객체 상태 기반 실시간 객체지향 시뮬레이션

이태동[0] 전범재 임강희 박상문 정창성
고려대학교 전자공학과
{lyadlove, bomjae, ikh, sangmoon}@snoopy.korea.ac.kr
csjeong@charlie.korea.ac.kr

Object State Based Real Time Object-Oriented Simulation

Tae-Dong Lee[0] Bom-Jae Jeon Kang-Hee Im Sang-Moon Park Chang-Sung Jeong
Dept. of Electronics Engineering, Korea University

요 약

시뮬레이션을 구현할 때 대부분의 경우 eventlist라는 자료구조를 사용하여 사건(event)를 처리하고 시간을 스케쥴링(scheduling) 한다. 그러나 eventlist를 사용하는 것은 객체 스스로가 사건을 처리하고 시간 스케쥴링을 하지 않는다는 점에서 객체지향적이지는 못하다. 그래서 본 논문에서는 객체 스스로가 상태를 가지며 사건을 처리하고 시간을 스케쥴링하는 전차대 전차, 전차대 헬기 교전을 위한 시뮬레이션을 설계 및 구현하였다. 설계는 상태변화를 쉽게 하고 시간 전진 문제를 쉽게 처리할수 있는 Façade Pattern방법을 사용하였으며. 시간전진 방법은 상태변화에는 논리시간(logical time)을 이용한 이산사건(discrete event) 전진방법을 사용하였고 Graphic Visualization에는 실시간(real time)을 이용한 이산(discrete) 전진방법을 사용하여 논리시간과 실시간을 병렬로 동기화 시켜 처리 하였다. 구현은 Visual C++ 의 MFC 라이브러리를 사용한 MDI구조로 구현하였다. 논문의 시뮬레이션은 교전모델을 응용하였고 객체지향(Objected-Oriented)으로 설계 및 구현되어 각 객체의 재사용과 확장 및 수정이 용이하다는 장점을 가진다.

1. 서론

현재 시뮬레이션 분야는 게임분야, 반도체 생산라인, 통신분야, 우주항공분야, 군사분야등 광범위 하게 사용되어지고 있다. 어떤 시스템을 시뮬레이션 하는 주요 이유는 여러 입력과 디자인 파라미터 값을 이용하여 동작하는 시스템의 예상된 반응을 분석학적인 표현으로 제공할 수 없기 때문이다[1]. 특히 군사분야에서는 인명 피해없이 시뮬레이션을 통해 결과를 얻어 분석할 수 있다는 큰 장점을 가진다.

먼저 2장에서는 시뮬레이션이 사건을 처리하기 위해서 사용하는 eventlist 에 대해서 설명하며 이를 사용할 때의 문제점을 설명하고 시간 관련 설명을 기술한다.

3장에서는 본 논문에서 적용하는 디자인 패턴으로 Façade Pattern을 소개한 후 설계과정에서의 적용을 보인다. 4장에서는 객체의 상태 종류를 설명하고 객체가 현재 상태에서 다음 상태로 어떻게 바꾸며 시간 스케쥴링을 어떻게 하는지를 보인 후 시간 전진 방법을 텍스트와 그래픽 모드로 기술한다. 5장에서는 구현을 기술하도록 하고, 끝으로 6장에서는 결론과 향후 발전방향에 대하여 기술하도록 하겠다.

2. 관련연구

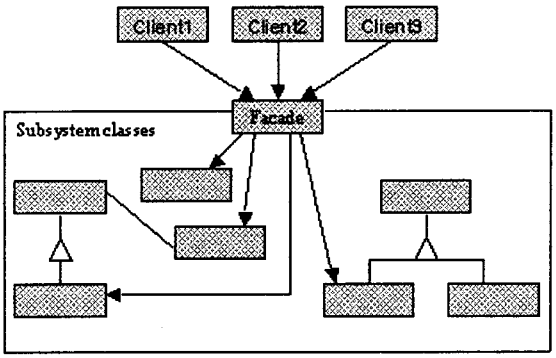
기존에 시뮬레이션은 사건을 처리하기 위해 eventlist라는 자료구조를 가지고 가장 작은 시간을 가진 사건을 꺼내서 처리하고 이에 따른 다음 사건이 있으면 다시 eventlist에 넣어 시간을 스케쥴링 하는 과정을 반복함으로 해서 시뮬레이션을 진행하였다. eventlist는 각각의 사건 종류가 발생할 다음 사건과 시간을 가지는 자료구조이며 시간과 사건의 형태들의 튜플(tuple)들의 Set 이다[2]. 그러나 이런 방법은 객체 스스로 사건을 처리해야 하고 시간을 스케쥴링해야 한다는 객체지향적인 면과는 상반된다. 따라서 좀 더 객체지향적인 면을 가질 필요가 있으며 객체 스스로가 사건을 처리하고 시간을 스케쥴링 하는 시뮬레이션 방법이 필요하다.

