# 3 차원 게임에서의 물리엔진에 기반한 인공지능 캐릭터에 관한 연구

최종화, 이병윤, 이주연, 신동규, 신동일
세종대학교 컴퓨터공학과
e-mail : {com97, over2015, gogo-jju}@gce.sejong.ac.kr,
{shindk, dshin}@sejong.ac.kr

# Research on Artificial Intelligence Character based Physics Engine in 3D Game

Jonghwa Choi, Byung Yoon, Lee, Juyoun Lee, Dongkyoo Shin, Dongil Shin
Dept. of Computer Engineering, Sejong University

**요 약**

이 논문은 게임물리엔진에서 게임세계의 물리적인 요소를 통하여 게임에 존재하는 캐릭터들에게 인공지능을 부여하기 위한 연구에 관해서 다룬다. 게임속에서의 물리적인 상황을 자동인식하기 위해서 신경망을 이용하였다. 게임속에서의 인공지능의 적용은 게임의 속도저하를 가져오게 되는데 이 논문에서는 그러한 단점을 보완하기 위하여 물리엔진에서 캐릭터의 움직임을 계산하는 수치적분 메서드들에 대한 각 물리상황에 따른 최적의 성능을 분석하여 각각의 물리 상황마다 다른 수치 적분 메서드를 적용하는 내부 구조를 취하였다. 수치적분 메서드에 대한 각각의 성능 분석은 세가지의 물리적 상황을 구분하여 그에 기반하여 실험되었다. 인공지능 캐릭터에 대한 실험은 신경망의 토폴로지에 대한 변화와 학습 횟수에 대한 변화 및 은닉층에 대한 변화로 신경망에서의 최적의 성능에 대한 평가를 실시하였다.

## 1. 서론

현재 3 차원 게임에 있어서 가장 중요한 이슈는 게임속 세계에 존재하는 모든 캐릭터들이 사실감과 생동감을 유지하면서 프로그램의 속도 감소 문제를 발생시키지 않는 방안을 찾는 것이다. 게임속 캐릭터의 사실감이란 캐릭터의 움직임이 현실의 물리현상과 같은 현상을 나타내는 것을 말하는 것이며 게임 물리엔진의 적용이 그 해결책으로 제시되고 있다[1]. 하지만 물리엔진이 적용된 게임은 캐릭터의 사실적인 움직임만을 표현할 뿐 캐릭터에게 생명력을 심어주지는 못한다.

이 논문에서는 물리엔진이 적용된 게임에서 주인공 캐릭터가 행동할 때 발생하는 물리적 요소를 그외의 캐릭터가 인식하여 자신의 행동을 스스로 결정하는 살아있는 캐릭터 생성을 위한 연구를 제시하고자 한다. 이러한 연구는 게임 인공지능의 한 분야하고 할 수있다. 하지만 지금까지의 게임 인공지능의 대부분의 연구는 캐릭터가 게임속에서 어떻게 행동해야 하는지에 대한 자신만의 규약을 가지고 행동하는 양식으로 진행되었다[2].

이 논문에서는 각각의 캐릭터가 주인공 캐릭터의 물리적 요소를 인식하기 위한 알고리즘으로 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 사용함으로써 게임에서의 성능을 보완한다[4].

2 장에서는 게임에서의 물리엔진에 관한 연구와 게임 인공지능에 관하여 지금까지 진행되었던 연구를 살펴보고 3 장에서는 이 논문에서 제시하는 컴포넌트 아키텍쳐를 제시한다. 4 장에서는 컴포넌트 아키텍쳐에서 제시된 각각의 컴포넌트에 대한 세부설명을 한다. 5 장에서는 상황에 따라 변화하는 물리요소를 입력받아서 현재의 물리현상을 결정하는

지능형 에이전트에 대한 실험 및 이로 인해 발생하는 성능감소의 해결책인 상황에 따른 적분 메서드의 성능평가를 제시한다. 6 장에서는 이 논문의 결론은 맺는다.

## 2. 관련연구

물리엔진은 상업적인 제품과 오픈소스를 기반으로 제공된 엔진으로 나뉜다. 상업제품으로는 Math Engine[5], Havok[6], Meqon[7]등이 존재하고 오픈소스로는 ODE[8]가 제공되고 있다.
물리상황에서의 인공지능에 관한 연구는 미비한 실정이다. 다만 이 연구에서와 유사한 방향으로는 지능을 가진 애완로봇에 관한 연구[9]와 로봇 축구에서의 주변 환경을 인식한 동작[10]에 관한 연구를 들 수 있다.

