한국컴퓨터정보학회지
제25권 제1호, 2017. 6.

컴퓨터 게임에서의 인공지능 기술에 관한 연구

• 송규진 (슬로스)

I. Introduction

인공지능에 관련된 기술의 필요성이 요구된 것은 1940년대 말이다. 이 시기에 미국의 대학과 기업 연구실에서는 하나의 목표를 달성하기 위해서 많은 노력을 집중하고 하였다. 그렇지만 그 시기의 연구에서는 큰 성과는 없었고, 그 기술이 완성되는 순간 인류의 역사는 새로운 세계를 맞이할 것이라는 확신을 가지고 있었다.

바로 그 기술은 생각하는 컴퓨터 즉, AI라고 알려져 있는 '인공지능'을 가진 컴퓨터를 만드는 것이었다. 그러한 능력을 갖춘 컴퓨터가 최종적으로 어떠한 모습을 띠게 될지는 알 수 없지만, 예를 들어 충분한 자료를 내장하고 있는 인공지능 컴퓨터는 구체적인 자료들을 분석하여 더욱 확실하고 실질적인 경제 이론을 만들어낼 수 있을지도 모른다. 또한 인간을 대신해 산업 현장에 뛰어드는 로봇의 두뇌를 활용된다면 인간은 훨씬 나은 삶을 추구할 수 있게 될 것이라고 생각했다.

최근 글로벌 기업의 인공지능 기술이 적용된 알파고와 바둑프로 기사의 대결이 세계의 관심을 받으며 인공지능 기술이 주목을 받고 있다.

과거 게임은 실감 향상을 위해 사실적인 그래픽 처리와 자연스러운 모션처리 기술 위주의 연구에 많은 노력을 기울여 왔다면 2016년부터는 인간처럼 사고하는 인공지능의 기술 접목이 새로운 관심을 모아지고 있는 것이다.

게임에서 인공지능은 플레이어 역할, 비 조정플레 이어 역할, 팀원역할 등을 사람처럼 대신 수행하는 부분이다. 여기에는 사람처럼 사고하는 학습능력을 포함하여 사람처럼 행동하는 지능적 행동도 포함되어 있다.

과거게임에서는 인간과 유사한 행동을 하는 캐릭터를 제작하기 위해 많은 노력과 비용을 기울였다면, 지금은 인간과 유사한 사고를 하도록 인공지능정보 기술이 중요한 부분을 차지하고 있다는 것이 획일하게 다른 점이라고 할 수 있다.

이제는 그 상대를 컴퓨터로 충분히 대신할 수 있게 된다. 이를 위해서는 게임요소 중에 서도

지능형 인공지능 분야의 지속적인 연구가 요구되고 있으며, 컴퓨터 게임에서도 마찬가지로 '인공지능'에 대해 관심이 모아지고 있다.

따라서 본 연구에서는 "컴퓨터 게임에서의 인공지능" 기술에 관한 연구에 대해 모색에 보고자 한다.

II. Preliminaries

1. Definition of Game

1.1 General definition

1960년을 전후해 등장한 컴퓨터 게임(computer game)은 1980년대 초반에 이르러 하나의 대중적인 문화형식으로 자리 잡은 이후 급속한 발전을 거듭하여 현재에는 가장 강력한 멀티미디어 문화형식의 하나로 영화, 텔레비전, 음반 등 기존의 미디어들과 경쟁하고 있다. 그러나 그 역사나 사회문화적 위상에 비하면 컴퓨터 게임에 대한 정의는 아직 뚜렷하게 정립되어 있지 못한 것으로 보인다. 이제까지 소개된 컴퓨터 게임에 대한 정의들은 대체로 피상적 정의, 법률적 정의, 이론적 정의 등 세 가지 유형으로 나누어질 수 있다.

먼저 피상적 정의를 살펴보면, 흔히 컴퓨터 게임은 "컴퓨터를 이용해 이루어지는 게임 또는 놀이" 정도로 막연하게, 동어 반복적으로 정의될 뿐이다. 한 예로 먼저 한국문화정책개발원은 《전자오락 게임의 문화 정책적 접근 방안》이란 보고서에서 다음과 같이 규정하고 있다.

두 번째는 법률적 정의이다. 법률적으로 컴퓨터 게임이 기

존의 비디오물과 독립적으로 정의된 것은 1999년 음반·비디오물 및 게임물에 관한 법률(1999.1.6.)이 처음이었다. 2002년 일부 개정된 동 법률(제6627호, 2002.1.26)에 따르면, "게임물"이라 함은 "컴퓨터프로그램 등 정보처리 기술이나 기계장치를 이용하여 오락을 할 수 있게 하거나 이에 부수하여 여가선용, 학습 및 운동효과 등을 높일 수 있도록 제작된 영상물 및 기기"를 말한다(제2조 3항).

