동적인 환경을 위한 다중 에이전트 구조

원 용 대, 이 수 원
숭실대학교 컴퓨터학부

A Multi-Agent Architecture for Dynamic Environments

Yongdae Won, Soowon Lee
School of Computing, Soongsil University

요 약

가상환경은 3D 컴퓨터 시뮬레이션을 통해 실제 작업환경에서 작업하는 것과 같은 몰입감을 사용자에게 줄 수 있지만, 동적인 가상환경은 발생하는 문제들이 복잡하기 때문에 하나의 단일 에이전트로 주어진 문제를 해결하기에는 한계가 있다. 따라서 복잡한 문제들을 작은 문제들로 모듈화 시켜서 해결해야 하는데, 이에 적합한 시스템이 다중 에이전트 시스템이다. 다중 에이전트 시스템은 여러 에이전트들이 협동하여 문제를 해결해야하기 때문에 에이전트들과의 통신 문제와 정보의 공유, 그리고 동적으로 변화되는 가상환경과의 효율적인 상호작용 방법 등이 중요한 문제가 된다. 본 논문에서는 Soar라는 인공지능 아키텍처를 이용하여 이러한 다중 에이전트 시스템을 모형화하는 방법에 대해 기술하고 가상현실 시스템과 동적으로 상호작용 하면서 추론/계획하고 행위를 생성하는 방법을 제시한다.

1. 서 론

최근 에이전트 기반 소프트웨어 기술은 소프트웨어 시스템을 설계하고 구현하는데 새로운 패러다임으로 많은 흥미를 유발하고 있다. 특히 가상환경이나 인터넷과 같은 동적이고 분산된 환경에서 동작하는 소프트웨어 개발 분야에서 이러한 기술은 더욱 더 매혹적이다. 가상환경은 3D 컴퓨터 시뮬레이션을 통해 실제 작업환경에서 작업하는 것처럼 사용자의 상호작용할 수 있기 때문에 사용자가 그 안으로 몰입되게 만들지만, 이러한 환경에서 발생하는 문제들은 복잡하기 때문에 하나의 에이전트가 다루기에는 한계가 있다. 이러한 문제를 해결하기 위해서는 복잡한 하나의 문제들을 여러 개로 분리하여 해결할 수 있는 모듈화가 필요한데, 다중 에이전트 구조는 이러한 모듈화에 이상적인 접근방법이다[9]

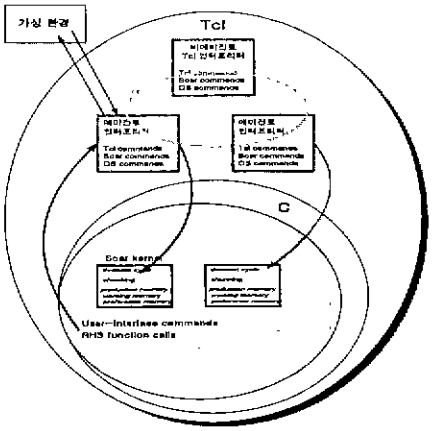
다중 에이전트 시스템은 에이전트 독자적으로 문제를 해결하던 단일 에이전트 시스템에 사회적인 요소를 추가하여 두 개 이상의 에이전트들이 각각 자신의 목표와 행위들을 가지고 서로 상호작용하면서 문제를 해결하는 시스템이다. 일반적으로 다중 에이전트 연구는 에이전트들이 다른 에이전트와의 통신에서 사용하게되는 에이전트 통신 언어(ACL)와 다른 에이전트와의 메세지 교환 방식, 그리고 에이전트의 수행 능력을 결정하는 에이전트 내부 아키텍처에 대한 연구로 나눌 수 있다. 에이전트 통신 언어는 각각의 에이전트 내부에서의 메세지 전달과 외부 에이전트들과의 메세지 전달에 사용되는 언어로 KIF, KQML 등이 많이 사용된다. 에이전트의 메세지 교환 방식은 에이전트들간에 직접 통신이 이루어지는 direct communication과 중재 프로그램을 통해서 통신이 이루어지는 assisted coordination의 두 가지 방법이 사용되고 있다. 에이전트 아키텍처로는 크게 동질적(homogeneous) 에이전트 시스템과 이질적(heterogeneous) 에이전트 시스템으로 구분된다[13]. 동질적 다중 에이전트 시스템은 각 에이전트들의 내부 구조와 행위 선택 절차가 같기 때문에 동일한 작업들을 분산 환경에서 수행하는 시스템이며, 이와는 달리 이질적 다중 에이전트 시스템은 각 에이전트가 서로 다른 내부 구조와 행위 선택 절차를 가지고 있으며, 서로 다른 목표를 성취하기 위해 각 에이전트의 목표에 맞는 행위들을 수행하는 시스템을 말한다. 전자로는 차량의 움직임을 지시하는 DVMT, 항공기 운항 관리를 위한 OASIS 등이 있으며, 후자로는 작업 프로세스 제어를 위한 ARCHON, 지능적으로 방을 관리해주는 SodaBot 등이 있다[9,10]

