 두번째 단계는 게임수행자의 이해를 향상시키기 위하여 시뮬레이션 모델의 구조를 탐색(search & view)할 수 있도록 허용하는 단계이다. 세번째 단계는 게임수행자로 하여금 시뮬레이션 모델의 구조를 자유롭게 변경시키도록 허용하여, 모델 구조의 변화에 따른 시뮬레이션 전개 양상을 학습하도록 허용하는 것이다. 두번째와 세번째 게임은 게임수행자가 모델의 구조를 탐색할 수 있다는 점에서 '화이트박스(white-box) 게임' 혹은 '투과적(transparent) 게임'이라고 불린다[7][12].

Machuca는 최근 두번째 단계의 투과성을 지닌 시뮬레이션-기반 학습게임의 구현을 발표한 바 있다[16]. 이와 함께 시스템 다이나믹스 학자들 및 시뮬레이션 소프트웨어 회사들은 투과성을 지닌 시뮬레이션-기반 학습게임의 구축을 발표하였다[1][11]. 그러나 이들의 해결책은 여전히 시뮬레이션 모델을 게임상호작용 모듈에 종속시키는 방식을 사용하고 있다. 즉, 시뮬레이션 모델에 관한 설명과 그림을 게임상호작용 모듈에 포함시키는 방식이다. 여전히 시뮬레이션 모델 자체는 게임상호작용 모듈에 종속되어 접근될 수 없다. 따라서 최근에 제시된 해결책하에서 게임수행자는 시뮬레이션 모델을 참조할 수는 있지만, 여전히 모델의 구조를 변경시킬 수는 없다.

3. 운영체제의 다중작업기능을 이용한 학습게임의 투과성 확보

3.1 개략적인 구조

최근에 보편화된 다중작업 운영체제는 시뮬레이션-기반 학습게임을 설계하는데 있어서 새로운 가능성을 제공해준다. 즉, 시뮬레이션 모델에 관한 모듈과 게임상호작용에 관한 모듈 그리고 시뮬레이터 모듈을 상호 독립시켜 병렬적으로 작동하게 만드는 방법이다.



〈그림 1〉 운영체제의 다중작업을 이용한 시뮬레이션-기반 학습게임

본 논문에서는 운영체제의 다중작업기능을 사용하여 〈표 1〉에서 3단계의 투과성까지 확보할 수 있는 시뮬레이션-기반 학습게임을 구축하였다. 이를 위하여 본 논문에서는 Windows 3.1의 운영체제와 Smalltalk/V를 사용하여 시뮬레이션-기반 학습게임을 설계할 수 있는 소프트웨어를 구성하였다.[1] 그 개념적인 구조는 〈그림 1〉과 같다.

〈그림 1〉에서 시뮬레이션-기반 학습게임은 시뮬레이터

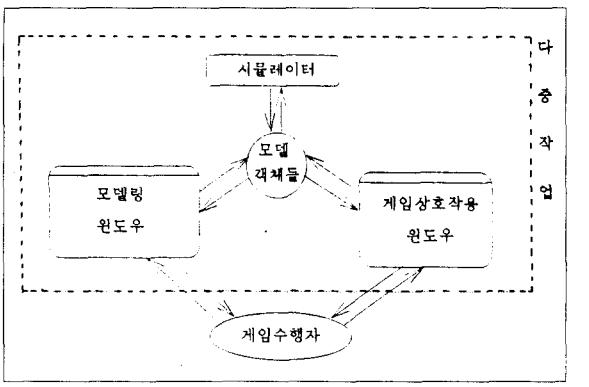
1) 본 소프트웨어는 EGO(Equations as Graphic Objects)라는 이름으로 1994년 영국에서 개최된 시스템 다이나믹스 국제학술대회에서 시연되었다. 본 소프트웨어의 일반적인 특징은 Icon기반의 모델링, 각 변수들 및 정의식의 그래픽 표현, 계층적 모델링의 지원, 신경망과 같은 복잡한 객체를 포함한 시뮬레이션, 시뮬레이션

