컴퓨터 게임을 위한 실시간 Timewarp Rigid body 시뮬레이션

민성환 김창헌
고려대학교 영상정보처리 협동과정 고려대학교 컴퓨터학과
{shmin, chkim}@cgvr.korea.ac.kr

Real-Time Rigid body Simulation By Using Timewarp for Computer Game

Sung-Hwan Min
Dept. of Visual Information Process
Korea University

Chang-Hun Kim
Dept. of Computer Science & Engineering,
Korea University

요 약

본 논문은 많은 수의 rigid body 물체들을 물리학에 기반하여 실시간으로 애니메이션하는 방법을 제안한다. rigid body 물체들의 움직임을 생성해내는 과정은 상당한 시간이 소요되며 또한 물체의 수가 증가함에 따라 계산시간이 급증한다. 본 논문에서는 Timewarp rigid body 시뮬레이션 알고리즘을 실시간 애니메이션에 적용하기 위해 시간 당 생성되는 프레임 수에 따른 다단계 롤백 범위 적용을 하는 방법을 제안하고 실험을 통하여 시뮬레이션 시스템의 효율성을 보인다.

1. 서론

컴퓨터 게임에서는 최근 캐릭터들의 사실적인 움직임들을 표현하는 것이 중요시되고 있다.

지금까지는 대부분의 게임에서 캐릭터의 움직임 생성은 사람이 수작업으로 일일이 지정하는 방식이나 키프레임을 이용한 보간 방식을 주로 사용하여 왔다. 전자의 경우, 사실적인 움직임의 생성은 가능하지만 작업량이 매우 많다는 단점이 있고, 후자는 작업량은 적지만 움직임이 그다지 사실적이지 못하다는 단점을 지니고 있다. 그래서 적은 작업량만으로 사실적인 애니메이션을 제작하기 위한 대안으로 제시된 것이 물리기반의 애니메이션 제작방식이다. 이 방식은 물체의 초기 위치와 마지막 위치 등과 같은 적은 양의 정보만을 설정한 후, 물리학적인 요소(마찰, 중력 등)들을 움직임에 적용하여 자동으로 애니메이션을 제작한다. 하지만 대부분의 경우 알고리즘이 복잡하여 프레임 당 생성시간이 길고, 결국 제작에 소요되는 시간이 많아진다는 것이 단점으로 지적되어 왔다.

그리고 게임에서 등장하는 대부분의 물체나 캐릭터들은 rigid body 또는 rigid body로 간주할 수 있는 경우가 많다. 따라서 자연스럽고 작업이 효율적인 rigid body 시뮬레이션하여 애니메이션하는 것은 넓은 범위에 걸쳐 적용될 수 있다.

지난 몇 년간 rigid body 시뮬레이션을 고려한 다수의 방법들이 제안되어 왔다. Milenkovic[2]는 회전하지 않는 다수의 다면체와 구를 시뮬레이션하는 방법을 제안하였고, Brogan[3]은 분산 환경에서 많은 수의 가축들을 시뮬레이션하였다. Miritich[5]는 물리 기반의 rigid body 애니메이션 생성에 걸리는 시간을 단축하기 위하여 Jefferson이 85년도에 제안한 Timewarp 알고리즘을 rigid body에 적용하는 방식을 제안하였다. 이 알고리즘은 다수의 rigid body 애니메이션 제작에 걸리는 시간을 크게 감소시켰다. 그러나 Timewarp rigid body 시뮬레이션은 다수의 물체의 움직임을 빠르게 제작할 수 있는 방법이었으나 실시간을 위한 것은 아니었다.

본 논문은 컴퓨터 게임과 같은 실시간 애니메이션

을 요구하는 시스템에서 캐릭터의 사실적인 움직임을 간단한 작업량으로 생성하는 방법을 제안한다. 본 논문에서 제안하는 방법은 기존의 Timewarp rigid body 시뮬레이션을 개선하여 실시간 애니메이션 생성이 가능하게 한다.

2. Timewarp rigid body 시뮬레이션

기존의 rigid body 시뮬레이션 알고리즘은 Retroactive Detection 방법과 Conservative Advancement방법 두 가지로 나눌 수 있다. Retroactive Detection 방법은 충돌과 같은 이벤트가 일어나면 그 시점에서 모든 물체가 다시 각자의 움직임을 재계산한다. 이에 따라 이벤트가 많아질수록 계산량이 커지게 된다. 반면 Conservative Advancement 방법은 일정 시간 뒤의 충돌 이벤트 발생을 미리 검사한 후 애니메이션을 진행하므로 전체 물체의 움직임을 재계산 하는 경우는 없다. 그러나 이벤트의 수가 늘게 되면 검사하는 시간 간격이 잦아져 계산 시간이 급증하게 된다.

