Proceedings of KFIS Spring Conference 2006 Vol. 16, No. 1

Dynamic-FSM을 위한 사용자 모델링 방법

User Modeling Method for Dynamic-FSM

윤태복[1], 박두경[2], 박교현[3], 이지형[4]

[123]경기도 수원시 성균관대학교 컴퓨터공학과
E-mail: {tbyoon[1],hardme[2],megagame[3]}@skku.edu

[4]경기도 수원시 성균관대학교 컴퓨터공학과
E-mail: jhlee@ece.skku.ac.kr

요 약

게임의 재미요소를 증대 시키고, 게임 생명주기(Life-Cycle)를 늘어나게 하기 위해 다양한 방법이 연구 중이다. 현실감 있는 그래픽 효과와 뛰어난 음향 효과 등과 함께 게임 플레이어의 게임 스타일이 반영된 게임을 만들기 위한 방법이 대표적이 예라 할 수 있다. 그 중 게임 플레이어의 스타일을 게임에 다시 이용하기 위해서는 플레이어의 인지과정이 요구되며, 인지된 결과를 이용하여 플레이어를 모델링(User Modeling)한다. 하지만, 게임의 종류와 특성에 따라 다양한 게임이 존재하기 때문에 플레이어를 모델링하기 어렵다는 문제를 가지고 있다. 본 논문에서는 게임에서 정의된 FSM(Finite State machine)을 이용하여 플레이어가 선택한 행동 패턴을 분석하고 적용하는 방법과 다양한 게임에서 이용 할 수 있는 스크립트 형태의 NPC 행동 패턴 변경 방법을 제안한다. 플레이어의 데이터를 분석하여 얻은 결과는 FSM을 변경하여 새로운 행동을 보이는 NPC(Non-Player Characters)를 생성하는데 사용되며, 이 캐릭터는 게임의 특성과 플레이어의 최신 행동 패턴 경향을 학습한 적응형 NPC라 할 수 있다. 실험을 통하여 사용자의 행동과 유사한 패턴을 보이는 NPC의 생성을 확인할 수 있었으며, 게임에서 상대적인 또는 적대적인 캐릭터로 유용하게 사용 될 수 있다.

Key Words : Game AI, User Modeling, Adaptive-NPC, Dynamic-FSM

1. 서 론

사용자 모델링(User Modeling)은 특정 시스템의 목적을 보다 효과적으로 수행할 목적으로, 시스템의 환경에서 발생한 사용자의 데이터를 수집하고 분석하는, 사용자 인지(User Cognitive)과정이라 할 수 있다[1]. 이 사용자 모델링 기술은 게임, 교육[2], 유비쿼터스, 웹 마이닝(Web Mining)등에서 사용자의 프로파일 정보를 분석하여 다양하게 사용되고 있는데, 특히 게임 환경에서는 게임 플레이 과정에서 수집된 플레이어의 게임 데이터 분석을 통한 서비스가 활발히 연구 중이다. 피터 몰리뉴[3]의 Black&White에서는 플레이어의 게임 운영에 따라 적응된 모습을 보이는 크리처라는 동반자 NPC를 볼 수 있으며, EA의 심즈라는 게임의 경우에도 플레이어의 게임 운영에 따라 상대적인 모습을 보이는 NPC를 확인 할 수 있다. 그러나 기존의 게임에서 사용자 모델링 기술은 주로 콘솔 게임이나 패키지 게임에 사용되었는데, 이는 다양한 게임이 가지고 있는 특성에 의존적인 모델링 방법이 필요하기 때문이다. 본 논문에서는 게임에서 초기에 정의된 NPC (Non-Player Characters)의 행동 패턴 설정을 플레이어 모델링을 이용하여 변경하거나 추가함으로써, 게임 초기에 설정되어있던 정적인 행동 패턴에서 게임 플레이어의 성향이 반영된 D-FSM(Dynamic Finite State Machine) 방법을 제안한다. 제안하는 방법은 다양한 종류와 특성을 가지는 게임에 대하여 유연하게 사용하기 위하여 행동 패턴을 정의하는 스크립트 언어를 제공한다. 플레이어의 모델링에 사용된 방법은 기계학습(Machine Learning)의 한 방법인 의사 결정 트리(Decision Tree)를 이용하였고, 의사 결정 트리 방법에 사용된 속성은 스크립트 언어를 통해

정의된 행동 결정 속성을 이용하였다. 여기서 속성은 플레이어의 행동을 결정하는 요인을 제공한 요소라 할 수 있다. 분석하여 얻은 정보는 게임 플레이어와 동반자 역할을 하는NPC나 적대적인 관계를 갖는 NPC생성에 사용 할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구에 대해 소개하고, 3장에서는 제안 하는 방법인 D-FSM을 위한 사용자 모델링 방법을 소개 한다. 4장에서는 제안하는 방법을 위한 시뮬레이션과 결과를 확인한다. 마지막으로 5장에서는 결론과 향후 연구로 맺는다.

