컴퓨터 게임을 위한 물리 엔진의 성능 향상 및 이를 적용한 지능적인 게임 캐릭터에 관한 연구

최 종 화[†] · 신 동 규[††] · 신 동 일[†††]

요 약

이 논문은 컴퓨터 게임을 위한 물리 엔진의 성능 향상 및 이를 적용한 지능적인 게임 캐릭터에 관한 연구를 서술한다. 물리적 상황을 자동으로 인식하는 알고리즘으로는 Momentum back-propagation을 적용하였다. 또한 우리는 각 상황에 따른 적분 방식의 실험 결과를 제시한다. 실험을 위하여 Euler Method, Improved Euler Method, 및 Runge-kutta Method의 세 가지의 적분 방식을 적용하였다. 각 적분 방식의 실험 결과에서 충돌이 없는 상황에서는 Euler Method가 최적의 성능을 보여주었다. 또한 충돌 상황에서는 세 가지 방식이 모두 비슷한 성능을 보여주었지만, Runge-kutta Method가 최적의 정확도를 보여주었다. 물리 상황인식에 대한 실험결과에서는 입력 층과 출력 층이 고정된 상태에서 은닉 층이 3일 때 가장 좋은 성능을 보여주었고, 또한 학습횟수가 30000일 때 최적의 성능을 보여주었다. 앞으로 우리는 다른 장르의 게임에 이러한 물리적 컨텍스트(context)를 인식하는 연구를 진행할 것이며 또한 전체 게임의 성능을 증가할 수 있도록 M-BP이외의 인식 알고리즘을 적용할 것이다.

키워드 : 물리엔진, 인공지능, 인공지능 게임 캐릭터

Research on Intelligent Game Character through Performance Enhancements of Physics Engine in Computer Games

Jonghwa Choi[†] · Dongkyoo Shin[††] · Dongil Shin[†††]

ABSTRACT

This paper describes research on intelligent game character through performance enhancements of physics engine in computer games. The algorithm that recognizes the physics situation uses momentum back-propagation neural networks. Also, we present an experiment and its results, integration methods that display optimum performance based on the physics situation. In this experiment on integration methods, the Euler method was shown to produce the best results in terms of fps in a simulation environment with collision detection. Simulation with collision detection was shown similar fps for all three methods and the Runge-kutta method was shown the greatest accuracy. In the experiment on physics situation recognition, a physics situation recognition algorithm where the number of input layers (number of physical parameters) and output layers (destruction value for the master car) is fixed has shown the best performance when the number of hidden layers is 3 and the learning count number is 30,000. Since we tested with rigid bodies only, we are currently studying efficient physics situation recognition for soft body objects.

Key Words : Physics Engine, Artificial Intelligence, Intelligence Game Character

1. 서 론

현재 3차원 게임에 있어서 가장 중요한 이슈는 게임 속 세계에 존재하는 모든 캐릭터들이 사실감과 생동감을 유지하면서 프로그램의 속도 감소 문제를 발생시키지 않는 방안을 찾는 것이다. 게임 속 캐릭터의 사실감이란 캐릭터의 움직임이 현실의 물리현상과 같은 현상을 나타내는 것을 말하는 것이며 게임 물리엔진의 적용이 그 해결책으로 제시되고 있다[1]. 그러나 물리엔진만이 적용된 게임은 캐릭터의 사실적인 움직임만을 표현할 뿐 캐릭터에게 생명력을 심어주지는 못한다. 이 논문에서는 물리엔진이 적용된 게임에서 주인공 캐릭터가 행동할 때 발생하는 물리적 요소를 지능적 캐릭터가 인식하여 자신의 행동을 스스로 결정하는 살아있는 캐릭터 생성을 위한 연구를 제시하고자 한다. 이러한 연구는 게임 인공지능의 한 분야하고 할 수 있다. 하지만 지

※ 본 연구는 서울시 산학연 협력사업(Seoul R&BD Program)에 의하여 이루어진 것임.
† 준 회 원 : 세종대학교 컴퓨터공학부 박사과정
†† 종신회원 : 세종대학교 컴퓨터공학과 부교수
††† 종신회원 : 세종대학교 컴퓨터공학과 조교수(교신저자)
논문접수 : 2005년 9월 12일, 심사완료 : 2006년 1월 10일

금까지의 게임 인공지능의 대부분의 연구는 캐릭터가 게임 속에서 어떻게 행동해야 하는지에 대한 자신만의 규약을 가지고 행동하는 양식으로 진행되었다[2].

