3차원 가상환경에서 동작하는 지능형 에이전트의 구조와 경로 찾기 행위

김 인 철† · 이 재 호††

요 약

본 논문에서는 대표적인 3차원 일인칭 액션 게임인 Unreal Tournament 게임과 이것에 기초한 지능형 에이전트 연구용 테스트베드인 Gamebots 시스템을 소개한다. 그리고 이들이 제공하는 3차원 가상환경에서 동작하는 지능형 NPC인 KGBot의 설계와 구현에 대해 설명한다. KGBot는 Gamebots 시스템 내의 하나의 보트 클라이언트이다. KGBot는 3차원 가상환경 안에 숨겨진 목표점들을 찾아 효과적으로 점령하는 임무를 수행한다. KGBot는 범용의 BDI 에이전트 구조인 UM-PRS를 제어엔진으로 채용하고 있으며, 복잡한 행위들을 여러 계층으로 표현한 계층화된 지식베이스를 가지고 있다. 본 논문에서는 미지의 월드를 효과적으로 탐색함으로써 숨겨진 목표점들의 위치를 빨리 파악하고, 흩어져 있는 이동점들과 경로들을 찾아내어 경로 그래프를 작성하며, 이것에 기초하여 특정 목적지까지 최적의 이동 경로를 계획하는 KGBot의 지능 행위들에 대해 자세히 설명한다. 그리고 끝으로 다양한 3차원 지도를 이용한 실험을 통해 KGBot의 월드 탐색 전략과 제어엔진의 성능을 분석해본다.

Architecture and Path-Finding Behavior of An Intelligent Agent Deploying within 3D Virtual Environment

Incheol Kim† · Jaeho Lee††

ABSTRACT

In this paper, we introduce the Unreal Tournament (UT) game and the Gamebots system. The former is a well-known 3D first-person action game and the latter is an intelligent agent research testbed based on UT. And then we explain the design and implementation of KGBot, which is an intelligent non-player character deploying effectively within the 3D virtual environment provided by UT and the Gamebots system. KGBot is a bot client within the Gamebots System. KGBot accomplishes its own task to find out and dominate several domination points pre-located on the complex surface map of 3D virtual environment. KGBot adopts UM-PRS as its control engine, which is a general BDI agent architecture. KGBot contains a hierarchical knowledge base representing its complex behaviors in multiple layers. In this paper, we explain details of KGBot's intelligent behaviors, such as locating the hidden domination points by exploring the unknown world effectively, constructing a path map by collecting the waypoints and paths distributed over the world, and finding an optimal path to certain destination based on this path graph. Finally we analyze the performance of KGBot's exploring strategy and control engine through some experiments on different 3D maps.

키워드 : 지능형 에이전트(intelligent agent), 가상환경(virtual environment), 에이전트 구조(agent architecture), 경로 찾기 행위(path-finding behavior)

1. 서 론

최근 들어 개인용 컴퓨터와 인터넷 보급이 확산됨에 따라 인터넷을 이용한 온라인 컴퓨터 게임과 게임을 즐기는 이용자의 수가 급격히 늘어나고 있다. 대표적인 온라인 컴퓨터 게임들로는 Doom, Quake, Tomb Raider 등과 같은 액션게임(action game), Diablo와 Ultima Online, Everquest 등과 같은 롤플레잉게임(role-playing game), King's Quest, Full Throttle, Grim Fangango와 같은 어드벤처게임(adventure game), Starcraft, Myth, Close Combat과 같은 전략게임(strategy game), SimCity와 같은 God 게임, FIFA와 같은 팀 스포츠 게임(team sports game) 등이 있다[12, 13, 20]. 이와 같은 다양한 장르의 온라인 게임들에서 공통적으로 그 중요성이 증가되고 있는 것은 휴먼 플레이어로부터 직접적인 제어를 받지않고 스스로 동작하는 컴퓨터 제어 캐릭터(computer controlled character), 즉 NPC(non-player character)들의 역할이다. 컴퓨터 게임을 위한 자율 캐릭터 에이전트들인 NPC들은 게임의 유형에 따라 다양한 역할을 수행한다. 컴퓨터 게임에서 NPC들이 수행하는 대표적인 역

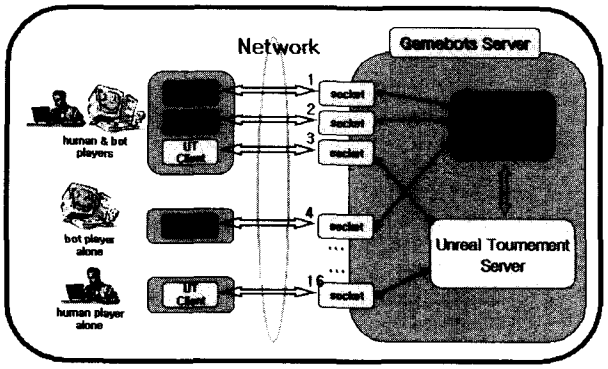
※ 본 연구는 첨단정보기술 연구센터를 통하여 과학재단의 지원을 받았음.
† 종신회원 : 경기대학교 정보과학부 교수
†† 정 회 원 : 서울시립대학교 전자전기컴퓨터공학부 교수
논문접수 : 2002년 7월 29일, 심사완료 : 2003년 1월 7일

할들로는 지능적인 적(enemy)으로서의 역할, 휴먼 플레이어의 파트너(partner)로서의 역할, 게임 진행을 지원하는 도우미 캐릭터(support character)의 역할, 애완동물이나 차량, 우주선, 미사일과 같이 제한적인 자율성을 갖는 특수 아이템(special item)의 역할, 스포츠게임이나 액션게임 등에서 진행상황을 자연어로 설명해주는 해설자(commentator)로서의 역할 등이 있다[6, 14]. 이처럼 게임에서 다양한 역할을 수행하는 지능형 NPC의 중요성이 널리 인식됨에 따라, 최근 게임 개발자들 사이에 이러한 지능형 NPC들을 구현하기 위한 인공지능 및 지능형 에이전트 기술에 관한 관심이 더욱 높아졌다[1, 21, 22].

본 논문에서는 대표적인 온라인 3차원 일인칭 액션 게임인 Unreal Tournament 게임과 이것에 기초한 지능형 에이전트 연구용 테스트베드인 Gamebots 시스템을 소개한다. 그리고 이들이 제공하는 3차원 가상환경에서 효과적으로 동작하는 지능형 NPC인 KGBot의 설계와 구현에 대해 설명한다. KGBot는 Gamebots 시스템에서 동작하는 하나의 보트 클라이언트(bot client)로서, 3차원 가상환경 안의 복잡한 지도 위에 놓여진 특정 목표점(domination point)들을 찾아 자율적으로 점령하는 자신의 임무를 수행한다. KGBot는 범용의 BDI 에이전트 구조인 UM-PRS[4]를 제어엔진으로 채용하고 있으며, 복잡한 행위들을 효과적으로 기술하기 위해 계층화된 지식베이스를 가지고 있다. 본 논문에서는 특히 지도상에 흩어져 있는 이동점들(waypoints)에 대한 제한적인 센서정보에만 의존하여 자신이 놓여진 월드의 지도를 작성하고, 이동 경로를 찾는 KGBot의 행위에 대해 자세히 설명한다. 그리고 끝으로 서로 다른 3차원 지도상의 실험을 통해 이러한 KGBot의 경로 찾기 행위의 성능을 분석해본다.

2. Unreal Tournament 게임과 Gamebots 시스템

Epic사의 Unreal Tournament 게임은 대표적인 3D 일인칭 액션 게임이다[4]. 이 게임은 온라인 상용게임(commercial game)이면서도 대부분의 소스가 공개되어 있다. UT 게임은 견고한 3D 시뮬레이션 엔진, 다양한 게임 유형과 지도(map)를 제공하고 있으며, 게임의 확장성을 위해 전용 스크립트 언어인 Unreal Script도 제공하고 있다. 일인칭 액션 게임이므로 UT 게임에서 각 휴먼 플레이어는 3차원 가상환경에서 자신의 캐릭터와 동일한 관점에서 시각정보와 청각정보를 제공받게 되므로 게임의 현실감과 몰입감이 한층 높다. 한편, Gamebots 시스템[16]은 지능형 에이전트 및 다중 에이전트 연구를 목적으로 USC의 ISI 연구소와 CMU에 의해 공동으로 개발된 UT 게임 기반의 지능형 에이전트 연구용 테스트베드(testbed)이다. 그동안 에이전트 연구에 성공적으로 이용되어온 2차원 가상 시뮬레이션 환경인 RoboCup(Robot World Cup) 축구 서버에 이어, 보다 복잡하고 현실적인 물리법칙이 적용되는 3차원 공간과 뛰어난 그래픽, 그리고 다양한 작업목표와 해결 문제들이 존재하는 새로운 연구환경으로 온라인 3차원 전략게임과 액션게임들을 고려하게 되었고, 이 중에서 UT 게임이 이러한 목적으로 선정되어 확장되었다[15, 19].