본 연구는 실세계를 그대로 모형화한 것과 같은 가상 물리실험 환경에서 동질적 에이전트들을 이용해 사용자와 상호작용 하면서 사용자가 보다 지능적인 느낌을 받을 수 있도록 현실감을 증대시킬 수 있는 다중 에이전트 시스템 설계에 관한 연구이다. 이 다중 에이전트 시스템은 기존의 Soar 기반의 하이브리드 아키텍처로 구현된 동질적 구조의 에이전트들이 동적 환경에서 효과적인 계획과 추론을 수행하고, 타에이전트와의 협동을 통하여 공동의 목표를 성취할 수 있도록 확장한 것이다[14]. 기존에는 각각의 하이브리드 에이전트가 분리된 각자의 가상공간 안에서만 행위를 수행했지만, 다중 에이전트 시스템에서는 통합된 하나의 가상 공간 안에서 서로 상호작용 하면서 모든 행위를 수행해야 한다. 따라서 각 에이전트들은 가상 물리 실험실에서 발생하는 변화들을 분석하여 자신의 목표를 수정하고 목표 성취를 위한 계획을 생성하여 다음 행위를 결정해야 한다. 이러한 시스템 설계를 위해서 2절에서는 Soar 아키텍처를 사용한 다중 에이전트 아키텍처에 대해 살펴보고, 3절에서는 동적인 가상환경에 효율적으로 대응하기 위한 에이전트의 행위 계획 방법에 대해 설명한다. 4절에서는 시스템 구현에 대해 설명하고 5절에서는 결론 및 향후 연구 과제를 제시한다.

2. 다중 에이전트 아키텍처

본 연구에서 제안하는 가상 환경을 위한 다중 에이전트 시스템은 인식 모듈과 추론/계획 모듈로 구성되어 있다. 인식 모듈은 동작 생성기로부터 가상환경의 상태(state) 변화와 가상 행위자나 인간 참여자의 행위에 의한 이벤트들에 대한 메세지를 받아서 이를 분석하여 추론/계획 모듈이 사용할 수 있는 형태로 정보를 변형시켜 전달해 준다. 추론/계획 모듈은 인식 모듈로부터 받은 가상환경에 대한 정보들에서 가상 행위자가 수행해야할 적절한 목표를 선택하고, 그 목표들을 성취하기 위한 계획을 생성하고 필요한 행위들을 선택하여 동작 생성기로 보낸다. 이러한 작업들을 효율적으로 수행하기 위해서 인식 모듈은 Tcl/Tk와 C로 구현되고, 추론/계획 모듈은 Soar를 사용한 생성규칙(production rule)들로 이루어진다

Soar를 사용한 다중 에이전트 아키텍처에서는 각각의 에이전트들이 자신만의 Soar 커널을 하나씩 소유하고, Tcl 인터프리터를 통해 외부와의 통신을 가능하게 한다(<그림 1>)[2]. 이 다중 에이전트 시스템은 Soar 생성 규칙으로 만들어진 여러 개의 에이전트들과 Tcl로 구현된 하나의 비에이전트로 구성되어 있다. 각 에이전트는 각각의 단기적 기억장소(Working Memory : WM)와 선호 기억장소(Preference Memory : PM)를 가지고 실행되며, 다른 에이전트와의 통신이나 가상환경과의 메세지 교환은 에이전트 인터프리터를 통해 이루어지게 된다. 에이전트는 에이전트 인터프리터를 통해 가상환경과 직접 정보를 주고받기 때문에 가상환경에 대한 정보는 모든 WM에 공통적으로 가지고 있게 되지만, 각각의 에이전트들이 현재 어떤 목표를 성취하기 위해 어떤 계획들을 수행하고 있는지에 대한 정보들은