이 논문에서 각각의 캐릭터가 주인공 캐릭터의 물리적 요소를 인식하기 위한 알고리즘으로는 momentum back-propagation NNs[3]이 사용되었다. 하지만 게임 속에서의 물리현상에 대한 패턴 인식 알고리즘의 적용은 게임의 성능에 저하를 가져온다. 이에 대한 해결방안으로 물리엔진에서의 상황에 따른 최적의 적분기를 사용하여 물리 엔진의 성능을 향상하는 방식으로 게임에서의 반응속도를 최적화한다[4].

2장에서는 게임에서의 물리엔진에 관한 연구와 게임 인공지능에 관하여 지금까지 진행되었던 연구를 살펴보고, 3장에서는 이 논문에서 제시하는 컴포넌트 아키텍처를 제시한다. 4장에서는 컴포넌트 아키텍처에서 제시된 각각의 컴포넌트에 대한 세부설명을 한다. 5장에서는 상황에 따라 변화하는 물리요소를 입력 받아서 현재의 물리현상을 결정하는 알고리즘의 성능에 대한 평가를 제시한다. 6장에서는 이 논문의 결론은 맺는다.

2. 관련연구

현재 물리엔진은 상용엔진과 오픈 소스 기반의 엔진으로 제공되고 있으며 실제 게임에서는 성능상의 문제로 인해 제한적으로 적용되고 있다. Mach Engine[5], Havok[6], 그리고 Meqon[7]은 상업적으로 폭넓게 사용되고 있는 물리엔진이다. 또한 ODE(Open Dynamics Engine)는 잘 알려진 오픈 소스 기반의 물리엔진이다 [8]. 이들 엔진은 물리적 계산에서의 효과적인 성능 제공을 위하여 정확도와 속도에서의 적절한 분배를 기반으로 구현되어 있다.

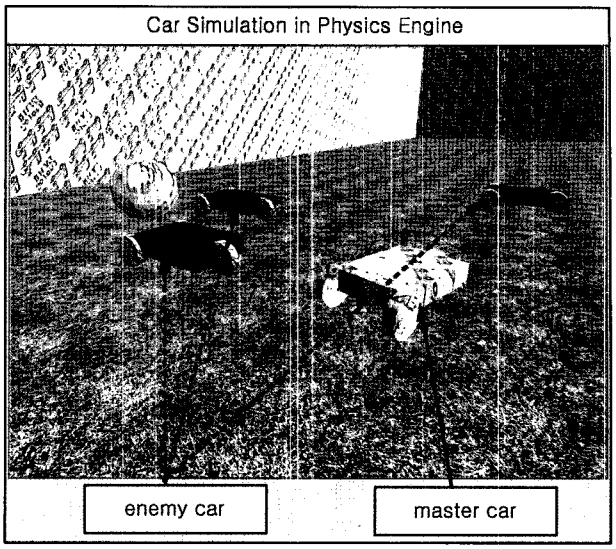
성능 개선을 위한 연구로써 실제 세계에서의 물리적 문제에 대한 효율적인 모델에 관한 연구가 진행되었고, 엔진구조에 대한 효율적인 알고리즘에 대하여는 [1]에서 제시하였다. 충돌 상황에서의 속도 개선을 위한 연구도 진행 되었는데, Voxel의 효과적인 구조 제시가 이에 해당된다 [9]. 또한 제한된 Boxtree로 충돌 감지에 대한 성능 개선 알고리즘이 연구되었다[10].

전반적으로 게임에서 물리엔진을 적용하여 제시된 인공지능 엔진에 관한 연구는 미비한 실정이다. 학습 방법을 적용한 지능적 캐릭터에 관한 연구는 실제 게임에서의 적용이 부분적으로 이루어질 수 있는 좋은 모델이 된다[11]. 이 논문은 물리엔진을 적용한 게임에서 물리적 수치에 대한 학습을 통하여 지능적으로 동작하는 게임 캐릭터에 대한 모델을 제시한다.