(그림 1) Gamebots 시스템 구조

Gamebots 시스템의 핵심은 제어 프로그램이 게임서버에 존재하는 내장형(built-in) NPC들과는 달리 (그림 1)과 같이 네트워크 소켓을 통해 접속하는 원격의 NPC - Gamebots 시스템에서는 보트 클라이언트(bot client)라고 부름 - 들에게 휴먼 플레이어와 동일한 센서정보를 제공하고 역으로 원격 NPC의 명령을 받아 수행해주는 Gamebots 서버모듈이다. Gamebots 서버가 각 원격의 NPC에게 전달하는 메시지들은 크게 초기 메시지(initial message), 동기 메시지(synchronous message), 비동기 메시지들(asynchronous message)로 구성된다. 각 보트 클라이언트의 연결 초기 단계에서는 초기화 메시지의 교환을 통해 서버측에서는 보트 클라이언트에게 게임 타입과 게임의 승리조건 등을 알려주며, 보트 클라이언트는 자신이 제어하는 캐릭터의 이름, 소속 팀 등을 서버측에게 알려준다. 일단 연결이 되고 나면, 서버는 동기 메시지 모드와 비동기 메시지 모드를 옮겨 다니며 캐릭터들의 지각정보(sensory information)를 해당 보트 클라이언트들에게 보내준다. 각 메시지는 메시지 유형(message type)과 다수의 속성-값(attribute-value) 쌍들로 구성된다. 동기 메시지는 약 1/10초 간격으로 계속해서 해당 캐릭터 에이전트의 감각기관(눈, 귀, …)을 통해 들어오는 시각정보와 자신의 위치 및 신체정보, 그리고 게임상황정보(팀별/개인별 득점)를 묶어 전달하는 것이다. 동기 메시지는 하나의 BEG 메시지와 하나의 END 메시지를 경계로 삼아 상기의 정보들을 담은 하나의 메시지 블록을 만들어 보내진다. 이에 반해 비동기 메시지는 해당 캐릭터가 게임 중 어떤 특별한 사건에 직면했을 때만 보내지는 것으로서, 벽에 부딪히거나 아이템을 줍거나 무기를 바꾸거나 늪지대와 같은 새로운 지역으로 진입했을때 관련 정보를 전달해

준다. 한편, 원격의 에이전트로부터 Gamebots 서버에 전달되는 명령 메시지 유형에는 RUNTO, TURNTO, JUMP, SHOOT, STOP, CHANGEWEAPON 그리고 에이전트간 대화를 위한 MESSAGE 등이 있다. 특히 원격의 보트 클라이언트들은 MESSAGE 명령에서 팀 전용 채널 또는 글로벌 채널을 지정함으로써 팀 동료들끼리만 메시지를 교환하거나 가상환경내의 모든 캐릭터들과의 메시지 교환이 가능하다. (그림 2)는 Gamebots 서버와 각 보트 클라이언트간에 교환되는 메시지 유형들을 보여주고 있다. UT 게임과 Gamebots 서버모듈이 제공하는 이러한 환경 안에서 원격 NPC 들이 DeathMatch, Domination, CaptureTheFlag 같은 게임들을 성공적으로 진행해나가기 위해서는 매우 복잡하고 다양한 지능적 행위와 능력을 보여줄 수 있어야 한다. 예컨대 제한적인 시각정보로부터 자신을 둘러싸고 있는 3차원 공간에 대한 전체지도를 작성하고 이를 바탕으로 원하는 목적지까지 효율적인 이동경로를 계획하는 능력에서부터 적들을 고려하여 동료들과 효과적으로 협조하거나 전략적 계획을 수립하는 등의 수준 높은 팀 행위(team behavior)까지 보여줄 수 있어야 한다

Initial Handshake Messages
- Game Type/Team/ID : INI

Synchronous Messages
- MSG Boundary : BEG, END
- Game State : GAM
- Player State : SLF, PLR
- Perceived Obkects : NAV, MOV, DOG, PLG

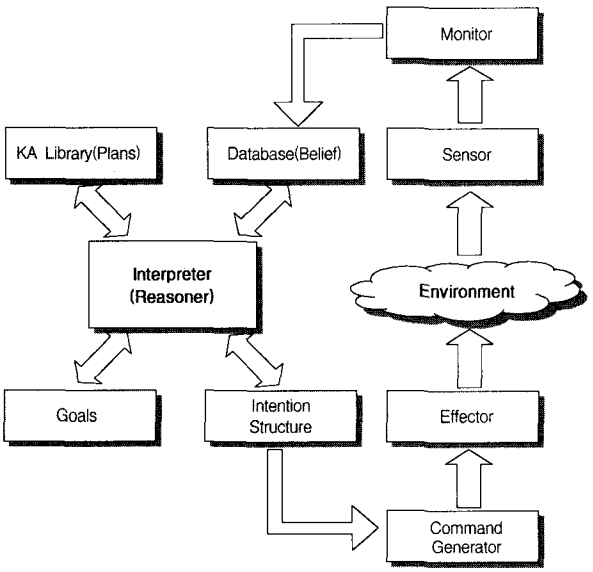
Asynchronous Messages
- Received Message : VMS, VMT
- Changed Zone : ZCF, ZCH
- Changed Weapon : CWP, AIN
- Collision/Fall : WAL, FAL, BMP
- See : SEE
- Damage : PRJ, KIL, DIE, DAM
- End Game : FIN

Commands
- Movement : SETWALK, STOP, JUMP, ROTATE, RUNTO, STRAFE, TURNTO
- Shoot : SHOOT, STOPSHOOT
- Change Weapon : CHANGEWEAPON
- Sending Message : MESSAGE
- Check Reachability : CHECKREACH
- Get Path : GETPATH

(그림 2) Gamebots 서버와 보트 클라이언트간의 전송 메시지

3. BDI 기반의 제어엔진

본 연구에서는 Gamebots 시스템이 제공하는 3차원 가상환경에서 동작하는 지능형 NPC인 KGBot의 효율적인 구현을 위해 대표적인 BDI 에이전트 구조인 UM-PRS[5]를 제어엔진으로 채용하였다. UM-PRS는 목표-지향적인 추론(goal-directed reasoning)과 반응적 행위(reactive behavior)를 통합한 범용의 BDI(Belief-Desire-Intention) 에이전트 구조이다. UM-PRS는 전통적인 숙고형 계획 시스템(deliberative planning system)들과는 달리, 변화하는 실세계에 대한 자신의 믿음을 바탕으로 자신이 내린 결정을 테스트하고 예측하지 못한 환경변화에 대해서는 동적으로 자신의 결정을 바꿀 수 있다. UM-PRS는 (그림 3)와 같이 실세계 모델을 저장하는 데이터베이스(database), 에이전트가 추구하는 목표들(goals), 목표를 달성하기 위한 절차적 지식을 표현한 지식영역(Knowledge Area, KA)들 - 계획(plan)이라고도 부른다 -, 그리고 이들로부터 끊임없이 새로운 의도들을(intentions) 결정하는 인터프리터(interpreter)들로 구성된다. 하나의 지식영역(KA)은 하나의 시스템 목표(system goal)나 질의(query)를 어떻게 만족할 수 있는지를 절차적으로 명세해놓은 것으로서, 계획(plan)이라고도 불린다.



