에이전트 기반 지능형 게임 캐릭터 구현에 관한 연구[1)]

이재호, 박인준[2)]

On the Development of Agent-Based Online Game Characters

Jaeho Lee, In Jun Park

요약

개발 적인 측면에서 온라인 게임 환경에서의 NPC(Non Playable Character)들은 환경인식능력, 이동능력, 특수 능력 및 아이템의 소유 배분 등을 원활히 하기 위한 능력들을 소유해야 하며, 게임 환경을 인식, 저장하기 위한 데이터구조와 자신만의 독특한 임무(mission)를 달성하기 위한 계획을 갖고 행위를 해야 한다. 이런 의미에서 NPC는 자신만의 고유한 규칙과 행동 패턴, 그리고 목표(Goal)와 이를 실행하기 위한 계획(Plan)을 소유하는 에이전트로 인식되어야 할 것이다. 그러나, 기존 게임의 NPC 제어 구조나 구현 방법은 이러한 요구조건에 부합되지 못한 부분이 많았다. C/C++ 같은 컴퓨터 언어들을 이용한 구현은 NPC의 유연성이나, 행위에 많은 문제점이 있었다. 이들 언어의 switch 문법은 NPC의 몇몇 특정 상태를 묘사하고, 그에 대한 행위를 지정하는 방법으로 사용되었으나, 게임 환경이 복잡해지면서, 더욱더 방대한 코드를 만들어야 했고, 해석하는데 많은 어려움을 주었으며, 동일한 NPC에 다른 행동패턴을 적용시키기도 어려웠다. 또한, 대부분의 제어권을 게임 서버 측에서 도맡아 함으로써, 서버측에 많은 과부하 요인이 되기도 하였다. 이러한 어려움을 제거하기 위해서 게임 스크립트를 사용하기도 하였지만, 그 또한 단순 반복적인 패턴에 사용되거나, 캐릭터의 속성적인 측면만을 기술 할 수 있을 뿐이었다. 이러한 어려움을 해소하기 위해서는 NPC들의 작업에 필요한 지식의 계층적 분화를 해야 하고, 현재 상황과 목표 변화에 적합한 반응을 표현할 수 있는 스크립트의 개발이 필수 적이라 할 수 있다. 또한 스크립트의 실행도 게임 서버 측이 아닌 클라이언트 측에서 수행됨으로써, 서버에 걸리는 많은 부하를 줄일 수 있어야 할 것이다. 본 논문에서는, 대표적인 반응형 에이전트 시스템인 UMPRS/JAM을 이용하여, 에이전트 기반의 게임 캐릭터 구현 방법론에 대해 알아본다.

Key words : NPC, 에이전트, 반응형 에이전트

1. NPC의 의미와 개념

NPC는 게이머가 직접 컨트롤(제어)하지 않는 모든 게임 캐릭터를 의미한다. 단순한 적군을 의미하던 기존 피시(PC)게임과 달리, 온라인 게임에서는 파트너, 유닛, 명령자 및 해설자 등의 다양한 역할을 수행한다. 인공지능 엔진의 경우 NPC를 개별적인 행동패턴과 사고를 지닌 에이전트로 볼 수 있다. 다음은 역할별로 구분한 NPC의 종류를 열거한 것이다.

1) 본 연구는 첨단정보기술 연구센터를 통하여 과학재단의 지원을 받았음.

2) 서울시립대학교 전자전기컴퓨터공학부 및 첨단정보기술연구센터(AITrc)

1.1 NPC의 종류

1.1.1 적군(Tactical Enemies)

이전 게임에서의 적군의 의미는 컴퓨터에 의해 제어되는 캐릭터에 불과했다. 이들은 그다지 지능적이지 못했으며, 사용자의 능력과 관계없이 일관적인 능력을 소유했다. 또한 게임의 레벨이 증가하더라도, 캐릭터의 지능적인 측면은 변화하지 않고, 무기나, 특수 능력과 같은 단순한 기능의 추가만을 함으로써, 게임을 쉽게 식상하게 만들었다. 하지만, 최근의 인공지능을 응용한 온라인 게임에서의 적군은 게이머의 능력치에 맞게 지능적 변화를 유도하며, 기존의 지식을 자신의 전술에 적용하는 등 보다 인간적인 캐릭터로 변모하였다.

