온라인 컴퓨터 게임에서 지능형 NPC 구현을 위한 제어구조

이경록[0] 김인철 이재호*
경기대학교 정보과학부, *서울시립대학교 전자전기컴퓨터공학부
{lkr0211, kic}@kyonggi.ac.kr, jaeho@ee.uos.ac.kr

Control Architecture for Implementing Intelligent Non-Player Characters in Online Computer Games

Kyoung-Rog Yi[0] In-Cheol Kim Jaeho Lee*
Dept. of Computer Science, Kyonggi University
*Dept. of Electronic Engineering, University of Seoul

요 약

본 논문에서는 온라인 컴퓨터 게임에서 지능형 NPC가 수행할 수 있는 역할과 이러한 지능형 NPC를 구현하기 위해 요구되는 제어구조의 특성을 살펴본다. 그리고 종래에 사용되던 NPC 제어 방법들과 이들이 갖는 문제점들을 살펴보고 새로운 대안으로서 NPC 제어를 위해 UM-PRS의 적용을 제안한다. 따라서 본 논문에서는 UM-PRS에 대한 간략한 소개와 NPC 제어구조에 적합한 UM-PRS의 특성을 설명한다. 또 본 논문에서는 대표적인 롤플레잉(role-playing) 게임인 Ultima Online에서 개발중인 Umtimabot의 설계를 통해 UM-PRS의 적용가능성과 효과를 알아본다.

1. 서론

최근 들어 인터넷을 이용한 온라인 컴퓨터 게임과 이용자의 수가 급격히 늘어나고 또한 이전보다 더욱 생동감 있고 지능적인 컴퓨터 제어 캐릭터(computer controlled character) - 통상 NPC(Non-Player Character) 라고 불림 - 에 대한 게이머들의 요구가 증가함에 따라, 게임 개발자들은 이러한 지능형 NPC들을 구현하기 위한 지능형 에이전트 기술에 더욱 주목하게 되었다. 이와 같이 기존 게임에 지능형 에이전트 기술이 도입되면 얻을 수 있는 가장 큰 잇점은 역시 이러한 지능형 NPC들로 인해 게임의 재미를 더욱 높일 수 있다는 점이다. 지능형 NPC와 더불어 혹은 지능형 NPC를 상대로 게임을 펼치는 것은 온라인 상의 다른 게이머들과 게임하는 것과 더욱 흡사한 박진감을 줄 수 있을 것이다. 그 뿐만 아니라 좀더 유연성이 높은 에이전트 제어구조를 도입하면 서로 다른 게임들에 공통적으로 적용할 수 있는 추론엔진과 재사용 가능한 지식베이스로 인해 게임 개발과 유지보수가 한층 수월해질 수 있다. 이러한 지능형 에이전트 기술이 컴퓨터 게임에 줄 수 있는 긍정적인 효과보다 더욱 지능형 에이전트 연구자들에게 관심을 끄는 부분은 오늘날 컴퓨터 게임이 기계학습, 지능형 에이전트 구조, 지식표현, 인터페이스 설계, 휴먼-컴퓨터 상호작용 등과 같은 많은 연구 주제에 관한 새로운 문제를 제공하고 해결책을 시도해볼 수 있는 좋은 도메인을 제공한다는 점이다.

종래의 온라인 컴퓨터 게임에서 NPC를 구현하기 위한 대표적인 방법은 게임서버와 마찬가지로 각 NPC를 C 코드로 작성하고 수많은 IF문과 SWITCH문을 통해 NPC의 행동을 제어하는 것이었다. 하지만 이러한 방법은 NPC의 행위가 더욱 지능적이고 복잡해지면, NPC를 구현하는 C 코드는 더욱 복잡해져 디버깅하거나 새롭게 개선시키기 어려워진다는 단점이 있다. 자주 쓰이는 또 다른 NPC 구현방법은 C 코드와는 다른 별도의 NPC 행동 스크립트 언어와 인터프리터를 정의해두고 이것을 이용해 각 NPC의 행위를 기술하고 실행하는 것이다. 하지만 기존의 게임 스크립트 언어와 인터프리터들은 표현 가능한 실행제어구조가 매우 제한적이고 특정 스크립트의 실행 중에 발생할 수 있는 상황변화에 좀더 신속히 반응할 수 없다는 단점을 가지고 있다.