(그림 3) UM-PRS의 구조

. 각 지식영역(KA)은 (그림 4)와 같이 크게 Name, Documentation, Purpose, Context, Body, Effect, Failure 부분으로 구성된다. Purpose 부분은 KA의 몸체부분을 성공적으로 수행했을 때 달성할 수 있는 목표를 나타내고, Context 부분은 해당 KA가 적용될 수 있는 실세계 상황을 명시한다. 그리고 KA가 실행되는 동안 계속해서 이 Context가 만족되는지 점검함으로써 KA가 여전히 적용 가능한지를 확인한다. Body 부분은 KA의 Purpose를 달성하기 위해 필요한 실행 단계들을 achieve, execute, assert 등의 다양한 기본함수(primitive function)들과 부속목표(sub-goal)들, 그리고 조건부 분기(conditional branch) 등으로 명시하는 것이다. UM-PRS의 인터프리터는 통상 현재의 목표를 달성할 수 있는 가장 적합한 KA를 찾아 그 KA에 기술된 절차들을 차례대로 수행해가지만 하나의 KA가 채완료되기 전이라도 Context가 만족되지 않는 환경변화가 발생하면 즉시 해당 KA의 수행을 중단하고 새로운 상황에 맞는 다른 KA를 찾

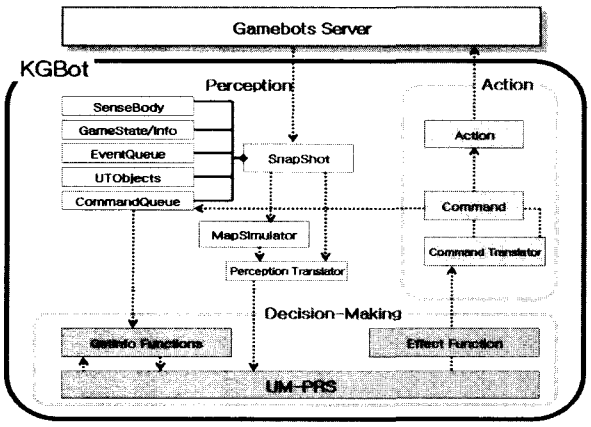
아 수행한다. 따라서 UM-PRS는 환경변화에 대한 높은 반응성(reactivity)과 이미 실행한 행동들과 일관성을 유지할 수 있는 문맥의존성(context specificity)을 가지고 있으며, 목표를 위해 고수준 또는 저수준의 다양한 행위를 선택할 수 있는 유연성(flexibility)을 가지고 있다. UM-PRS를 제어엔진으로 사용하는 NPC의 경우, 에이전트 행위의 모든 세부사항을 C나 Java와 같은 프로그래밍언어로 직접 구현하는 대신, 잘 구조화된 소수의 KA들로 NPC의 행위를 쉽게 기술할 수 있어 구현이 쉽다. 또한 NPC의 행위변경을 위해서는 일부 KA만을 변경하거나 추가함으로써 가능하며, 새로운 NPC의 개발을 위해 KA의 재사용도 용이하다는 장점이 있다.

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

(그림 4) KA의 주요 구성요소

4. KGBot의 구조

KGBot는 Gamebots 시스템에서 하나의 보트 클라이언트(bot client)로 동작하면서 지형이 복잡한 3차원 가상환경 안에서 적들에 대항해 아군과 연합하여 특정 목표 지점(domination point)들을 찾아 점령하는 지능형 NPC이다. KGBot의 내부구조는 (그림 5)와 같이 인식부(perception), 의사결정부(decision-making), 행동부(action) 등 크게 세 부분으로 구성된다. 인식부는 Gamebots서버로부터 센서정보를 담은 메시지를 받아들여 이것으로부터 자신의 신체 상태 정보(SenseBody), 게임 상태 정보(GameState/Info), 무기, 아이템, 이동점(waypoint)과 같은 각종 악세서리 개체들의 상태 정보(UTObject) 등을 추출해내어 의사결정부에 전달하는 역할을 수행한다. 의사결정부는 이러한 인식정보를 바탕으로 자신의 목표와 현재 상황에 적합한 행동계획을 세우거나 현재 취할 행동을 결정하는 역할을 수행하며, 행동부는 이렇게 결정된 행동에 대응하는 명령 메시지를 만들어 Gamebots 서버에게 보내는 역할을 수행한다.

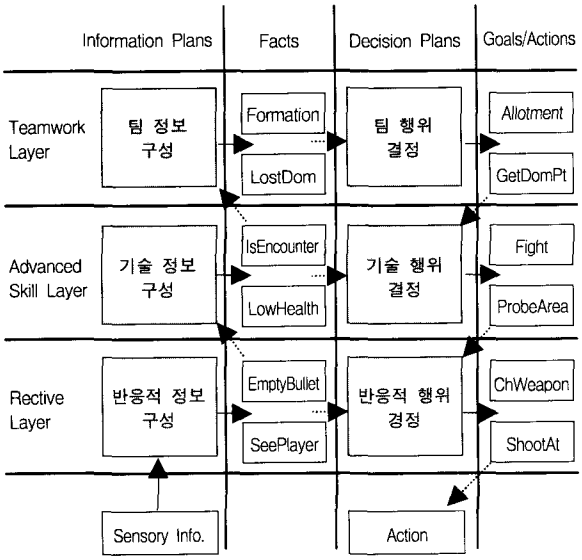


(그림 5) KGBot의 구조

KGBot는 하나의 Java 응용 프로그램으로서, 구성요소 중에서 주로 Gamebots 서버와의 통신을 담당하며 가상 게임 환경과의 인터페이스부를 형성하는 인식부와 행동부는 모두 Java로 구현된 반면, 계획을 세우고 행동을 결정하는 두뇌부분에 해당하는 의사결정부는 앞서 설명한 BDI 기반의 UM-PRS로 구현되었다. 센서정보의 도달간격과 이것을 바탕으로 의사결정을 하는 UM-PRS의 추론주기, 그리고 이미 결정된 행동에 대한 명령 메시지를 송신하는 간격 간의 차이를 극복하기 위해 인식부, 의사결정부, 행동부 각각을 별도의 스레드(thread)를 가지고 병행 수행토록 하였다. 즉 인식부는 센서정보가 도착하는 대로 자신의 큐(queue)에 담아두었다가 의사결정부인 UM-PRS의 내부 데이터베이스에 사실(fact) 형태로 직접 입력해주며, UM-PRS의 인터프리터는 자신의 실행주기에 맞추어 이 내부 데이터베이스를 참조하여 수행을 계속한다. 마찬가지로 행동부도 UM-PRS에서 내부적으로 결정된 행동들을 차례대로 자신의 큐에 내려받아 명령 메시지로 변환하여 전송한다.

한편 KGBot의 다양한 행위는 대부분 제어 엔진인 UM-PRS의 KA, 즉 계획(plan)들의 집합으로 표현되며, 이러한 KGBot의 지식베이스는 (그림 6)에서 보는 바와 같이 빠른 반응적 행위를 나타내는 반응적 계층(reactive layer), 개인 단위의 다양한 지능행위를 표현하는 기술 계층(advanced skill layer)과 팀 단위의 전략과 전술을 나타내는 팀 워크 계층(teamwork layer)으로 계층화 되어 있다. 그리고 각 계층은 하위 계층이 보유하고 있는 정보의 일부를 발췌하거나 새롭게 요약함으로써 그 계층에서 필요한 별도의 정보를 만들어 내는 계획들과 이러한 정보를 바탕으로 그 계층에 맞는 행위들을 결정하는 계획들로 구성된다. 특히 반응적 계층은 주로 위기상황에 대한 즉각적인 반응 행위(reactive behavior)나 비교적 의사결정이 간결한 기본 행위들이 포함된다. 예컨대 근접한 적의 공격을 빨리 피하거나 즉시 응사하는 행위 등이 이러한 행위들에 속한다. 이보다 상위 계층인 기술 계층에는 숨겨진 목표점들을 효과적으로 탐색하는 행위, 목적지까지 최단경로로 이동하는 행위, 적을 유인하거나 추적하는 행위, 적의 이동 경로 주변에 매복하는 행위 등 개인 단위의 다양한 지능 행위들이 포함된다. 최상위 계층인 팀워크 계층에는 주로 팀원들의 역할 배정과 조정에 관

한 결정이 표현된다. 예컨대 팀원들 중에 누구로 하여금 현재 점령중인 목표점들을 지키게 하고, 나머지 누구로 하여금 새로운 목표점들을 탐색하게 할 것인지에 관한 결정 등이 이 계층에서 표현된다.