(시뮬레이션 모델의 수행), 시뮬레이션 모델링, 게임상호작용 등 세가지 작업으로 구성된다. 윈도우 운영체제는 여러 작업들을 동시에 작동시킬 수 있으며, 따라서 위의 세가지 모듈들은 윈도우 환경에서 쉽게 다중처리될 수 있다. 여기에서 시뮬레이션 모델링 윈도우란 시뮬레이션 모델의 구조를 참조하며, 변화시킬 수 있는 윈도우를 의미한다.

〈그림 1〉에서 게임수행자는 시뮬레이션 모델링 윈도우와 게임상호작용 윈도우에 접근할 수 있다. 게임수행자는 시뮬레이터에 직접 접근할 수는 없다. 게임상호작용 윈도우에의 접근을 통하여 게임수행자는 시뮬레이션 진행에 따른 특정 변수들의 값의 변화를 관찰하며, 모델의 파라미터를 수정시킬 수 있다. 이는 1단계의 투과성을 의미한다. 모델링 윈도우에의 접근을 통하여 게임수행자는 시뮬레이션 모델을 참조할 수 있으며 동시에 시뮬레이션 모델의 구조를 변화시킬 수 있다. 이는 투과성의 2단계 및 3단계를 의미한다.

이들 세가지 모듈들간에는 긴밀한 메시지 전달이 유지되어야 한다. 즉, 시뮬레이터에 의해 모델이 시뮬레이션되면서 각 변수들의 값이 변화되면, 변화된 변수의 값은 게임상호작용 윈도우 및 모델링 윈도우로 전달되어야 한다. 또한 게임상호작용 윈도우에서 게임수행자가 모델의 파라미터를 수정하였을 경우, 수정된 파라미터를 반영하여 시뮬레이션이 수행되어야 한다. 만약 모델링 윈도우에서 게임수행자가 시뮬레이션 모델의 구조를 바꾸었을 경우, 즉 시뮬레이션 모델의 변수들간의 관계가 변화되는 경우, 시뮬레이션 모델링 윈도우는 이를 시뮬레이터에 전달해야 하며 동시에 게임상호작용윈도우에도 전달해야 한다.

결과적으로 어느 한 모듈에 변화가 발생할 경우, 세가지 모듈들은 모두 동시에 변화되어야 한다. 이는 세가지 모듈간의 상호의존성을 유지하기 위하여 복잡한 통제메카니즘이 필요하다는 점을 의미한다. 본 연구에서는 이를 해결하기 위하여 공동으로 접근할 수 있는 객체(object)를 사용하였다. 여기에서 공동으로 접근할 수 있는 객체란 광역변수를 의미하는 것은 아니다. 다만 모델을 구성하는 객체들과 세개의 모듈들간에 직접 메시지를 교환할 수 있도록 하는 것이다. 시뮬레이터, 모델링 윈도우, 게임 윈도우 등은 이들 객체들에게 메시지를 전달하며, 각 객체들은 전달받은 메시지를 자신의 고유한 메소드를 통해 처리한다. 모델을 구성하는 객체를 매개로 하여 세가지 모듈들간의 메시지 전달이 이루어지는 경우, 〈그림 1〉의 구조는 다음과 같이 변화된다.



〈그림 2〉 운영체제의 다중작업과 객체의 사용

〈그림 2〉에서 세가지 모듈들은 모델구성 객체들과 지속적으로 상호작용한다. 여기에서 모델구성 객체는 모델의 변수와 그 변수에 대한 정의식을 의미한다. 여러개의 변수들로 구성된 모델은 여러개의 객체들을 갖는다. 〈그림 2〉에서 어느 한 부분에서 변수의 값이나 구조에 변화가 발생된 경우, 그 변화는 곧장 모델구성 객체에게 전달된다. 그리고 모델구성 객체들은 자신의 변화된 값을 다시금 세 모듈들에게 전달한다. 따라서 시뮬레이션-기반 학습게임을 설계하는데 있어서 세가지 모듈들간의 복잡한 통제메카니즘을 구축할 필요가 없다.