3. 게임에서의 자동적인 물리적 상황 인식을 위한 컴포넌트 아키텍처

3.1 프로그램 시나리오

이 논문에서는 자동적인 물리적 상황 인식 중 자동차를

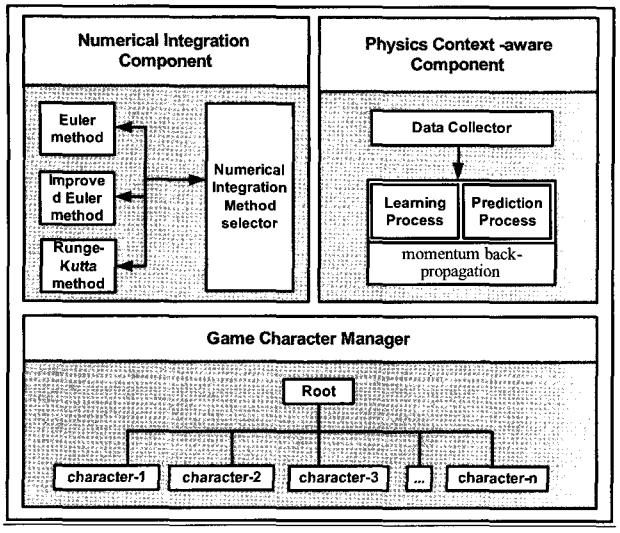


(그림 1) 물리엔진을 적용한 자동차 시뮬레이션

게임에 대해서 아키텍처를 제시하고 그에 대한 실험을 한다. 자동차 게임에서는 하나의 주인공 자동차와 주인공 자동차를 공격하는 많은 수의 적 자동차들이 존재한다. 주인공 자동차는 움직일 때 마다 변화하는 물리적 요소가 발생한다. 적 자동차는 주인공 자동차의 변화된 물리적 요소를 인식하여 주인공 자동차에게 가장 많은 손실을 줄수 있는 순간을 파악하고 주인공 자동차를 공격한다. 현재까지의 게임에서는 적 자동차들이 약속된 행동 및 주인공 자동차의 위치에 따른 행동을 할 뿐 주인공 자동차의 변화하는 물리적 요소를 파악하여 지능적으로 행동하는 적자동차에 관한 연구는 상당히 미비한 실정이다.

3.2 컴포넌트 아키텍처

(그림 2)는 게임에서의 자동적인 물리적 상황 인식을 위한 컴포넌트 아키텍처다. 컴포넌트 아키텍처는 3가지의 컴포넌트를 포함하며 각각의 컴포넌트는 서로 유기적인 관계를 맺고 있다. Physics Context-aware Component는 주인공 자동차가 동작할 때 발생하는 물리적 컨텍스트를 분석하여 적 자동차와 주인공 자동차가 충돌했을 때 일어날수 있는 상황을 예측한다. Physics Context-aware Component에서는 먼저 주인공 자동차와 적 자동차와의 여러 충돌 실험을 통해서 주인공 자동차의 손상수치를 학습한 후 그에 대한 학습 결과를 저장한다. 저장된 학습 결과를 바탕으로 실제 게임에서 각 상황에 따라 주인공 자동차가 움직일 때마다 적 자동차들은 최대의 피해를 줄수 있는 상황을 파악하여 행동하게 된다. 물리엔진에서는 모든 개체의 움직임을 게임에 적용할 때 Integration Method를 적용하여 개체의 움직임을 계산하는데 주로 Euler method, improved Euler method, Runge-Kutta method 중 하나가 사용된다. Numerical Integration Component에서는 주인공 자동차와 적 자동차 간의 상황에 따라서 최적의 성능을 발휘하는 Integration method를 적용한다. 실제 게임에서 물리적 상황을 인식하는



(그림 2) 컴포넌트 아키텍처

기능을 포함하면 게임전체의 성능 저하를 초래하는데 Numerical Integration Component는 이에 대한 성능감소 부분을 보완하는 역할을 담당한다.

Game Character Manager에서는 적 자동차 리스트 중에 주인공 자동차와 가장 근거리에 있는 적 자동차를 선택하여 적 자동차의 현재의 물리적 요소를 Physics Context-aware Component에게 전송하는 역할을 담당하고 그 결과값[주인공 자동차의 피해수치]을 적 자동차에게 다시 전달해 주는 역할을 담당한다.