Timewarp rigid body 시뮬레이션은 이벤트가 일어나면 그 이벤트에 영향받을 가능성이 있는 몇몇 물체들의 움직임만을 재계산함으로써 프레임 당 평균 계산시간을 대폭 줄일 수 있다. 이 방식은 각각의 rigid body마다 Local virtual time(LVT)을 두고 이에 따라 움직임을 생성한다. 이벤트가 발생하면 영향을 받는 rigid body들만이 이벤트가 발생한 시점으로 롤백하게 된다. 어떤 물체가 롤백되면 그 물체의 영향을 받았던 다른 물체들도 재귀적으로 롤백을 수행하게 된다.

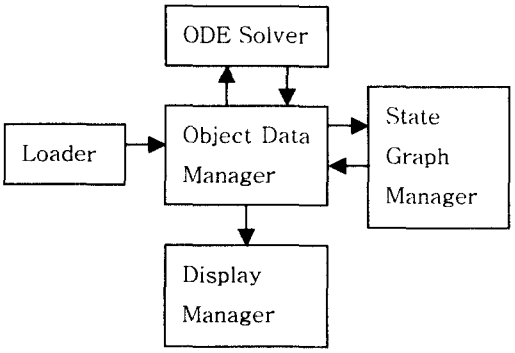
모든 rigid body들의 LVT 중에서 가장 느린 LVT값이 Global Virtual Time(GVT)로 정해지며 각 물체마다 GVT와 LVT사이의 해당 물체가 지나간 Bounding Volume을 생성하여 충돌검사와 같은 이벤트 계산에 사용된다.

3. Real-Time Timewarp Rigid Body 시뮬레이션

본 논문에서는 기존의 Timewarp rigid body 시뮬레이션을 실시간 애니메이션에 적합하도록 개선한 시뮬레이터 시스템을 제안한다

3.0 시스템 구성

본 논문에서 제안한 방법을 사용하여 구성한 샘플 시스템의 구성도를 <그림1>에 나타내었다. 물체를 불러오는 Loader와 물체의 다음 프레임 위치 계산을 담당해주는 ODE Solver, 전체 rigid body 데이터를 담당하는 Object Data Manager, 충돌에 의한 롤백에 필요한 정보를 관리하는 State Graph Manager, 실제 데이터를 화면에 디스플레이 하는 Display Manager로 구성되어 있다.



<그림.1 샘플 시스템 구성도>

3.1 Global Virtual Time을 이용

Timewarp 알고리즘은 각 물체의 움직임이 LVT를 따라서 생성된다. LVT는 각각의 물체마다 움직임이 생성되어 있는 가장 마지막 시간을 의미한다. 그러므로 애니메이션이 가능하기 위해서는 모든 물체의 움직임이 생성되어 있는 가장 낮은 LVT, 다시 말해서 GVT를 기준으로 애니메이션을 생성하여야 한다.

3.2 디스플레이 시점

실시간으로 애니메이션을 하기 위해서는 해당 프레임의 물체들의 위치와 방향 데이터를 생성해두고 그 데이터에 따라 화면에 물체의 위치를 그려주어야 한다.

Timewarp 알고리즘을 적용하게 되면 각각의 LVT를 기준으로 움직임을 생성해 나가기 때문에 어떠한 시점에 디스플레이하고 다음 프레임으로 넘어가야 하는지 기준이 모호하다. 본 논문에서는 그 기준을 예상되는 현 프레임의 디스플레이 타임 (이전 프레임 디스플레이 타임 + 정해진 프레임 시간간격)로 정하고 GVT가 그 시간을 넘어서는 순간 데이터 생성을 중지하고 프레임을 디스플레이 한다.

3.3 고정된 물체에 대한 처리

고정된 물체는 물리학 법칙에 직접적인 영향을 받지는 않는다. 그러나 전체 시뮬레이션에서 제외 시켜서는 안 된다. 왜냐하면 다른 움직이는 물체들과의 충돌이 일어나 물체들의 움직임에 영향 줄 수 있기 때문이다.