이러한 몇 가지 정의들을 검토하면서 컴퓨터 게임을 "컴퓨터(PC만을 일컫는 것이 아니라 정보처리 능력을 가진 장치로서의 컴퓨터)라는 하드웨어 상에서 흥미를 유발하는 내용물이 어떤 규칙에 의거한 선택 결정과정을 통해 진행되어 나가도록 컴퓨터 프로그램에 의하여 제작된 것"으로 정의한다. 이 정의는 이론적 논의를 기초로 한 것은 아니지만, 컴퓨터와 컴퓨터 프로그래밍, 규칙, 선택·결정과정 등 외적인 요소들을 포함하고 있다.

1.2 Theoretical definition

이러한 정의들과 달리 이론적 정의들은 게임 및 게임 진행과정을 구성하는 다양한 요소들을 가지고 컴퓨터 게임을 개념화하고자 한다. 먼저 컴퓨터 게임이라는 용어보다 비디오 게임이라는 용어를 선호하는 월프(Wolf, 2001b, pp. 14-19)는 비디오 게임을 정의하기 위해서는 "비디오"로서의 위상과 "게임"으로서의 위상을 고려해야 한다고 본다.

그에 따르면, 전자에는 스크린에서 상호작용적으로 플레이하기 위한 시각적 디스플레이 요소들이, 그리고 후자에는 갈등, 규칙, 플레이어의 능력 활용, 게임 결과에 대한 평가 등의 요소들이 포함된다. 그리고 게임을 가능케 해주는 컴퓨터는 참여자(또는 플레이어)로서, 그리고 심판원(referee)으로서의 위상을 갖는다고 본다.

한편 쥴(Juul, 1999)은 컴퓨터 게임을 "형식적으로 규정된 규칙에 기반 해 일어나는 활동이자 플레이어의 행위에 대한 평가를 수반하는 활동"으로 정의하고 있는데, 이것은 규칙과 행위 결과에 대한 평가를 강조하는 것이다.

이렇듯 여러 학자들은 게임에 대한 이론을 정의내리고 있지만, 여전히 게임에 대한 정의는 확립되지 못하고 있다.

2. History of the game

인류의 기원과 함께 놀이는 발생하였고, 필요에 따라서 존속하였다. 이러한 놀이를 지속적이고 보다 풍부하게 즐기기 위해 인간은 놀이기구를 발명한 것이다. 수공업 제품이었던 놀이기구는 한 헌신적인 과학자에 의해서 처음으로 전자 장비를 동반한 컴퓨터 게임으로서 그 모습을 나타내었다. 하지만 이러한 획기적인 발명에도 불구하고 초기의 컴퓨터 게임은 놀이를 돕는 무상성의 존재로 여겨졌으며 상업화의 대상이나 이윤추구의 대상이 아닌 공공의 소유물로 인지하였다. 그러나 이러한 컴퓨터 게임에 대한 관점은 새로운 시대를 맞는 미국의 사회적 분위기에서 매우 큰 변화를 맞이하게 된다(이형준, 2003).

이러한 컴퓨터 게임의 변화는 사람들이 순수하게 생각하는 것처럼 만인을 위한 놀이기구로서의 컴퓨터게임이 아닌 수익발생을 통해서 자본을 확장하고 발전하는 소비재의 가치를 가지는 자본주의의 새로운 형태의 상품으로 등장한다.

한편, 제 1단계 컴퓨터게임의 메사추세츠 공과대학(MIT)의 미디어 연구실에서 초대형 컴퓨터를 이용해 학생들이 만들었으며, 그 대표적인 것이 1962년 스티븐 러셀에 의해 제작된 "스페이스 위"라는 게임이다. 이 게임은 우주에서 벌어지는 전쟁을 모델로 한 것으로 뚱뚱한 시가모양의 두 대의 우주선을 컴퓨터가 창조해 낸 공간에서 전투를 벌이게 되며, 사용자는 스위치를 조작해 로켓의 좌우 방향을 선정하여 상대의 함정을 파괴하는 스토리의 게임이다(최유찬, 2002).

그러나 이 게임은 실험실 창작물로 그치게 되었지만 본격적인 게임의 역사의 문을 열게 한 것과 다름없다.

3. Formation and popularization of computer games

컴퓨터 게임의 가장 큰 형식적 변화를 일으킨 게임은 'Atari'사의 'Brick-Out'이다.

이 게임은 80년대 후반 우리나라 전자오락실이나 고속버스 터미널에 가면 흔히 볼 수 있었던 것으로 "블록깨기'게임의 최초의 형태였다. 이 게임은 현재까지도 여러 형식적 변화를 가하여 만들어지고 있다(문재영, 2003).

이러한 'Brick-Out'으로 시작되는 2단계 게임들의 탄생과 컴퓨터 게임의 몰입요소의 등장은 역사적 중요성이 어찌되었든지, 그 발생 배경에 있어서는 개발사들의 절박한 수익성을 위한 집착과 경쟁의 산물이며, 게임을 통해서 몰입과 중독을 사용자들에게 부여함으로써 끊임없이 화폐와 시간을 지속적으로 사용하게끔 유도하는 기재를 내포함으로써 경쟁에서 살

아남기 위한 개발사들의 의도적인 개발을 확장시킨 것이라고 볼 수 있다.

