컴퓨터 게임을 위한 지능형 아이콘 설계 및 구현

Design and Implementation of the Intelligent Icon for Computer Game

박인호[1], 이원형[2]

In-Ho Park[1], Won-Hyung Lee[2*]

요약

본 논문에서는 게임 내의 플레이어가 취할 수 있는 행동에 따라 다양한 기능을 분류하고 묶어서 상황에 따라 플레이어의 성향에 적합한 유저 인터페이스를 제공하는 지능형 아이콘을 설계하였다.
제안한 지능형 아이콘은 정보수집 / 필터링 모듈을 통해 게임 환경으로부터 플레이어의 현재 상태와 현재 상태에서 가능한 행동 목록을 받아온다. 받아온 상태와 행동의 쌍과 플레이어 모델에 저장된 선호도 특성값을 기반으로 플레이어가 가장 선호하는 행동을 사용자 인터페이스를 통해 제공한다. 사용자 인터페이스를 통해 입력된 플레이어의 입력은 인터페이스 에이전트의 학습모듈로 피드백 되어 입력된 행동에 대한 플레이어 모델의 선호도 특성값을 학습하게 된다. 앞에서 설계한 지능형 인터페이스 에이전트를 C#기반의 XNA Game Studio 2.0으로 제작된 실시간 경영시뮬레이션 게임 "The R-Day"에 실제로 적용하였다.

핵심어: 지능형아이콘, 플레이모델, 학습모듈, 게임플레이어, 컴퓨터게임

Abstract

In this paper, as various behaviors can be gained in game platform, various features were able to divided into categories. Thus 'Intelligent Icon' was developed to provide user interface, in which is suited to the propensity of player. Through gathering information and filtering module, 'Intelligent Icon' which is proposed to study further, is able to gain the current state of player and the list of possible behaviors on the current state from the environment of game. Based on value of specific character of preferences in which it has been stored on player model, and combined two status and behaviors in which they are received from gathered informations and filtering module, it is able to provide the most favored behavior of player through user interface. Input of the player which is inputted through user interface, can be used as the feedback of learning module of interface agent.
Thus in this 'Intelligent Icon', we are able to learn and analyse the specific value of player about input actions. Intelligent Interface Agent as I mentioned before, was applied to the real-time business simulation game.

Keyword: Intelligent Icon, play model, Learning model, Game Play, Computer game

1 Graduate School of Advanced Imaging Science, Multimedia & Film, Chung Univ, 221 Huksuk-dong, Dongjak-ku, Seoul, 156-756, Korea e-mail: jestint@korea.com
2 Graduate School of Advanced Imaging Science, Multimedia & Film, Chung Univ, 221 Huksuk-dong, Dongjak-ku, Seoul, 156-756, Korea; e-mail: whlee@cau.ac.kr (Corresponding author)
* 이 논문은 박인호의 2009년도 석사 학위논문에서 발췌 정리하였음
Received(May 31.2015), Review (June 11.2015), Accepted(June 30.2015)

1. 서론

컴퓨터 그래픽 기술의 발달로 인해 게임 그래픽의 수준이 실사에 근접할 정도로 좋아졌지만, 그래픽만으로는 실시간으로 변화하는 게임내의 상황을 플레이어에게 효과적으로 전달하고 플레이어를 게임에 몰입시키기에는 부족하다. '2006 대한민국 게임백서'에서 게임 이용시 불만사항 질의에 대한 응답에 따르면 가격이나 사양, 불안정 문제 등을 제외했을 때, 그래픽이 차지하는 비중(3.2%)보다 게임조작 및 인터페이스의 불편(5.4%)이 차지하는 비중이 더 높게 나타났다. 이는 게임의 흥미와 재미를 유지시키는 요소에서 그래픽이 차지하는 비중보다 유저 인터페이스(User Interface, UI)가 차지하는 비중이 더 크다는 것을 나타낸다.

게임 내에서 아바타의 상태나 사용가능한 행동, 기능 등을 얼마나 적절하고 빠르게 표시하여 플레이어에게 전달하느냐가 게임 진행에 미치는 영향이 크기 때문이다[1].

최근 게임 장르의 경계가 모호해지고 한 게임 안에서 즐길 수 있는 콘텐츠가 다양해지면서 게임 플레이 또한 복잡해지는 경향이 있다. 게임 시스템이 복잡해지고 플레이어가 할 수 있는 행동이 다양해짐에 따라 표시되는 정보가 많아지고 복잡해지면서 사용하기 불편하고 몰입도가 떨어지는 등 역효과가 나타나기 시작했다. 게임 내의 다양한 기능들을 한 화면에 표시하기가 힘들기 때문에 이를 잘 선별하여 효과적으로 플레이어에게 전달할 필요가 있다. 이중에서 모든 플레이어에게 만족할만한 UI를 제공하는 것은 현실적으로 매우 어려운 일이다. 이에 대한 좋은 해법은 지능형 인터페이스(intelligent interface)에 관한 연구에서 찾을 수 있다[2].