샘플 시스템에서는 움직이는 물체와 마찬가지의 속성을 고정된 물체에도 적용하여 다른 물체와의 충돌 검사와 그에 따른 반발력을 계산하여 각 물체에 적용시

켰다. 단지 고정된 물체는 위치 데이터는 변화 시키지 않도록 하였다.

3.4 다수의 충돌 발생시 롤백 범위를 한정

Timewarp 알고리즘을 적용하여 기존의 방법들보다 많은 속도 개선을 얻어 냈으나 많은 수의 이벤트가 발생하는 시점에서는 계산시간이 급증하여 느려지는 현상이 발생하였다. 이를 해결하기 위해서 계산 시간에 따라 롤백의 범위를 다단계로 적용하였다

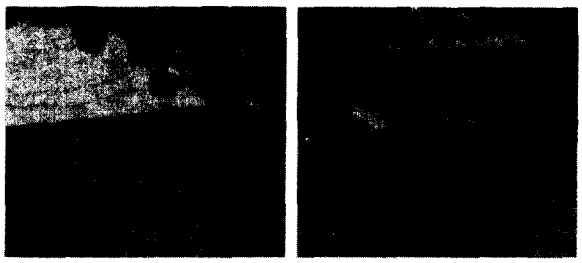
.즉, 프레임의 계산 시간이 한 프레임에 할당된 시간보다 높을 때는 낮은 단계의 롤백 범위를 적용하여 전체 롤백의 횟수를 줄여 계산량을 단축하였고 이것으로 인하여 보다 부드러운 움직임을 얻어냈다.

4. 실험 및 결과

본 논문에서 구현한 시스템은 펜티엄 III 866Mhz, 512MB RAM, 하드웨어 가속이 되는 비디오 카드를 갖춘 Windows 2000 OS의 PC에서 실험하였으며, OpenGL를 이용하여 구현하였다.

<그림2>는 30개의 물체를 자유낙하 시킨 애니메이션의 정지 영상이다. 20초 분량의 데이터를 실시간으로 생성하면서 디스플레이 하였으며 초당 생성된 프레임 수는 최소 15으로 설정 하였다

<표 1>은 기존의 Timewarp 알고리즘을 적용하여 애니메이션 데이터를 생성하였을 때와 본 논문에서 제안한 변형된 알고리즘을 사용하였을 때의 계산 시간을 비교한 것이다. 실험 결과 본 논문에서 제시한 방법은 전체 시간을 단축하였을 뿐 아니라 실시간으로 애니메이션이 가능하다.



<그림 2> 실험 시뮬레이션 정지 영상

	20개의 물체	50개의 물체
TW	2.5 sec	6.1 sec
Real-time TW	2.1 sec	5.2 sec

<표 1> 실험 결과의 계산 시간 비교

5. 결론 및 향후 연구

본 논문에서는 Timewarp rigid body 시뮬레이션 알고리즘을 단일 프로세스 환경에서 실시간 시뮬레이션에 적용하는 방법을 제안하였고 시뮬레이션 시스템을 구성하였다. 또한 보다 부드러운 시뮬레이션을 위해 계산시간에 따라 롤백 적용범위를 다단계로 제한하는 방법을 제안하였다.

향후 연구로는 현재 시스템을 확장하여 articulated body에 적합하도록 변형하고 이에 따라 알고리즘을 개선하고 멀티 프로세서 환경에 적합한 시스템으로 전환하는 연구방향을 모색하고 있다.

6.참고문헌

[1] Matthew Moore, "Collision Detection and Response for Computer Animation," SIGGRAPH 88 Conference Proceedings
[2] Victor J. Milenkovic. "Position-Based Physics : Simulating the Motion of Many Highly Interacting Spheres and Polyhedra," SIGGRAPH 96 Conference Proceedings, pp. 129-136, August 1996
[3] David C. Brogan, Ronald A. Metoyer, and Jessica K. Hodgins, "Dynamically Simulated Characters in Virtual Environments," IEEE Computer Graphics and Applications, 18(5):58-69, September 1998
[4] Stephen Chenney, Jeffrey Ichnowski, and David Forsyth., "Dynamics Modeling and Culling." IEEE Computer Graphics and Application, 19(2) 79-87, March/April 1999
[5] Brian Miritich, "Timewarp Rigid Body Simulation," SIGGRAPH 2000 Conference Proceedings