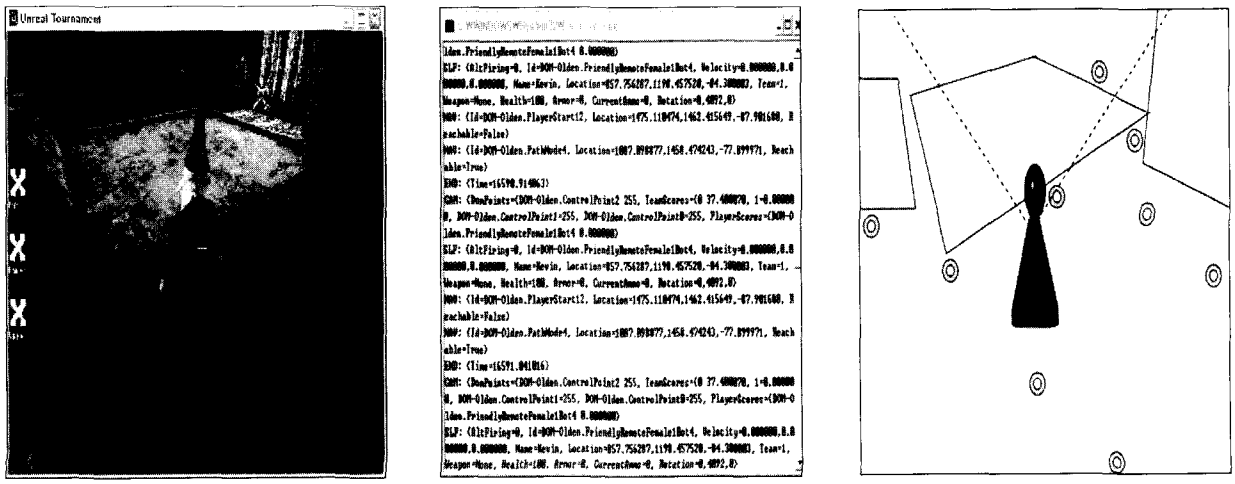
(그림 6) KGBot의 계층화된 지식베이스

5. KGBot의 경로 찾기 행위

UT 게임의 기존 내장형(built-in) NPC들은 게임 서버측에 존재하면서 속임수(cheating)를 통해 자신이 놓인 환경의 전체적인 경로 그래프, 목표점들(domination points)의 위치, 동료와 적들을 포함한 다른 캐릭터들의 현재 위치, 그리고 임의의 목적지까지 최단 이동 경로 등 휴먼 플레이어에게는 숨겨진 많은 정보를 게임 서버로부터 직접 이용하고 있다[20]. 하지만 휴먼 플레이어와 마찬가지로 원격의 한 클라이언트에서 제어하는 KGBot는 이러한 속임수없이 제한적인 실시간 센서정보에만 의존하여 스스로 이와 같은 유용한 정보를 알아내고 게임에 이용하는 지능적 행위가 요구된다.

전통적으로 온라인 컴퓨터 게임에서는 휴먼 플레이어와 같은 이미지 기반의 시각(vision) 인식 능력을 NPC들에게 기대할 수 없으므로, 그대신 NPC가 경로 찾기에 쉽게 이용할 수 있는 매우 간결한 정보로서 지도상에 분산되어 있는 이동점(waypoint)을 이용해왔다. 본 연구의 적용영역인 UT 게임과 Gamebots 시스템에서도 KGBot에 제공되는 이동경로와 관련된 공간정보는 단지 현재 시야에 들어오는 이동점들의 위치와 어떤 것들이 현재 위치에서 직접 도달 가능한 것인지에 관한 정보뿐이다. (그림 7)은 이동중인 KGBot의 한 실행장면과 그때 KGBot에 전달되는 시각정보 메시지들을 나타내고 있다. 이 시각정보 메시지에는 현재 위치에서 KGBot의 시야에 들어오는 이동점들을 포함하고 있다.

KGBot가 Domination 게임을 성공적으로 진행해가기 위해 가장 필수적이면서 기본적인 지능 행위들은 앞서 언급한대로 자신이 놓여진 미지의 3차원 환경을 효과적으로 탐색하여 숨겨진 목표점들의 위치를 빨리 파악하는 행위와 목표점들을 포함한 월드 전체의 경로 그래프를 작성하는 행위, 그리고 이러한 경로 그래프에 기초하여 실시간으로 임의의 목적지까지 최단 경로로 이동하는 행위들이다. 이중에서 첫번째 지능 행위인 목표점들을 찾아 미지의 환경을 효과적으로 탐색하는 행위는 기존의 많은 그래프 탐색 알고리즘 연구들과 몇 가지 차이점들이 있다. 첫째는 어떤 이동점을 직접 방문하기 전에는 그 이동점과 인접한 다른 이동점들의 연결상태 즉 부분 경로 그래프를 사전에 알고 있지 못하다는 점과 둘째는 비록 특정 이동점을 방문해도 이 이동점에 연결된 모든 인접 이동점들을 파악하는 일은 에이전트의 인지범위의 제한을 받는다는 점과 셋째는 찾고자 하는 목표점들의 위치좌표를 알지 못하기 때문에 각 목표점까지

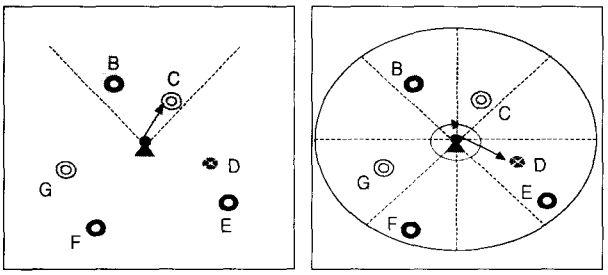


(그림 7) KGBot의 한 실행장면과 이동점들을 포함한 센서정보

의 거리에 대한 아무런 추정치나 휴우리스틱(heuristic) 정보가 없다는 점, 그리고 마지막으로 이 단계에서는 목표점의 위치를 파악하는 것이 주된 목적이므로 찾고자 하는 목표점까지 반드시 최단 경로로 도달할 필요가 없다는 점 등이다. 따라서 본 연구에서는 KGBot의 효과적인 월드 탐색을 위해서 현재 위치에서 인접한 미 방문 이동점과 방문한지 가장 오래된 이동점들을 우선적으로 탐색해가는 일종의 실시간 깊이-우선 탐색(Real-time Depth-First Search, RDFS)을 적용하였다. 하지만 여기에 부가적으로 고려해야 할 한가지 중요한 문제는 각 이동점에서 인접 이동점들을 파악할 수 있는 KGBot의 인지범위의 한계성이다. 현재 허용되는 KGBot의 시야범위는 (그림 7)과 같이 전방 약 90도이다. 따라서 월드 탐색중인 KGBot에게는 이 시야범위 안에 놓여 있는 이동점들만 인접한 노드로 인식됨으로써 탐색해가는 범위가 좁아 일정한 시간 내에 목표점을 찾는데 실패하거나 찾더라도 매우 긴 탐색시간을 소모할 가능성이 있고, 한번의 이동점 방문으로 얻는 경로정보 수집이 매우 비 효율적이므로 목표점들을 찾을때까지 월드 탐색을 통해 얻게 되는 경로 그래프 역시 매우 불충분하고 불완전할 수 있다.

이 문제를 해결하기 위해 본 연구에서는 월드 탐색과정의 매 단계마다 한 이동점을 방문한 KGBot는 현재 시야에 들어오는 인접 이동점들에 국한해서 다음 방문점을 선택하지 않고, 현재 방문중인 이동점에서 다음 이동점으로 이동하기 전에 그 지점에서 스스로 몸을 360도 회전하며 좌우 및 후방에 놓여 있는 인접 이동점들까지 모두 파악한 후 이들 중에 아직 방문한 적이 없는 미 방문점을 다음 방문점으로 선택하는 방식을 채택하였다. 본 논문에서는 편의상 전자를 Go & Go 탐색전략, 후자를 Rotate & Go 탐색전략이라 부른다. (그림 8)(a)와 (그림 8)(b)는 각각 이 두 전략을 예시하고 있다. 그림에서 이동점 D는 찾고자 하는 목표점 중의 하나를 나타내고, 현재 이동점 A를 방문중인 KGBot의 시야에 들어오는 이동점 B, C 중에 B는 이미 과거에 방문한 적이 있는 이동점을, C는 아직 방문한 적이 없는 이동점을 나타낸다. (그림 8)(a)와 같이 Go & Go 탐색전략을 적용하면, 시야에 들어오는 전방의 이동점들 중에서 아직 미 방문 이동점인 C가 다음 방문 지점으로 선택되는 반면 (그림 8)(b)와 같이 Rotate & Go 탐색전략을 적용하면 시계방향으로 회전하면서 주변에 인접한 모든 이동점 B, C, D, E, F, G를 찾아내고 이중에서 특히 미 방문 이동점 중의 하나인 목표점 D를 다음 방문 지점으로 선택함으로써 효율적인 목표점 탐색이 가능하게 된다. (그림 9)는 본 논문에서 제안하는 Rotate & Go 전략을 결합한 깊이-우선 월드 탐색 알고리즘을 간략히 표현하고 있다.