1.1.2 파트너(partner)

게이머의 미션이나, 최종 목적을 달성하는데 도움을 주는 캐릭터. 슈팅게임의 경우엔, 게이머가 위급한 경우에, 특수능력을 이용하여 이를 해결해 주는 역할을 하기도 하며, RPG 게임의 경우, 게이머의 문제를 해결하기 위한 실마리를 제공해 주는 역할을 하기도 한다. 적군역할을 하는 캐릭터가 컴퓨터에 의해 제어되는 만큼 자율성(autonomy)을 중요시하는 반면, 파트너 캐릭터는 게이머와의 상호작용(cooperation)을 중요시한다. 이 캐릭터를 위해서, 키보드의 단축키나 메뉴를 이용하던 기존의 방식에 반해, 최근 온라인 게임에서는 캐릭터가 게이머의 상태 및 능력, 레벨, 팀 플레이를 고려해, 자연스럽고, 인간적인 접근을 유도한다.

1.1.3 유닛(Units)

게이머와 컴퓨터 모두에 의해 제어(컨트롤)되는 캐릭터. 주로 게이머에 의해 제어되며, 자신만의 미션을 소유하고 있는 캐릭터이다. 이 캐릭터는 반자율적(semi-autonomy)인 속성을 지니고 있다. 즉, 대부분 자신의 미션을 자율적으로 수행하다가, 게이머의 명령을 받는 순간, 명령을 수행하는 방식으로 움직이는 캐릭터이다. RPG나 전략 시뮬레이션 게임의 경우에 이 캐릭터는 수백여개나 등장하므로, 메모리나 CPU에 과부하에 주의하여야 하며, 특히, 다른 유닛들과의 상호 작용이 중요한 이슈이다.

1.1.4 해설자(Commentator)

게이머가 게임을 유지해 나가는데 필요한 정보를 제공해 주는 캐릭터. 대부분의 경우, 게이머의 상태를 계속적으로 체크하면서, 게이머가 원하는 상태로 이동하는데 필요한 정보를 텍스트 문자열로 생성하는 역할을 수행한다. 이러한 행위는 게이머의 행동과 행동사이에 순간적인 정보로 제공 될 수 도 있으며, 때로는 고난이도의 이해력을 요구하는 정보로 제공되기도 한다.

위에서 보듯, 온라인 게임의 캐릭터들은, 게이머와 게임환경의 상태를 파악하고, 그에 합당한 진화를 거듭해야 하며, 자신들의 목표(Goal) 도달하기 위한 고유 행동양식을 소유해야 한다. 즉, 캐릭터 자신의 행동 목표와 그에 합당한 계획(Plan)을 소유하고 있는 에이전트(agent)라 할 수 있다.

다음은 게임 캐릭터로서의 에이전트에게 요구되는 기능 및 속성을 나열한 것이다.

1.2 NPC에 요구되는 속성

1.2.1 자율성(Autonomy)

주위의 환경을 인식하여 어떤 행동을 취하기 위해 능동적인 판단을 하는 능력. 에이전트 이전의 게임 캐릭터들은 고정적인 상태(state)들을 단순하게 이동하는 수준이었다. 이 경우에 게이머나 게임환경에 대한 고려가 없이, 고정적인 패턴을 유지함으로써, 게이머 들이 NPC의 다음 행위를 쉽게 예상하거나, 무력화(cheat라고 표현됨) 시키는데 노출되었으며, 게임을 식상하게 만드는 주 요인이기도 했다. 하지만 에이전트 기반의 캐릭터는, 게이머와 게임환경의 상태를 인식하여, 자신의 행동양식에 적용되는 행위를 결정하므로써, 이런 문제에 비교적 잘 적응 할 수 있다.

1.2.2 지능성(Intelligence)

이전 상태 및, 현재 상태를 저장하여, 다음 상태를 도출해 내는 능력 및 게임 환경의 상태 인식과 사용자의 의도를 파악하여 필요한 행위를 유도해 낼 수 있는 학습 능력을 말한다. 에이전트의 이러한 속성은 게임 환경의 현실성(리얼리즘)과 직결되며,

---

3) 본 연구는 첨단정보기술 연구센터를 통하여 과학재단의 지원을 받았음.

4) 서울시립대학교 전자전기컴퓨터공학부 및 첨단정보기술연구센터(AITrc)

보다 능동적 캐릭터를 만드는데 필수 적인 요소이다. 특히, 자신이 취해 왔던 행위(action)에 대한 기록을 기반으로 새로운 지식(knowledge)을 만들어 내는 것은, 다양한 장르의 게임 캐릭터에 공통적으로 요구되는 속성이라고 할 수 있다.