그러나 세가지 모듈들에서 두개 이상의 메시지가 동시에 전달되는 경우 모델구성 객체들은 어떠한 메시지를 먼저 처리할 것인가를 결정해야 한다. 모델구성 객체들을 사용하는 경우, 세가지 모듈들간의 메시지 전달에 관한 규칙을 마련할 필요는 없는 반면, 모델구성 객체들은 어느 모듈에서 전송된 메시지를 먼저 처리해야 하는가를 결정해야 한다.

결과를 해석하기 위한 도표지원, 피드백 루프에 관한 분석 등이다. 본 소프트웨어를 사용하여 신경망과 시뮬레이션 모델을 연결한 예는[8][9]를 참조할 수 있다.

3-2. 작업들간의 우선순위

병렬적인 작업들간의 상호작용에는 충돌해소가 필요하며, 이를 위하여 각 작업들간의 우선순위 부여가 필요하다. 게임수행자의 간섭이 없는 경우, 〈그림 2〉에서 세가지 모듈들간의 상호작용에는 충돌이 발생되지 않는다. 게임수행자의 간섭이 없는 경우, 모델링 윈도우와 시뮬레이터 모듈간에는 메시지 전달이 필요하지 않으며, 시뮬레이터 모듈은 일방적으로 게임상호작용모듈에 메시지 전달을 수행하기 때문이다.

게임수행자가 게임상호작용윈도우에서 파라미터의 값을 변화시키는 경우, 이는 곧 바로 시뮬레이션 수행에 (즉, 시뮬레이터에) 반영되어야 한다. 또한 게임수행자가 모델링 윈도우에서 변수들간의 관계를 변화시키는 경우 이 또한 곧 바로 시뮬레이터에 반영되어야 한다.

〈표 2〉 다중작업들간의 우선순위에 관한 Smalltalk 프로그램 코드

```
1  egoGame := EGOgame new open: EGOgameGraphs.
2
3  [ [ (EGOvariables at: #TIME) < endTime ]
4        whileTrue:
5        [ self updateTIME.
6          self externalValue.
7          self levelValue.
8          self auxValue.
9          self save.
10         egoGame draw.
11         [egoGame isRun] whileFalse: [].
12         lastTime := Time millisecondClockValue.
13       [ speed < ( (Time millisecondClockValue) - lastTime)]
14                 whileFalse: [].
15       ].
16 self endSimulation.
17 egoGame endMessage. ] forkAt: 3.
```

따라서 시뮬레이터는 모델링모듈이나 게임상호작용모듈보다 낮은 우선순위를 부여받아야 한다. 또한 게임수행자가 시뮬레이션수행 도중에 간섭할 수 있도록 허용하기 위하여, 시뮬레이터는 게임수행자의 개입행위 보다 적은 우선순위를 부여 받아야 한다. 이와같은 점들을 반영한 시뮬레이터는 〈표 2〉와 같다.

〈표 2〉는 시뮬레이션-기반 학습게임을 수행시키는 Smalltalk의 메소드중 일부이다. 〈표 2〉의 메소드가 수행되기 이전에 이미 모델링 윈도우는 활성화되어있는 상태이며, 〈표 2〉의 1번째 줄에서 게임상호작용 윈도우가 활성화된다. 〈표 2〉에서 시뮬레이션을 수행시키는 부분은 5번째 줄에서 9번째 줄에 해당된다. 10번째 줄은 시뮬레이션을 통해 변화된 값을 게임상호작용 화면에 전달하는 역할을 한다.