KGBot는 이와 같은 방법으로 미지의 3차원 환경을 돌아다니며 숨어있는 목표점(domination point)들을 찾아 그 위치를 파악하는 한편, 자신이 방문하는 이동점들간의 연결상태를 파악하여 나중에 임의의 목적지까지 효과적인 경로 찾기에 이용할 경로 그래프를 작성한다. 이때 주목해야 할 점은 Gamebots 시스템에서 KGBot가 실시간 경로 그래프 작성을 위해 이용할 수 있는 유일한 센서정보는 언제나 현재 위치에서 인접한 각 이동점들의 위치와 이들이 현재 위치에서 접근 가능한지 즉 단위 경로를 조사할 수 있는 정보뿐이다.



(a) Go & Go (b) Rotate & Go

(그림 8) 월드 탐색 전략들

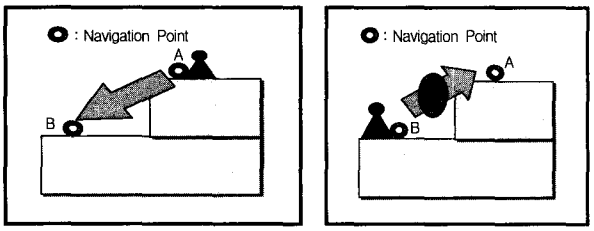
```
EXPLORE(p)

/* V : a set of visible nodes
   R : a set of reachable nodes
   P : a set of unit paths, i.e., arcs
   G : the directed path graph                    */

IF the current position is not on the node p,
      THEN move to the waypoint p

V ← { }, R ← { }, P ← { }
θ ← 0

WHILE (θ < 360), DO
     V ← VISIBLE-NODES(p)
     R ← REACHABLE-NODES(p, V)
     P ← P ∪ {(p, r) | r ∈ R}
     IF there exists a domination node d in R,
          THEN record the location of d
     Rotate right in △θ degree
     θ ← θ + △θ
END

G ← G ∪(V, P)

IF found all hidden domination nodes, THEN return G

FOR each reachable node r in R
     IF r is an unvisited node, THEN call EXPLORE(r)
END
```

(그림 9) 월드 탐색 알고리즘

일반적인 2차원 환경에서는 한 이동점 A에서 다른 이동점 B로의 접근이 가능한 경우, 그 역 방향인 B에서 A로의 접근도 가능한 경우가 대부분이다. 즉 인접한 두 이동점들

간에는 접근성(reachability) 면에서 대칭성(symmetry)을 가진다. 이러한 대칭성을 이용하면 에이전트가 실제로 두 이동점들 중 하나만 방문하여 한쪽 방향의 접근성만 확인할 수 있으면 두 이동점들간에 양방향으로 접근가능한 경로 그래프를 작성할 수 있다. 따라서 에이전트는 적은 탐색비용으로도 각 에지(edge)가 두 노드간의 양방향 접근성을 암시하는 무향 그래프(undirected graph)를 효과적으로 생성할 수 있다.

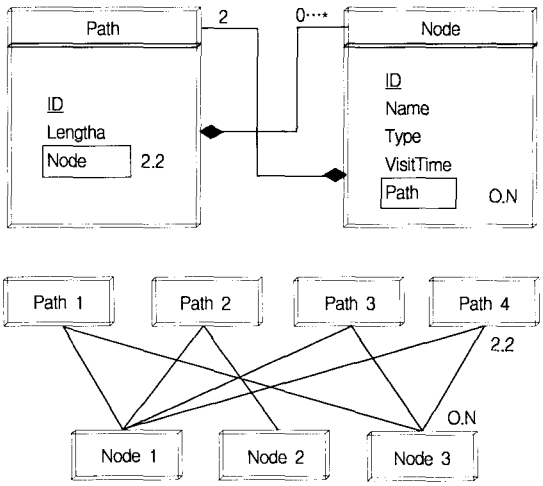
하지만 KGBot가 활동하는 UT 게임의 복잡한 3차원 환경은 이러한 두 이동점들간의 접근 대칭성 가정을 만족하지 못한다. 예컨대 (그림 10)(a) 경우처럼 높은 위치에 놓인 이동점 A에서 낮은 위치의 이동점 B로의 이동은 가능한 반면, (그림 10)(b)의 경우처럼 그 역방향인 B에서 A로의 이동은 불가능한 경우가 매우 흔하다. 따라서 탐색중인 KGBot는 현재 방문하고 있는 이동점을 중심으로 인접한 이동점들 각각으로의 접근 가능성을 조사한 다음 단방향 경로를 나타내는 에지들을 추가함으로써 하나의 유향 경로 그래프(directed path graph)를 작성한다. 그리고 각 경로의 역방향 접근성을 조사하려면 추가적인 탐색을 통해 경로의 다른쪽 이동점을 실제로 방문해야만 가능하다. 대신 월드 탐색을 위해 KGBot가 채용하고 있는 Rotate & Go 탐색전략에 따르면, 한 이동점 A에서 접근 가능한 인접 이동점들과 단위 경로들 중에서 다음 방문대상으로 선택된 이동점 B로의 단위 경로는 KGBot가 다음순간 이동점 B로 이동한 직후 다시 모든 인접 이동점들의 접근 가능성을 조사할때 그 역 방향의 경로에 대한 조사도 함께 이루어질 수 있다. 그런 면에서 Rotate & Go 탐색전략은 효율적인 경로 그래프 작성에도 큰 도움이 된다.



(a) 접근 가능 (b) 접근 불가능

(그림 10) 이동 경로의 비대칭성 문제

(그림 11)은 KGBot가 작성하는 경로 그래프를 표현하는 내부 자료구조들을 나타내고 있다. 경로 그래프는 각 이동점을 나타내는 노드(node) 객체들과 두 이동점들간의 단위 경로를 나타내는 경로(path) 객체들로 구성된다. 각 노드에는 노드 식별자, 이름, 유형, 방문시간 등이 저장되고, 각 경로에는 경로 식별자, 경로 길이, 경로의 종단노드 등이 저장된다. 경로 객체들은 해당 경로의 시작노드가 일치하는 것끼리 모아 시작노드 객체에 함께 보관된다. 그리고 각 노드 객체는 대응되는 이동점들이 KGBot의 시야에 들어올 때 생성되며, KGBot가 현재 위치하고 있는 노드에서 접근 가능하다는 판단이 이루질 때마다 해당 단위 경로 객체가 생성된다.



(그림 11) 경로 그래프를 구성하는 내부자료구조

미지의 환경에 대한 탐색과정을 통해 일단 숨겨진 목표점들과 그들 사이를 잇는 부분 경로 그래프를 생성하고 나면, KGBot는 찾아진 목표점들 사이를 오가며 목표점을 새로이 점령하거나 적이 점령중인 목표점을 탈환하거나 자신이 현재 점령중인 목표점들을 적으로부터 지켜냄으로써 자신의 게임 점수(game score)를 높이려고 노력한다. 이 단계에서는 이미 놓여진 위치와 현재 소유주를 알고 있는 목표점들을 대상으로 어느 목표점을 우선적으로 공격하거나 사수할 것이냐를 결정하는 전략적 판단과 전술적 판단이 승리의 가장 큰 관건이 된다. 하지만 이 단계에서도 목표점을 포함한 특정 목적지까지 얼마나 빠르게 최단 경로로 이동할 수 있느냐 하는 경로 찾기 문제는 여전히 중요한 문제가 된다. 하지만 이 단계에서의 경로 찾기는 목적지의 위치좌표가 주어진다는 점, 현재 위치와 목적지를 포함한 부분 경로 그래프(partial path graph)를 이미 가지고 있다는 점, 그리고 목적지까지 최단 경로(shortest path)로 도달해야 한다는 점에서 앞서 설명한 미지의 월드 탐색의 경우와는 다르다.