본 논문에서는 종래의 NPC 구현 방법들이 갖는 문제점들을 살펴보고, NPC 제어를 위해 UM-PRS의 적용을 제안한다. 본 논문에서는 NPC 제어구조에 적합한 UM-PRS의 특징을 살펴보고, 대표적인 롤플레잉(role-playing) 온라인 게임인 Ultima Online에서 현재 개발중인 지능형 NPC Ultimabot의 설계를 통해 UM-PRS의 적용 가능성과 효과를 알아본다.

2. 온라인 게임과 지능형 에이전트

대표적인 온라인 게임 장르들로는 Doom, Quake, Tomb Raider 등과 같은 액션게임(action game), Diablo, Ultima 등과 같은 롤플레잉게임(role-playing game), King's Quest, Full Throttle, Grim Fangango

와 같은 어드벤처게임(adventure game), Starcraft, Myth, Close Combat 과 같은 전략게임(strategy game), SimCity와 같은 God게임, FIFA와 같은 팀 스포츠게임(team sports game) 등이 있다. 이와 같은 다양한 장르의 온라인 게임들에서 공통적으로 그 중요성이 증가되고 있는 것은 어떤 휴먼 플레이어로부터도 직접적인 제어를 받지않고 스스로 동작하는 NPC들의 역할이다. 이러한 NPC들은 일종의 게임 에이전트들로서, 게임의 유형에 따라 다양한 역할을 수행한다. NPC 즉 게임 에이전트가 수행하는 대표적인 역할들로는 액션게임 등에서 스스로 효과적인 경로계획을 하는 등 보다 지능적인 적(enemy)으로서의 역할, 휴먼 플레이어 캐릭터의 파트너(partner)로서의 역할, 게임 중 독특한 나름의 기능을 수행하는 도우미 캐릭터(support character)의 역할, 애완동물이나 우주선과 같은 게임아이템(game item)의 역할, 스포츠게임이나 액션게임 등에서 진행상황을 자연어로 설명해주는 해설자(commentator)로서의 역할 등이 있다.

NPC들이 게임에서 자신의 역할을 현실감 있게 수행해내기 위해서는 좀더 높은 지능성을 갖추어야 하며 이러한 지능성은 NPC의 행위를 기술하고 실행하는 제어구조에 크게 의존한다. 일반적으로 게임 에이전트인 NPC는 인지(sense), 사고(think), 행동(act)의 결정주기(decision cycle)를 반복한다. 인지단계에서는 게임서버로부터 자신의 현재 위치에서 시각과 청각에 감지되는 센서정보를 받아들여 현재 게임상황을 파악하고, 사고단계에서는 현재 상황에서 자신의 역할에 가장 부합되는 행동을 결정하고, 행동단계에서는 선택된 행동을 수행하도록 게임서버에 명령을 내린다. 특별히 게임 에이전트 제어구조에 요구되는 대표적인 특성들로는 환경변화에 대한 신속한 반응성(reactivity), 현재의 행동이 이미 실행했거나 실행을 계획하고 있는 행동들과 일관성을 가지는 문맥의존성(context specificity), 현재의 목표를 위해 다양한 고수준 또는 저수준의 행위를 선택할 수 있는 유연성(flexibility), 휴먼 플레이어와 동일한 수준의 강점과 약점을 가짐으로써 휴먼 플레이어와 닮았다는 느낌을 주는 사실감(reality), 그리고 끝으로 게임개발의 용이성(ease) 등이 있다.

종래의 대표적인 NPC 구현 방법은 NPC 전체를 C (혹은 C++) 코드로 작성하고 수많은 IF문과 SWITCH 문을 통해 NPC의 행동을 제어하는 것이었다. 하지만 이러한 방법은 NPC의 행위가 복잡해지면, NPC를 구현하는 C 코드도 더욱 복잡해져 디버깅하거나 새롭게 개선시키기에 더욱 어렵다는 단점이 있다. 또한 이렇게 구현되는 NPC들은 과거에 수행한 행동들이나 상태정보를 별도로 저장하지 않고 단지 매 단계마다 현재 게임상황에 따라서만 반응하는 자극-반응형(stimulus-response) 에이전트들이 대부분이었다. 일반적으로 이러한 단순 반응형 NPC들은 상황변화에는 매우 신속하게 대응할 수 있으나 문맥정보(contextual information)의 부재로 좀더 높은 수준의 지능행위를 보여주기 어렵다. 널리 쓰이는 또 다른 NPC 구현방법은 C 코드와는 다른 별도의 NPC 행동 스크립트 언어와 인터프리터를 이용해 각 NPC의 행위를 기술하고 실행하는 것이다. 이러한 스크립트 기반의 NPC들은 행동들의 시퀀스인 스크립트로 인해 자연스럽게 보다 풍부한 문맥정보를 이용할 수 있어 행동의 일관성을 유지할 수 있다. 하지만 기존의 게임 스크립트 언어와 인터프리터들은 표현 가능한 제어구조가 매우 제한적이고 특정 스크립트의 실행 중에 발생할 수 있는 상황변화에 좀더 신속히 반응할 수 없다는 단점을 가지고 있다.