시뮬레이션에서 중요한 요소들 중 하나로 시간이 있다. 시간 종류로는 논리시간과 실시간이 대표적이다. 논리시간은 각 객체들이 가지고 진행되는 시간을 말하며, 실시간이란 실세계에서 진행되는 시간을 말한다. 또한 시간을 진행시키는 방법으로 이산 전진 방법과 이산 전진 방법이 있다. 이산 시간 전진은 같은 시간 단위 간격으로 일정하게 진행하는 것을 말하며 이산사건 시간 전진은 사건이 일어날 때마다 시간을 전진 시키는 것을 의미한다.

3. 객체지향 시뮬레이션 설계

3.1 Facade Pattern

Facade Pattern은 Structural Pattern의 하나로 사용하는
목적으로는 하나의 객체(Object)가 다른 객체의 Set에 접근할 때 이를 좀더 편하기 하기 위해서 이다. 또한 복잡성(Complexity)와 의존성(Dependency)을 줄인다[3].



[그림 1] Facade Pattern Structure

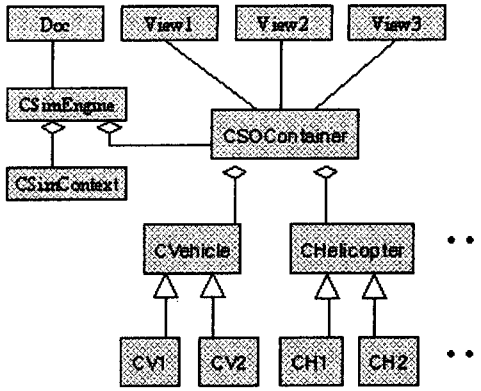
3.2 객체지향 시뮬레이션 설계

시뮬레이션을 설계하는데 있어서 가장 중요한 부분이 엔진이 될 것이다. 시뮬레이션 엔진은 효과적인 시뮬레이션 모델의 구조에 결정적인 기능을 지원하며 여러 컴포넌트(Component)들의 집합이다. 게다가 시뮬레이션 엔진은 더 큰 응용프로그램의 서브컴포넌트로 사용되어 질 수 있다. 좋은 시뮬레이션 엔진이란 모델 개발자들에게 시스템의 모델을 이해하기 쉽고 재사용이 쉽도록 제공하는 효율성과 기능을 가지는 것이다[4]. [그림 2] 에서 엔진 부분은 Façade인 CSOContainer 부분과 서브 클래스들을 포함한 부분이 되며 엔진 구동을 CSimEn gine 클래스가 하게 된다. CSimEngine 은 CSOContainer 를 집합관계(Aggregation)로 관계를 가지고 있다. 또한 CSOContainer도 각 객체를 집합관계로 관계를 가지게 되어 서로 유기적으로 동작을 하게 된다.

논문의 구현은 MDI 구조로 되어있다. 이는 여러 뷰(View)를 가지게 되는 것으로 각 뷰는 각 객체에 접근하여 Graphic Visualization 시켜야 한다. 그러므로 먼저 서술한 Façade Pattern을 사용하여 각 뷰가 Façade에게 접근하도록 하여 각 객체의 정보를 얻어 디스플레이 시킨다면 효율적일 것이다.

[그림 2] 에서 보듯이 CSOContainer 클래스가 위에서 서술한 Façade를 대행하는 역할을 하게 된다. 따라서 각 뷰들은 CSOContainer 클래스를 거쳐 서브시스템의 모든 객체들의 정보를 가져와 정보를 Visualization하면 된다. 또한 서브시스템에 마음대로 다른 객체를 확장 할 수 있으며 수정이 용이하다.