4. 내부 컴포넌트 구조

4.1 Physics Context-aware Component

4.1.1 물리 컨텍스트 정의

Physics Context-aware Component는 적 자동차가 주인공 자동차의 움직임을 파악하여 주인공 자동차에게 최대의 피해를 줄수 있는 상황을 판단하여주는 역할을 담당한다. 이 논문에서는 상황 판단을 위하여 M-BP알고리즘을 적용하여 상황을 학습하고 판단한다. <표 1>은 상황판단을 위한 주인공과 적 자동차에서 추출되는 물리적 컨텍스트를 정의한다. 테이블1에 정의된 물리적 컨텍스트는 물리상황인식 알고리즘의 입력 값으로 적용된다.

자동차 시뮬레이션에서 자동차가 움직일 때 마다 변화는 물리적 요소는 Position, Linear Velocity, Variable Velocity가 있고 자동차간의 충돌시 충돌반응에 영향을 미치는 요소는 Mass, Center of Mass가 있다. <표 1>에서 제시된 5가지의 컨텍스트는 자동차의 충돌에 있어서 영향을 미치는 요소이며, 물리엔진에서 추출 가능한 요소이기도 하다. 우리는 <표 1>에서 제시된 5가지의 충돌에 관여하는 요소를 기반으로 두 차의 충돌 시 발생하는 충돌 값을 기반으로 최대의 충돌이 일어날 수 있는 상황을 예측하는 실험을 제시한다. 그리고, 모든 컨텍스트는 0.1~0.9사이의 값으로 정규화(normalization value) 하였다. 컨텍스트를 정규화하는 이유는 모든 컨텍스트를 우리가 실험에서 사용하는 M-BP의 입력 값(input value)으로 적용하여 출력값(output value)의 정확도를 높이기 위한 것이다. Position은 Game World를 9sector 설정하여 9등분 하였고 Linear Velocity는 그 범위를 0-90으로, Center of Mass는 개체의 면적 각각 9등분 하였다. Mass는 한계 질량이 45를 기준으로 하였고 Variable Velocity는 45를 기준으로 9등분 하여 실험 데이터를 설정하였다.

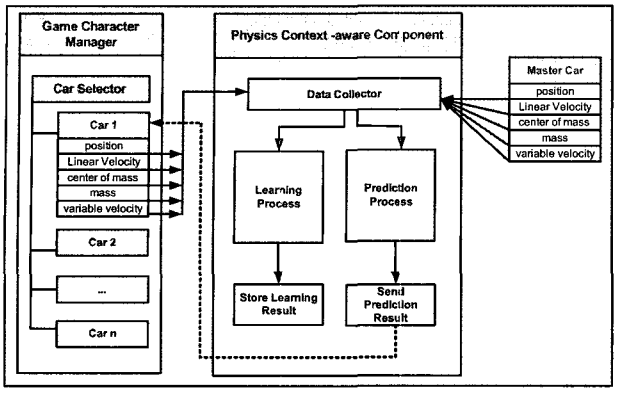
4.1.2 Physics Context-aware Component의 구조도

Physics Context-aware Component는 <표 1>에서 정의된 주인공 자동차와 적 자동차의 물리 컨텍스트를 입력 받아서 학습과 예측에 대한 결과를 생산한다. (그림 3)은 Physics Context-aware Component에 대한 구조도이다.

Physics Context-aware Component는 학습 시에는 주인공 자동차와 적 자동차에서 물리 컨텍스트 및 그 상황에서의 주인공 자동차의 피해수치를 입력 값으로 받고 예측 시에는 주인공 자동차와 적 자동차의 물리 컨텍스트만을 입력 값으로 받는다. 학습시에는 Learning Process에서 해당 학습을 한 후에 M-BP의 가중치 값을 변경시킨다. 예측은 학

〈표 1〉 주인공자동차와 적자동차의 물리적 컨텍스트
〈P:Position, L:Linear Velocity, C: Center of Mass, M:Mass, V:Variable Velocity〉