컴퓨터 처리속도의 증가와 인터넷 속도의 증가에 따라 사용자가 원하는 정보를 제공하는 검색 서비스에서 발전해 사용자에 따라 각기 다른 정보를 제공하는 맞춤형 서비스가 등장하고 있다. 이러한 서비스는 사용자의 나이별, 직업별 특성이나 선호 등을 고려하여 사용자가 관심 있어할 만한 적합한 정보나 사용자의 특성에 따라 유용한 정보를 제공한다. 사회 전반적인 분야에서 맞춤형 서비스에 대한 관심이 높아지고 있는 가운데 게임에서도 맞춤형 서비스의 필요성이 대두되고 있다[3].

본 논문에서는 게임 내의 플레이어가 취할 수 있는 행동의 패턴을 연구하여 플레이어의 상황에 적합한 유저 인터페이스, 즉 필요한 아이콘을 화면상에 표시하는 지능형 인터페이스 에이전트(intelligent interface agent)를 구현하였다.

2. 관련연구

2.1 Voting EM 알고리즘

Voting EM 알고리즘은 데이터로부터 파라메터를 실시간으로 학습하는 방법으로, 베이지안 네트워크의 구조가 정해져 있고, 확률 변수들이 불연속 값을 취할 때 적용할 수 있다[22][23]. xi를 $\{x_i^1, x_i^2, \cdots, x_i^{Pi}\}$의 p_i개의 값을 갖는 베이지안 네트워크의 노드라 하고, π_i를 $\{x_i^1, x_i^2 \cdots, x_i^{qi}\}$의 q_i개의 상태 조합을 갖는 xi의 부모 집합이라고 하면, xi에서의 CPT값 $\theta ijk = P(x_i = x_i^k | \pi_i = \pi_i^j)$로 정의할 수 있다. 온라인 베이지안 네트워크 파라메터 학습은 주어진 데이터 집합 D=$\{d_1, d_2, \cdots, d_t, \cdots\}$와 현재의 파라메터 집합 θ^t를 이용하여 적합한 파라메터 θ^{t+1}을 다음수식과 같이 구한다.

$$\theta^{t+1} = \arg MAX_\theta [\eta L(\theta|D) + d(\theta, \theta^t)] \quad (1)$$

여기서 $L(\theta|D)$는 파라메터 값 θ와 데이터 집합 D간의 로그 가능도(log likelihood, LL)를 나타내고, $d(\theta,\theta^t)$는 두 파라메터 값 θ와 θ^t간의 거리를 나타낸다. 즉, θ^{t+1}은 주어진 데이터 집합 D와의 LL값이 높으면서 이전의 파라메터 집합 θ^t의 성질을 유지한다. η는 거리에 대한 LL의 중요도, 즉 데이터의 학습률을 의미한다. Bauer 등은 모든 I, j에 대해서 $\sum_k \theta_{ijk} = 1$을 만족해야하는 제약조건을 이용하여 [수식 1]의 최대화 문제를 풀었다[4].

Cohen 등은 Bauer의 EM(η) 알고리즘을 온라인 학습에 적용시켜 Voting EM알고리즘을 제안했다[5][6]. Voting EM 알고리즘은 시점 t에서의 데이터와 파라미터 θ^t를 이용하여 t+1 시점에서의 파라미터 값 θ^{t+1}을 [수식 2]와 같이 구하였다.