4. Various developments in the game

1990년대에 들어서면서 비디오 콘솔이 1980년대 초반처럼 다시 가정용 컴퓨터 게임의 기반으로 확대되어갔다. 이 이유 중의 하나는 당시 PC는, 하드웨어적 한계도 있었지만, 1980년대에 보여준 게임기로서의 잠재력에도 불구하고 일반인들에게 게임 기구가 아닌, 워드프로세싱, 회계처리, 출판 등에만 활용되는 범용 또는 업무용 컴퓨터로 인식되었고, IBM 계열의 PC들에서 보듯 실제로도 그렇게 활용되었기 때문이었다.

그러나 보다 직접적인 이유는 게임 소프트웨어가 카트리지(cartridge) 형태로 제공되었고, 게임기인 콘솔 자체의 가격도 상대적으로 저렴했기 때문이었다. 카트리지형 콘솔은 바로 바로 게임을 할 수 있다는 것 이외에도 다른 게임을 하기 위해 디스켓을 갈아 끼워야 했던 당시 컴퓨터와 달리 게임을 로딩 하는데 거의 시간이 소요되지 않는다는 이점을 갖는다.

또한, 반응시간(response)과 플레이(playability)의 즉각성이라는 강점과 더불어 보다 강력한 성능을 갖는 시스템으로 지속적인 개선이 이루어지면서 홈비디오 콘솔은 상당 기간 동안 지배적인 게임 기반으로 활용되었고, 그에 따라 기기의 판매와 게임의 개발이라는 측면에서 엄청난 성장을 거듭했다.

한편, 1990년대 초반 콘솔 시장은 닌텐도(Nintendo)와 세가(Sega)라는 두 일본 회사에 의해 거의 독점적으로 지배되고 있었다. 두 회사는 끊임없이 하드웨어의 성능 개선을 주도해 나갔고 자사의 콘솔과 게임에 대한 홍보와 마케팅도 강력하게 추진하였다. 중요한 것은 개선된 하드웨어의 성능을 그 기반에서 플레이되는 게임을 통해 보여주는 것이었다.

닌텐도는 수퍼 NES(수퍼 패미콤)을 통해 <마리오 브라더스>를, 세가는 제네시스(메가 드라이브)를 통해 <소닉: 고슴도치>를 제공하게 되는데, 이런 게임들은 기존의 게임들과 달리 본격적으로 게임에 캐릭터를 선보였다는 점에서 획기적인 일이었다.

또한, 지속적으로 보급이 확대된 PC 또한 1990년대에 들어서면서 강력한 CPU, CD-ROM, 사운드 카드, 그래픽 가속기 등 게임에 필요한 제반 시스템들을 갖춘 명실상부한 멀티미디어로 발전하여 이제는 그래픽, 사운드, 상호작용성 등의 측면에서 비디오게임 전용 콘솔 못지않은 정교한 게임, 소위 CD 게임을 가능케 해주었다. 그리고 무엇보다도 중요한 것은 1990년대 후반 이후 인터넷의 대중화와 고도화로 PC는 이제 다수의 게이머들 사이의 상호작용, 즉 멀티플레이어 게임이 가능한 온라인 게임(online game)의 기반이 되고 있다는 것이다.

게임 장르 측면에서 볼 때, PC 게임의 등장으로 슈팅, 격투, 플랫폼, 경주 등과 같은 아케이드형 게임의 비중이 축소되고 판타지, 어드벤처, 퍼즐 등과 같은 장르들이 늘어나면서 상대적인 균형이 회복하는 듯했지만, 1990년대 초반 이후 홈비디오 콘솔의 보급 확대로 액션 게임들의 비중은 다시 크게 늘어났다.

컴퓨터 게임은 1960년을 전후에 등장한 이후 10년도 채 안 돼 하나의 문화 산업(culture industry)으로 성장하였고, 20여년이 지나서는 현대 멀티미디어 문화의 주요한 한 축이 되었다. 그리고 게임은 이제 하드웨어 및 소프트웨어 생산의 산업화를 단계를 넘어 프로게이머에서 보듯 게임 행위 자체가 인기 스포츠처럼 산업화되는 단계로 진입하였다. 이는 다른 문화산업과 비교하면 엄청나게 빠른 성장이다. 게임이 산업화되면서 초창기 게임 회사들이 사라지고 새로운 게임기 제조 및 소프트웨어 개발 회사들이 등장하고 있으며, 이로 인해 인공지능 게임도 등장하게 된다.

III. Artificial intelligence applied to games

1. AI theory

1.1 The advent of artificial intelligence

오늘날의 컴퓨터관련 학문과 산업은 인간의 지능이 필요한 모든 분야에 사용하기 위해 인간두뇌의 영역으로 그 범위를 확장해가고 있으며 더 나아가 인간과 유사한 지능을 가진 첨단 컴퓨터 이론과 그 응용에 대해 많은 관심을 가지고 있다. 이런 관심으로부터 출발한 학문이 '인공지능'이다.