1.2.3 상호협력(Cooperative Coordination)

게임 에이전트는 인간(게이머)이나 다른 NPC와의 원활한 상호 작용을 통하여, 정보를 습득하거나, 자신의 목표(Goal)를 수행 할 수 있어야 한다. 대개의 경우, Unit NPC들에 요구되는 속성이며, 게임환경을 cheat하는 방식으로 설계되었던 기존 NPC와는 달리, 에이전트기반 NPC들은 사용자와 기타 다른 에이전트NPC의 상태 및 능력에 맞는 진화를 거듭하는 속성을 소유해야 한다.

게임에서의 NPC는 지식 표현 및 추론, 자율적인 행동패턴을 지닌 에이전트이다. 이러한 게임 에이전트를 설계하기 위해서는 그에 합당한 설계 및 제어구조가 필요하며, 게임환경에 필요한 인공지능 엔진이 필수 적이다

2. 기존 NPC 제어 방법

2.1 C/C++를 이용한 제어

C/C++을 이용한 NPC제어는 switch문이나 if-else 등의 구문을 이용하여 NPC의 상태를 나열 한 후, 조건에 맞는 구문을 선택, 수행하는 작업을 기본으로 한다. NPC가 처할 수 있는 상황이 비교적 단순한 게임에서는 쉽게 구현이 될 수 있으나, 온라인 게임과 같이 NPC의 수가 많고, 다양한 상황이 제공되는 프로그램의 경우 제어루틴이 거대한 조건-판단문에 집중되어야 하므로, 좋은 효과를 기대할 수 없고, NPC 상태 파악에 많은 어려움이 있다.

2.2 게임 스크립트를 이용한 제어

NPC를 제어하기 위한 스크립트는 대부분 규칙기반 시스템(Rule Based System)으로 구현이 가능하다. 규칙기반시스템은 많은 규칙을 나열하고 현 상황에 해당하는 규칙을 골라내어 결과를 얻는 시스템을 말한다. 그러나 기존의 온라인 게임에서의 NPC 제어 엔진으로 사용되던 스크립트 언어의 경우, 규칙기반시스템으로의 역할을 충분히 수행하지 못하였다. 규칙기반 시스템은 에이전트의 속성과, 규칙, 행위를 구현 할 수 있어야 하지만, 기존 게임의 스크립트는 NPC의 이름이나, ID등과 같은 속성(attribute)을 묘사하는데 많은 비중을 두었으며, 규칙이나, 행위 묘사에는 효과적이지 못했다.

3. 에이전트기반 NPC 제어 방법론

3.1 반응형 계획 에이전트

기존 NPC제어의 구조적, 구현적 문제점을 줄이기 위해서는, NPC를 에이전트로 표현하기 위한 방법론과, 에이전트들의 목표를 달성하기 위한 계획의 명시, 그리고 NPC의 현재 상태 및 게임환경을 저장할 수 있는 스크립트의 개발이 필수 적이다. 이를 위하여 반응형 계획 에이전트 시스템을 이용 할 수 있다. 일반적으로 계획기를 내부 핵심 요소로 채용하고 있는 에이전트 구조를 계획 에이전트 구조라 부른다. 계획기 및 계획 시스템은 계획생성(plan generation)과 계획실행(plan execution)이 완전히 별도의 독립적 프로세스로 분리되어 있다고 가정한다. 즉 계획 생성 단계에서는 계획기가 달성하고자 하는 목표(goal)와 현재상태(current state), 가능한 동작(action)들을 기호논리로 표현한 뒤, 이들을 바탕으로 목표를 달성할 수 있는 동작의 순서를 결정하고, 계획실행 단계에서는 이렇게 생성된 계획을 별도의 실행기에 의해 실행하는 순차적 제어구조로 되어 있다. 계획기들을 일반적으로 반응형(reactive planner)이라고 부른다. 이러한 반응형 계획기로는 PRS(Procedural Reasoning System)이 있다. 본 장에서는 에이전트 기반 NPC제어구조를 구현하는데 사용될 수 있는 반응형 에이전트 시스템인 JAM에 대하여 서술한다.