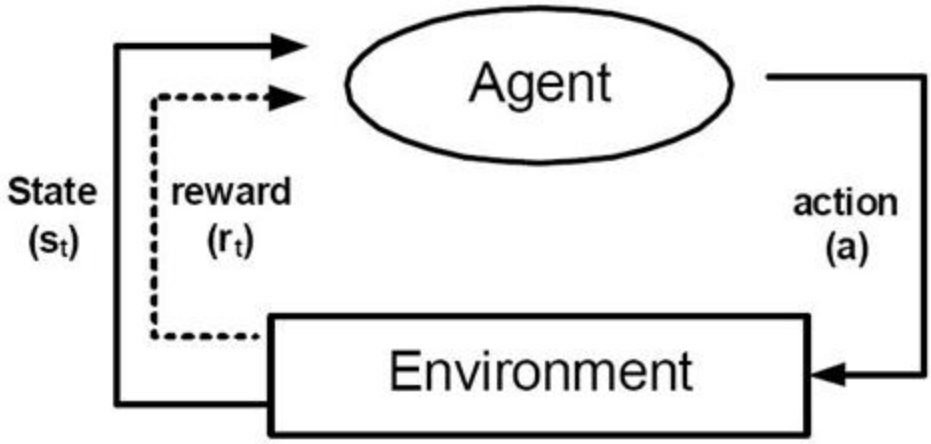
$$\theta_{ijk}^{t+1} = \begin{cases} (1-\eta)\theta_{ijk}^t + \eta \dfrac{P(x_i^k, \pi_i^j | d_t, \theta^t)}{P(\pi_i^j | d_t, \theta^t)} & \text{for} \quad P(\pi_i^j | d_t, \theta^t) \neq 0 \\ \theta_{ijk}^t, & otherwise \end{cases} \quad (2)$$

완전한 데이터의 경우, [수식 2]를 이용하여 파라미터를 학습하면 θ_{ijk}^{t+1}는 학습률 η를 따라 t시점에서의 데이터 d_t에서 θ_{ijk}^t의 상황($x_i = x_i^k, \pi_i = \pi_i^j$)이 나타나면, 1에 가까운 값으로 갱신되고, θ_{ijk}^t의 상황이 나타나지 않으면, 0에 가까운 값으로 갱신된다.

Voting EM 알고리즘은 학습률을 따라 t시점에서의 파라미터를 추측하여 학습한다. 학습률의 효과는 파라미터의 수렴 속도와 파라미터의 변화량에 영향을 미친다. 학습률 = 1일 때 가장 빠른 수렴 속도를 보이지만 파라미터의 변화량이 매우 큰 편차를 보이게 된다. 따라서 학습률이 작을수록 실제 CPT값에 가까운 파라미터를 학습할 수 있다. 또한 Voting EM 알고리즘에 의해 추측된 파라미터는 실제 데이터의 확률 값에 따라 수렴 정도가 달라진다.

2.3 강화 학습(reinforcement learning)

강화 학습은 환경과 에이전트와의 상호 정보교환을 통하여 에이전트의 행동을 학습해 나가는 학습방법이다. 일반적인 교사 학습은 예상되는 상황에 대해 예측하고, 예측된 상황에 대한 대응 방법을 제시하는 반면, 강화학습은 주어진 상태에 대해 자유로운 선택을 하고 선택의 결과를 제시함으로서 차후에는 보다 나은 선택을 할 수 있게 유도한다는 점이다[7][8].



[그림 1] 강화 학습 수행 과정
[Fig. 1] Reinforcement learning process

[그림 1]은 강화 학습의 수행 과정을 간략히 도식화 한 것이다. 현재의 시간(t)에 대한 환경의 정보(st)를 통해 현재의 상황에서 행동 가능한 모든 행동(A(st))에 대한 정보를 에이전트에 전달한다. 에이전트는 이러한 정보를 바탕으로 최적의 행동(a)을 선택하고, 이것을 환경에 적용시킨다.

강화 학습 에이전트는 변경되는 새로운 상태 정보(st+1)와 환경에 적용시킨 행동(a)에 대한 평가 값인 보상(rt)을 받는다. 이러한 과정을 반복해서 수행하면서 에이전트가 점차 학습 되어 나간

다. 이론 적으로는 학습한 지식과 받은 보상간의 차이를 고려하여 학습하지만, 실제 구현된 강화 학습 에이전트의 대부분은 최적의 행동(a)을 선택할 때 보상(rt)을 미리 구하여 강화 학습 에이전트를 학습한다[9].

Q-learning 알고리즘은 Q(s,a)의 값을 통해 최적의 정책(policy)을 찾아내는 강화 학습 알고리즘이다. Q(s,a)의 값은 주어진 상황에서의 행동(a)로 인해 얼마나 좋은지를 수치화 한 것이다[10][11].