이 논문은 현실세계의 물리적 요소를 게임속에서 발견하여 그로 인한 게임속의 캐릭터에게 지능을 주기 위한 연구로 게임연구에서는 아직 연구가 이제 시작 단계로 여겨지는 분야이다.

## 3. 게임에서의 자동적인 물리적 상황 인식을 위한 컴포넌트 아키텍쳐

### 2.1 시나리오

이 논문에서는 자동적인 물리적 상황 인식 중 자동차를 이용한 게임에 대해서 아키텍쳐를 제시하고 그에 대한 실험을 한다. 자동차 게임에서는 하나의 주인공 자동차와 주인공 자동차를 공격하는 많은수의 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.

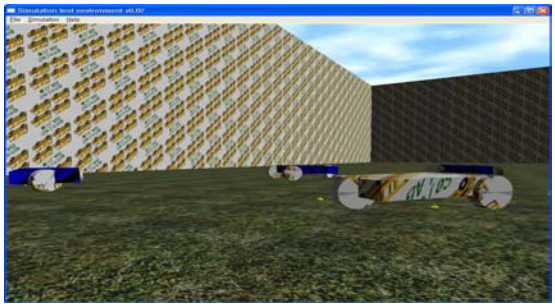

**그림 1 물리엔진에서의 인공지능 케릭터의 스크린샷**

### 2.2 컴포넌트 아키텍쳐

그림 1 은 게임에서의 자동적인 물리적 상황 인식을 위한 컴포넌트 아키텍쳐이다. 컴포넌트 아키텍쳐는 3 가지의 컴포넌트를 포함하며 각각의 컴포넌트는 서로 유기적인 관계를 맺고 있다. Physics Situation Recognition Component 는 주인공 자동차가 동작할 때 발생하는 물리적 컨텍스트를 분석하여 적 자동차와 주인공 자동가가 충돌했을 때 일어날수 있는 상황을 예측한다. Physics Situation Recognition Component 에서는 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 따른 주인공 자동차가 움직일 때마다 적 자동차들은 최대의 피해를 줄 수 있는 상황을 파악하여 행동하게 된다. 물리엔진에서는 모든 개체의 움직임을 게임에 적용할 때 Integration Method 를 적용하여 개체의 움직임을 계산하는데 주로 Euler method, improved Euler method, Runge-Kutta method 가 사용된다. Numerical Integration Component 에서는 주인공 자동차와 적 자동차 간의 상황에 따라서 최적의 성능을 발휘하는 Integration method 를 적용한다. 실제 게임에서 물리적 상황을 인식하는 기능을 포함하면 게임전체의 성능 저하를 초래하는데 Numerical Integration Component 는 이에 대한 성능감소 부분을 보완하는 역할을 담당한다.
적 자동차 매니저에서는 적 자동차 리스트 중에 주인공 자동차와 가장 근거리에 있는 적 자동차를 선택하여 적 자동차의 현재의 물리적 요소를 Physics Environment Recognition Component 에게 전송하는 역할을 담당하고 그 결과값[주인공 자동차의 피해수치]을 적 자동차에게 다시 전달해 주는 역할을 담당한다.

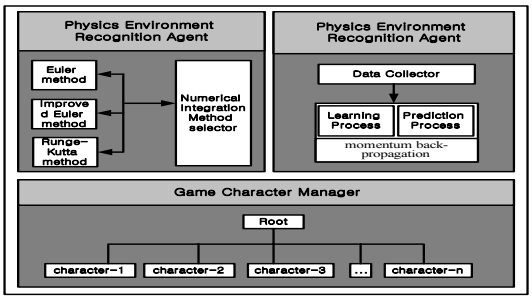


**그림 2 컴포넌트 아키텍쳐**

## 4. 컴포넌트 구조

### 4.1 물리상황인식 컴포넌트

#### 4.1.1 물리 컨텍스트 정의

Physics Environment Recognition Component 는 적 자동차가 주인공 자동차의 움직임을 파악하여 주인공 자동차에게 최대의 피해를 줄수 있는 상황을 판단하여 주는 역할을 담당한다. 이 논문에서는 상황 판단을 위하여 M-BP 알고리즘을 적용하여 상황을 학습하고 판단한다. 테이블 1 은 상황판단을 위한 주인공과 적 자동차에서 추출되는 물리적 컨텍스트를 정의한다. 테이블 1 에 정의된 물리적 컨텍스트는 물리상황인식 알고리즘의 입력값으로 적용된다.