기계가 인간의 지적활동을 대신하도록 하는데 관심을 갖고 있는 인공지능은 1956 년 미국의 Dartmouth 대학에서 열 명의 과학자가 모여 인공지능에 대한 모임을 가진 것이 그 시초이다. 이 모임에서 인공지능(Artificial Intelligence)이라는 용어가 처음으로 사용되었다. 이러한 인공지능에 관한 연구는 1950 년대부터 미국의 대학들을 중심으로 시작되었는데, 초기에는 주로 게임이나 서로 다른 언어들 간의 번역, 수학적

정리의 증명 등의 작업을 기계를 통해 수행하는 실험적인 성격이 강했다.

1960 년에 민스키의 "퍼셉트론즈"가 발표된 이후로 인공지능과 라이벌 관계에 있던 신경망에 관한 연구는 급격하게 시들해졌고 다시 인공지능에 관한 연구가 활기를 띠게 되었다.

1.2 Traditional methods of gaming artificial intelligence

게임에서 적들의 움직임을 인공지능적으로 구현하기 위해 기존의 게임들에서 자주 사용되던 방법 중 하나가 바로 FSM 이다. 이 FSM은 오래전부터 다양한 방법으로 널리 사용되어 온 컴퓨터공학, 수학적 개념이며, 용어 뜻 그대로 유한한 개수의 상태들로 구성된 하나의 간단한 기계를 말한다. 여기서 하나의 상태(State)라는 것은 그냥 하나의 조건을 뜻을 나타내는 것이다.

FSM은 크게 1)현재 상태, 2)입력, 3)출력상태, 전이함수의 4가지 요소로 나눌 수 있다. 현재 상태는 말 그대로 현재 FSM의 상태를 말하는 것이고, 입력은 FSM에 들어온 입력정보, 출력상태는 다음 단계의 상태, 전이함수는 입력정보와 현재 상태를 기반으로 출력상태를 결정하는 함수를 뜻하는 것이다.

다시 말하자면, FSM이라는 것은 유한한 개수의 상태들을 가진 하나의 기계이고, 그 상태들 중 하나가 현재 상태인 것이다. FSM은 입력을 받고 어떠한 상태전이함수에 기반 해서 '현재 상태'로부터 '출력상태'로의 '상태전이'를 일으킨다.

그리고 출력상태는 새로운 현재 상태가 되는 것이다. 컴퓨터 역시 크게 보면 FSM의 한 예라 할 수 있다.

2. Applications of Artificial Intelligence and Computer Games

2.1 With the monster using the FSM Artificial artificial intelligence

이러한 FSM은 게임 내에서 인공지능을 구현하기 위해 다양하게 사용될 수 있는데, 그 중 대표적인 사용예가 적 몬스터의 행동을 결정하는 인공지능의 구현이라 할 수 있다. FSM의 각각의 상태를 몬스터의 감정이라서 하면, 이는 현재 상태와 FSM으로의 입력에 따라 변하며, 어떤 식으로 변할 것인지는 상태전이 규칙들에 의해 결정된다. 몬스터의 감정에 기반 해서 서로 다른 행동을 수행하는 코드를 만들면 몬스터가 마치 감정을 가진 실제 괴물처럼 움직이게 될 것이다.

복잡한 캐릭터라면 훨씬 더 많은 상태들과 입력들, 그리고 상태전이 규칙들을 가질 것이다. 어떤 상태들과 입력들을 구분할 것이며 그것들이 어떤 방식으로 전이될 것인지를 결정하는 것이 바로 FSM에 기반 한 인공지능을 만드는 기본적인 과정이다.

FSM(Finite State Machine)은 현재 가장 널리 사용되는 인공지능 처리 방식이다.

아래 그림1은 FSM의 기본 개념을 나타낸 것이며, 거의 아래 그림과 같은 원리로 처리되고 있다. 그림1에서 표시 한 바와 같이 여러 개의 상태로 나누어지면 캐릭터의 현재 상태(state)에 따라 외부에 대처하는 방식이 결정되는 것이다. 즉, 외부의 상황이 변화하게 되면 state도 변화되는 것이다.

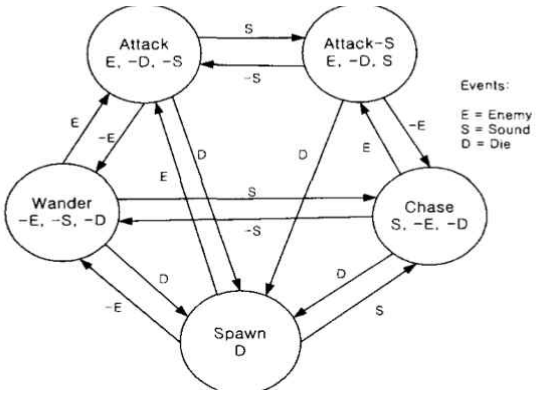

Fig. 1. Basic concept of FSM

위의 그림에서 보는 바와 같이 캐릭터는 특정한 상태에 있을 경우에는 항상 같은 방식으로 행동한다. 예를 들어 Wander state에서 적이 발견되는(E) 상황이 발생할 경우 Attact state로 이동하게 되며, 이 state에 있는 동안 계속 같은 행동을 한다.

