게임 NPC 지능 개발 플랫폼 구조 비교 및 분석

임차섭[0] 엄상원 김태용
중앙대학교 첨단영상대학원
imcs@gametech.cau.ac.kr sangwon@imagelab.cau.ac.kr kimty@cau.ac.kr

Comparison and Analysis of Platform Architecture for Game NPC Intelligence Development

Cha-Seop Im[0] Sang-Won Um, TaeYoung Kim
Dept. of Image Engineering, Graduate School of Advanced Imaging Science, Multimedia & Film

요 약

최근 컴퓨터 게임에 있어 인공지능에 관한 관심이 높아지고 있다. 이에 따라 게임에서 인공지능 구현을 지원하는 게임 NPC 지능 개발 플랫폼에 관한 연구도 많이 진행되고 있다. 본 논문에서는 게임에서 NPC의 지능을 개발하는 측면에서 게임 NPC 지능 개발 플랫폼 설계시에 요구되는 사항들을 알아보고, 실제 게임 NPC 지능 개발 플랫폼들-FEAR, SOAR, Gamebots-의 구조를 비교, 분석한다. 또한 기존 게임 NPC 지능 개발 플랫폼에서 발생하는 문제점을 알아보고, 이를 해결하기 위한 방법을 제안한다.

1. 서 론

컴퓨터 사용자들은 자신들의 컴퓨터를 통해 자신만의 세계를 구축하고 지배하기를 원한다[1]. 이것은 컴퓨터 게임에서도 예외는 아니다. 게임 속에서 플레이어는 실제 세계에서는 가질 수 없는 능력을 가지게 되고, 자신만의 세계를 구축하고, 그 세계의 지배자가 되길 원한다. 그리고, 멀티 플레이어 게임에서 플레이어는 상대방과의 경쟁에서 이기면서 즐거움을 느끼게 된다[2]. 또한, 이런 즐거움은 상대방이 컴퓨터에 움직이는 캐릭터가 아닌 자신과 같은 인간 플레이어일 경우 더욱 큰 즐거움을 느끼게 된다[3]. 하지만, 게임에서 NPC(Non-Player Character)를 완전히 배제할 수는 없다[4]. 이에 따라 게임에서 NPC들은 더욱 세련되고 사실적인 인공지능이 필요하게 되었다. 특히, Quake II 와 같은 일인칭 슈팅(FPS) 게임에 있어 NPC들의 행동은 게임의 성공과 직결된다. 만약, 게임 속에 등장하는 NPC들이 단순한 패턴으로 행동한다면 사용자는 NPC들의 행동을 쉽게 알아채고, 흥미를 잃게 된다. 더불어, NPC들이 게임에서 사용되는 모든 데이터들의 정보를 미리 알고, 사용자보다 훨씬 더 우월한 행동을 보인다면 사용자들은 게임을 어렵게 느끼고, 기피하게 된다. 사용자들은 NPC들도 주어진 게임 환경 내에서 사용자와 똑같은 수준의 정보를 가지며 사람의 행동과 유사한 지능과 행동을 요구한다.

이러한 상황에서 게임내의 캐릭터들의 인공지능을 개발하고 테스트할 수 있는 환경에 대한 연구가 있어왔으며 Gamebots, SOAR, FEAR등의 플랫폼들이 있다.

본 논문의 2절에서는 게임 환경에 나타나는 NPC들의 인공지능을 표현할 수 있는 플랫폼 설계시 고려되어야 하는 사항은 어떤 것들이 있는지 알아본다. 3절에서는 기존 게임 NPC 지능 플랫폼이 어떻게 설계되었는지 알아본다. 그리고, 4절에서는 기존 게임 NPC 플랫폼들의 장단점과 문제점을 알아보고, 이를 해결하기 위한 방법을 제안한다.

2. 게임 NPC 지능 플랫폼 요구사항

게임에서 NPC의 지능을 높이기 위해 일반적으로 사용하는 인공지능 기법은 다음과 같다[5].

●**규칙 기반 시스템(Rule-Based Systems)** : 인공지능을 표현하는 가장 간단한 형태 중 하나이며, 일반적으로 IF-THEN 문으로 이루어진다.

●**퍼지 논리(Fuzzy Logic)**: 퍼지 로직은 애매한 상황에서 사람들이 결정하는 방법을 모델링하였다.

●**유한 상태기(Finite State Machines)**: 유한 상태기는 많은 게임에서 사용되고 있으며, 각 NPC가 바뀔 수 있는 상태들을 이산적인 그래프형태로 표현한다.