[그림 2]는 Q-learning 알고리즘을 나타낸다. Q-learning은 정책을 기반으로 하여 현재 상황에 적합한 행동을 선택하여 수행한 뒤에 보상 값을 받아 Q-function 값을 학습하는 과정을 통해 Q-learning이 이루어진다.

```
α : 학습률
γ : 감쇠률

while learning
    excute action a
    collect reward r
    determine new state s'
    best = -∞
    for each a' in s'
        if Q(s',a') > best
            best = Q(s',a')
        end if
    end for
    Q(s,a) += α * (r + γ * best - Q(s,a))
    s = s'
end while
```

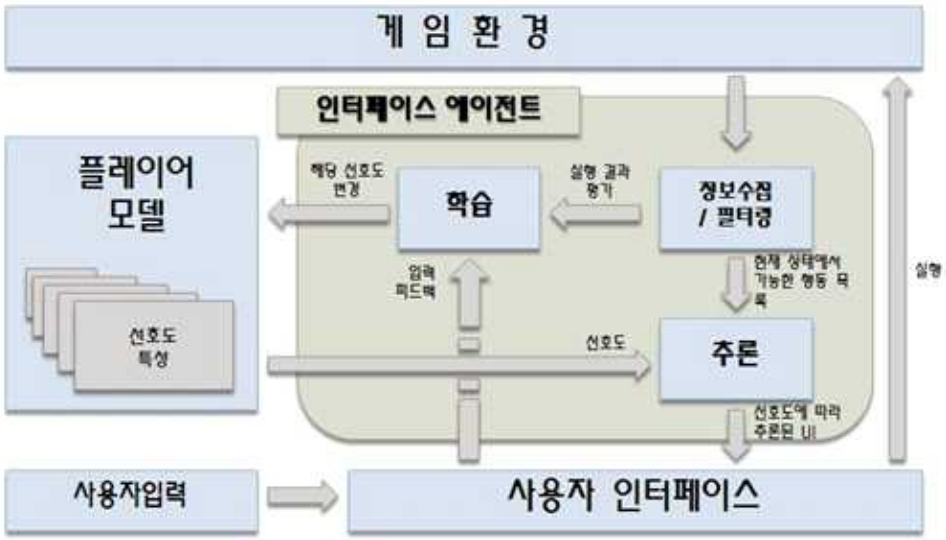
[그림 2] Q-Learning 알고리즘
[Fig. 2] Q-Learning Algorithm

Q-function은 Q-table을 가지고 있다. Q-table은 단순한 저장 공간으로 행동 가능한 모든 경우의 행동에 대한 Q-function의 값을 1:1 매칭되게 저장하고 있다.

이럴 경우 상태 공간이 큰 문제일 경우 저장 공간이 점차 커지는 문제를 가지고 있다. 또한 Q-table이 커질수록 조건에 맞는 Q-function 값을 검색하는 시간은 점차 많아진다.

3. 제안하는 지능형 아이콘

지능형 인터페이스의 핵심 요소인 사용자 모델링, 베이지안 네트워크, 강화 학습 알고리즘을 기반으로 동적으로 플레이어의 선호도를 학습하여 적절한 게임 유저 인터페이스를 제공할 수 있는 플레이어 모델을 기반으로 한 지능형 인터페이스 에이전트를 구현한다.



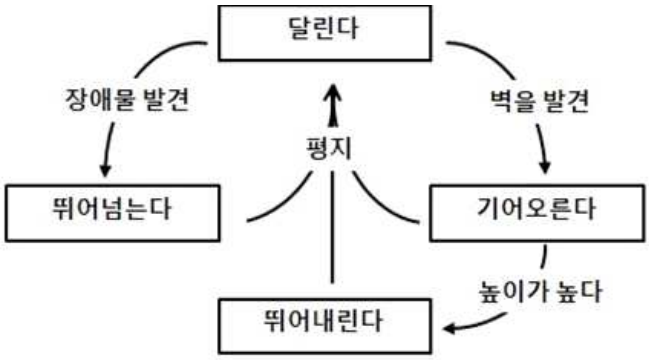
[그림 3] 게임을 위한 지능형 인터페이스 에이전트 구조도
[Fig. 3] Intelligent interface agent structure for game

3.1 지능형 인터페이스 에이전트의 모듈별 설계

1. 정보수집 및 필터링 모듈

정보수집 / 필터링 모듈은 게임 환경으로부터 플레이어의 상태 정보를 수집하고, 유한상태머신을 이용하여 현재 상태에서 가능한 행동만 필터링한 후 추론모듈에 전달하는 역할을 담당한다[12].