2. 관련 연구

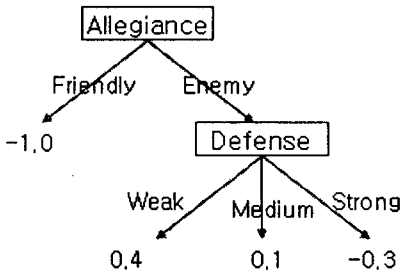
2.1 의사 결정 트리(Decision Tree)

의사 결정 트리(decision tree)는 데이터 마이닝의 분류 작업에 주로 사용되는 기법으로, 과거에 수집된 데이트의 레코드들을 분석하여 이들 사이에 존재하는 패턴, 즉 분류별 특성을 속성의 조합으로 나타내는 분류모형을 트리의 구조로 표현하는 것이다. 사용자가 트리를 쉽게 이해할 수 있고, 학습 및 검증이 쉽다는 장점 때문에, 데이터의 분류 및 예측 그리고 학습에 자주 사용된다. 앞서 설명한 Black & White 라는 게임에 적용되어 유저로부터 큰 호응을 얻었는데, 게임에서 플레이어는 크리처를 쓰다듬거나 때림으로써 크리처에게 긍정적인 또는 부정적인 피드백을 제공한다. 그런 피

<표1> 게임에서 수집된 데이터

	Attributes			Target
	Allegiance	Defense	Tribe	
D1	Friendly	Weak	Celtic	-1.0
D2	Enemy	Weak	Celtic	+0.4
D3	Friendly	Strong	Norse	-1.0
D4	Enemy	Strong	Norse	-0.2
D5	Friendly	Weak	Greek	-1.0
D6	Enemy	Medium	Greek	+0.2
D7	Enemy	Strong	Greek	-0.4
D8	Enemy	Medium	Aztec	0.0
D9	Friendly	Weak	Aztec	-1.0

드백에 의해, 크리처가 경험으로부터 배운 것을 반영하는 의사 결정 트리가 만들어진다. 크리처는 그 의사결정 트리를 통해서 주어진 물체가 자신의 배고픔을 해결하는 데 사용될 수 있을 지를 판단한다. 표1과 그림1은 Black & White에서 크리처가 다른 크리처를 공격했을 때 얻은 피드백(target) 데이터와 해당 데이터를 학습한 결정트리의 모습이다.



<그림 1> 결정 트리의 생성

D2나 D6에서처럼 상대 크리처가 적대적(enemy)이고 방어도가 낮거나(weak) 보통(medium)일 때에는 긍정적인 피드백(target > 0)을 받았었기 때문에 위와 같은 결정트리를 갖은 크리처는 앞으로도 같은 상황이 벌어지면 상대 크리처를 공격 할 것이다.

2.1 FSM(Finite State Machine)

상태기계(state machine) 또는 유한상태기계(finite state machine)는 널리 쓰이는 소프트웨어 설계 패턴의 하나로, 유한한 개수의 상태들로 구성된 기계를 말한다[4]. FSM은 상태들과 전이들의 유한한 집합으로 정의하며, 주어진 시점에서 오직 하나의 상태만 활성화된다. 실제 적용에서 각 상태는 하나의 특정한 행동을 나타내는데, NPC는 현재 활성화된 행동을 자신의 과제로 삼아서 그것을 수행한다. 특정한 사건이 일어나면 한 상태는 다른 상태로 전이되며, 이 때 사건은 상태가 환경을 주기적으로 점검해서 인지할 수도 있고, 사건이 상태에게 통지할 수도 있다. 단순한 구조를 가지고 있기 때문에 게임에서 가장 널리 사용되는 기법이고, 종종 다른 인공지능 테크닉과 혼합되어 사용되기도 한다. 그림6은 RPG 게임에서 몬스터의 감정을 FSM으로 표현한 것이다.