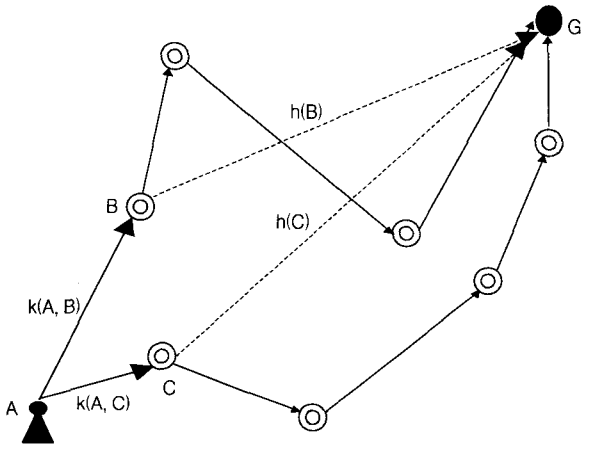
이미 경로 그래프를 알고 있을때 목적지까지 최단경로를 찾는 그래프 탐색 알고리즘에 대한 연구는 인공지능 분야에서 오랫동안 연구되어 왔으며 이미 A*와 IDA* 등 매우 효율적인 휴우리스틱 탐색 알고리즘들이 개발되어 있다. 하지만 이러한 A*와 IDA* 알고리즘 등은 일종의 오프라인(off-line) 탐색 알고리즘들로서, 에이전트가 계획된 경로의 첫번째 단계를 실행하기 전에 미리 목적지까지 하나의 완전한 경로를 계획하는 방식이다. 이와는 달리 RTA*(Real-Time A*)와 LRTA*(Learning Real-Time A*) 알고리즘과 같은 실시간 온라인(on-line) 탐색 알고리즘들은 목표노드에 도달할

때까지 매 단계마다 이동할 다음 노드를 선택하는 이동계획과 계획된 이동에 대한 실제 실행이 번갈아 수행되는 방식이다[17, 23]. 이러한 실시간 온라인 알고리즘들은 언제나 최적의 해 경로(optimal solution path)를 보장할 수는 없지만, 전통적인 오프라인 탐색 알고리즘들에 비해 차적의 해 경로(sub-optimal solution path)를 매우 빨리 찾을 수 있다고 알려져 있다. 본 연구에서 목적지까지 가능한 빨리 도달해야 하는 KGBot의 경우에도 실제 이동에 소요되는 시간뿐만 아니라 이동계획에 소요되는 시간까지 실시간 제약성을 가지므로, 오프라인 탐색 방법인 A*와 IDA* 알고리즘 대신 실시간 온라인 탐색 방법인 LRTA* 알고리즘을 적용하였다.

(그림 13)은 월드 탐색을 통해 작성된 부분 경로 그래프를 이용하여 현재 위치의 노드에서 출발하여 목표노드에 도달할 때까지 반복되는 KGBot의 실시간 최단 경로찾기 알고리즘인 LRTA* 알고리즘을 간략히 표현하고 있다. 이 알고리즘에서 경로 그래프상의 각 노드 x의 초기 휴우리스틱 함수값

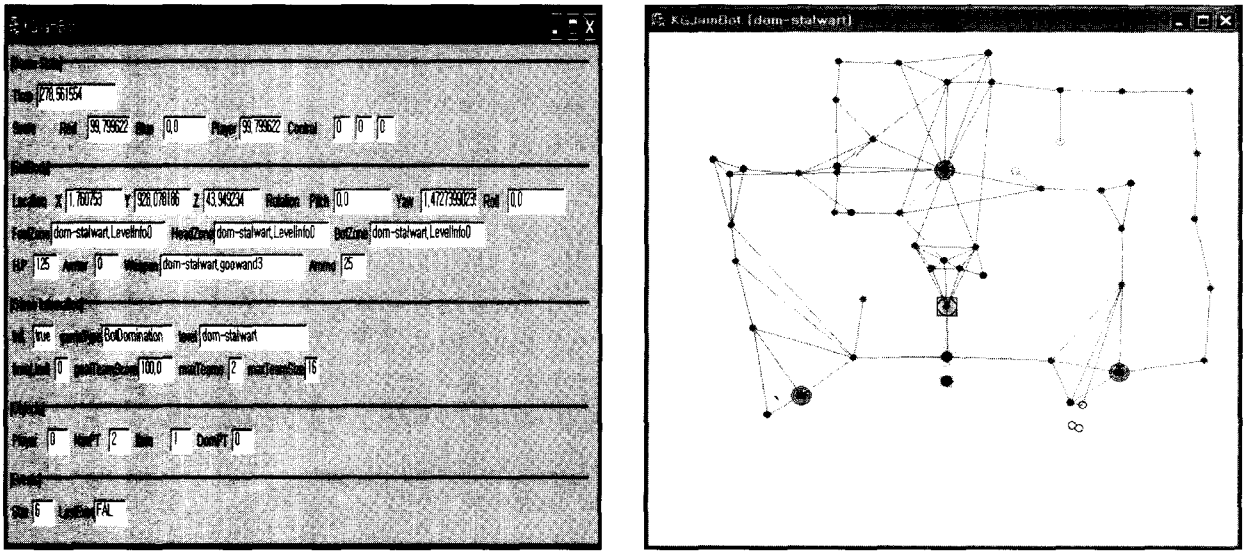
```
FIND-PATH-LRTA(x, goal, G)

/*   x : the current node,
     goal : the goal node
     G : the directed path graph
     k(x, x') : the edge cost from x to x' */

 IF the current node x is the goal node, THEN return with
 success

/* Lookahead */
FOR each neighbor x' of x
     Calculate f(x') = k(x, x') + h(x')
END

/* Consistency maintenance */
Update the value of h(x) to the minimum f(x') value as follows
       h(x) ← min { f(x') } for all neighbors of x

/* Action selection */
Move to neighbor x' that has the minimum f(x') value
x ← x'
Call FIND-PATH-LRTA (x, goal, G)
```

(그림 13) 실시간 경로 찾기 알고리즘

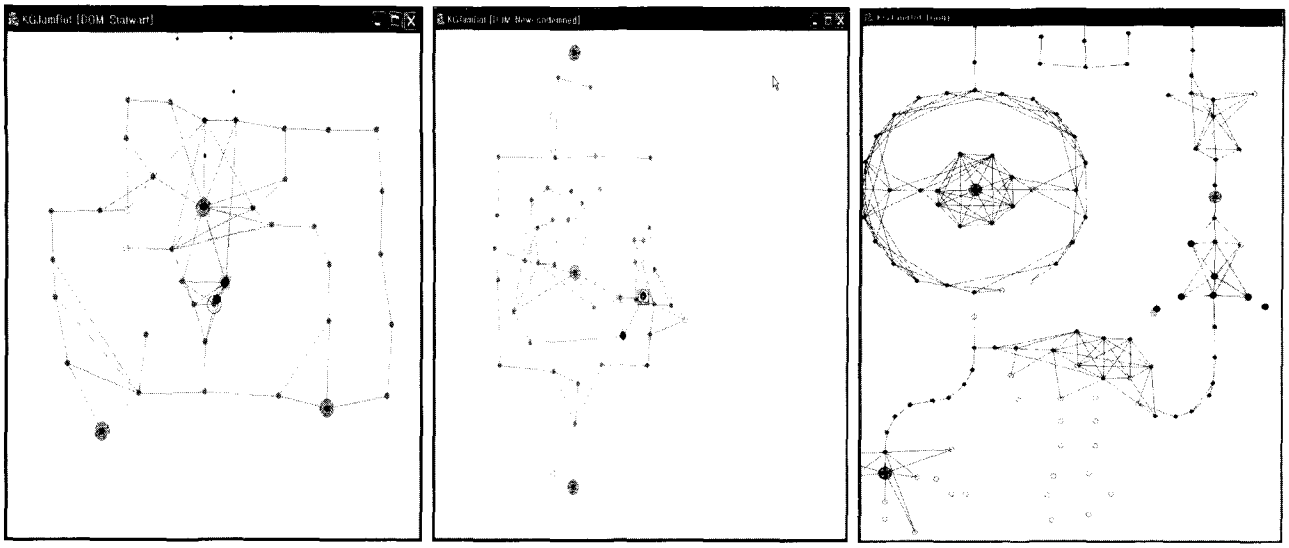


(그림 12) 실시간 경로 찾기의 예

h(x)은 그 이동점에서 목적지까지의 직선거리를 의미하는 유클리드 거리(Euclidian distance)로 한다. 그러면 이 함수값 h(x)는 목적지까지 실제 거리보다 작은 양의 값이 되므로, 이 알고리즘의 완전성(completeness)를 위한 충분조건을 만족한다. 따라서 이 알고리즘에 따라 이동하는 에이전트는 반드시 목적지에 도달할 수 있다. (그림 12)는 LRTA* 알고리즘에 따라 KGBot가 수행하는 실시간 경로 찾기 행위를 예시하고 있다. 그림에서 현재 이동점 A를 방문중인 KGBot는 접근 가능한 인접 이동점 B와 C중에 하나를 다음 방문지점으로 선택하여 이동하여야 한다. 이때 이웃 이동점 B의 함수값 f(B)는 현재 이동점 A에서 B로의 단위 경



(그림 14) KGBot의 행위 분석 도구



(a) Dom-Stalwart (b) Dom-Condomned (c) Dom-Leadworks

(그림 15) 실험에 사용된 지도들

로의 길이인 k(A, B)와 이동점 B에서 목표점 G까지 직선거리인 h(B)를 더함으로써 계산된다. 또 다른 이웃 이동점 C의 함수값 f(C)도 같은 방법으로 계산되며, 이렇게 계산된 함수값 f(B)와 f(C)중 최소값을 가지는 이웃 이동점으로 KGBot는 이동하게 된다. KGBot는 이와 같은 과정을 반복함으로써 최적 경로에 근사한 방법으로 보다 빨리 목표점 G에 도달한다.