<그림 1> Tcl을 사용한 Soar의 다중 에이전트 환경

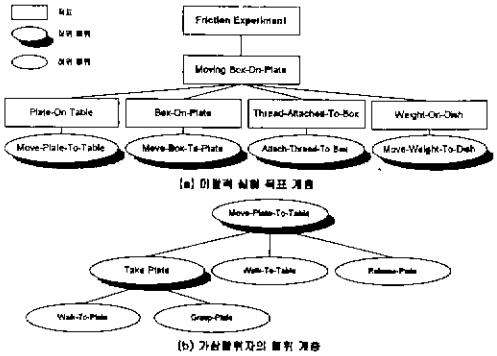
자신만이 가지고 있게 된다 Tcl로 구현된 비에이전트는 다른 에이전트들처럼 WM, PM을 기지고 있지는 않지만 각각의 에이전트들을 생성하고, 인식 모듈과 그 에이전트들이 순차적으로 실행될 수 있도록 흐름을 제어하여 에이전트와 다른 시뮬레이터 프로그램을 함께 사용할 수 있도록 한다 즉, 비에이전트는 GUI를 사용해 에이전트들과 사용자가 편리하게 상호작용할 수 있도록 제어하는 것이다

다중 에이전트 시스템에서의 에이전트들은 각기 다른 목표들을 실험하고 동적으로 변화되는 가상환경의 정보들을 분석하여 현재 목표나 계획을 수정하면서 필요한 행위들을 생성한다. 생성된 행위들은 Soar의 RHS 함수를 통해 에이전트 인터프리터로 전달되고, 에이전트 인터프리터는 이 명령을 받아서 소켓을 통해 가상환경의 동작생성기로 보낸다.

3. 다중 에이전트의 추론

3.1 행위 표현과 목표 계층

가상 물리실험에서 다중 에이전트는 사용자의 실험 진행을 돕거나 사용자와는 다른 형태의 실험을 동시에 수행해서 그 결과를 비교하여 사용자의 실험 이해를 돕는 역할을 한다. 이러한 과정에서의 추론은 실험 목표계층을 사용해서 실험 종류에 따라 자신이 선택해야 하는 도구나 실험에서의 역할 등을 결정하거나 실험 도중에 예기치 못한 이벤트가 발생했을 경우 이를 분석하여 적절한 목표를 생성하는 것 등이 있다 마찰력 실험에서는 사용자와 실험 결과를 비교할 수 있도록 실험에 사용하는 판이나 상자의 무게 등을 다르게 설정하는 것이 필요하며, 탄성 충돌 실험에서는 사용자의 실험 진행을 도와 하나의 목표를 함께 성취해야 하기 때문에 사용자의 선택에 맞춰 자신의 역할을 결정하는 등의 추론이 일어나는 것이다. 예를 들면, 가상 물리실험의 두 에이전트는 사용자가 마찰력 실험을 선택했을 때, 사용자의 다른 판이나 상자를 선택해서 나중에 실험 결과를 비교할 수 있도록 해야하기 때문에 사용자가 어떤 판과 상자를 선택하는지를 지켜보면서 자신들의 판과 상자를 결정해야 한다 다른 실험에서도 각 실험의 특성에 맞게 각자의 실험 초기화나 역할을 결정하게 된다 <그림 3>의 (a)는 에이전트의 이러한 추론을 돕는 마찰력 실험 목표계층의 일부를 나타낸 것이며, (b)는 에이전트의 행위 계층을 나타낸 것이다 에이전트의 모든 행위들은 <그림 3>의 (b)에서처럼 계층적 구조를 갖는 상위(high-level) 행위와 하위(low-level) 행위의 두 종류로 나뉘어 표현된다. 상위 행위들은 여러 개의 하위 행위들이 조합되어 이루어지는 추상적인 개념의 행위들을 말하며, 하위 행위들은 실제로 에이전트에게 RHS 함수를 통해 행위를 생성하게 만드는 motor command와 같은 것을 말한다 가상 물리실험에서는 "Move", "Take", "Attach" 등의 상위 행위들과 "Walk", "Grasp", "Release", "Connect", "Stop" 등의 하위 행위들이 있다 상위 행위들은 에이전트에게 당장 어떤 행위를 하게끔 명령을 내리는 것은 아니지만, 일련의 하위 행위들이 끝났을 때 최종적으로 목표를 성취하게 될 수 있도록 한다 예를 들면, <그림 3>의 (b)에서 "Move Plate to Table"의 경우 "Take Plate", "Walk to Table", "Release Plate"라는