몬스터는 6가지 상태를 갖고 있고 각 상태는 다른 상태로 천이 될 수 있음을 볼 수 있다. 예를 들어 대기 상태에 있던 몬스터가 플레이어로부터 공격을 받으면 돌격 상태로 천이가 이루어지고, 다치게 되면 도망 상태로 변할 것이다.

3. D-FSM을 위한 사용자 모델링

게임 플레이어를 적응한 NPC를 생성하기 위한 방법은 다양하게 연구되고 있는데, 가장 일반적인 방법은 게임을 진행한 플레이어의 프로파일 데이터를 분석하고 그 결과를 이용하여 게임 플레이어를 모델링하고 그 모델링된 데이터는 적응형 NPC를 생성하는 기반 지식으로 사용 할 수 있다.

3.1 적응형 NPC(Non-Player Characters)

NPC는 게임내의 요소중 플레이어에 반응하는 에이젼트를 말하며, 상점내 상인이나 길안내, 팀원, 몬스터 또는 환경적인 요소로 기후, 아이템, 지형 변화 등을 말하다. 또한 이런 NPC는 소스내 직접 코딩하여 생성하고 제어하는 방법을 이용하거나 스크립트를 이용한다. 그리고 좀더 개선된 방법으로는 사용자의 성향에 따라 다양한 모습을 보이는 적응형NPC를 예로 들 수 있는데, 표 2와 표 3는 제공되는 NPC에 따른 기능과 응용 게임을 나타내고 있다.

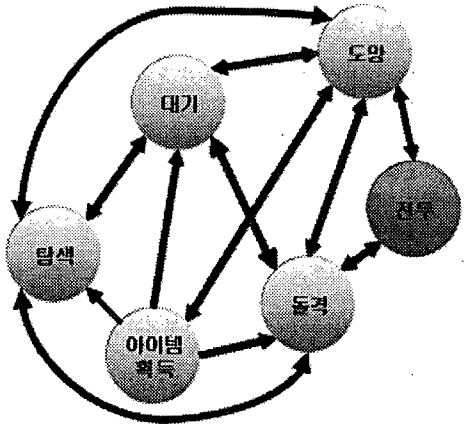
<표 2> 초보적인 NPC

초보적 NPC
● 플레이어 능력 고려 않됨 ● 동일한 상황에 대해서 동일한 반응을 보임 ● 게임 패턴이 파악됨 ● 상황에 따라서 고정된 행동 패턴 선택 예) Space Invaders, Pac-Man , Donkey Kong 등

<표 3> 개선된 NPC

적응형 NPC	
능력(수치) 적응 NPC	행동 적응 NPC
● 플레이어의 능력에 따라 NPC의 능력 변함 (명중률,속력,에너지 등) ● 게임 난이도 동적조절 ● 게임 수명 연장 예) FIFA, Call of Duty, Medal of Honor 등	● 플레이어의 성향이나 능력에 따라 NPC 행동 패턴 변화 ● 동적인 게임 전개 ● 게임의 현실감을 높여 재미 요소 기여 예) Black & White, The Sims 등

표에서와 같이 정적인 NPC보다는 플레이어의 게임 스타일에 따라 수치적인 변화를 보이는 NPC가 게이머로 하여금 더 흥미를 유발시키고, 수치적인 행동적인 변화를 보이는 NPC보다는 행동적인 반응을 보이는 NPC에 대해 플레이어는 더 많은 재미를 느낀다. 그리고, 이런 NPC생성을 위해서는 플레이어의 인지와 모델링과정이 매우 중요한 요소로 작용한다. 게임에서 NPC의 행동은 게임 제작과 함께 결정되며 스크립트(Script)형태로 정의되어 사용된다. 스크립트 형태로 제공되기 때문에 게임 서비스 중간에 게임 기획자에 의해 게임 밸런스 조절이나 난이도 조절을 위해 사용되기도 하지만 이는 게임 플레이어의 성향이 반영 되었다기 보다는 기획자의 주관적이 판단이 반영된다. 그림2는 FPS(First Person Shooting)이나 MMORPG (Massive Multi-player On-line Role Playing Game)에서 일반적으로 사용할 수 있는 NPC의 상태 전이를 표현한 예를 보여주고 있다.