3번째 줄에서 17번째 줄까지의 명령은 하나의 블럭(block)으로 묶여져 있으며, 이는 하나의 작업으로서의 역할을 한다. 이 블럭은 〈그림 2〉에서 시뮬레이터에 해당된다. 17번째 줄에서 이 시뮬레이터 블럭은 우선순위가 3으로 지정되어 있다. 이는 시뮬레이터의 우선순위가 사용자 개입의 우선순위인 4보다 낮아야 하기 때문이다[2][3].

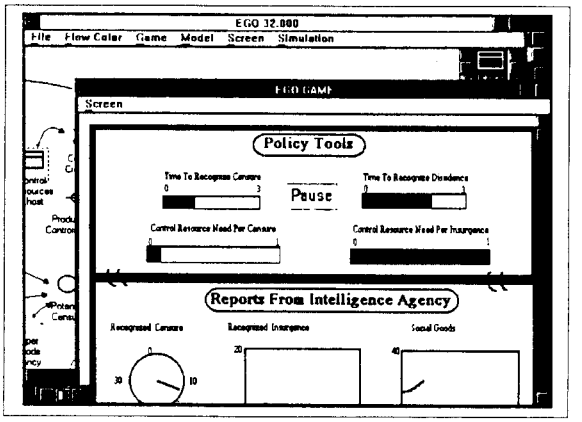
게임수행자는 모델링윈도우와 게임상호작용 윈도우상에서 필요하다고 느끼는 경우 언제든지 간섭하여 모델구조나 모델파라미터를 변경시킬 수 있다. 이러한 게임수행자의 개입행위는 시뮬레이터 블럭(모듈)보다 우선순위가 높으며, 따라서 이러한 행위는 시뮬레이터와 동시에 그러나 높은 우선순위로 이루어진다. 게임수행자의 개입행위가 끝나고 모델구조나 모델파라미터가 변화되는 경우, 그 결과는 즉시 모델객체들에 전달된다. 그리고 시뮬레이터가 위의 5번째 줄에서 10번째 줄까지의 작업을 수행하고자 할 때, 시뮬레이터는 이미 변화된 모델객체들을 참조하여 작업을 수행하게 된다.

4. 투과적 시뮬레이션-기반 학습게임의 사례

본 논문에서 사례로 들은 시뮬레이션 게임은 Khalid Saeed의 정부통제모델이다. 〈그림 3〉의 화면은 이 모델을 사용하여 시뮬레이션-기반 학습게임을 작성하고 이를 작동시킨 예이다.

〈그림 3〉에서 화면의 뒷부분에 위치한 윈도우는 모델링 윈도우이며, 전면에 위치한 윈도우는 게임상호작용 윈도우이다. 〈그림 3〉에 시뮬레이터는 표현되어 있지 않다. 이는 게임수행자가 시뮬레이터에까지 직접 접근할 필요는 없기 때문이다. 게임수행자는 〈그림 2〉에서와 같이 모델링 윈도우와 게임상호작용 윈도우에만 접근할 수 있다.