**표 1 물리 컨텍스트 정규화**

| 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 |

자동차 시뮬레이션에서 자동차가 움직일 때 마다 변화는 물리적 요소는 Position, Linear Velocity, Variable Velocity 가 있고 자동차간의 충돌시 충돌반응에 영향을 미치는 요소는 Mass, Center of Mass 가 있다. Position 은 Game World 를 9sector 설정하여 9 등분 하였고 Linear Velocity 는 그 범위를 0 - 90 으로, Center of Mass 는 개체의 면적 각각 9 등분 하였다. Mass 는 한계 질량이 45 를 기준으로 하였고 Variable Velocity 는 45 를 기준으로 9 등분 하여 실험 데이터를 설정하였다.

4.1.2 물리상황인식 알고리즘

Physics Situation Recognition Process 는 Table 1 에서 정의된 주인공 자동차와 적 자동차의 Physics Context 를 입력받아서 학습과 예측에 대한 결과를 생산한다. 그림 2 는 Physics Situation Recognition Process 에 대한 구조도이다.

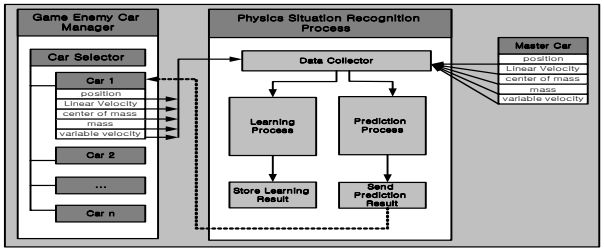


**그림 3 물리상황인식 프로세스 흐름도**

Physics Situation Recognition Process 는 학습시에는 주인공 자동차와 적 자동차에서 Physics Context 및 그 상황에서의 주인공 자동차의 피해수치를 를 입력값으로 받고 예측시에는 주인공 자동차와 적 자동차의 물리적 컨텍스트만을 입력값으로 받는다. 학습시에는 Learning Process 에서 해당 학습을 한 후에 가중치 값을 변경시킨다. 예측은 학습이 이루어진 후 가능하다. 예측시에는 학습된 결과를 바탕으로 Prediction Process 에서 Physics Context 를 분석하여 적 자동차와 주인공 자동차가 충돌했을경우 피해상태의 예측결과를 적 자동차에게 알려준다. 이 결과에 따라서 적 자동차는 주인공 자동차와의 충돌 여부를 결정한다.

```
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) ∑ 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 실험에서 적용된 알고리즘의 의사코드**

4.2 속도 보완을 위한 적분 메서드

이 논문에서는 캐릭터의 움직임을 찾아내기 위하여 물리엔진에서 사용되는 세가지 적분메서드의 성능 분석 실험을 행하였고, 세가지 적분 메서드는 충돌상황, 충돌이 없는 상황, 가속도 상황 등으로 구분하여 각각의 성능을 평하가였다. 다음은 이 물리엔진에서 사용된 적분 메서드인 오일러메서드(1), 향상된 오일러메서드(2), 룬지쿠다 메서드(3)에 대한 방정식이다.

$$\frac{dy}{dx} = f(x, y),\ y(x_0) = y_0 \tag{1}$$

$$\frac{dy}{dx} \cong \frac{y(x_0 + \Delta x) - y(x_0)}{\Delta x} \cong f(x_0, y_0)$$

$$f(x_i, y_i) \cong \frac{y(x_{i+1}) - y(x_i)}{\Delta x}$$

$$y(x_{i+1}) \cong y(x_i) + hf(x_i, y_i)$$

$$h = \Delta x, x_i = x_0 + ih, i = 1, 2, 3 \cdots, n$$

$$y_{i+1} = y_i + (1 - b)k_1 + bk_2, (b = \frac{1}{2} or\ 1) \tag{2}$$

$$k_1 = hf(x_i, y_i)$$

$$k_2 = hf(x_i + \frac{h}{2b}, y_i + \frac{k_1}{2b})$$

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{3}$$

$$k_1 = hf(x_i + y_i)$$

$$k_2 = hf(x_i + \frac{h}{2}, y_i + \frac{k_1}{2})$$

$$k_3 = hf(x_i + \frac{h}{2}, y_i + \frac{k_2}{2})$$

$$k_4 = hf(x_i, h, y_i + k_3)$$

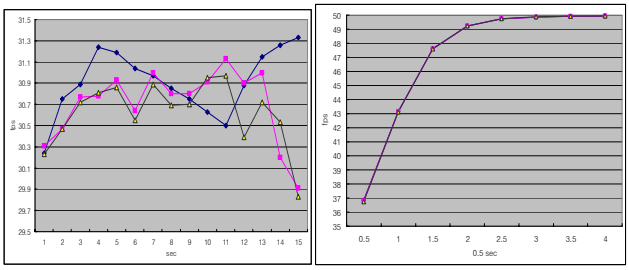

**그림 5 물리엔진에서 적용된 적분메서드의 성능평사 결과 그래프**

충돌이 존재하지 않는 상황에서는 세가지 메서드가 FPS 에서는 비슷한 성능을 보였지만 정확도에 있어서는 룬지쿠타가 좋은 성능을 보였다. 충돌이 존재하는 상황에서는 오일러 메서드가 가장 좋은 성능을 보였다.