Norm Value	Master Car					Enemy Car				
	P	L	C	M	V	P	L	C	M	V
0.1	1	0-10	0.0-1.0	1	1-5	1	0-10	0.0-1.0	1	1-5
0.2	2	11-20	1.1-2.0	2	6-10	2	11-20	1.1-2.0	2	6-10
0.3	3	21-30	2.1-3.0	3	11-15	3	21-30	2.1-3.0	3	11-15
0.4	4	31-40	3.1-4.0	4	16-20	4	31-40	3.1-4.0	4	16-20
0.5	5	41-50	4.1-5.0	5	21-25	5	41-50	4.1-5.0	5	21-25
0.6	6	51-60	5.1-6.0	6	26-30	6	51-60	5.1-6.0	6	26-30
0.7	7	61-70	6.1-7.0	7	31-35	7	61-70	6.1-7.0	7	31-35
0.8	8	71-80	7.1-8.0	8	36-40	8	71-80	7.1-8.0	8	36-40
0.9	9	81-90	8.1-9.0	9	41-45	9	81-90	8.1-9.0	9	41-45



(그림 3) Physics Context-aware Component 의 구조도

```
1. member field set
   i_to_h   ---------- random value initialize (-0.5 < i_to_h < 0.5)
   h_to_o   ---------- random value initialize (-0.05 < h_to_o < 0.05)
2. Error max value and learning rate initialize
   learning rate = alpha -------- alpha initialize (0.1 > alpha > 0)
3. momentum value initialize
   momentum value = beta ------ beta initialize ( 0 < beta < 0.8)
4. learn algorithm start
   4.1 compute NET(hidden Layer)
      1 / 1 + exp(-NET(hidden layer))
   4.2 compute NET(output Layer)
      1 / 1 + exp(-NET(ouput layer))
   4.3 compute output error value
      error value <- (target value - output value) ^ 2 / 2 + error value
   4.4 compute error output layer error signal / hidden layer error signal
      output layer error signal <- (target value - output value) * output
      value (1 - output value)
      hidden layer error signal <- hidden layer output value (1 - hidden
      layer output vlaue) Σ ouput layer error signal * hidden layer weight
      value
   4.5 update weight
     output layer-hidden layer weights update
     hidden layer-input layer weights update
   4.6 save weight variation
   4.7 test condition
     if error value < Error max
       stop learn algorithm
     else
       goto 4
```

(그림 4) 알고리즘의 의사코드

습이 이루어진 후 가능하다. 예측 시에는 학습된 결과를 바탕으로 물리 컨텍스트를 분석하여 적 자동차와 주인공 자동차가 충돌했을 경우 피해상태의 예측결과를 적 자동차에게 알려준다. 이 결과에 따라서 적 자동차는 주인공 자동차와의 충돌 여부를 결정한다. 우리는 인공지능 게임 캐릭터 실험을 위하여 모멘텀 역전파 알고리즘(Momentum Back Propagation Algorithm)을 사용하였다. (그림 4)는 우리의 실험에서 적용된 M-BP의 의사코드를 보여준다.

4.2 속도 보완을 위한 Numerical Integration Component

물리엔진을 적용한 게임은 많은 연산작용으로 인해 게임의 성능 저하를 초래한다. 우리의 연구는 물리엔진을 적용한 게임에서 물리 컨텍스트를 인식하여 자신의 행동을 지능적으로 판단하는 캐릭터에 관한 것이다. 이는 물리엔진의 계산 연산뿐만 아니라 인공지능 알고리즘에 대한 연산도 수행해야 하기 때문에 게임의 성능에 있어서 많은 장애를 가져온다. 이러한 연산 장애를 극복하고자 우리의 논문에서는 Numerical Integration Component를 제시하였다. 이 논문에서 우리는 지능적인 캐릭터의 적용 대상으로 자동차 시뮬레이션을 구현하였다. 물리엔진에서는 개체의 이동에 따른 개체의 방향 및 위치계산에 Euler Method, Improved Euler Method, Runge-kutta Method를 적용한다. 하지만 각각의 적분 메서드는 물리적 상황에 따라 서로 다른 성능을 나타낸다. Numerical Integration Component는 게임 속의 물리적인 상황에 따라 가장 좋은 성능을 나타내는 적분메서드를 적용함으로써 물리엔진을 이용한 인공지능의 적용에 따른 성능 장애를 보상해 줄 수 있도록 한다. 세 가지 적분 메서드의 정의는 다음과 같다.