<그림 2> 일반적인 NPC의 FSM

대기, 탐색, 돌격, 전투, 도망, 아이템 획득의 상태는 상황에 따라 전이를 발생하여 각각의 행동을 보여주게 되는데 가장 대표적인 상황변수는 HPC(Human Player Character)의 에너지와 NPC의 에너지 그리고 두 캐릭터간 공격력의 차이, 이동 속력의 차이, 이동 거리의 차이가 가 기본 지표로 사용될 수 있다. 추가적인 변수로는 캐릭터가 가지고 있는 아이템이나 공격력과 이동 속력의 능력외에 다른 능력 수치나, 주변의 다른 동료 캐릭터 존재 여부 등이 있다. 본 예제에서는 이해를 돕기 위해 가장 기본적인 구조의 상태 전이와 전이 상황 변수를 선택하여 설명한다. 표 4는 그림 2의 상태 전이를 결정하는 변수의 타입을 나타내고 있다.

<표 4> FSM에 사용되는 변수의 속성

변수	속성 (값)
플레이어 체력(P_{hp})	**Low(L), Mid(M), High(H)**
적 체력(E_{hp})	**Low, Mid, High**
플레이어와 적의 체력 차이(HP_d)	**0**($P_{hp} < E_{hp}$), **1**($P_{hp} > E_{hp}$)
플레이어와 적 사이의 거리(**D**)	**Near(N), Inside(I), Outside(O)**
플레이어와 적의 이동 속도 차이(**S**)	**-1, 0, 1** S>0면 플레이어가 더 빠르고 S<0이면 적의속도가 더 빠르다
플레이어와 적의 공격력 차이(**P**)	**-1, 0, 1** (P>0면 플레이어가 더 강하고 P<0이면 적이 더 강하다.)

<표 5> 변수 속성에 따른 전이 규칙

before	P_{hp}	E_{hp}	HP_d	D	S	P	after
탐색	H	H	0	O	1	1	탐색
탐색	M	H	0	I	-1	0	돌격
탐색	M	M	1	I	1	-1	대기
탐색	H	L	1	I	0	0	도망
돌격	H	H	1	I	0	-1	돌격
돌격	H	M	1	I	1	-1	대기

before	P_{hp}	E_{hp}	HP_d	D	S	P	after
돌격	L	L	1	N	1	-1	전투
돌격	M	M	0	I	-1	0	돌격
돌격	L	M	0	O	1	0	탐색
대기	H	L	1	O	-1	1	아이템
대기	M	M	1	N	0	0	돌격
대기	M	H	0	I	-1	-1	돌격
대기	L	H	0	O	0	1	탐색
도망	L	L	1	I	1	1	도망
도망	H	L	1	I	1	1	아이템
도망	L	L	0	N	0	0	전투
도망	M	M	1	O	0	0	대기
도망	M	L	1	I	-1	-1	돌격
도망	H	H	0	O	1	1	탐색
전투	L	M	0	N	-1	-1	전투
전투	H	L	1	N	1	0	도망
전투	L	H	0	I	0	-1	돌격
아이템	H	M	1	I	0	1	도망
아이템	L	M	0	I	0	1	돌격
아이템	M	M	0	O	0	0	대기
아이템	M	H	0	O	1	1	탐색

표 5는 그림 2의 상태전이에 대하여 전이 조건을 나타내고 있으며, 그림 2에서와 같이 6가지 상태에 대하여 발생 할 수 있는 전이와 조건을 정의하였다. 표 5의 경우 생성될 수 있는 전이 규칙의 개수는, 상태 전이 수 * P_{hp}*E_{hp}*HP_d*D*S*P (= 6 * 3 * 3 * 2 * 3 * 3 * 3)= 2916개 이어야 한다. 하지만, 이 모든 규칙을 정의하여 사용하기에는 너무 수가 많

```
<NPC_Script>
    <FSM_define>
        <FSM1> 탐색 </FSM1><FSM2> 돌격 </FSM2>
        <FSM3> 대기 </FSM3><FSM4> 도망 </FSM4>
        <FSM5> 전투 </FSM5>
        <FSM6> 아이템획득 </FSM6>
    </FSM_define>
    <Attribute_define>
        <Attribute1> P_hp : H, M, L </Attribute1>
        <Attribute2> E_hp : H, M, L </Attribute2>
        <Attribute3> HP_d : 0, 1 </Attribute3>
        <Attribute4> D : N, I, O </Attribute4>
        <Attribute5> S : -1, 0, 1 </Attribute5>
        <Attribute6> P : -1, 0, 1 </Attribute6>
    </Attribute_define>
    <NPC_rule_define>
        <rule1> 탐색, H, H, 0, O, 1, 1, 탐색 </rule1>
        <rule2> 탐색, M, H, 0, I, -1, 0, 돌격 </rule2>
        <rule3> 탐색, M, M, 1, I, 1, -1, 대기 </rule3>


      <rule25>아이템획득, M, M, 0, O, 0, 0,대기</rule25>
      <rule26>아이템획득, M, H, 0, O, 1, 1,탐색</rule26>
    </NPC_rule_define>>
</NPC_script>
```