3.2 JAM 에이전트

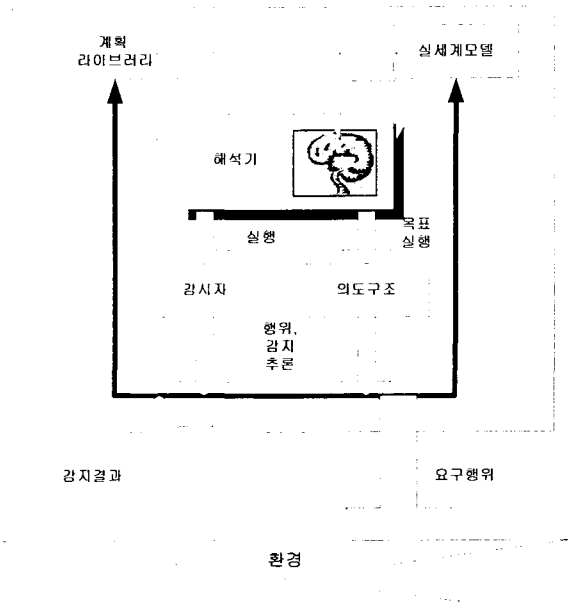
JAM은 PRS(Procedural Reasoning System) 계획 에이전트 구조에 몇 가지 기능을 확장하고, JAVA 언어로 구현하였다. JAM에는 5가지의 중요한 요소가 있으며, 이 요소들의 상호 작용에 의해 외부 환경으로부터의 변화를 내부적으로 수행하며, 실 세계(환경)와 반응한다.

3.2.1 실세계모델(World Model)

현재 에이전트가 가지고 있는 실세계에 대한 정보

를 저장하는 데이터 베이스 이다. 저장 정보는 에이전트의 상태 및 센서 정보 등이 포함되며, 이것들은 에이전트의 믿음(Belief)으로 활용된다.

그림 1 Jam agent architecture



3.2.2 계획라이브러리(Plan Library)

에이전트가 목표(goal)를 성취하는데 필요한 계획들로서, 설계자에 의해 사전에 주어지는 추상 계획(abstract plans)들의 집합이다.

3.2.3 인터프리터(Interpreter)

에이전트의 뇌에 해당하는 컴포넌트이다. 에이전트 설계자에 의해 초기 실세계 모델과 최종 목표, 그리고 이용 가능한 추상 계획들의 집합이 선언적으로 주어지면, 이들을 바탕으로 부속 목표들과 세부 계획들을 생성하고 이들을 실행하는 역할을 한다.

3.2.4 의도구조(Intention Structure)

최종 목표 달성을 위해, 에이전트가 현재 추구하는 부속 목표와 현재 실행중인 세부 계획 상태를 나타낸다.

3.2.5 감지기(sensor)

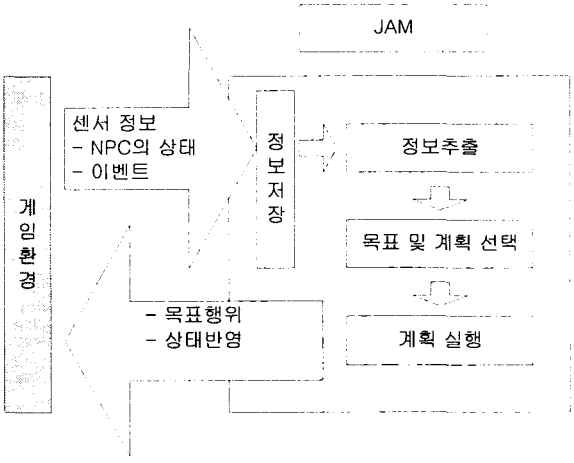
인터프리터의 추론 주기 사이에 외부세계의 변화 및 에이전트의 상태 변화를 감지하는 기능을 수행한다.

4. JAM을 이용한 에이전트 기반 NPC 제어 방법론

4.1 개념

게임 환경으로부터 NPC의 상태정보를 받은 후, 데이터 변환 과정을 거쳐서, JAM내부에 소유하고 있는 저장소에 저장한다. 저장된 환경정보는 JAM 내부에 존재하는 감시기를 통하여 인식되고, 감시기는 환경정보를 기반으로, 적절한 목표를 선택하고, 실세계 모델에 목표 등록과정을 수행한다. 그렇게 되면 해당 목표의 계획기가 활성화되고 목표를 달성하기 위한, 액션들이 수행된다. 이러한 액션들은 게임 환경을 위한 인터페이스(Interface)를 통하여 게임 환경에 반영된다. NPC들은 이러한, 상태정보 습득, 정보 저장, 목표 설정, 계획실행, 행위 및 상태 반영을 반복하면서, 게임 환경에 적응해 나가게 된다.