Euler Method는 초기 조건이 방정식(1)에서 주어졌을 때 테일러 급수에 의해 일차 미분 계수까지 정리한다. Euler Method는 개체의 위치 산정의 정확도는 Runge-Kutta Method 보다 떨어지지만 계산 속도에 있어서는 세가지 메서드 중에서 가장 좋은 효과를 보인다.

$$\begin{aligned}
&\frac{dy}{dx} = f(x,y), y(x_0) = y_0 \\
&\frac{dy}{dx} \cong \frac{y(x_0+\Delta x)-y(x_0)}{\Delta x} \cong f(x_0,y_0) \\
&f(x_i,y_i) \cong \frac{y(x_{i+1})-y(x_i)}{\Delta x} \\
&y(x_{i+1}) \cong y(x_i) + hf(x_i,y_i) \\
&h = \Delta x, x_i = x_0 + ih, i = 1,2,3\cdots,n
\end{aligned} \tag{1}$$

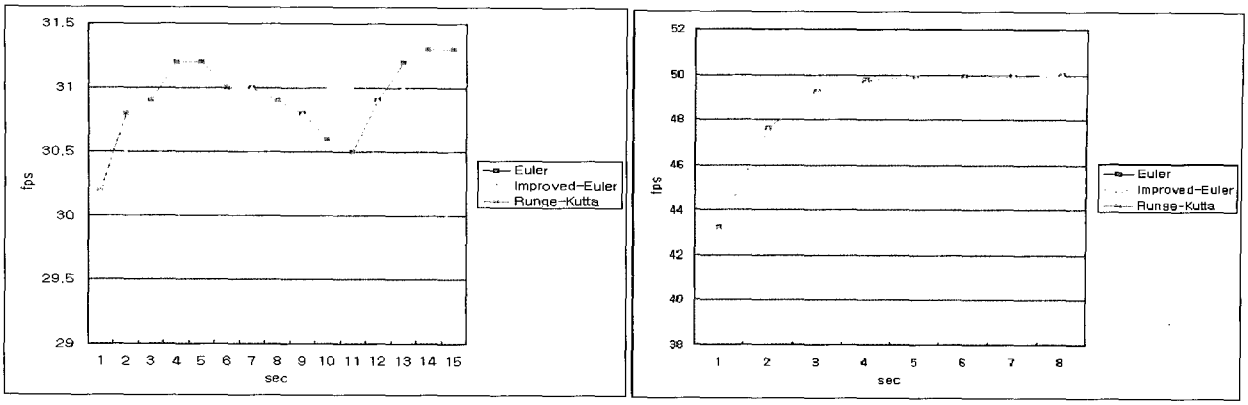
Improved Euler method는 아래 방정식 (2)에서 보이는 바와 같이 이차 미분 계수까지 정리된다.

$$\begin{aligned}
&y_{i+1} = y_i + (1-b)k_1 + bk_2, (b = \frac{1}{2} or 1) \\
&k_1 = hf(x_i,y_i) \\
&k_2 = hf(x_i + \frac{h}{2b}, y_i + \frac{k_1}{2b}
\end{aligned} \tag{2}$$

Runge-Kutta method는 테일러 급수에 의해 4차 미분계수까지 정리되며 Euler Method와 비교하여 정확도에서 좋은 성능을 보인다.

$$\begin{aligned}
&y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\
&k_1 = hf(x_i + y_i) \\
&k_2 = hf(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}) \\
&k_3 = hf(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}) \\
&k_4 = hf(x_i, h, y_i + k_3)
\end{aligned} \tag{3}$$

우리의 실험은 20개 이하의 캐릭터에 대한 적용 결과 이며 실제 게임에서는 수많은 캐릭터가 존재하기에 아래의 작은 차이가 실제 게임에서는 엄청난 성능 차이를 발생시킨다.



(그림 5) 적분 방식에 따른 성능 결과

(그림 5)의 왼쪽은 충돌이 존재하는 않는 상황에서의 개체(x좌표)에 따른 frame per seconds(y좌표)에 관한 실험이다. 개체의 개수가 늘어 남에 따라서 Euler Method의 성능이 현격하게 좋음을 알 수 있다. (그림 5)의 오른쪽은 충돌이 존재하는 상황에서의 개체(x좌표)에 따른 frame per seconds(y좌표)에 관한 실험이다. 충돌이 존재하는 상황에서는 모든 메서드가 FPS의 성능 비교에 있어서 비슷한 성능을 나타내므로 정확도가 뛰어난 Runge-Kutta Method가 최적의 성능을 보인다고 말할 수 있다. Numerical Integration Component는 자동차(주인공 및 적)의 방향 및 위치 계산에 있어서 충돌이 존재하지 않는 상황에서는 Euler Method를 적용하고 충돌이 존재하는 상황에서는 Runge-Kutta를 적용해주는 역할을 담당한다.