<그림 3> 표5에 따른 스크립 파일

다. 더구나 속성 하나가 추가 될 때마다 규칙의 개수가 지수적으로 증가한다면 게임에서 추가 속성을 적용하는 것은 매우 어려운 작업일 것이다. 그렇게 때문에 상태 전이도<그림 2>에 나타난 부분에 대해서만 규칙을 정의하고 다른 조건에 대해서는 처리 하지 않는다. 정의되지 않은 규칙에 대해서는 게임 진행중에 플레이어로부터 얻은 정보를 이용하여 규칙을 자동적으로 추가/갱신 할 수 있다.

표 5와 같이 정의된 상태 전의 규칙은 그림 5와 같은 형태의 스크립트로 나타내어 게임에서 사용된다. 스크립트에 정의된 상태 전이 속성에 따른 플레이어의 게임 데이터를 수집하고 결정 트리를 이용하여 분석한다. 우선 플레이어의 게임 데이터를 수집하고 분석하는 과정에서 플레이어가 선택한 행동에 대한 상태 정의가 이루어 져야 한다. 표 6은 일반적인 게임에서의 상태가 어떤 원인을 가지고 있을 것인가에 대한 가설을 기반한 정의를 나타내고 있다. 정의된 가설을 기반으로 플레이어의 행동과 상황을 표 6의 코드상 의미와 비교하여 일치 한다면 그 때의 상태를 표에서와 같이 정의한다. 예를 들어 플레이어의 시야에 적이 있고, 플레이어와 적이 점점 가까워 진다면, 표 6에 의해 그 상태는 돌격으로 정의하고 그때의 게임데이터(P_{hp}, E_{hp}, HP_d, D, S, P)를 수집하여 저장한다.

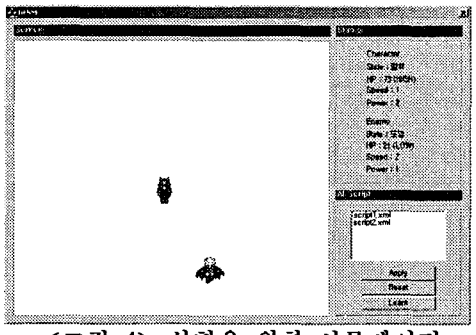
<표 6> 가설을 기반한 각 상태의 정의

상태	행동적 의미	코드상 의미
대기	캐릭터가 특정 행동을 보이지 않을 경우	T 시간동안 캐릭터 환경변수가 변하지 않을 때 (T : threshold)
탐색	적을 찾는 행위	적이 시야 바깥에 있고, 캐릭터가 이동 할 때
돌격	적을 향해 이동	적이 시야 안에 있고, 캐릭터와 적과의 거리가 가까워 질 때
전투	적과의 전투	공격 키 값이 입력되어 졌을 때
도망	적으로부터 벗어남	적이 시야 안에 있고, 캐릭터와 적과의 거리가 멀어 질 때
아이템 획득	체력을 회복하기 위한 아이템을 찾는 행위	도망 중 캐릭터와 아이템의 거리가 가까워 질 때

수집된 데이터는 동일한 데이터와 잘못된 데이터(Noise)를 제거하는 전처리(Prepocessing) 과정을 거쳐 결정 트리를 이용하여 분석한다. 분석 후에 나온 결과는 그림 5의 <NPC_rule_define> 부분을 변경하거나 새로 추가하는 정보로 활용한다. 새로이 생성된 NPC 규칙은 게임 제작시에 생성된 정적인 규칙이 아닌 플레이어의 게임 데이터를 이용하여 생성된 정보이며, 적응형 NPC를 생성하는 기반 자료로 사용할 수 있다.