<그림 3> 마찰력 실험의 목표 계층 및 행위 계층

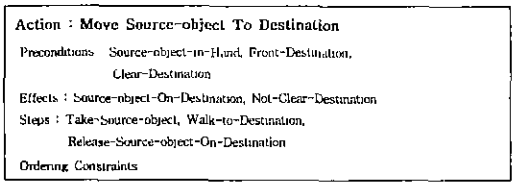
행위들이 끝났을때 "Move Plate to Table"이라는 하나의 상위 행위가 끝나게 되는 것이다 하나의 상위 행위 안에는 하위 행위들뿐만 아니라 보다 작은 개념의 상위 행위가 있을 수도 있다. <그림 3> (b)의 "Take Plate"가 그런 경우로, 이 역시 "Walk to Plate", "Grasp Plate"라는 두 개의 하위 행위가 수행되는 것으로 행위가 끝나게 되는 것이다.

3.2 행위 계획

행위 계획이란 에이전트가 주어진 문제를 해결하기 위해 실행해야 하는 행위들의 순서를 결정하는 것을 말한다 하지만 단일 에이전트만이 있는 경우와는 달리 다중 에이전트가 존재하는 동적인 환경에서는 행위 계획에 많은 제약이 따르게 되어 복잡하다 이러한 환경에서의 에이전트는 인간 참여자나 다른 에이전트의 행동을 계속해서 인지하고 있어야 하며, 현재 자신이 성취해야하는 목표에 대한 계획뿐만 아니라 우발적으로 발생하는 예기치 못한 많은 이벤트들에 대처하면서 생성된 계획을 유지, 변경하여 작업을 수행해야 하기 때문이다 이 밖에도 가상물리실험에서의 다중 에이전트는 인간 참여자의 일련의 동작을 분석하여 자신의 목표를 수정할 수 있어야 하는데, 이것은 인터럽트의 일종으로 사용자가 갑자기 자신의 행위를 수정했을 경우 이를 인식하여 에이전트 자신의 목표와 행위도 수정할 수 있어야 한다 또한 인간 참여자나 다른 에이전트와 충돌하지 않도록 그들의 행위를 미리 예측하여 다음 행위를 계획할 수 있어야 한다

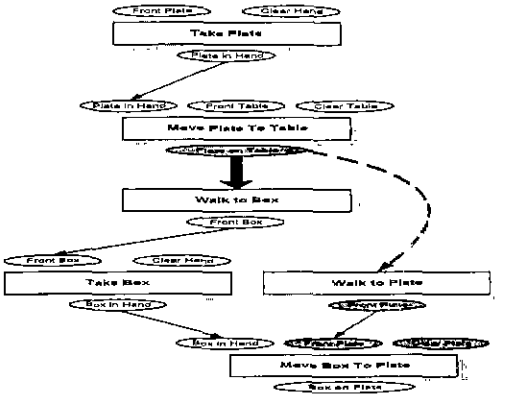
기존의 일반적인 계획 방법에서는 목표가 주어졌을 때, 초기 상태에서부터 목표 상태까지의 전과정에 대한 계획이 끝났을 경우에 각 행위에 대한 실행을 가능하게 한다 하지만 동적인 환경에서는 목표나 현재 상태가 계속해서 변화될 수 있기 때문에 하나의 목표에 대한 전체 계획이 끝날 때까지 기다린다는 것이 비효율적이다[6,8] 이를 개선하기 위해 본 연구에서 제안하는 방법은 행위들의 계층적 표현에 의한 목표의 부분적 계획 방법이다 행위들을 <그림 3>의 (b)에서처럼 계층적으로 표현했을 경우 오브젝트들 사이의 관계를 분석하는 제어지식을 사용하여 상위 행위들 사이의 순서는 탐색을 하지 않고도 결정할 수 있으며, 상위 행위들은 하위 행위들의 순차적인 실행으로 이루어지기 때문에 대부분의 상위 행위들이 이루어지기 위한 계획은 서로 독립적으로 이루어질 수 있다 이런 이유에서 계층적 구조에 의한 계획 방법은 복잡한 가상환경에서도 빠른 시간에 현재 상태에 적용할 수 있는 행위를 결정할 수 있다