5. 실험 및 평가

이 논문에서 제시하는 실험은 물리엔진이 적용된 게임 속에서 주인공 및 적 자동차가 움직일 때 발생하는 상황에 따른 최대의 충격량을 M-BP를 이용하여 정확하게 예측할 수 있는지에 대한 실험이다. 실험은 두 가지 방식으로 정확도를 실험하였다. 첫 번째가 은닉 층의 변경으로 인한 알고리즘 정확도 실험이고, 두 번째가 학습회수의 통제에 의한 알고리즘 정확도 실험이다. 주인공 자동차와 적 자동차로부터 5가지의 물리 컨텍스트를 입력 값으로 전달 받기 때문에 입력 층의 수는 고정되어서 실험되었다. 또한 출력층의 개수도 주인공 자동차의 파괴 값이 100을 기준으로 10등분하여 총 10층의 출력 층이 고정되어 실험되었다. <표 2>는 실제 두 자동차 간에 발생하는 충격량을 0.1-0.9로 정규화 한 값이고, 이 값은 M-BP의 학습 시에 5가지의 물리 컨텍스트의 결과 값으로 적용된다.

<표 3>은 은닉 층의 변경에 따른 예측 정확도를 표로 나타낸 것이다. 예측 정확도란 M-BP알고리즘에 의해 예측된 정확도와 실제 물리엔진에 의해서 계산된 두 개체간의 충격량의 정확도가 일치하는 것인지에 대한 테스트이다.

테이블 3에서 나타난 것 과 같이 은닉 층은 3계층이 가장 좋은 성능을 보여주었다. 테이블 4는 학습횟수에 따른 정확

〈표 2〉 출력 층 정의 값 (Master Car의 파괴 값)

Power value of Master Car	0-10	11-20	21-30	31-40	41-50	51-60	61-70	71-80	81-90	91-100
Norm Value	0.0-0.1	0.11-0.2	0.21-0.3	0.31-0.4	0.41-0.5	0.51-0.6	0.61-0.7	0.71-0.8	0.81-0.9	0.91-0.99

〈표 3〉 은닉 층의 개수에 따른 정확도 실험 결과

Hidden Layer	Success Rate(%)	Cross Validation error signal value by hidden layer	Cross validation error signal value by output layer	Test error signal value by output layer
1	63	31.23532	231.34353	234.12622
3	92	13.23243	164.23213	166.09230
5	84	24.12153	128.64009	130.25030

〈표 4〉 학습횟수에 따른 정확도 실험 결과

Learning Count	Success Rate(%)	Cross Validation error signal value by hidden layer	Cross validation error signal value by output layer	Test error signal value by output layer
10000	71	28.23553	224.23242	225.55020
20000	81	25.25939	130.96954	131.08834
30000	92	13.23243	164.23213	166.09230
40000	85	23.25534	127.94204	128.03253
50000	85	23.25254	127.89042	128.01223

도 실험 결과를 나타내었다. 테이블 4에서는 학습횟수 30000일 경우에 가장 좋은 성능을 보여주었다.

실험에서 나타난 바와 같이 Physics Context-aware Component는 입력 층이 5로 고정되어있고 출력 층이 10으로 고정된 상황에서 실험되었고 은닉 층이 3계층이고 학습횟수가 30000정도 일 때 가장 좋은 성능(정확도 92%)을 보여주었다.