[그림 2] 에서 보듯이 각 MDI 의 View 는 CSOContainer를 통해 전차와 헬기의 정보를 가져와 Graphic Visualization 하게 된며 전차와 헬기는 스스로 상태를 변화시키면서 시간을 스케쥴링하게 된다.



[그림 2] 시뮬레이션 설계

4. 실시간 시뮬레이션 설계

4.1 객체 상태 종류

[표 1] 에서 보듯 각 객체는 Ready, aQuire, Allocate, Swing, Lock_on, Fire, rEload 라는 객체 상태를 가지고 있으며 시간 전진에 따라 객체의 상태를 변화하며 시뮬레이션이 진행하게 된다. Ready 는 준비상태, aQuire 는 탐색, Allocate는 표적할당, Swing 은 포구를 전환, Lock_on 은 정조준, Fire는 사격, rEload는 재장전을 나타낸다.

[표 1] 에서 O, X 는 객체가 해당하는 상태를 가지는지 못가지는지를 표시하였다.

[표 1] 객체의 상태종류

상 태	R	Q	A	S	L	F	E
정찰헬기	O	O	O	X	X	X	X
공격헬기	O	O	O	O	O	O	O
지휘전차	O	O	O	O	O	O	O
전투전차	O	O	O	O	O	O	O

4.2 상태 스케쥴링

어떤 객체가 자신이 가장 작은 시간을 가진 것을 알게 되면 객체는 자신의 현재상태에 맞는 행동(Action)을 취하고 다음 상태로 스케쥴링한다. 또한 다음 시간을 스케쥴링하게 된다.

4.3 시간 전진 방법

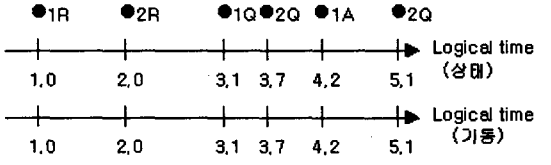
시간 전진 방법은 상태변화를 위한 시간 전진과 3D Graphic Visualization 을 위한 시간전진 방법을 병렬로 처리하게 된다. 먼저 상태변화를 위한 시간전진은 논리시간을 이용한 이산사건 전진방법을 사용하였고 Graphic Visualization을 위한 시간전진은 디스플레이 기능이 필요 없으면 이산사건 시간 전진 방법을 사용하였고 디스플레이 기능이 필요할 때는 실시간을 이용한 이산 시간 전진 방법을 사용하였다. 이를 하나의 시간축상에서 보인 것이 [그림 3] 이 된다. 앞의 숫자는 객체의 번호가 되고 뒤의 문자는 논리시간에서의 상태가 된다.

4.3.1 텍스트 모드(Text Mode)

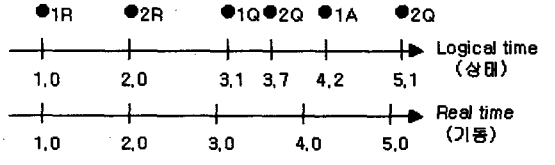
실시간으로 디스플레이 하지 않으며 상태가 처리되는 순간에 기동도 처리되도록 한다. [그림 3]의 (a)가 해당 그림으로 이산사건 처리가 된다.

4.3.2 그래픽 모드(Graphic Mode)

실시간으로 디스플레이한다. 기동이 discrete하게 처리되면서 기동시간 사이의 상태를 처리하도록 하였다. 예를 들어 [그림 3]에서 (a)는 상태가 처리되는 시간에 기동도 같이 처리할 수 있지만 (b)는 사람의 눈으로 보면서 처리해야 하므로 실시간과 논리시간의 동기화가 필요하게 된다. 예를 들어 기동시간 3.0에서 4.0으로 실시간으로 진행되면 상태 시간인 3.1 단위시간과 3.7 단위시간의 상태가 처리되어진다. 이를 그래픽모드라고 부른다.



(a) 텍스트 모드 일 때 (이산사건 시간진행)



(b) 그래픽 모드 일 때 (이산 시간진행)

[그림 3] 시간 진행

[그림 3] (b)의 동기화 알고리즘은 다음과 같다.