●**결정트리(Decision Tree)**: 결정 트리는 복잡한 IF-THEN 규칙을 트리로 표현한 것이다.

●**신경망(Neural Networks)**: 인간의 뇌의 구조를 기초로한 학습 시스템이다. 가중치를 통해 학습을 할 수 있다.

위와 같은 인공지능 기법을 구현하기 위해서는 NPC 지능 플랫폼은 아래와 같은 요구조건을 만족해야 한다.

●**Reactivity** : 게임 환경에서 상황 변화를 NPC 지능 플랫폼이 인지하고 반응할 수 있어야 한다.

●**Independence** : 인공지능 엔진은 게임 환경에 독립적으로 작동해야한다. 즉, 모든 게임 환경에서 사용가능하여야 한다.

●**World Interface** : 게임환경에서 인지하는 정보를 인공지능 플랫폼은 이용 가능해야 하고, 인공지능 플랫폼에서 수정한 정보는 게임환경에 적용가능해야 한다.

●**Embodied** : NPC가 이용할 수 있는 게임환경의 정보는 인간 플레이어가 얻을 수 있는 수준의 정보이어야 한다.

●**Flexible Architecture** : 하나의 전략을 위한 여러 인공지능 컴포넌트간의 조합을 할 수 있어야 한다.

●**Easy to Develop** : 인공지능 디자이너가 요구한 사항을 개발자가 쉽게 구현할 수 있어야 한다.

●**Module** : 모듈화를 통해 새로운 인공지능 기법의 추가/수정이 용이해야한다.

●**Real-time** : 게임 NPC 지능 플랫폼이 게임 성능을 저하시켜서는 안된다. 즉, 게임 NPC 인공지능 플랫폼이 자원을 선

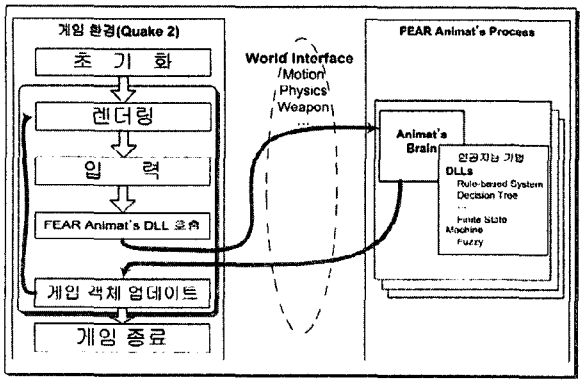
점한 상태를 유지하지 않도록 해야한다.

3. 기존 게임 NPC 지능 개발 플랫폼

본 절에서는 이전에 연구, 개발된 게임 NPC 지능 개발 플랫폼들의 구조와 특징을 알아본다.

3.1 FEAR

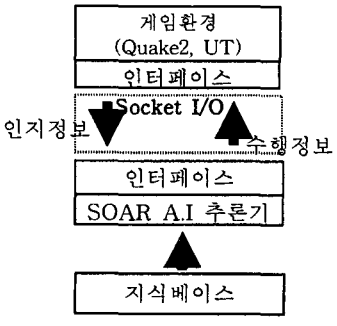
FEAR는 지능적인 NPC 개발을 위한 플랫폼으로 Flexible Embodied Animat aRchitecture를 의미한다. 이 프로젝트는 현재 오픈 소스 프로젝트로 제공되고 있다[6]. (그림 1)에서와 같이 FEAR의 전체적인 구조는 일반 사용자의 입력을 처리한 후, 인공지능을 설계한 FEAR Animat의 DLL을 호출하고, 그 결과를 게임 환경에 반영한다[6]. FEAR의 Animat은 각각 독립적으로 구현되어 있는 인공지능 기법 DLL을 사용하여 NPC에 지능을 부여한다. 이와 같은 인공지능 기법 컴포넌트는 계층적인 구조로 유연한 구조를 엔지니어에게 제공한다. 또한, FEAR는 게임환경과의 통신을 위해 일반적으로 사용하는 인지(sensors)와 효과(Effectors) 인터페이스를 Wepon, Movement, Physics 등과 같이 기능별로 월드 인터페이스를 구성한다. 마지막으로 FEAR에서는 NPC의 지능을 구현함에 있어 XML과 scons[7]를 이용하여 지능을 담당하는 핵심 부분만을 제외한 나머지 소스 코드를 자동으로 생성하여 개발을 도와주고 있다.