만약 이 state에서 플레이어가 죽게(D)되면 Spawn state로 바꾸며 적이 보이지 않는(-F) 경우에는 다시 wander state 상태로 변하게 되는 것이다.

이러한 FSM 모델은 이해하기가 쉽고 프로그램으로 구현하기 쉬운 것이 장점이다.

FSM은 특별히 뛰어난 인공지능을 필요로 하지 않으며, 대부분의 게임에서 사용된다. 그러나 FSM의 단점은 state 수가

늘어나면 state diagram 을 정리하기가 어렵고 state 변화를 가능하게 하는 외부 센서 입력 루틴이 급속도로 복잡해지며, 이를 이용한 캐릭터의 행동을 예측하기 쉽다는 점이다. 이러한 문제를 부분적으로 해결하기 위해 하나의 state를 몇 개의 substate로 나누어 해결 할 수 있다.

2.2 GVGP(General Video Game Playing)

GVGP는 하나의 에이전트로 여러 가지의 게임을 플레이하는 것을 뜻한다. GVGP를 하기 위해서, 에이전트는 자신의 행동에 게임이 어떻게 반응할 것인지, 이로써 어떤 보상을 얻게 되는지, 결론적으로 어떻게 게임에서 이길 수 있는지를 찾아야만 하는데 목적이 있다.

또한, 에이전트가 하나의 게임에 대한 도메인 지식을 이용하여 만들어지게 되면, 그 에이전트는 해당 게임을 제외한 다른 게임에 대해서는 좋지 않은 성능을 발휘하게 되므로 전반적인 측면에서의 개발이 필요하다.

Tom Schaul은 GGP(General Game Playing)을 위한 환경을 제공하기 위해 VGDL(Video Game Description Language)을 제안하였. VGDL은 2차원 비디오 게임을 간단히 제작할 수 있는 개발언어로써, 이미지만 있다면 간단하게 여러 가지 컨셉의 게임을 제작할 수 있는 것이 장점이다. 실제로 GVG-AI 대회의 모든 게임은 VGDL을 이용하여 만들어진 것이다.

VGDL를 사용하면 게임의 간단한 승리조건(예를 들면, 단순 길 찾기 문제)부터 복잡한 로직이 요구되는 승리조건 (예를 들면, 일정개수의 다이아몬드를 획득한 후 출구로 나가야 하는 문제)까지 다양한 승리조건을 몇 줄의 언어로 정의가 가능하다.

Fig. 2. Example of GVG - AI games made with VGDL

VGDL로 만들어진 게임들은 각 게임마다 에이전트가 가능한 행동은 다르다. 좌우로만 움직일 수 있는 게임이 있는 반면, 상하좌우와 공격까지 가능한 게임도 존재한다.

본 논문에서는 에이전트가 공격이 가능할 시, 공격 쪽의 확률을 조금 더 주어 결과적으로 공격위주의 선택이 중심이 되게 가상하여 보았다. 왜냐하면, 보통의 경우 공격이 가능한 게임에서는 제한된 시간 안에 NonPlayer Character를 해치우거나 목표물을 없애야 하기 때문이다.

하지만 에이전트에게 '바로 앞에 적이 있을 때 공격을 하라'라고 가르치는 것은 GVGP 에서 벗어날 수 있다라는 것이다. 이러한 이유로 공격을 선택할 확률을 높여서 공격을 하면 좋은 성능을 보여준 것이 된다.

AI는 점점 더 중요한 비중을 차지하고 있다. 사실 초기 Game AI는 거의 전부가 hard coding(if문 등으로 처리)로 처리하는 것이 대부분 이었다. 하지만 이제는 좀 더 체계적이고 학술적인 측면으로 접근하고 있다. 이전까지 AI는 학술적인 연구분야의 전유물처럼 어렵게만 느껴졌지만 점차 우리 생활과 지금 얘기하려는 Game에서도 점점 AI요소가 적용되고 있다.

Game에서 AI가 들어가는 부분은 일단은 유저가 조작하지 않는 NPC(Non-Player Character)를 Control 하는 데에 기본적으로 적용되고 있다. 예전에 나왔던 간단한 2D Game에서는 단순한 랜덤 알고리즘도 많이 사용되었지만 지금의 Game에서는 다양한 장르를 소화할 수 있는 적절한 AI 기법들이 적용되고 있다.

해외 게임들을 보면 인공지능의 비중이 늘고 있다는 것을 확인할 수 있다. 하드웨어의 발전에 따른 것도 있을 것이고 게임에서의 재미요소로서 인공지능의 중요성이 높아지고 있는 것도 하나의 이유일 것이다.