그림 2 에이전트기반 NPC 제어구조

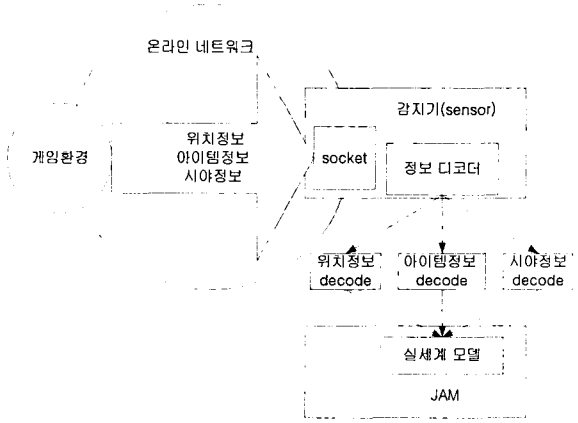


4.2 감지기 (Observer) 의 구현

감지기는 게임 환경의 정보를 받아서 Jam 에이전트에 전송하는 역할을 한다. 감지기 구현에 필요한 컴포넌트는 소켓(socket), 정보디코더(Information decoder)와 함께, JAM 실세계모델에 접근하기 위

한 인터페이스(interface)이다. 감지기는, 실제 온라인 네트워크와 연결되어 있으므로, JAM에서 고유액션(primitive action)을 통해 접근하는 방법보다는, 자바(JAVA)언어를 이용하여 클래스(class)로 구현한다. 또한 감지기(senor)가 네트워크와의 통신을 위해 사용되는 메소드(method)는 JAM에서도 직접 access할 수 있도록 고유액션(primitive action)으로도 구현되어야 한다.

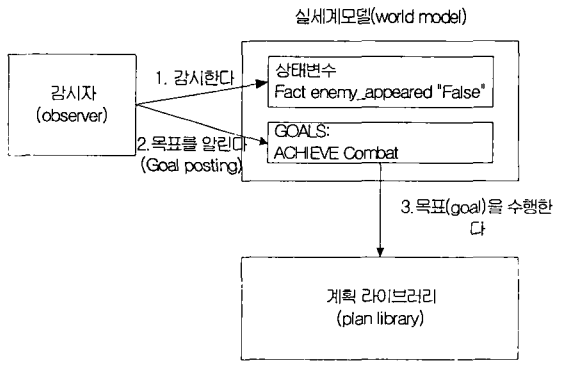
그림 3 감지기의 구조



4.3 감시자(observer)의 구성

감시자(observer)는 실세계 모델의 상태 변수를 감시하는 역할을 수행한다. 게임 환경의 NPC에 대한 정보는 실세계 모델에 상태 변수로 저장되며, 감시자는 이를 감시하는 프로시져(procedure) 역할을 수행하고, 상태(state)에 맞는 목표(goal)를 실세계 모델에 알린다.

그림 4 감시자의 구성



4.4 계획라이브러리의 구성

NPC의 지능성 부여에 가장 중요한 부분이다. NPC가 자신의 상태와 무관하게, 미리 정해진 목표(Goal)행위를 지속하는 것은 사용자에게 cheat행위를 당하거나, 게임을 식상하게 만드는 원인이 된다. 따라서, NPC제어 에이전트는 현재 수행중인 목표(Goal)를 지속할지, 포기할지의 결정을 내려야 하며, 자신의 현재 상태와 실행조건을 비교해 보아야 한다. JAM에이전트의 계획기는 이러한 요구조건을 만족시키기 위한 실행조건(context)을 소유하고 있다. JAM에이전트는 계획기가 액션(action)을 실행하며 작업하는 동안, 계속해서 실행조건과 현재 상태(state)를 비교함으로써, 목표수행의 수행 여부를 결정하는 구조로 되어 있다. 따라서 NPC 제어를 위한 계획기는 NPC의 종류와 행위 목표 그리고 이들의 현재 상태를 실행조건(context)으로 사용하여 구성한다.

4.5 실세계모델의 구성

실세계 모델은 JAM 에이전트가 환경을 인식하기 위한 지식정보가 저장되는 데이터베이스(database)이다. 환경으로부터 입력된 정보들은 predicate와 그것의 속성 값 형태로 저장되며, 에이전트의 행동패턴을 결정짓는 중요자료들 이다. NPC 제어를 위한 기본적인 환경 정보로는 위치정보, 시야정보, 아이템 정보 등이 있으며, 감지기와 JAM action interface가 데이터를 직접 입력하는 구조로 되어 있다. 실세계 모델은, 환경 정보 이외에도, 에이전트가 수행해야 할 목표(goal)들을 명시하고, 에이전트의 상태변수(state variable)들이 초기화되어 있다.