(그림 1) FEAR 아키텍처

3.2 SOAR

Soar의 구조는 (그림 2)와 같이 크게 추론기(Inference Machine), 인터페이스, 지식 베이스(Knowledge Base)로 구성되어 있다[8]. 추론기는 현재 상태에 내외부적인 행동을 구현하기 위해 지식베이스로 부터 지식을 적용한다. SOAR는 게임환경과 NPC간의 통신을 위해 Socket I/O를 사용하여 NPC 지능 개발 플랫폼과 게임환경이 같은 머신(Machine)에 있을 때뿐만 아니라, 네트워크로 연결도 가능하다.

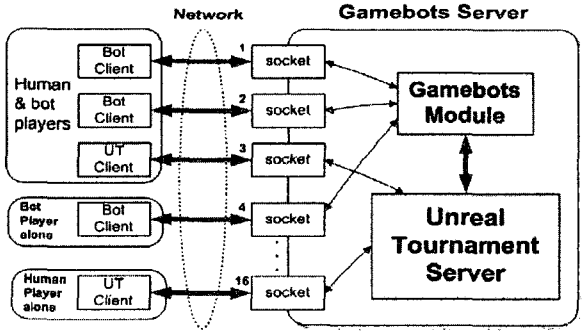


(그림 2) SOAR 아키텍처

3.2 Gamebots

Gamebots는 University of Southern California's Information Sciences Institute와 Carnegie Mellon University에 의해 공동 개발 진행된 프로젝트이다[9]. 이 프로젝트는 인터넷을 통한 다중 플레이어를 지원하는 언리얼토너먼트[10] 게임 플랫폼을 기반으로 인공지능 개발을 지원하는 멀티 에이전트(multi-agent) 시스템이다.

(그림3)에서 보는 것과 같이 Gamebots는 네트워크 소켓을 통해 보트(bot) 클라이언트에 연결되어 캐릭터를 조정할 수 있게 해주는 UT Module에 있다. Gamebots 서버는 네트워크를 통해 캐릭터의 인지 정보를 제공한다. 이때 에이전트가 인지할 수 있는 정보는 인간 플레이어가 인지하는 범위를 벗어나지 않는다. 제공되어진 정보에 기반하여 클라이언트(보트 또는 인간 플레이어)는 캐릭터가 어떤 행동을 할 것인지 결정할 수 있고, 네트워크를 통해 이동하거나 전투하기, 대화하는 등의 명령을 전달할 수 있다.



(그림 3) Gamebots 아키텍처 (Human Player는 UT Server에 직접적으로 연결되어 있고, bot들은 Gamebots Module에을 통해 연결되어 있다.)

4. 게임 NPC 지능 개발 플랫폼 비교

3절에서 알아본 세가지 플랫폼 모두 전체적인 구조는 게임 환경으로부터 인지 정보를 얻은 후, 그 정보를 이용하여 NPC에 지능을 담아 NPC가 게임 환경에서 행동을 하는 구조를 가지며, 게임 환경 변화에 반응한다. 또한, 게임 환경의 모든 데이터 정보를 알 수 없고, 인지되어지는 정보만을 이용한다.

하지만, FEAR와 SOAR 플랫폼은 게임 환경에 독립적으로 설계되어진 반면, Gamebots 언리얼토너먼트 환경에 의존적으로, 게임 환경과의 독립성을 보장하지 못하기 때문에 Gamebots상에서 개발된 NPC를 다른 게임 환경에 적용할 수 없는 단점을 지닌다.

또한, FEAR, SOAR, Gamebots는 일반적인 경우 Real-time을 보장하는 최대 NPC 클라이언트의 수를 각각 32, 13~15[11], 16이다. 이것은 게임환경과 통신하는 방법과 관계있다. FEAR

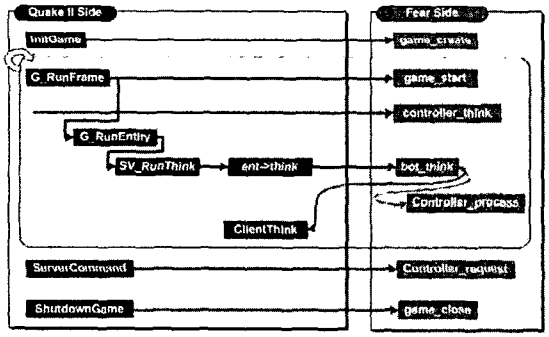
는 직접 Animat DLL을 호출하기 때문에 가장 많은 수의 클라이언트를 처리할 수 있으며, SOAR와 Gamebots는 소켓을 사용하기 때문에 클라이언트 처리능력이 떨어진다.