그럼 최근 게임들을 통해서 게임에 사용된 AI 기법들을 소개하고 간단한 설명을 해 보도록 하겠다.

2.3 Black &White

피터 몰리뉴가 파퓰러스 등의 게임을 거쳐서 최근에 내놓은 화제작. God 게임의 창시자로서 이 게임에서는 특히 크리쳐의 인공지능이 돋보인다.

이 게임에서는 A-Life(Artificial Life 인공생명) 기술이 도입되었는데 크리쳐 자신의 욕구와 상태에 따라 유저의 행동에 대한 반응이 다이나믹하게 나타난다. 더 놀라운 것은 유

저가 크리쳐를 가르칠 수 있다는 점이다. 어떤 행동을 크리쳐에게 반복해서 보여 주었을 경우 그 행동을 똑같이 따라 하는 것과 칭찬과 벌을 병행함으로써 선과 악 양 방향으로 교육이 가능하다.

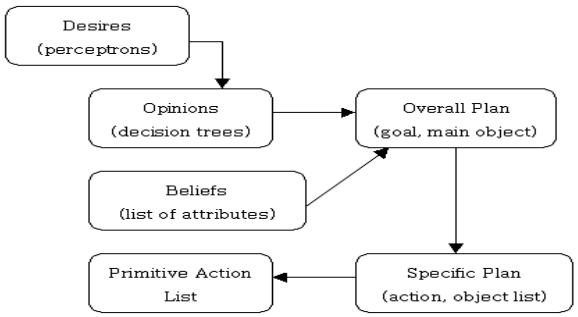

Fig. 3. Black & White

위의 아키텍처를 살펴보면 크리쳐의 욕망과 의견, 믿음 등이 행동을 결정하는데 영향을 준다. 특히 Opinions에 해당하는 Decision tree를 이용해서 교육의 효과를 구현했다. Black &White에서는 Dynamically Building Decision Trees (다이나믹하게 제작된 결정 트리)라는 방식을 이용했는데 일반적인 Decision Tree 방식을 응용해서 좀 더 다이나믹한 형태로 바꾸었다. 먼저 Decision Tree 방식을 설명하자면 게임에서 어떤 논리적인 계획을 좀 더 효율적으로 구현하기 위한 구조이다. Hard coding(if 문 등)으로 처리할 수도 있으나 좀 더 효율 적인 구조로 만들어서 코드를 직접 변경하지 않고도 외부에서 변경 가능한 구조로 만드는 것이다.

아래의 계획을 Decision Tree로 작성한다고 가정해보자.

\레이어가 가까이 있고 손상정도가 적으면 플레이어에 공격한다.
플레이어가 멀리 있고 연료가 많으면
플레이어가 있는지 수색한다.

변경 가능한 입력 요소
Distance : 플레이어와 NPC 사이의 거리
Fuel : 남은 연료
Damage : 현재 손상 정도

행동
Search : 플레이어 수색
Attack : 플레이어를 공격

노드 실행 시 사용할 수 있는 구조

Table 1. Decision Tree

```
typedef struct DECNODE_TYP
{
int operand1, operand2;    // the first   operands
                int   comp1;                              //  the
comparison operator

                 int    operator;                         //  the
conjuctive operator

              int operand3, operand4;    // the second pair of
operands
                 int   comp2;                              //  the
comparison to perform

ACTION *act_true;              // action lists for true and false

ACTION *act_false;

 DECNODE_PTR *dec_true;                // branches to take if
true or false

DECNODE_PTR *dec_false;
} DECNODE, *DECNODE_PTR;
```

이와 같은 Decision Tree는 단순한 방식이지만 응용하기에 따라서 편리하게 사용할 수 있을 것이다.

2.4 The Sims

EA에서 내놓은 시뮬레이션 장르의 게임이다. Sim City시리즈로 유명한 Will Write가 심혈을 기울여 만든 게임이다. 우리의 실생활을 게임에 재미있게 옮겨놓았는데 각 Character의 욕구가 있어서 그것들을 잘 충족시켜 주어야지 Game Play를 순조롭게 할 수 있다. Sims에서 특징적인 것은 캐릭터의 욕구에 기반한 AI기법이 사용된 것과 어떤 행동에 대한 정보들을 각각의 오브젝트가 가지고 있는 방식이 독특하다. 욕구 기반의 AI 기법은 Gamasutra에도 소개되었지만 내부상태에 대한 욕구한계점(threshold)을 조절해서 적절

한 효과를 얻을 수 있는 방식으로서 생물의 행동과 유사하도록 그 한계점들을 적절히 조절하면서 적용했다.