한편 본 연구에서는 동작중인 KGBot의 내부에서 실시간으로 관리되는 자신의 신체정보(위치 값, 위치하고 있는 영역, 체력), 무기정보, 경로 그래프 정보, 게임상태정보(게임서버 시간, 팀 별 점수) 등을 보여주면서, 동시에 이러한 정보에 기초한 KGBot의 월드탐색 및 이동행위를 시각화하여 보여주는 (그림 14)와 같은 행위 분석 도구를 함께 개발하고 이용하였다. 이와 같은 행위 분석 도구를 이용함으로써 에이전트 설계자나 이용자는 KGBot가 수행되고 있는 외부환경의 상태 정보와 KGBot의 내부 상태 정보를 비교하면서, KGBot가 현재 목표에 맞는 행위를 정상적으로 수행하고 있는지를 분석하고 평가 할 수 있게 해준다.

6. 실 험

본 논문에서는 KGBot의 월드 탐색 전략인 Rotate & Go와 제어엔진인 UM-PRS의 효율성을 검증하기 위한 실험을 전개하였다. 먼저 월드 탐색 전략의 효율성 분석을 위해서는 본 논문에서 제안한 Rotate & Go 전략과 단순한 Go & Go 전략간의 비교 실험을 전개하였고, 제어엔진의 효율성 분석을 위해서는 BDI 제어엔진인 UM-PRS와 순수 반응형(pure reactive) 제어엔진간의 비교 실험을 전개하였다. 그리고 이 실험들에는 각각 (그림 15)(a), (그림 15)(b), (그림 15)(c)에 나타나 있는 Dom-Stalwart, Dom-Condomned, Dom-Leadworks 등 복잡도가 서로 다른 세 개의 3차원 지도들이 사용되었다. (그림 15)은 실험에 사용된 세 개의 지도상의 경로를 KGBot가 탐색하는 모습을 행위 분석 도구를 이용하여 실시간으로 보여준 것이다. 실험에서 사용된 세 개의 지도 중 특히 (그림 15)(a)의 Dom-Stalwart는 이동점의 수가 적고 지도가 매우 단순하여 어떤 유형의 에이전트도 목표점과 연결 경로를 비교적 쉽게 찾을 수 있는 지도이다. 반면에 (그림 15)(b)의 Dom-Condomned와 (그림 15)(c)의 Dom-Leadworks는 계단 위와 아래의 2층으로된 복잡한 공간구조에 많은 수의 이동점들이 분산되어 있고 곳곳에 용암지대와 같은 위험 지역들이 놓여 있는 복잡한 지도들이다.

월드 탐색 전략 비교 실험에서는 각각 다른 탐색전략을 적용하였을때 에이전트가 주어진 지도를 얼마나 효율적으로 탐색하였는지를 알아보기 위해, 지도상에 숨어있는 모든 목표점(domination point)들과 그들을 연결하는 경로들의 그래프를 작성하는데 소모하는 총 시간을 측정하여 보았다. 동일 전략을 각 지도별로 10회씩 반복 실험하여 탐색에 걸린 최대 시간, 최소 시간, 그리고 평균 시간을 비교하여 보았다. 그리고 매 실험마다 월드 탐색 제한 시간을 600초로 설정하였는데, 600초가 지나도 목표점들을 다 찾지 못한 경우에는 실시간 액션 게임의 특성상 사실상 이미 승패가 판가름난 상태여서 더이상 탐색을 계속하는 것이 무의미하다고 판단되어 실험을 멈추었다.

월드 탐색 전략간의 비교 실험 결과는 <표 1>과 같다. Go & Go 전략은 가장 단순한 지도인 Dom-Stalwart를 제외하고 나머지 모든 경우에 탐색 제한 시간을 초과할 때까

지 목표점들을 찾아내지 못한 반면, Rotate & Go 전략은 실험에 사용된 모든 지도에 대해 탐색 제한 시간 내에 모든 목표점들을 탐색하는데 성공하였다. 또한 Rotate & Go 전략의 경우, 지도의 복잡도가 증가할수록 탐색에 소요된 시간은 증가하였지만 전체적으로 매우 효율적인 탐색으로 인해 지도의 복잡도에 비해 총 탐색시간의 증가는 완만하였다. Dom-Stalwart와 같이 장애물이 없고 지도의 구성이 매우 간단한 경우에는 Go & Go 전략이 오히려 Rotate & Go 전략보다 빨리 목표점들을 탐색할 수 있었지만, Dom-Condomned와 Dom-Leadworks처럼 지도가 두 개의 층으로 구성되거나 경로의 굴곡이 심한 경우에는 Rotate & Go 전략이 매우 효과적임을 알 수 있었다.

〈표 1〉 월드탐색 실험결과

(단위 : 초)

지 도	이동점 개수	탐색전략	최대 시간	최소 시간	평균 시간
Dom-Stalwart	39	Roatate & Go	123	76	97
		Go & Go	75	22	46
Dom-Condomned	51	Roatate & Go	214	137	193
		Go & Go	> 600	> 600	> 600
Dom-Leadworks	149	Roatate & Go	443	318	387
		Go & Go	> 600	> 600	> 600

한편, 제어엔진의 효율성 분석을 위한 실험에서는 BDI 제어엔진인 UM-PRS와 순수 반응형(pure reactive) 제어엔진 간의 비교 실험을 전개하였다. 대표적인 BDI 제어엔진인 UM-PRS의 특징중의 하나는 목표지향적(goal-oriented) 추론 능력과 문맥 의존적(context-dependant) 행위이다. 따라서 일반적으로 환경 변화가 매우 심한 경우가 아니라면 UM-PRS 기반의 에이전트는 자신의 목표를 향해 일관성 있는 행위를 보여줄 수 있다. 이에 반해 순수 반응형 제어엔진은 추구하는 명시적인 목표가 없고, 과거상태에 대한 기억이나 미래상태에 대한 예측도 없으며 오직 환경의 현재 상태에 따라 행동을 결정한다[3]. 따라서 이러한 제어엔진을 채용한 순수 반응형 에이전트는 환경변화가 심한 경우 매우 빠른 반응성을 보일 수 있어 효과적이다. 하지만 일반적으로 순수 반응형 에이전트는 약간의 환경변화에도 목표 추구를 위한 일관성 있는 행위를 지속하기 어렵고, 과거 상태들을 기억하지 못함으로써 복잡한 상황에 맞는 정교한 의사결정을 하기 어렵다는 단점이 있다.

본 연구에서는 복잡한 3차원 가상공간에서의 일대일 Domination 게임을 위해 UM-PRS 제어엔진과 순수 반응형 제어엔진 중 어느 쪽이 더 효과적인지 비교하여 보았다. 이를 위해 게임서버로부터 전달되는 실시간 동기 및 비동기 센서정보에 따라 즉각적으로 행동을 결정하는 순수 반응형 에이전트를 별도로 구현한 다음, UM-PRS 제어엔진을 채용한 KGBot와 동일한 지도안에서 경쟁하며 일대일 Domination 게임을 하도록 시행하여 보았다. 게임은 어느 한쪽이 제한 점수인 100점을 먼저 취득하여 승리할때까지 계속되었고, 따라서 별도의 게임 제한시간은 설정하지 않았다. 제어엔진 간의 비교 실험에서도 앞서 설명한 세 가지 지도들을 사용하였으며, 각 지도별로 10회씩 실험하여 각 에이전트가 획득한 최대 점수, 최소 점수, 평균 점수들을 비교하였다.