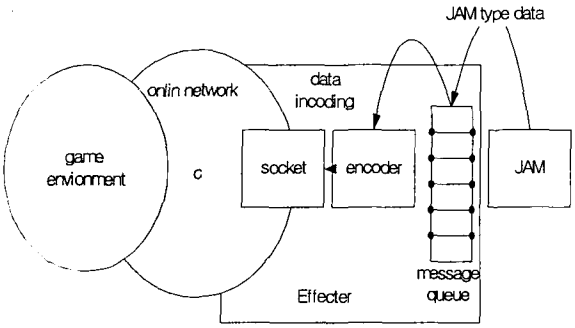
4.6 효과기(Effector)의 구현

효과기는 JAM에서 수행된 행위의 결과를 게임환경에 반영하는 컴포넌트(component)이며, 감지기의 상대적인 역할을 수행한다. 효과기 역시 네트워크와 연결되어 있으므로, JAM에서 직접 연결하는 것보다는 자바클래스(Java class)로 구현하는 것이 바람직하다. 효과기 구현에 필요한 컴포넌트는 네트워크 환경으로 데이터를 송신하기 위한 소켓과 JAM에 의해 구조화된 데이터를 게임환경에 맞는 인코딩(encoding)작업을 수행하기 위한 인코더(encoder) 및 JAM과의 인터페이스(interface)를 위한 JAM action interface가 필요하다.

JAM으로부터 전달된 메시지는 효과기 내부에 있는 메시지 큐(Message Queue)로 전달되어 저장 된

다. 효과기는 스레드(thread)를 감시하고, 데이터를 추출 한 후, 인코더로 전달한다. 인코더는 메시지를 해석하여, 게임 환경에 정의된 데이터 타입으로 전환하고, 소켓을 통하여 네트워크로 송신한다.

그림 5 효과기의 구조



5. 결론 및 향후 과제

본 논문에서는 반응형 에이전트 시스템인 JAM을 이용하여, 에이전트 입장에서 NPC를 제어하는 방법론에 대하여 기술하였으며, 이를 기반으로, 실제 게임상에서, NPC가 게임 환경의 2차원 공간을 이동 및 탐색하는 간단한 NPC 에이전트를 구성하였다. 그 결과, 기존의 저수준 프로그래밍 언어를 사용하여 NPC를 제어하는 방법에 비해, 에이전트의 전략수립이나, 행동패턴의 교체면 에서 보다 수월한 접근을 할 수 있었다. 그러나, JAM 에이전트가 게임을 위한 특별한 라이브러리를 제공하지 못하였으므로, 많은 사용자 정의 라이브러리를 필요로 하게 된 점은 간과 할 수 없는 문제점이기도 했다. 따라서, 본 논문에서 제안한 방법론을 효과적으로 적용시키기 위해서는, 게임 환경을 위한 라이브러리 및 NPC API와 함께, JAM 에이전트와 게임 환경을 연결시키기 위한 플래폼의 구현이 선행되어야 할 것이다.

참고문헌

[1] 고광현, 한국 정보과학회 정보과학지, 2002 : 온라인 게임 개발 사례

[2] 이만재, 한국 정보과학회 정보과학지, 2002 : 온라인 게임 엔진 기술동향

[3] 조성삼, 정문경, 정보처리학회지 2002: 온라인 게임 개발 현황

[4] 이헌주, 김준애, 임충규, 김현빈. 정보처리학회지 2002: 온라인 3D 게임엔진 표준화

[5] 박철제, 최 성. 정보처리학회지 2002: 네트워크 기반 게임의 다중시스템 운영 기술

[6] 이만재. 정보처리학회지 2002: 게임에서의 인공지능 기술

[7] Stuart Russell and Peter Norvig. Artificial Intelligence : A Modern Approach

[8] R. Adobbati, Andrew N.Marshall, Gal Kaminka, Steven Schaffer, Chris Sollitto. Gamebots : A 3D Virtual World Test-Bed For Multi-Agent Research

[9] Michael van Lent, John Laird, Josh Buchman, Joe Hartford, Steve Houchard, Kurt Steinkraus, Russ Tedrake. Intellignet Agents in Computer Games

[10] John E. Laird and Michael van Lent. Human-level AI's Killer Application : Interactive Computer Games

[11] Michael van Lent, John Laird. Developing an Artificial Intelligentce Engine