6. 결 론

이 논문에서 제안되고 실험된 컴퓨터 게임을 위한 물리엔진의 성능 향상 및 이를 적용한 지능적인 게임 캐릭터에 관한 연구는 물리엔진의 적용으로 인해 게임의 사실성을 높여주고, 물리엔진에 기반한 게임 속 캐릭터들의 물리 수치를 인식함에 따라 게임 속 캐릭터의 생명력을 느끼게 해 주는 역할을 한다. 우리는 본 연구를 위한 전체 아키텍처를 제안하였고 그에 대한 실험을 하였다. 제안된 아키텍처는 Physics Context-aware Component, Numerical Integration Component, 그리고 Game Character Manager로 구성되었다. Physics Context-aware Component에서는 물리 컨텍스트를 기반으로 그에 대한 학습과 예측을 실행한다. Numerical Integration Component는 물리엔진 기반에서의 지능적 캐릭터 적용이 게임에서의 많은 속도 감소를 유발하는데, 이를 보완하기 위한 최적의 성능을 발휘하는 적분 메서드 선택 함으로써 게임의 성능장애를 극복하는 역할을 한다.

현재 많은 게임에서 물리엔진이 사용되고 있으므로, 향후에는 이러한 물리엔진이 적용된 게임에서의 물리적 상황인식이 게임에서의 중요한 요소로 자리잡을 것이다. 이 논문에서의 연구는 자동차 게임에 한정되어 있다. 앞으로 우리는 다른 장르의 게임에서 이러한 물리 컨텍스트를 인식하는 연구를 진행할 것이며 또한 전체 게임의 성능을 증가할수 있도록 M-BP이외의 인식 알고리즘을 적용할 것이다.

참 고 문 헌

[1] Kook, H.J, Novak, G. S., Jr, "Representation of models for solving real world physics problems," Proceedings of the Sixth Conference on Artificial Intelligence for Applications, pp.274-280, 1990.

[2] Chen, Z., An, Y., Jia, K., Sun, C, "Intelligent control of alternative current permanent manage servomotor using neural network", Proceedings of the Fifth International Conference on Electrical Machines and Systems, Vol.2, pp.18-20, August, 2001.

[3] Chen Zhifei, An Yuejun, Jia Kebing, Sun Changzhi, "Intelligent control of alternative current permanent magnet servomotor using neural network," Electrical Machines and Systems, 2001. ICEMS 2001. Proceedings of the Fifth International Conference on, Vol.2, pp.18-20 Aug., 2001, Vol.2, pp.743-746.

[4] Munthe-Kaas, H, "High order Runge-Kutta methods on manifolds," Journal of Applied, Number Math, pp.115-127, 1999.

[5] Math Engine, http://www.mathengine.com

[6] Havok, http://havok.com

[7] Meqon, http://www.meqon.com

[8] Open Dynamics Engine, http://ode.org

[9] Lawlor, O.S., Kalee, L.V, "A Voxel-based Parallel Collision Detection Algorithm," Proceedings of the 6th international conference on Supercomputing, pp.285-293, June, 2002.

[10] Gabriel Zachmann, "Minimal Hierarchical Collision Detection," Proceedings of the ACM symposium on Virtual reality software and technology, pp.121-128, November, 2002.

[11] Qian. Hu, Ming-Rui. Fei, Xia-Ei. Feng, "Machine Learning and Cybernetics," Proceedings of International Conference on , Vol.2, pp.1034-1037, Nov., 2002.

최 종 화

e-mail : jhchoi@gce.sejong.ac.kr

2001년 세종대학교 컴퓨터공학부(공학사)

2005년 세종대학교 컴퓨터공학부(석사)

2005년~현재 세종대학교 컴퓨터공학부 박사과정

관심분야 : 상황인식 미들웨어, 패턴인식, HCI, 게임물리엔진, 게임 인공지능

신 동 규

e-mail : shindk@sejong.ac.kr

1986년 서울대학교 계산통계학과(학사)

1992년 M.S. in Computer Science, Illinois Institute of Technology

1997년 Ph.D in Computer Science, Texas A&M University

1986년 2~1991년 8월 한국국방연구원 연구원

1997년 8월~1998년 2월 현대전자 멀티미디어연구소 차장 (책임연구원)

1998년 3월~현재 세종대학교 컴퓨터공학과 부교수

관심분야 : XML보안, 전자상거래, MPEG

신 동 일

e-mail : dshin@sejong.ac.kr

1988년 연세대학교 전산과학과(이학사)

1993년 M.S. in Computer Science, Washington State University

1997년 Ph.D in Computer Science, University of North Texas

1997년 9월~1998년 2월 시스템공학연구소 선임연구원

1998년 3월~현재 세종대학교 컴퓨터공학과 조교수

관심분야 : 무선인터넷, HCI, 게임엔진, CSCW