제어엔진 간의 비교 실험 결과는 <표 2>와 같다. 표에서 알 수 있듯이 본 연구에서 시행한 모든 실험 게임에서 UM-PRS를 제어엔진으로 채용한 KGBot가 먼저 제한 점수 100점을 취득함으로써 순수 반응형 에이전트를 이겼다. 또한 지도의 복잡도가 커질수록 두 엔진간의 점수차가 더욱 벌어진 것을 알 수 있다. 물론 KGBot와 순수 반응형 에이전트가 취득한 게임 점수만을 단순 비교하여 두 제어엔진의 효율성을 비교하는 것은 분명 많은 문제점을 내포하고 있다. 게임에서 취득하는 점수는 제어엔진의 차이 이외에도 보다 다양한 요인들에 의해 영향을 받을 수 있기 때문이다. 하지만 그럼에도 불구하고 일대일 Domination 게임환경 안에서 두 제어엔진의 효율성을 실증적으로 비교할 수 있는 현실적인 방법은 실제로 그 게임환경 안에서 직접 두 에이전트를 서로 경쟁시켜 보는 것이다. 따라서 이러한 의미에서 본 연구의 실험 결과는 순수 반응형 제어엔진에 비해 UM-PRS 제어엔진의 우수성을 보여주었다. 하지만 이러한 결과는 본 연구에서 가정하는 일대일 Domination 게임타입의 특성에도 그 원인이 있는 것으로 판단된다. 보다 많은 에이전트들이 상호 동작하는 팀별 Domination 게임이나 팀별 CaptureTheFlag 게임에서는 본 연구의 일대일 Domination 게임과는 달리 게임 진행이 보다 빠르게 이루어지고 따라서 상태 변화도 심한 환경적 특성을 가진다. 이러한 다중 에이전트 환경에 비해 일대일 Domination 게임 환경은 비교적 게임 상태의 변화가 느릴뿐 아니라 에이전트 자신이 이러한 게임상태 변화를 주도할 수 있다. 따라서 이러한 환경에서는 단순히 환경변화에 반응하는 순수 반응형 제어엔진에 비해 목표-지향적인 행위를 실현하는 UM-PRS가 더 효과적인 것으로 판단된다.

〈표 2〉 제어엔진 실험결과

지 도	이동점 개 수	제 어 엔 진	최 대 점 수	최 소 점 수	평 균 점 수
Dom-Stalwart	39	UM-PRS	100	100	100
		Pure Reactive	51	19	26
Dom-Condomned	51	UM-PRS	100	100	100
		Pure Reactive	19	7	12
Dom-Leadworks	149	UM-PRS	100	100	100
		Pure Reactive	22	10	12

7. 결 론

본 논문에서는 3차원 일인칭 액션 게임을 지능형 에이전트 연구용 테스트베드로 확장한 Unreal Tournament 게임과 Gamebots 시스템을 소개하였다. 그리고 이들이 제공하는 3차원 가상환경 안에서 적과 일대일 Domination 게임을 효과적으로 진행해가는 지능형 NPC인 KGBot를 설계하고 구현하였다. KGBot는 범용의 BDI 에이전트 구조인 UM-RS를 제어엔진으로 채용하고 있다. UM-PRS는 목표-지향성과 더불어 환경변화에 대한 높은 반응성도 가지고 있으며, 높은 수준에서 에이전트의 행위를 기술할 수 있어 에이전트의 개발과 수정이 용이하다는 장점이 있다. 또한 본 논문에서는 지도상에 흩어져 있는 이동점들을 따라 돌아다니며 숨어있는 목표점들의 위치를 파악하는 행위, 이와 같은 탐색과정을 통해 자신이 놓여진 월드의 경로 그래프를 효과적으로 작성하는 행위, 끝으로 이렇게 작성된 경로 그래프에 기초하여 목적지까지 최적 경로를 실시간적으로 찾아가는 행위 등 KGBot의 지능행위에 대해 자세히 설명하였다. 그리고 끝으로 서로 다른 3차원 지도상의 실험을 통해 이러한 KGBot의 제어엔진과 경로 찾기 행위의 우수성을 입증하여 보았다. 계획하고 있는 향후 연구로는 현재까지 구현된 기초적인 지능 행위를 바탕으로 팀별 Domination 게임이나 팀별 CTF 게임과 같은 다중 에이전트 환경에서도 효과적으로 동작하는 KGBot 팀 에이전트를 개발하는 것이다. 이를 위해서는 먼저 팀원들간의 메시지 교환을 위한 통신 프로토콜과 팀 단위 전술행위를 구현하기 위한 효과적인 조정 프로토콜에 관한 연구가 필요할 것으로 생각된다.

참 고 문 헌

[1] AAAI : Papers from the AAAI 2000 Spring Symposium on Artificial Intelligence and Interactive Entertainment, Technical Report SS-00-02, AAAI Press, 2000.

[2] A. Stentz, "Optimal and Efficient Path Planning for Partially Known environments," Proceedings of ICRA-94, pp. 3310-3317, 1994.

[3] D. Shapiro, "Controlling Gaming Agents via Reactive Programs," Proceedings of the AAAI Spring Symposium on AI and Computer Games, pp.73-76, 1999.

[4] J. Gerstmann, Unreal Tournament : Action Game of the Year, GameSpot, 1999.

[5] L. Jaeho, M. J. Huber, E. H. Durfee and P. G. Kenny, "UM-PRS : An Implementation of The Procedural Reasoning System for Multirobot Applications," Proceedings of CIRFFSS-94, pp.842-849, 1994.

[6] J. E. Laird, M. Lent, "Human-level AI's Killer Application : Interactive Computer Games," Proceedings of AAAI-2000, August, 2000.

[7] J. E. Laird, A. Newell and P. S. Rosenbloom, "Soar : An Architecture for General Intelligence," Artificial Intelligence, Vol.33, No.3, pp.1-64, 1987.

[8] J. Funge, AI for Games and Animation : A Cognitive Modeling Approach, A. K. Peters, 1999.

[9] K. Knight, "Are Many Reactive Agents Better Than a Few Deliberative Ones," Proceedings of IJCAI-93, pp.432-437, 1993.

[10] K. Perlin and A. Goldberg, "IMPROV : A System for Scripting Interactive Actors in Virtual Worlds," Proceedings of SIGGRAPH-96, pp.205-216, 1996.

[11] L. Gasser, "MAS Infrastructure Definitions, Needs, and Prospects," Proceedings of the Workshop on Scalable MAS Infrastructure, Barcelona, Spain, 2000.

[12] M. DeLoura, Game Programming Gems, Charles River Media, 2000.

[13] M. DeLoura, Game Programming Gems 2, Charles River Media, 2001.

[14] M. Freed et al., "Towards More Human-Like Computer Opponents," AAAI Spring Symposium on AI and Interactive Entertainment, pp.22-26, 2000.

[15] R. B. Calder, J. E. et al., "ModSAF Behavior Simulation and Control," Proceedings of the 2nd Conference on Computer Generated Forces and Behavioral Representation, STRICOM-DMSO, 1993.

[16] R. Adobbati et al., "Gamebots : A 3D Virtual World Test-Bed For Multi-Agent Research," Proceedings of Agents-01, May, 2001.

[17] R. E. Korf, "Real-time Heuristic Search," Artificial Intelligence, Vol.42, No.3, pp.189-211, 1990.

[18] R. Stern, "Optimal Path Search in Unknown Physical Environments," MSc thesis, CS Dept., Bar-Ilan University, Israel. 2001.

[19] S. Hanks, M. E., Pollack and P. Cohen, "Benchmarks, Test Beds, Controlled Experimentation, and the Design of Agent Architectures," AI Magazine, Vol.14, pp.17-42, 1993.

[20] S. Rabin, AI Game Programming Wisdom, Charles River Media, 2002.

[21] S. Woodcock, "Game AI : The State of the Industry 2002," Game Developer, Vol.9, No.7, 2002.

[22] S. Woodcock, "Game AI : The State of the Industry 2001," Game Developer, Vol.8, No.8, 2001.

[23] Y. Kitamura, K. Teranishi, and S. Tatsumi, "Organizational Strategies for Multiagent Real-Time Search," Proceedings of ICMAS-96, pp.150-156, 1996.

김 인 철

e-mail : kic@kyonggi.ac.kr

1985년 서울대학교 수학과(학사)

1987년 서울대학교 계산통계학과 (이학석사)

1995년 서울대학교 계산통계학과 (이학박사)

1989년~1995년 경남대학교 전산통계학과 전임강사, 조교수

1996년~현재 경기대학교 정보과학부 부교수

관심분야 : 지능형 에이전트, 기계학습, 데이터마이닝, 게임AI

이 재 호

e-mail : jaeho@ece.uos.ac.kr

1985년 서울대학교 계산통계학과(학사)

1987년 서울대학교 계산통계학과 (이학석사)

1997년 미시간대학교 전산공학(공학박사)

1996년~1998년 ORINCON(미국 샌디에고 소재) 수석엔지니어

1998년~현재 서울시립대학교 전자전기컴퓨터공학부 전임강사 조교수

관심분야 : 에이전트 시스템, 온라인 게임, 시맨틱 웹 응용