모델링 윈도우에서 게임수행자는 모델객체들의 정의식을 변화시키거나 모델객체들간의 연결관계를 변화시킬 수

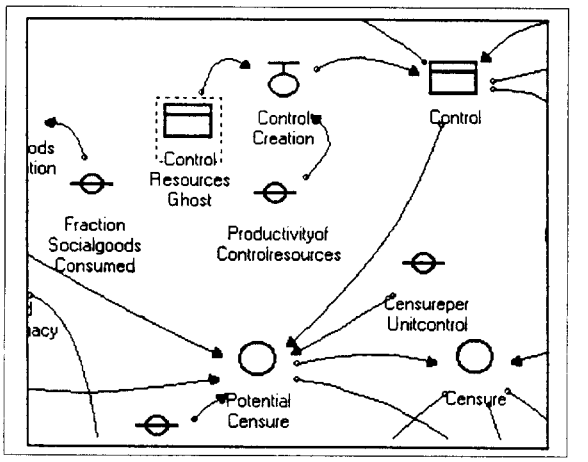


〈그림 3〉 다중작업을 통한 시뮬레이션-기반 학습게임의 화면

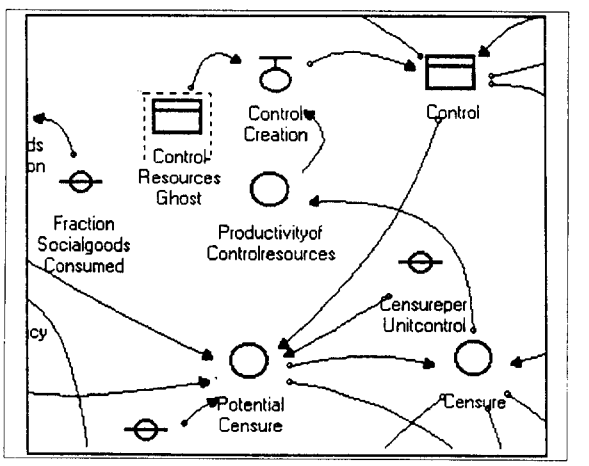
있다. 게임상호작용 윈도우의 윗부분에는 게임수행자가 개입할 수 있는 변수들이 나열되어 있으며, 하단 부분에는 게임수행자에게 시스템의 상태를 보여주는 화면이 나열되어 있다. 게임수행자는 게임상호작용 윈도우를 통하여 모델의 파라미터를 수정한다. 물론 모델의 파라미터는 모델링 윈도우를 사용하여서도 수정할 수 있다. 이러한 수정으로 인한 시뮬레이션의 변화는 곧 바로 게임상호작용 윈도우의 하단 부분에 표시된다.

여기에서는 게임수행 도중에 모델의 구조를 변경시켜 보았다. 〈그림 4〉는 변화시키기 이전의 모델의 구조이며, 〈그림 5〉는 변화시킨 이후의 모델의 구조이다. 모델구조의 변경은 시뮬레이션 시각 25 시점에서 이루어졌다. 〈그림 4〉에서 ProductivityOfControlresources는 상수로 표현되어 있다. 이 상수는 ControlCreation이라는 비율변수(rate variable)에 영향을 준다. 게임수행도중에 ProductivityOfControlresources를 상수에서 변수(보조변수)로 변화시켰으며, 그 정의식도 'ProductivityOfControlresources = Censure'로 변화시켰다. 즉, 원래의 모델에서 ProductivityOfControlresources와 Censure는 상호독립적이었지만, 모델의 변경이후에 ProductivityOfControlresources의 값은 Censure의 값에 따라 비례적으로 결정된다.

〈그림 6〉과 〈그림 7〉은 게임수행중에 모델의 구조를 변경시켜본 결과를 보여준다. 〈그림 6〉은 모델의 구조를 변경시키지 않은채 시뮬레이션을 수행했을 때, SocialGoods의 값이 어떻게 변화되는가를 표시한 것이며, 〈그림 7〉은 게임수행중에 모델의 구조를 〈그림 5〉와 같이 변경시켰을 때 SocialGoods의 값이 어떻게 변화되는가를 표시한 것이