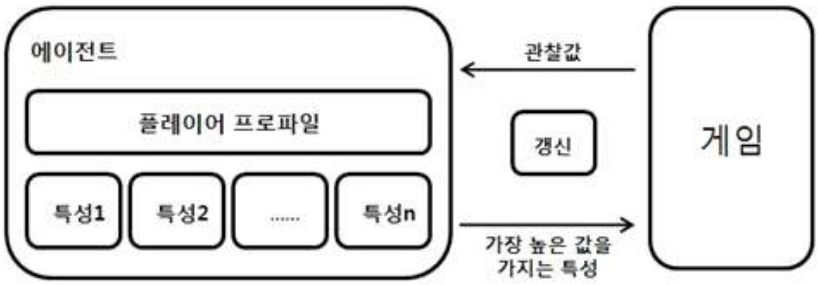
유한상태머신(Finite-State Machines, FSM)은 게임에서 인공지능을 프로그래밍 하는데 널리 사용되는 기술이다. FSM은 개념적으로 간단하면서도 효율적이고, 확장이 용이하다. 간단히 설명하자면 FSM은 어떤 한 객체의 상태가 환경이나 시간이 지남에 따라 바뀌는 방식을 간결하고 비선형적인 형태로 서술한 것이다.



[그림 4] 게임에서 유한상태머신의 예
[Fig. 4] Examples of finite state machines in games

2. 플레이어 모델

시스템이 자신의 목적을 보다 효과적으로 수행하기 위해 사용자의 행동양식, 취향, 성격 등을 파악할 수 있는 사용자 모델링 개념을 응용한 플레이어 모델에서는 플레이어의 행동에 대한 선호도 특성을 게임 과정에 동적으로 학습하여 갱신한다. 이러한 특성은 플레이어 행동들의 발생 빈도에 대한 통계적 수치를 기록 한 것으로 학습 모듈과 추론 모듈 단계에서 참조하게 된다[13].



[그림 5] 플레이어 모델링 구조
[Fig. 5] Player modeling structure

(1) 강화 학습을 이용한 플레이어 모델

강화 학습을 이용한 플레이어 모델은 모델의 특성값을 각각 현재 상태 s와 현재의 상황에서 행동 가능한 행동 a에 대한 쌍으로 계산한다. Ryan이 제안한 알고리즘에서의 관찰 값을 현재 상태 s에서 a라는 행동을 했을 때의 보상 값으로 두면 플레이어 모델의 학습에 Q-Learning 알고리

즘을 적용할 수 있다. 플레이어 모델은 모든 상태 집합 S와 모든 행동 집합 A의 2차원 배열의 상태 공간으로 표현된 Q-Table로 나타낼 수 있다.

[표 1] Q-Table로 표현된 플레이어 모델
[Table 1] Player model represented by Q-Table

	행동0	행동1	행동2	행동3	행동4	행동5
상태0	43.238	79.404	0	0	0	0
상태1	0	0	49.308	-0.923	0	0
상태2	0	0	0	0	70.579	33.871

(2) 베이지안 네트워크를 이용한 플레이어 모델

베이지안 네트워크를 사용한 플레이어 모델링은 플레이어 모델의 구축 단계에서 설계자의 사전지식을 활용하기 쉽기 때문에 기대치만큼의 성능을 쉽게 얻을 수 있다. 플레이어 모델의 선호도 특성값 갱신을 게임 중의 온라인 학습에만 의존하기보다는 플레이어가 선호하게 되는 원인을 반영한 플레이어 모델의 초기값을 설정하는 것이 플레이어의 선호도를 더 빨리 학습할 수 있다.

베이지안 네트워크를 이용하여 플레이어 모델을 설계하기 위해서는 먼저 게임의 상태 전이도를 기반으로 추론 하고자 하는 변수와 각 변수가 가질 수 있는 상태들을 정의한다. 상태에 따라 각 변수들 간의 의존 관계를 정의하여 베이지안 네트워크의 구조를 설계한 후, 각 노드들의 조건부 확률을 정하면 된다.

3. 유저 인터페이스 모듈

유저 인터페이스는 게임 내에서 가능한 행동들을 플레이어에게 표시해주고 플레이어로부터 행동을 입력받는 역할을 담당한다. 유저 인터페이스로부터 입력받은 행동을 학습 모듈에서 플레이어 모델에 학습시키게 되는데, 유저 인터페이스로부터 입력받은 행동을 학습에 피드백하는 방법에 따라 명시적 피드백과 암시적 피드백으로 구분할 수 있다. 명시적 피드백은 플레이어로부터 해당 행동을 직접 입력 받는 경우이고, 암시적 피드백은 추론 모듈로부터 추론된 결과를 재입력 받는 경우이다. 이를 위해 유저 인터페이스는 명시적 피드백을 위한 일반 유저 인터페이스와 암시적 피드백을 위한 추론 유저 인터페이스로 구성된다.

4. 추론 모듈

추론 모듈은 정보수집 / 필터링 모듈로부터 현재 상태까지 가능한 행동 목록들을 전달받는다. 추론 모듈은 전달 받은 현재 상태 S와 현재 상태에서 가능한 행동집합 A를 기준으로 플레이어 모델에 해당하는 베이지안 네트워크의 CPT값 $P(A|S)$를 찾는다. 이를 이용하여 행동 A의 모든 원소 중에 가장 높은 확률값을 가진 행동을 플레이어의 선호도가 가장 높은 행동으로 추론하여 해당 행동을 사용자 인터페이스에 제공한다.