오브젝트가 이벤트를 발생시키는 방식을 Will Write는 “smart terrain”이란 말로 표현을 했는데 그 비유가 적절한 것 같다. 주체가 아니라 객체인 Object들이 각각 주체가 해야 할 행동들에 대한 이벤트 정보를 발생시킨다. 말하자면 배고픈 캐릭터가 지나가면 냉장고에서 “꺼내 먹어라”라는 명령을 내리는 것과 같은 방식이라고 말할 수 있다. 어떻게 보면 웃기지만 아주 적절하게 그 효과를 보고 있다고 말할 수 있다. 이 게임은 A-Life 와 Fuzzy-State Machine을 이용한 AI를 선보이고 있다. 특히 이 게임에서는 “smart terrain”이라는 Will Write의 말처럼 각 오브젝트들이 상태와 행동을 발생시키는 방식. 즉 object-oriented 방식으로 각 오브젝트가 캐릭터에게 여러 가지 행동에 대한 이벤트를 발생시키는 event-driven 방식이다.

Black&White나 The Sims에 A-Life라는 말이 언급되는데 조금 설명을 해보자면 지금까지의 AI는 위에서 아래로 그러니까 위에서 계획한 대로 아래에서 행해지는 계획되지 않은 행동은 일어날 수 없는 방식인데 A-Life 는 밑에서 위로라는 말로 표현되는데 각각의 세부적인 단순한 규칙들이 모여서 하나의 커다란 특성을 나타내는 것으로 정해주지 않은 행동이 창발적으로 발생하는 특징을 가지고 있다. 좀 더 자연의 생물을 흉내낸 방식이라고 할 수 있다.

2.5 Half Life

Half Life는 게임 중에서도 명작으로 꼽히는 게임이다. 이 게임은 무엇보다 스토리와 연출로 플레이어들을 사로잡았다. 이 게임 또한 AI가 Game에 재미를 더해 주고 있는데 몬스터나 특수부대원들이 나올 때면 정말 만만한 상대들이 아니다라는 느낌을 받을 수 있는데 그 들의 움직임 또한 모두 AI로 처리되었다.

Half Life에서 적용한 방식은 일련의 행동들을 Schedule로 묶어서 처리한 방식인데, 이것은HFSM(Hierarchy Finite State Machine)이란 방법으로 볼 수 있는데 FSM을 개념에 따라 계층을 나누어 놓았다고 볼 수 있다. 하나하나의 구체적인 행동들을 Task State라고 하면 그 task들이 여러 개 모여 있는 State가 Schedule State인 것이다.

여기서 간단하게 State Machine이란 개념을 소개하자면 State들을 어떤 조건에 따라 연결해 놓은 것이다. 각각의 State에는 여러 가지가 들어갈 수 있다. 캐릭터의 내부상태가 될 수도 있고 행동 패턴이 될 수도 있고 심리 상태가 될 수도 있다. 이런 각각의 State들이 어떤 조건에 따라 연결되어 있는 것이 State Machine 이다.

State Machine 중에서도 유한한 개수의 상태를 가진 것을 FSM(Finite State Machine)이라고 하는데, 이것의 한 예를 들어 보겠다.

Table 2. Monsters State Transition Rules

	현재 상태	입력	출력 상태
①	보통	플레이어 등장	불쾌
②	보통	플레이어의 공격	흥분
③	흥분	몬스터 다침	분노
④	흥분	몬스터 치료됨	보통
⑤	분노	몬스터 다침	광분
⑥	분노	몬스터 치료됨	불쾌
⑦	광분	몬스터 다침	광분
⑧	광분	몬스터 치료됨	분노
⑨	불쾌	플레이어 떠남	보통
⑩	불쾌	플레이어의 공격	분노
⑪	불쾌	몬스터 치료됨	보통

위의 전이 규칙을 FSM으로 표현하면 다음과 같다.

위의 예에서 보듯이 FSM은 현재의 State에서 입력값에 따라 다음의 State가 결정되는 형태로 동작한다.

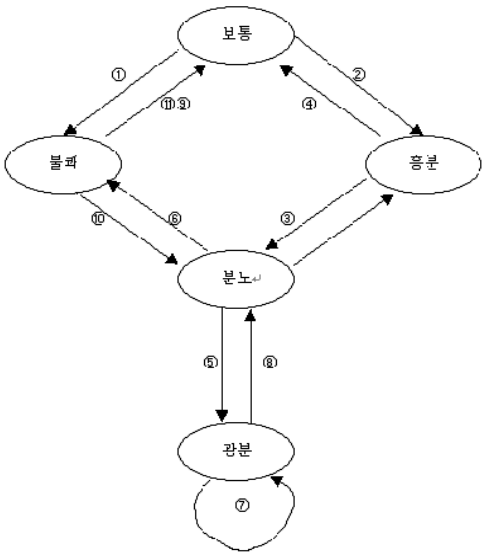

Fig. 4. Action of FSM

다음은 FSM의 하나의 상태를 표현하는 데 쓰이는 클래스이다.