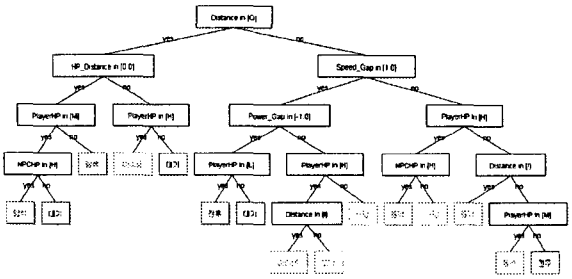
3. 실험

제안하는 방법의 실험을 위한 그림 3의 스크립트를 읽어 NPC의 상태 및 행동 규칙을 설정하고 게임 진행중에 플레이어의 게임 데이터를 수집 할 수 있는 시뮬레이터를 구현하였다 <그림 4>.

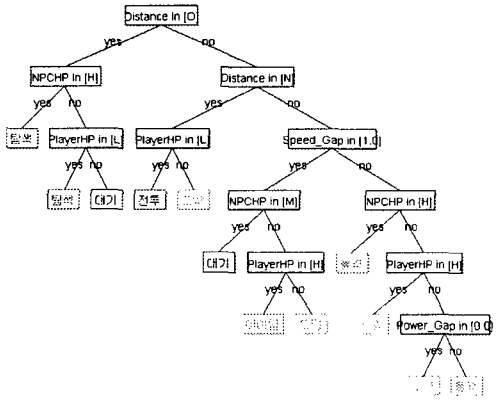


<그림 4> 실험을 위한 시뮬레이터

시뮬레이터는 플레이어의 상태가 바뀔 때 마다 표 4에서 정의한 플레이어 체력, 적 체력, 플레이어와 적의 체력 차이, 플레이어와 적 사이의 거리, 이동 속도 차이, 공격력 차이를 얻어 오고, 그때의 상황을 표 6의 가설에 기반한 상태로 정의한다. 그림 5은 초기에 정의된 상태 전이 조건인 표 5를 분석하여 얻은 결과 이며, 그림 6은 시뮬레이터를 이용하여 얻은 결과를 분석하여 얻은 결과이다.



<그림 5> 초기에 정의된 NPC의 상태 정의 분석결과



<그림 6> 게임에서 수집한 플레이어의 게임 데이터를 이용한 분석 결과

3. 결론 및 향후 연구

게임에서 제공되는 NPC는 제한적인 동작과 정적인 행동 패턴을 보인다는 것이 일반적인 특징이다. 이런 정적인NPC의 행동은 게임 플레이어로 하여금 쉽게 게임에 대하여 재미를 떨어트리는 요소 중에 하나이다. 본 연구는 게임 플레이어의 게임 데이터를 수집하고 분석하여 새로운 NPC생성에 사용함으로써 창조적이고 다양성을 가지는 NPC를 제공할 수 있는 기술이다. 실험을 통하여 게임 데이터를 수집하고 분석하여 새로운 규칙을 얻었다. 새로운 규칙은 스크립트 형태로 변환하여 플레이어와 유사한 행동 패턴을 가지는 NPC를 생성할 수 있는 기술에 사용 가능하다.

본 논문에서는 플레이어의 상태를 인지하기 위하여 가설을 기반으로 상태들을 정의 하였다. 이 가설 기반의 상태 정의는 다양한 게임에 적용하기에 부적절하다는 단점과 이론적 설득력이 떨어진다는 단점이 있다. 향후 연구로는 전문가 지식(Expert Knowledge)을 반영한 방법을 이용하여 가설을 보완하고, 검증하는 방법에 대한 연구가 필요하겠다.

참 고 문 헌

[1] Brajnik G., Guida G., Tasso C., "User modeling in expert man-machine interfaces: a case study in intelligent information retrieval,"Systems, Man and Cybernetics, IEEE Transactions on, Volume 20, Issue 1, Jan.-Feb. 1990, 166 185.

[2] Hyun J. C., Yong S. K, Jee H. L, Tae B. Y, "An Adaptive Learning System with Learning Style Diagnosis based on Interface Behaviors", International Conference on E-learning and Games, Edutainment 2006.

[3] Steve R., AI Game Programming Wisdom 2, Charles River Media, 2002.

[4] Eric D., Game Programming Gems, Charles River Media, 2000.

[5] John E. L., "Using a Computer Game to Develop Advanced AI," Computer IEEE Journal, Volume 34, No. 7, July 2001, 70-75.

[6] Pieter S., Ida S., Eric P., "Online Adaptation of Game Opponent AI in Simulation and in Practice,"Proceedings of the 4th international Conference on Intelligent Games and Simulation, GAME-ON 2003, 93 100.