<그림 4>는 계층적 구조로 표현된 행위들 중 "Move"라는 상위 행위에 대한 표현이며, "steps"에는 "Move"를 완성하기 위한 행위들이 첨가되고, 이러한 행위들의 순서는 "ordering-constraints"에 표시된다 예를 들어 마찰력 실험에서는 "Move-Box-On-Plate"라는 행위가 완전히 성취되기 위해서는 3개의 하위 행위들이 실행되어야만 한다 처음에는 "steps"와 "ordering constraints"가 null 상태이며, 선행 조건을 성취하기 위한 상위 행위들이나 하위 행위들을 "steps"에 추가한 후, 이들 사이의 threat를 검사하여 전체적인 하위 행위들의 순서를 결정하기 위한 "ordering-constraints"를 결정한다 모든 하위 행위들의 순서가 결정되면 가상행위자는 동작 생성기로 하위 행위 명령들을 차례로 보낼 수 있으며, 만일 도중에 다른 interrupt가 발생하여 계획을 수정해야 한다면, 연관이 없는 다른 상위 행위들의 순



<그림 4> 가상행위자의 행위 표현의 예

서까지 다시 재계획하지 않을 수 있게 된다. <그림 5>는 이러한 ordering을 결정하는 과정의 일부를 나타낸 것이다 "Plate on Table"과 "Box on Plate"라는 두 상위 행위의 순서를 결정하기 위해 두 오브젝의 관계를 분석해야 한다. 즉, 모든 오브젝트는 "relation" 이라는 속성을 사용해 다른 오브젝트와의 관계를 표현하고 있기 때문에 "Box on Plate"의 "Plate"는 <그림 5>에서 " ↗"가 표시하는 것처럼 이미 "Table"이라는 오브젝트와 "On"의 관계를 가지고 있는 "Plate"이어야 함을 알 수 있게 된다 이것을 사용해 "Plate on Table"이 "Box on Plate" 보다 먼저 성취되어야 하는 것임을 추론하여 <그림 5>에서 "⬇"의 링크를 만들게 된다 이러한 방법으로 제어 지식을 사용해 "Move Plate To Table"이라는 상위 행위가 "Move Box To Plate"보다 먼저 실행되어야 함을 알 수 있으며, 이렇게 모든 행위들의 순서 계획이 끝나면 하위 행위를 하나씩 동작 생성기로 보내면서 가상환경의 변화되는 상태를 지켜보며, 새로운 문제가 생기면 다시 step을 첨가하거나 새로운 행위를 만들어 이들 사이의 순서만 재귀적으로 계획하면 동적으로 변화되는 가상환경에 대응하여 효과적으로 행위 계획을 생성할 수 있다

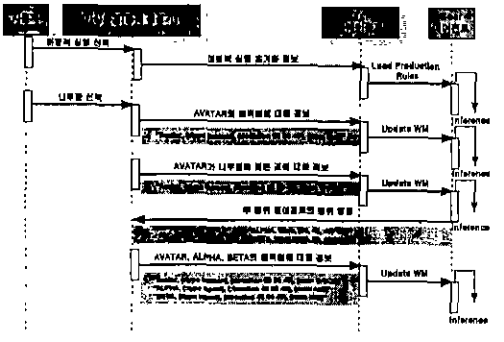


<그림 5> 계층적 구조를 사용한 행위 계획

4. 구 현

본 논문의 다중 에이전트는 Soar와 Tcl을 사용하여 구현되었다. Soar는 계획, 학습, 추론 등의 지능적인 행위를 구현하기에 적합한 통합적 인공지능 아키텍처이며, Tcl은 다중 에이전트 환경을 지원하기 위해 사용되었다.

이렇게 생성된 에이전트는 인식 모듈인 비에이전트를 통해 가상환경의 정보를 인식하고, 주어진 목표를 성취하기 위한 행위 계획을 생성하고 현재 상태에서 실행할 수 있는 행위를 결정한 후, 선택된 행위 명령을 RHS 함수를 사용해 다시 동작 생성기로 보내게 된다 <그림 6>은 가상환경과 다중 에이전트 시스템 사이의 이러한 메세지 전달 과정을 나타낸 것이다 편일 사용자가 마찰력 실험을 선택하고 나무판을 선택하게 되면, 동작 생성기는 GUI에 사용자의 분신인 "AVATAR"가 나무판을 잡기 위해 이동하는 과정을 그래픽으로 출력하고, 다중 에이전트 시스템의 비에이전트로 "AVATAR"의 위치 변화를 알려준다 비에이전트는 이를 분석하여 에이전트의 WM을 갱신한다 그러면 에이전트 "ALPHA"와 "BETA"는 이러한 위치 변화와 "AVATAR"의 정보를 추론하여 사용자가 나무판을 선택해서 가져가고 있다는 것을 인식하게 된다 에이전트 "ALPHA"는 이것으로부터 자신이 선택해야할 판의 종류를 선택하여 에이전트 "BETA"에게 이를 알려주고, 판을 잡기 위한 행위를 계획한 후 가장 처음 행위인 판 앞으로 걸어가라는 행위 명령 "Walk"를 동작 생성기로