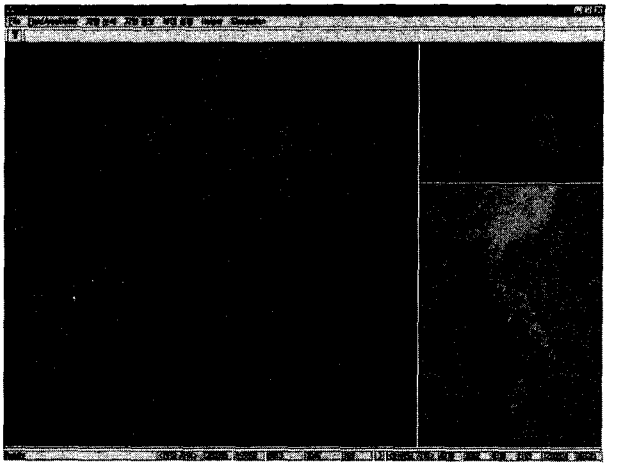
```
If(get_current_simulation_time() < get_simulation_period_time()
{
   while(get_current_simulation_time() <= get_advance_time())
   {
     // 가장 작은 시뮬레이션 시간을 가진 객체를 찾음
     // 가장 작은 시뮬레이션 시간을 가진 객체의 행동
   }
   set_advance_time(get_advance_time()+ move_time_interval)
   visual_update_all_object_position()
}
```

위의 알고리즘에서는 먼저 현재의 시뮬레이션 시간과 시뮬레이션의 종료 값을 비교해서 현재의 시뮬레이션 시간이 작다면 move_time_interval 값만큼 계속적으로 시뮬레이션을 진행시키는 알고리즘이다. 이 때 move_time_interval 시간 사이의 상태를 모두 처리하고 다음 시간을 스케쥴링하고 객체의 위치 정보를 가져와서 Graphic Visualization 하게 된다.

5. 실시간 객체지향 시뮬레이션 구현

시뮬레이션 구현은 Visual C++의 MFC 라이브러리를 사용하여 MDI 구조로 만들었다. View는 3개로 구성되며 전차와 전차, 전차와 헬기들이 시작지점에서 목표지점으로 이동하면서 교전을 벌리게 되며 시뮬레이션이 끝난 후 결과를 나타나게 된다.

[그림 6] 에서 오른쪽 위의 창이 객체의 눈으로 바라본 창이 되고 오른쪽 밑의 창은 한국지도를 보이며 왼쪽 창은 한국 지형을 10km * 10km 로 띄운 위에 객체의 기동을 시간에 따라서 Visualization 하는 모습이 된다. 객체는 스스로 상태를 변화시키며 시간을 스스로 스케쥴링하게 된다.



[그림 5] 시뮬레이션 구현

6. 결론 및 향후 발전 방향

시뮬레이션이 사건을 처리하기 위한 방법으로 eventlist라는 자료구조를 사용하는 것은 객체 스스로 사건을 처리하고 시간을 스케쥴링해야 한다는 객체지향적인 면에서는 좋은 방법이 아니다. 따라서 본 논문에서는 객체의 상태를 이용하여 스스로 사건을 처리하고 시간을 스케쥴링 하는 객체지향 실시간 시뮬레이션을 설계하고 구현하여 객체지향적인 면에서 훨씬 나은 방법이라는 것을 보였다. 디자인으로 Façade Pattern을 사용하였고 이 디자인 패턴은 객체지향 시뮬레이션을 효율적으로 설계 및 구현하였다. 구현상의 시간 전진 방법은 실시간으로 디스플레이를 하는가의 여부에 따라서 달리하였다. 시간 전진 방법에 있어서 논리시간과 실시간을 동기화 시키는 방법을 사용하여 3D Graphic Visualization 까지 구현하였다.

본 논문은 한대의 PC 에서 실행되는 것을 보였다. 따라서 향후 발전 방향으로는 분산구조의 구현이나 또는 기존에 완성된 분산구조를 가지고 본 논문이 제시한 디자인과 구현 방법이 좋은 방법이라는 것을 증명하는 것이 필요하다.

7. 참고 문헌

[1]. Wayne J. Davis, " Simulation: Technologies in the New Millennium" . Proceedings of the 1999 Winter Simulation Conference.

[2]. Jayadev Misra, " Distributed Discrete -Event Simulation" . ACM Computing Surveys, Vol. 18(1) pages 39-65, 1986.

[3]. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, " Design Patterns" . Addison-Wesley, 1994.

[4]. Herb Schwetman, " CSIM18" . Proceedings of the 1996 Winter Simulation Conference.