본 논문에서 제안하는 학습 알고리즘은 다음과 같다.

```
state s : 현재 상태
action a : 행동
reward R :  action의 결과에 대한 보상값
FW : 유저 인터페이스로부터 전달받은 학습률
γ : 감쇠율
s' : action의 결과로 전이된 다음 상태
a' : 다음 상태에서 가능한 행동
best = - ∞ : 다음 상태에서 받을 수 있는 보상의 최대값
sumQ = 0 : 현재 상태에서 가능한 모든 행동 a의 Q(s,a) 합

while learning
   excute action a
   collect reward r
   determine new state s'

   for each a' in s'
      if Q(s',a') > best
         best = Q(s',a')
      end if
   end for

   for each a' in s
      if a' = a
         Q(s,a') = (1 - FW)Q(s,a') + FW * (R + γ * best)
      else
         Q(s,a') = (1 - FW)Q(s,a')
   end for
   s = s'
```

```
    for each a' in s
        sumQ += Q(s,a')
    end for

    P(a|s) = Q(s,a) / sumQ

end while
```

[그림 6] 지능형 아이콘의 학습 알고리즘
[Fig. 6] Learning algorithm of intelligent icons

4. 지능형 아이콘 활용을 위한 컴퓨터 게임제작 및 고찰

3장에서 설계한 지능형 인터페이스 에이전트를 실제 게임 제작과정에 적용함으로써 제안한 알고리즘에 대한 검증을 하였다. 플레이어 모델의 학습 알고리즘을 기존의 Voting EM 알고리즘과 비교하여 플레이어의 선호도 학습에 대한 성능을 평가 하였다.

4.1. 제안한 지능형 아이콘의 컴퓨터 게임 적용

C#기반의 XNA Game Studio 2.0으로 제작된 게임 "The R-Day"는 실시간 경영시뮬레이션 게임이다. 플레이어는 제약회사를 운영하며, 백신을 개발하고 바이러스를 퇴치해야만 한다. 게임을 진행하면서 플레이어는 다양한 도시에서 의약품을 공급해야하기 때문에 플레이어는 지속적으로 대상 도시에 대한 정보를 수집해야하고 관리할 필요가 있다. 여기서 에이전트는 플레이어가 주로 거래하는 도시와 도시에서의 행동에 대한 플레이어의 선호도 특성을 학습하여 상황에 맞는 행동에 대한 GUI를 제공하는 것이 목표이다.

XNA Game Studio는 Microsoft사의 제공하는 Windows / XBox360 용 게임 SDK이자 통합개발환경이다. C# 기반의 .NET Framework 컴팩트 에디션 환경에서 동작한다. XNA는 DirectX에 비해서 훨씬 진입장벽이 낮고, 프레임워크를 제공함으로써 게임 제작의 전반적인 과정을 간단화 시켜 빠른 시간 내에 아웃풋을 얻어낼 수 있다는 점, Windows나 XBox360 게임을 동시에 릴리즈 할 수 있다는 점을 장점으로 들 수 있다. XNA Creators Club(http://creators.xna.com)에 가입하면 다양한 정보를 제공받을 수 있다.

[그림 7] The R-Day 게임 화면
[Fig. 7] The R-Day game screen

4.2 제안한 학습 알고리즘의 성능 평가 및 고찰

학습 알고리즘의 성능을 평가하기 위해서 Voting EM 알고리즘과 제안된 알고리즘을 비교하였다. 비교를 위해 상태 전이도를 바탕으로 Voting EM 알고리즘과 제안된 알고리즘에서 행동0, 행동2, 행동4을 선택할 선호도 확률을 0.7, 행동1, 행동3, 행동5를 선택할 선호도 확률을 0.3으로 가정한 상태에서, 각기 다른 학습률(0.01, 0.05)을 적용해 각각 2000번의 반복 학습을 실행하였고, 실제 선호도가 변화할 때 알고리즘에 의한 선호도 학습이 어떻게 되는지 확인하기 위해 학습률 0.01에서 실제 선호도가 0.3에서 0.7로 증가했을 때, 실제 선호도가 0.7에서 0.3으로 감소했을 때 각 알고리즘을 2000번의 반복 학습하였다.

학습률 0.01 기준으로 한 Voting EM 알고리즘의 실험 결과는 다음과 같다.