<그림 6> GUI와 다중에이전트 시스템의 정보 전달

전달한다. 에이전트 "BETA"는 에이전트 "ALPHA"의 판 선택에 대한 메세지를 받으면, 자신이 사용할 판을 선택하여 역시 동작 생성기로 행위 명령을 보내게 된다. 동작 생성기는 이 명령들을 GUI에서 그래픽으로 출력하고, 또다시 에이전트의 위치 변화와 "AVATAR"의 위치 변화에 대한 정보를 비에이전트에게로 보내게 된다. 가상환경에서는 그 안에 존재하는 모든 물체들의 변화되는 정보들을 다중 에이전트 시스템으로 보내고, Soar 에이전트는 이를 분석하여 에이전트의 다음 행위나 목표를 결정한다 이러한 반복된 과정을 통해 실험이 끝나면 비에이전트는 실험 결과를 비교, 분석하여 사용자에게 설명 해줌으로써 모든 작업이 끝나게 된다.

5. 결론 및 향후 연구

본 논문에서는 다중 에이전트가 동적으로 변화하는 가상환경과 상호작용 하면서 목표 성취를 위한 계획을 생성하고, 계속적으로 변화되는 가상환경의 정보들을 분석하여 자신의 목표를 수정할 수 있도록 모형화 할 수 있었다 또한 각 에이전트는 계층적인 행위 표현을 통해 탐색을 하지 않고도 상위 행위들 사이의 순서를 알 수 있는 제어 지식을 사용해 계획을 좀 더 효율적으로 할 수 있었다 하지만 사용자나 타 에이전트의 일련의 행위 과정들을 분석하여 서로간의 충돌을 회피한다거나 공통의 목표를 분할하여 성취하는데 있어 타 에이전트의 목표를 고려하여 자신의 계획을 생성하는 다중 행위 계획에 대한 연구가 부족했다 가상환경에서 적응적으로 다른 에이전트들과 협동하여 복잡한 문제들을 해결하기 위해서는 다른 에이전트가 가지고 있는 정보들을 교환하여 불완전한 자신의 지식을 보완하고 효율적인 행위 계획을 생성할 수 있도록 하는 다중 행위자 계획기에 대한 연구가 더 이루어져야 할 것이다.

참고 문헌

[1] Allen Newell, The Knowledge Level *Computation & Intelligence Collected Readings*, 1995.
[2] Clare Congdon, K. Schwamb, *The Soar Advanced Applications Manual Version 7*, 1995
[3] Craig Knoblock, Josh Tenenberg Qiang Yang, *In Proceedings of the Ninth National Conference on Artificial Intelligence*, 1991
[4] Daniel Weld, An Introduction to Least Commitment Planning, *AI Magazine*, 1994
[5] David Wilkins, Karen Myers, A Common Knowledge Representation for Plan Generation and Reactive Execution, *Journal of Logic and Computation*, 1995
[6] Earl Sacerdoti, Planning in a Hierarchy of Abstraction Spaces, *Artificial Intelligence*, 1974
[7] Jeff Rickel, Lewis Johnson Task-Oriented Dialogs with Animated Agents in Virtual Reality, *In Proceedings of AAAI-98 Workshop on Representations for Multi-Modal Human-Computer Interaction*, 1998
[8] Kiren Haigh, Manuela Veloso High-Level Planning and Low-Level Execution Towards a Complete Robotic Agent, *In Proceedings of the First International Conference on Autonomous Agents*, 1997
[9] Katia Sycara, Multiagent Systems, *AI Magazine*, 1998.
[10] Michael Coen, Sodabot A Software Agent Environment and Construction System, *AI Technical Report 1493 MIT* 1994
[11] Nils Nilsson, Teleo-Reactive Programs for Agent Control, *Journal of Artificial Intelligence Research 1*, 1994
[12] Pattie Mnes, Rodney Brooks Learning to Coordinate Behaviors, *AAAI*, 1990
[13] Peter Stone, Manuela Veloso, Multiagent Systems' A Survey from a Machine Learning Perspective, *IEEE Transactions on Knowledge and Data Engineering(TKDE)* 1996
[14] 원용대, 이정진 이수원, 가상 물리실험을 위한 하이브리드 에이전트, *제24회 학술정보처리회 가을 학술발표 논문집*, 1997