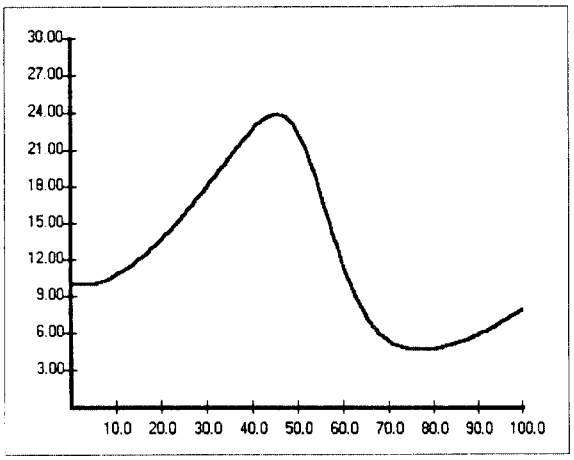
〈그림 4〉 변화시키기 이전의 모델구조



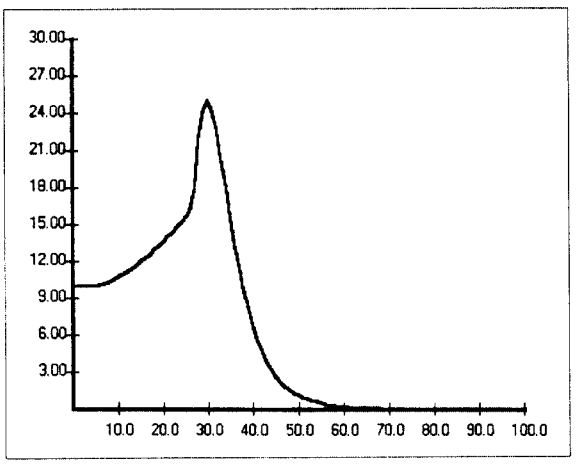
〈그림 5〉 게임수행시에 변화시킨 모델의 구조

다. 〈그림 6〉과 〈그림 7〉에 있어서 횡축은 시뮬레이션 시간의 흐름을 의미하며, 종축은 SocialGoods의 값을 의미한다. 〈그림 6〉과 〈그림 7〉은 25시점까지는 동일한 형태를 보이지만, 모델의 구조를 변경시킨 25시점 이후부터는 상이한 형태를 보인다.

이러한 결과는 운영체제의 다중작업기능을 사용한 시뮬레이션-기반 학습게임상에서 모델의 구조를 변화시킴으로써 게임수행자는 시뮬레이션의 전개방향을 변화시킬 수 있다는 점을 보여준다.



〈그림 6〉 모델구조를 변화시키지 않았을 때의 SocialGoods 값의 변화



〈그림 7〉 25 시점에서 모델구조를 변화시켰을 때의 SocialGoods 값의 변화

5. 앞으로의 연구과제

본 연구에서는 운영체제의 다중작업기능을 사용하여 투과성이 높은 시뮬레이션-기반 학습게임을 설계하는 방법에 관하여 논의하였다. 본 논문의 연구결과는 Machuca가 제안한 세번째 단계의 투과성까지도 쉽게 확보할 수 있음을 보여주었다.

그러나 비록 낮은 투과성으로 인해 학습장애가 발생된다는 점은 많은 연구를 통하여 밝혀졌지만, 아직까지 투과성이 높은 시뮬레이션-기반 학습게임이 게임수행자들의 학습효과를 증진시킬 것인지에 관한 검증은 이루어지지 않았다. 이는 아직까지 투과성이 높은 시뮬레이션-기반 학습게임을 설계할 수 있는 도구가 제공되지 못했기 때문이기도 하다. 앞으로의 연구과제로 몇가지 질문들을 제안할 수 있다. 과연 투과성의 증가가 학습효과의 증가를 가져올 수 있는가? 모델구조의 변경을 통한 학습효과와 모델파라미터의 변경을 통한 학습효과는 어느 것이 보다 큰가? 모델파라미터의 변경을 금지한 채 모델구조의 변경만을 허용했을 경우, 학습효과는 어떻게 나타날 것인가?

앞으로 본 논문에서 연구된 결과는 이러한 연구질문들을 탐색하는데 중요한 도구를 제공할 수 있으리라 기대한다. 또한 이러한 연구질문들에 대한 탐구를 통하여, 컴퓨터 시뮬레이션을 통한 학습이 보다 활성화될 수 있기를 기대한다.