[표 2] 학습률 0.01에서 Voting EM 알고리즘의 실험 결과
[Table 2] Experimental results of Voting EM algorithm at learning rate of 0.01

	행동0	행동1	행동2	행동3	행동4	행동5
상태0	0.7381688	0.2615705	0	0	0	0
상태1	0	0	0.6959419	0.30327794	0	0
상태2	0	0	0	0	0.6857611	0.30508405

[그림 8] 학습률 0.01에서 Voting EM 알고리즘의 선호도 변화 그래프

[Fig. 8] The graph of preference change of Voting EM algorithm at learning rate of 0.01

실험결과 [그림 8]에서 선호도 확률 0.7로 가정한 행동0, 행동2, 행동4의 선호도 그래프는 실제 선호도 확률에 가깝게 수렴하는데 약 1400회, 선호도 확률 0.3으로 가정한 행동1, 행동3, 행동5의 선호도 그래프는 실제 선호도 확률에 가깝게 수렴하는데 약 700회 가량 걸리는 것을 확인할 수 있다.

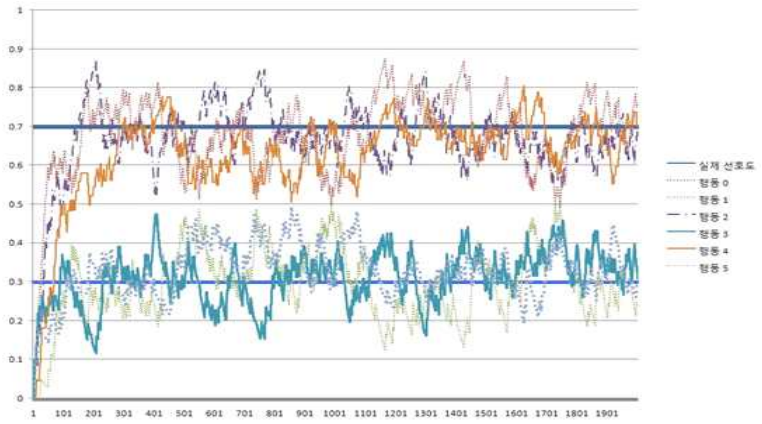
반면에 제안된 알고리즘에서 감쇠율 = 0.7, 행동에 대한 보상값 R을 동일하게 주었을 때, 학습률 0.01에 대한 실험 결과는 다음과 같다.

학습률 0.05 기준으로 한 Voting EM 알고리즘의 실험 결과는 다음과 같다.

[표 3] 학습률 0.05에서 Voting EM 알고리즘의 2000회 실행에 대한 CPT

[Table 3] CPT for 2000 runs of the Voting EM algorithm at a learning rate of 0.05

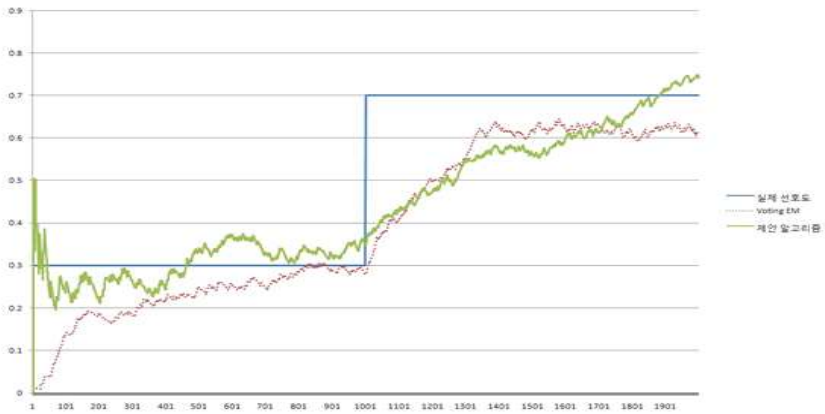
	행동0	행동1	행동2	행동3	행동4	행동5
상태0	0.76131004	0.23868984	0	0	0	0
상태1	0	0	0.692283	0.30771708	0	0
상태2	0	0	0	0	0.70171636	0.29828358



[그림 9] 학습률 0.05에서 Voting EM 알고리즘의 선호도 변화 그래프
[Fig. 9] The Graph of Affinity Change of Voting EM Algorithm at Learning Rate of 0.05

실험결과 [그림 9]에서 선호도 확률 0.7로 가정한 행동0, 행동2, 행동4의 선호도 그래프는 실제 선호도 확률에 가깝게 수렴하는데 약 300회, 선호도 확률 0.3으로 가정한 행동1, 행동3, 행동5의 선호도 그래프는 실제 선호도 확률에 가깝게 수렴하는데 약 150회 가량 걸리는 것을 확인할 수 있다.