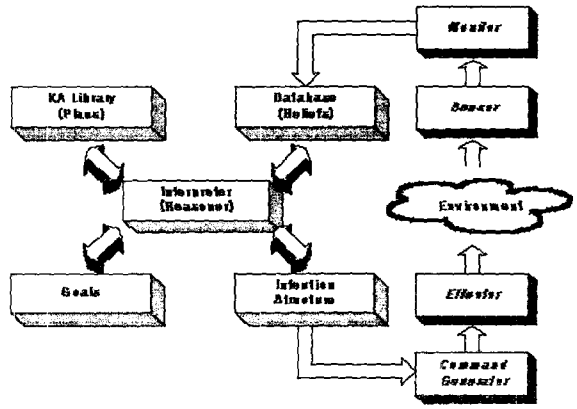

그림 1 UM-PRS의 구조

- **Name**
- **Documentation:** comment
- **Purpose:** goal
- **Context:** condition
- **Body:** procedure
  Action: achieve, execute, query, test, assert, retract
- **Effect:** simulation mode
- **Failure**

그림 2 KA의 구성요소

3. NPC 제어구조로서 UM-PRS

UM-PRS는 목표-지향적인 추론(goal-directed reasoning)과 반응적 행위(reactive behavior)를 통합한 범용의 에이전트 구조인 Procedural Reasoning System(PRS)을 C++로 구현한 것이다. UM-PRS는 (그림 1)과 같이 추구하는 목표들(goals), 실세계 모델을 저장하는 데이터베이스(database), 목표를 달성하기 위한 절차적 지식을 표현한 지식영역(Knowledge Area, KA)들의 집합, 그리고 이들로부터 끊임없이 새로운 의도들을(intentions) 결정하는 인터프리터(interpreter)들로 구성된다. 에이전트의 행위를 절차적 지식형태로 표현하는 각 지식영역(KA)은 (그림 2)와 같이 크게 Name, Documentation, Purpose, Context, Body, Effect, Failure 부분으로 구성된다. 특히 이 중에 Purpose부분은 이 KA가 달성해줄 수 있는 목표를 나타내고, Context부분은 이 KA가 실행되는 동안 만족되어야 하는 환경조건을 명시하는 것이다. 실제 수행되어야 할 절차 또는 계획을 표현하는 Body부분은 achieve, execute, assert 등의 다양한 동작(action)과 이들에 대한 순차, 반복, 분기 등의 풍부한 제어구조를 제공한다. UM-PRS의 인터프리터는 통상 현재의 목표를 달성할 수 있는 가장 적합한 KA를 찾아 그 KA에 기술된 절차들

을 차례대로 수행해가지만 하나의 KA가 채 완료되기 전이라도 Context가 만족되지 않는 환경변화가 발생하면 즉시 해당 KA의 수행을 중단하고 새로운 상황에 맞는 다른 KA를 찾아 수행한다. 이러한 UM-PRS의 반응성, 문맥의존성, 유연성 등은 앞서 언급한 NPC 제어구조에 대한 요구 사항들을 충분히 만족한다. UM-PRS기반의 NPC의 경우, 기존의 단순 스크립트나 자세한 C 코드를 직접 쓰는 대신 잘 구조화된 다수의 KA와 달성하고자 하는 최종목표로 NPC의 행위를 쉽게 기술할 수 있다. 그리고 단순히 몇 개의 KA만을 변경하거나 추가함으로써 NPC의 행위를 쉽게 변경하거나 개선시킬 수 있으며, 이미 잘 정의된 기본적인 KA를 재사용함으로써 새로운 게임이나 새로운 NPC의 개발도 용이하다.