참고문헌

[1] Davidsen, P.I., "The system dynamics approach to computer-based management learning environments: Implications and their implementation in POWERSIM," J.D.W. Morecroft & J.D. Sterman (eds.), *Modeling for learning organizations*, Productivity Press, 1994, 301-315

[2] Digitalk, *Smalltalk/V for Windows Turorial and Programming Handbook*, 1992, Digitalk Inc., 1992.

[3] Goldberg A. & D. Robson, *Smalltalk-80: The Language*, Addison-Wesley Co., 1989.

[4] Graham A.K. & Senge P., "Computer-based case studies and learning laboratory projects," *System Dynaics Review*, Vol. 6, No. 1, 1990, 100-105.

[5] Graham A.K., J.D.W. Morecroft, P.Senge & J.D. Sterman, "Model-supported case studies for management education," J.D.W. Morecroft & J.D. Sterman (eds.), *Modeling for learning organizations*, Productivity Press, 1994, 219-241.

[6] Issacs W.N. & Senge P., "Overcoming learning limits in CBLE's," *European Jurnal of Operational Research*, Vol. 59, No. 1, 1992, 195.

[7] Kemeny J.M. & Kreutzer B., "An archetype based Management Flight Simulator," *Proceedings Internation-*

al System Dynamics Conference, Utrecht, Netherlands, 1992, 305.

[8] Kim D.H. & I.J. Chung, "Neural Network Heuristics for Controlling System Dynamics Models," *1994 International System Dynamics Conference*, 43-52.

[9] Kim D.H. & I.J. Chung, "Neural network as a policy maker in dynamic social systems," *Proceedings of International Conference on Neural Information Processing*, 1994, 1873-1878.

[10] 김승연, 김동환, 시스템 시뮬레이션과 시뮬레이션 언어, 홍릉과학 출판사, 서울, 1993.

[11] Lyneis J.M., K.S. Reichelt & T. Sjoblom, "Professional DYNAMO: Simulation software to facilitate management learning and decision making," J.D.W. Morecroft & J.D. Sterman (eds.), *Modeling for learning organizations*, Productivity Press, 1994, 349-358.

[12] Machuca J.A., "Are we losing one of the best features of System Dynamics?," *System Dynamics Review*, Vol. 8, No. 2, 1992, 175-177.

[13] Machuca J.A., et al., "Systems thinking learning for management education, what aare we going about it in Sevilla?," *1993 International System Dynamics Conference*, Mexico, 1993.

[14] McLeod J., "Computer Simulation: A personal view," *Behavioral Science*, Vol.34, 1989, 1-15.

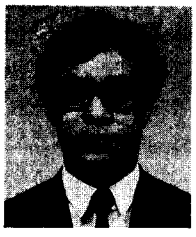
[15] Pegden C.D., R.E. Shannon & R.P. Sadowski, *Introduction to Simulation Using SIMAN*, McGraw-Hill Inc., 1990.

[16] Ruiz, J.C. & Machuca J.A., "Creation of a new interface for transparent- box games," *European Simulation Multiconference 94*, Barcelona, 1994.

[17] Senge P. & Sterman J., "System Thinking and Organizational Learning," *European Jurnal of Operational Research*, Vol. 59, No.1, 1992, 137-150.

[18] Sterman J., "Modelling managerial behavior: Misperceptions of feedback in a Dynamic Decision-Making Experiment," *Management Science*, Vol. 35, No.3, 1989, 321-339.

● 저자소개 ●



김동환

1985년 고려대 경영학과 학사
1987년 고려대 행정학과 석사
1991년 고려대 행정학과 박사
현재 경기대, 대전산업대 강사
전자통신연구소 기술경제연구부 초빙연구원
관심분야: 시스템 다이나믹스, 시스템 시뮬레이션, 신경망, 혼돈이론, 게임이론, 공공선택론, 정책 및 의사결정론, 객체지향 프로그래밍(Small-talk).