선호도의 변화에 따른 학습결과 제안된 알고리즘에서 행동에 대한 보상값을 서로 다르게 주었을 때의 실험 결과는 다음과 같다.



[그림 10] 학습률 0.01에서 선호도 증가에 따른 학습결과
[Fig. 10] Learning result with increasing preference in learning rate 0.01

[그림 11] 학습률 0.01에서 선호도 감소에 따른 학습결과

[Fig. 11] Learning result by decreasing preference in learning rate 0.01

실험결과에서 Voting EM 알고리즘에 비해 제안된 알고리즘이 초기 학습속도가 빠르다는 것을 알 수 있다. 이후 선호도에 대해 충분한 학습이 이루어진 상태에서 선호도가 급격하게 변화하였을 경우 두 알고리즘 모두 변화한 선호도에 느리게 적응하는 모습을 보인다. 이러한 선호도의 급격한 변화에 빠르게 적응하기 위해서는 보다 높은 학습률을 적용할 필요가 있다.

5. 결론 및 향후 연구과제

본 논문에서 제안한 지능형 아이콘은 정보수집 / 필터링 모듈을 통해 게임환경으로부터 플레이어의 현재 상태와 현재 상태에서 가능한 행동 목록을 받아 온다. 받아온 상태와 행동의 쌍과 플레이어 모델에 저장된 선호도 특성값을 기반으로 플레이어가 가장 선호하는 행동을 사용자 인터페이스를 통해 제공한다. 사용자 인터페이스를 통해 입력된 플레이어의 입력은 지능형 아이콘의 학습모듈로 피드백되어 입력된 행동에 대한 플레이어 모델의 선호도 특성값을 학습하게 된다. 본 논문에서 제안한 학습 알고리즘의 실험 결과는 기존의 Voting EM 알고리즘에 비해 빠른 초기 학습 속도를 보여준다. 0.01의 학습률을 적용한 Voting EM 알고리즘을 이용하여 선호도를 학습하였을 경우 실제 선호도에 수렴하기 위해서 약 700~1400회의 학습을 진행해야하지만 본

논문에서 제안한 알고리즘의 경우 동일한 학습률 0.01에서 약 300회로 더 빠르게 수렴한다. 이는 플레이어의 선호도 특성을 게임 인터페이스에 효과적으로 반영할 수 있다는 것을 뜻한다. 하지만 낮은 학습률의 경우 선호도의 변화에 빠르게 적응할 수 없기 때문에 이러한 선호도의 온라인 학습 알고리즘을 실제 게임에 적용하기 위해서는 고정된 낮은 학습률을 사용하는 것보다는 상황에 따라 학습률을 가변적으로 적용할 필요가 있다. 본 논문에서는 명시적 피드백과 암시적 피드백을 나누어 플레이어의 피드백 방법에 따라 서로 다른 학습률을 적용하였다. 또한 베이지안 네트워크를 적용하여 설계한 플레이어 모델은 게임 디자인 단계에서 모델의 선호도에 대한 초기값을 쉽게 넣을 수 있기 때문에 초기 학습 시간을 줄여줄 수 있다.

ISSN: 2508—9099 JDAEM

References

[1] P. Maes, In Software Agents-Papers from the 1994 Spring Symposium, (1994), March 21–23; Palo Alto, California.

[2] M. J. Pajjani, IEEE INTELLIGENT SYSTEMS. (2000).

[3] A. Jameson, User Modeling and User-Adapted Interaction. (1996), Vol.5, No.3-4, pp.193-251.

[4] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, USA, (1988).

[5] T. Paek, E. Horvitz, Conversation as action under uncertainty, Proc. 16th Conf. on Uncertainty in Artificial Intelligence. (2000) San Francisco, USA.

[6] T. Stephenson, IDIAP-Rearch Report. (2000).

[8] I. Cohen, A. Bronstein, F. G. Cozman, HPL-2001-55. (2001).

[9] I. Cohen, A. Bronstein, F. G. Cozman, HPL-2001-156. (2001).

[10] S. Z. Zhang, An application of onlice learning algorithm for Bayesian network parameter, 2003 International Conference on Machine Learning and Cybernetics, (2003), November 5-5, Xi'an, China.

[11] I. Ghory, Department of Computer Science, University of Bristol, (2004)

[12] T. M. Mitchell, Machine Learning, McGraw Hill, USA, (1997).

[13] I. H. Park, W. H. Lee, Journal of the Korean society for computer game. (2008), Vol.115, pp.43-47.