Table 3. Status class of FSM

```
class FSMstate
{unsigned   m_usNumberOfTransistions;
  // 이 상태가 담을 수있는 상태전이// 들의 개수
                                                    int
*m_piInputs;

// 상태전이를 위한 입력 배열
                                                    int
*m_piOutputState;
// 출력 상태 배열
                                                    int
m_iStateID;
        // 이 상태의 고유 ID

public:
    // 생성자는 이 상태에 대한 ID와 이 상태가 지원하는 상태 전
이들의 개수를 받는다.
    FSMstate (int   iStateID,

unsigned   usTransitions);
    // 소멸자는 할당된 모든 배열들을 해제한다.
    ~FSMstate ();

    // 상태 ID를 돌려준다.
    int GetID () { return   m_iStateID; }
    // 배열에 하나의 상태 전이를 추가한다.
    void AddTransition (int iInput, int iOutputID);
    // 배열로부터 상태 전이를 제거한다.
    void DeleteTransition (int iOutputID);
    // 상태 전이를 일으키고 출력 상태를 돌려준다.
    int GetOutput (int iInput);}
```

Ⅳ. Conclusions

게임에서의 인공지능을 구현하기 위해서는 한두 가지 방법만을 사용하기 보다는 게임의 수준이나 장르에 따라 몇 가지 방법을 조합하여 구현하는 것이 일반적이다. (black&white)에서는 인공생명이라는 기술이 적용되었는데, 캐릭터 자신의 욕구와 상태에 따라 사용자의 행동에 대한 반응이 다이나믹하게 나타날 수 있도록 하였다. 또한 사용자가 캐릭터에게 어떤 행동을 반복해서 보여주게 되면 그 행동을 똑같이 따라하고 상이나 벌을 제시함으로써 선과 악을 구분하고, 교육이 가능하도록 하였다.

EA사의 심시티 시리즈로 유명한 심즈(The Sims)는 윌라이트(Will Write)가 개발한 것으로 일상적인 실생활을 게임으로 옮겨놓은 것으로, 각 캐릭터의 욕구가 있어서 그것들을 잘 충족시켜줘야 순조롭게 플레이를 할 수 있다.

또한, 사용자가 캐릭터에게 어떤 행동을 반복해서 보여주게 되면, 그 행동을 똑같이 따라 하고 상이나 벌을 제시함으로써 선과 악을 구분하고, 교육이 가능하도록 하였다.

EA사의 심시티 시리즈로 유명한 심즈(The Sims)는 윌라이트(Will Write)가 개발한 것으로 일상적인 실생활을 게임으로 옮겨놓은 것으로, 각 캐릭터의 욕구가 있어서 그것들을 잘 충족시켜줘야 순조롭게 플레이를 할 수 있다.

컴퓨터를 이용한 게임의 역사는 그리 길지 않다. 하지만 그 짧은 역사에도 불구하고 수많은 사람들의 기대 속에 1인칭 슈팅게임부터 스포츠, 각종 시뮬레이션 게임 등 다양한 장르의 게임들로 발전하고 있다.

컴퓨터 게임의 매력은 혼자서도 컴퓨터가 만들어내는 가상의 상대와 함께 게임을 즐길 수 있다는 점인데 이는 바로 컴퓨터에서 구현되는 인공지능의 결과물인 것이다. 실제로 어떻게 하면 게이머의 우주선이 발사하는 미사일을 좀 더 잘 피해내는 적 비행기를 구현할 것인가, 어떻게 하면 더욱 다양한 전술을 구사하는 군대를 만들어 낼 것인가 하는 과제가 인공지능 분야를 발전시키는데 큰 공헌을 하였다.

앞으로는 더욱 다양한 여러 가지 인공지능 이론들의 구현한 게임이 등장할 것으로 보인다.

REFERENCES

[1] Jang Young Beom. (http://www.techleader.net)
[2] Lee Hyung-jun, 2003, in the history of computer game development Consideration - From a Consumption Perspective Perspective -.
[3] Kookmin University Graduate School of Techno Design, A master's degree p.10.
[4] Choi Yoo Chan, Understanding the computer games, cultural scientists, 2002. p.66.
[5] Moon Jae Young 2003, The effect of Mole and Addiction on Online Gaming and Immobility of

User Satisfaction, Economics and Economics in Economics, Sungkyunkwan University.
[7] Lee Man Jae, Artificial Intelligence at the Games, Professor of Information and Communication at the University of Korea.
[8] J. Levine, C. B. Congdon, M. Ebner, G. Kendall, S. M. Lucas, R. Miikkulainen, T.
[9] Schaul, and T. Thompson, “General Video Game Playing,” Dagstuhl Follow- Ups, vol. 6, pp. 77-83, 2013.
[10] Kim Hyun-Tae, Game analysis and using MCTS Video Games General Artificial Intelligence, 2014.
[11] Andre La Mothe, “tricks of the WINDOWS Game Programmin Gurus”.
[12] belief/desire/intention architecture.

저 자 소 개

Song Gyu Jin developed programs for network and communication control while working for Arche in 2015.

Song Gyu Jin develops and operates game applications together with four other partners at Sloth.

He is interested in game programming, IoT development and applications.