하지만, DLL을 직접 호출하는 방식의 FEAR의 경우 정상적인 경우 1/10초에 한번씩 NPC의 지능 루틴을 호출하게 된다. 하지만, 하나의 NPC 지능 루틴이 많은 자원을 소요하게 되면, 초당 NPC 호출 수는 2~3으로 real-time을 보장하지 못한다. 이런 문제는 (그림 4)에서 보여주는 내부구조를 통해 알 수 있다. Quake2 게임환경에 의해 bot_think가 호출되면 제어권은 FEAR에게 넘어가고, 1/10초내에 제어권이 ClientThink로 넘어가지 못하면 Real-time을 보장하지 못하게 된다. FEAR에서 나타난 이런 문제점의 해결하기 다음과 같은 방안들을 제시한다.

●다중 스레드(Multi-Threads) 기법 : *Robocode*[12]에서 각 로봇을 호출시 사용하는 방법으로 각 NPC 프로세스를 병렬적으로 처리한다. 즉, bot_think 함수를 병렬적으로 호출하는 기법이다.

●공유메모리(Shared-Memory)를 이용한 IPC (InterProcess Communication) 기법 : 게임환경과 게임 NPC 지능 개발 플랫폼간의 통신에 이용되는 구조체를 공유메모리를 통해 양 플랫폼간의 통신을 지원한다. 즉, *ent->think* 함수에서 bot_think 함수를 직접 호출하는 것이 아니라, 공유메모리를 읽고, bot_think는 게임환경과 FEAR의 통신에 이용되는 구조체를 공유메모리에 쓰게 된다.

위와 같은 두가지 기법은 게임환경과 병렬적으로 처리되기 때문에, 하나의 NPC 지능 처리 루틴이 자원을 선점하는 문제를 해결할 수 있다. 그리고, SOAR와 Gamebots가 사용하는 소켓방식보다 많은 NPC 클라이언트를 처리할 수 있다.



(그림 4) Quake2와 FEAR의 내부함수 연결(bot_think가 호출되면 제어권은 FEAR에게 넘어간다. 1/10초내에 제어권이 ClientThink로 넘어가지 않으면 Real-time을 보장하지 못한다.)

5. 결론 및 향후 연구 과제

최근 게임에서 NPC의 지능은 점차 중요해지고 있다. 본 논문에서는 게임 NPC 지능 개발 플랫폼을 설계시 고려사항을 알아보고 기존 NPC 지능 개발 플랫폼을 비교 분석한 결과, 대체적으로 FEAR가 많은 장점을 보여주고 있지만, real-time 조건을 만족 시키지 못하는 문제점을 발견하였다. 이런 문제점에 대해 Robocode에서 사용된 다중 스레드 기법과 공유메모리를 이용한 IPC 기법을 제안하였고, 향후 이에 대한 연구가 필요하다. 또한, 현재 게임 개발시 사용하는 다양한 게임 환경과 NPC 지능 개발을 위한 플랫폼간의 완전한 독립화를 위해서는 이들 간의 인터페이스 표준화 문제도 앞으로 해결해야할 문제이다.

6. 참고문헌

[1]Turkle, S. Constructions and Reconstructions of Self in Virtual Reality: Playing in the MUDs, Mind, Culture and Activity, vol. 1, no. 3, 1994.

[2]Zubek, R. & Khoo, A., Making the Human Care: On Building Engaging Bots, Proceedings of the 2002 AAAI Spring Symposium on Artificial Intelligences and Interactive Entertainment. Palo Alto, California, pp. 103-108., 2002.

[3]West, N., The Way Games Ought to Be, Next Generation, Imagine Media Inc., Brisbane, CA, pp.104-105, 1998.

[4]Schreiner, T., Arificial Intelligence in Game Design, http://ai-depot.com/GameAI/Design.html, 2004.

[5]Penny Baillie-de Byl, Programming Believable Characters for Computer Games, Charles river media, 2004.

[6]Alex J. AI Game Development Synthetic Creatures with Learning and Reactive Behaviors. 2003.

[7]http://www.scons.org/.

[8]van Lent, M. C. and Laird, J. E. Developing an Artificial Intelligence Engine. In Proceedings of the 1999 Game Developers's Conference, 577-587. 1999.

[9] R. Adobbati, Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS.2001.

[10]http://www.unrealtournament.com/.

[11]Laird, J. E., M. Assanie, et al. A Test Bed for Developing Intelligent Synthetic Characters. AAAI 2002 Spring Symposium Series: Artificial Intelligence and Interactive Entertainment. 2002.

[12]http://www-903.ibm.com/developerworks/kr/robocode/robocode.html.