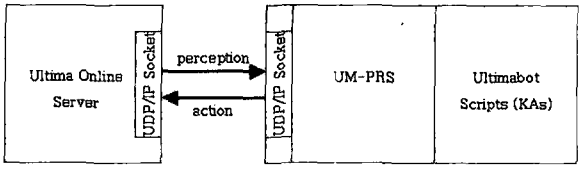

그림 3 게임서버와 UM-PRS Ultimabot의 인터페이스

4. UM-PRS Ultimabot

NPC 제어구조로서 UM-PRS의 적용 가능성과 효과를 알아보기 위해 현재 상용 서비스되고 있는 대표적인 롤플레잉 온라인 게임인 Ultima Online 도메인상에 새로운 지능형 NPC인 Ultimabot를 개발하고 있다. (그림 3)은 게임서버와 Ultimabot의 인터페이스 및 Ultimabot의 내부구조를 보여준다. 특히 Ultimabot는 게임서버와 강결합된 상태에서 서버내의 필요한 정보를 임의로 접근할 수 있는Ultima Online의 기존 NPC들과는 달리, 독립된 하나의 클라이언트 프로그램으로 구현함으로써 역시 하나의 클라이언트 프로그램을 통해 게임에 접근하는 휴먼 플레이어와 동일한 인지 및 정보의 제약을 갖도록 설계하였다. 서버로부터 메시지를 통해 전달 받는 Ultimabot의 인지정보와 서버에게 전달할 수 있는 기본 동작들은 (그림 4)와 같다. Ultimabot는 돌아다니며 나쁜 monster들을 공격함으로써 플레이어 캐릭터들을 보호하는 전사의 역할을 수행한다. Ultimabot가 보여줄 최상위 행위들은(top-level behaviors) 다음과 같다 :

Wander, Guard_region, Enhance_skill, Buy_skill, Train_skill, Wait_healing, Attack_monster, Chase_monster

```
Sensor Information :
    Visible items/characters
        itemID/charID, Location (x, y, z) , Direction , Skin_color
    Near characters
        Spoken messages
    Itself
        Location (x, y, z), Direction, Skills, Strength, dexterity, and intelligence, items

Primitive Action Commands :
    Attack $target
    Change_war_mode $flag
    Move $direction $sequence_num
    Talk $speech_type $color $font $message
    Buy $vendorID $item
    Sell $shopkeeper $itemID $quantity
    Pickup $itemID $num_of_items
    Drop $itemID $x_loc $y_loc $z_loc $containerID
```

그림 4 UM-PRS Ultimabot의 센서정보와 기본동작

```
KA (
    NAME    "Guarding KA"
    DOCUMENTATION  "Gurading a certain region by attacking bad monsters"
    PURPOSE  ACHIEVE guard $map_region
    CONTEXT  FACT safe $map_region  "False"
    BODY
        EXECUTE next_character $map_region $temp_char;
        WHILE  TEST (!= $temp_char NULL)
        (
          EXECUTE calc_distance $temp_char $distance;
          EXECUTE get_NPC_AI_type $temp_char $type;
          EXECUTE get_status $temp_char $alive;
          TEST (and (<= $distance 10) (== $type 0x02) (== $alive "True"));
          ACHIEVE attack_monster $temp_char;
          EXECUTE next_character $map_region $temp_char;
        )
        UPDATE (safe $map_region) (safe $map_region "True");
)
```

그림 5 UM-PRS Ultimabot의 한 KA

(그림 5)는 Ultimabot의 행위 일부를 UM-PRS의 KA로 기술한 예이며, (그림 6)은 게임 진행 중 Ultimabot가 monster를 만나 전투를 벌이는 장면이다.

그림 6 Monster를 공격하는 Ultimabot의 실행화면

5. 결론

본 논문에서는 새로운 NPC 제어방법으로서 UM-PRS 에이전트 구조의 적용을 제안하였고 Ultimabot의 설계를 통해 UM-PRS의 적용가능성을 알아보았다

참고 문헌

[1] 이만재, "온라인 게임 엔진 기술 동향", 한국정보과학회지, 제20권 제1호, pp.12-18. 2002.

[2] Jaeho Lee, Marcus J. Huber, Edmund H. Durfee, and Patrick G. Kenny, "UM-PRS: an implementation of the procedural reasoning system for multirobot applications", Proc. of CIRFFSS-94, pp.842-849, 1994.

[3] John E. Laird, Michel van Lent, "Human-level AI's Killer Application : Interactive Computer Games", Proc. of AAAI-2000, August 2000.

[4] UOX Freelancers Group, The UOX Hackers' Guide, 2001.