## 5. 실험 및 평가

Physics Situation Recognition Process 에서의 실험은 세가지 방법으로 정확도가 실험되었다. 주인공 자동차와 적 자동차로부터 10 가지의 물리 컨텍스트를 입력값으로 전달 받기 때문에 입력값의 수는 고정되어서 실험되었다. 또한 출력층의 개수도 주인공 자동차의 파워량이 100 을 기준으로 5 등분하여 결정하였기 때문에 총 5 층의 출력층이 고정되어 실험되었다

**표 2 인식 알고리즘에서의 출력값 정의**

| 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 |

실험은 두가지 방식으로 Physics Situation Recognition Algorithm 의 정확도를 실험하였다.
첫번째는 히든레이어 층의 변경으로 인한 알고리즘 정확도 실험이고 두번째는 학습회수의 통제에 의한 알고리즘 정확도 실험이다. 테이블 3은 히든레이어의 변경에 따른 예측 정확도를 표로 나타낸 것이다.

**표 3 은닉층 변경에 따른 정확도**

| 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 |

테이블 3 에서 나타난 것 과 같이 히든레이어는 3 계층이 가장 좋은 성능을 보여주었다.

테이블 4 는 학습횟수에 따른 정확도 실험 결과를 나타내었다. 테이블 4 에서는 학습횟수 3000 일 경우에 가장 좋은 성능을 보여주었다.

**표 4 학습횟수 변경에 따른 정확도**

| 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 |

실험에서 나타난바와 같이 물리적 컨텍스트 인식 알고리즘은 입력층이 6 단계로 고정되어있고 출력층이 5 단계로 고정된 상황에서 실험되었고 히든레이어가 3 계층이고 학습횟수가 30000 정도 일 때 가장 좋은 성능(정확도 92%)을 보여주었다.

## 6. 결론

이 논문에서 제안되고 실험된 Physics Context Recognition 에 관한 연구는 게임의 생명력을 느끼게 해 주는 중요한 요소이다. 우리는 Physics Context Recognition 을 위한 아키텍쳐를 제안하였고 그에 대한 실험을 하였다. 제안된 아키텍쳐에서는 Physics Context Recognition Component, Integration method Component, Game Contents Manager 로 구성되었다. Physics Context Recognition Component 에서는 물리적 컨텍스트를 기반으로 그에 대한 학습과 예측을 실행한다. Physics Context Recognition 적용은 게임에서의 속도 감소를 유발하는데 그에 대한 보완으로 Physics Situation 에 따른 최적의 성능효과를 나타내는 Integration Method Component 를 제시하였다.
현재 많은 게임에서의 물리엔진적용이 이루어지고 있다. 앞으로는 이러한 물리엔진이 적용된 게임에서의 물리적 상황인식은 게임에서는 중요한 요소로 자리잡을 것이다.
이 논문에서의 연구는 자동차 게임에 한정되어 있다. 앞으로 우리는 다른 장르의 게임에 이러한 물리적 컨텍스트를 인식하는 연구를 진행할 것이며 또한 전체 게임의 성능을 증가할수 있도록 M-BP 이외의 인식 알고리즘을 적용할 것이다.

## 참고문헌


[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, (1990) 274-280

[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, Volume 2, August(2001) 18-20

[3] Jameson, A, Baker, T.J, “Improvements to the Aircraft Euler Method”, AIAA, January (1987)

[4] Munthe-Kaas, H, ”High order Runge-Kutta methods on manifolds”, Journal of Applied Number Math, (1999) 115-127

[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] Kubota. N, Kojima. F, Fukuda. T, IFSA World Congress and 20th NAFIPS International Conference, July (2001) 2786 - 2791

[10]Qian. Hu, Ming-Rui. Fei, Xia-Ei. Feng, “Machine Learning and Cybernetics”, Proceedings of International Conference on , Volume 2,Nov(2002) 1034 - 1037