# 컴퓨터 게임 환경에서 에이전트들의 효율적인 협력을 위한 통신 프로토콜과 역할 배정☆

# Communication Protocol and Role Assignment for Efficient Agent Cooperation in Computer Game Environments

김 인 철*
In-Cheol Kim

## 요 약

본 논문에서는 컴퓨터 게임 환경에서 에이전트들의 효율적인 협력과 팀워크를 위한 메시지 기반의 통신 프로토콜과 동적 역할 배정 방법을 제시한다. 본 논문에서 제시하는 동적 역할 배정 방식은 설계단계 혹은 실행단계에서 역할 배정에 필요한 모든 사항을 결정하는 기존의 정적 역할 배정 방식이나 동적 역할 배정 방식과는 다른 새로운 역할 배정 방식이다. 이 역할 배정 방식에 따르면, 설계단계에서 가능한 역할집합들을 미리 결정하지만, 이 역할집합에 명시된 각각의 역할을 어느 에이전트가 맡아 수행할 것인가는 실행단계에 가서 결정한다. 이 방법은 역할배정에 필요한 실행단계의 협상을 최소화할 수 있는 방법으로서, 실시간 멀티 에이전트 환경에 매우 효과적인 방법이다. 본 본문에서는 실험을 통해 새로운 동적 역할 배정 방식의 우수성을 보인다.

## Abstract

In this paper, we suggest a message-based communication protocol and a dynamic role assignment mechanism for efficient cooperation and teamwork of agents in computer game environments. This role assignment mechanism is a new one different from both existing static and dynamic mechanisms, in which all decisions related with role assignment are made at once in design phase or execution phase. According to our mechanism, all possible role sets are determined in design phase. Detail decisions regarding which agent takes what role, however, are made in execution phase. This mechanism for role assignment can minimize the negotiation effort in execution phase. Therefore, this mechanism is quite effective especially in real-time multiagent environments. Through experiments, we show the superiority of the new dynamic role assignment mechanism.

☞ Keyword : Agent Cooperation, Computer Games, Role Assignment, Negotiation

## 1. 서 론

오늘날 대부분의 인터렉티브 컴퓨터 게임들에서는 다수의 지능형 캐릭터 에이전트, 즉 NPC (Non-Player Character)들이 등장하며, 이들이 휴먼 플레이어 또는 다른 NPC들과 한 팀을 이루어 서로 협력하거나 경쟁해야 하는 멀티 에이전트 환경을 제공한다. 멀티 에이전트 환경에서는 팀 구성원 에이전트들 사이에 효율적인 협력 메커니즘이 반드시 필요하다. 주어진 환경의 특성과 에이전트들의 능력에 따라 멀티 에이전트 협력을 위한 방법은 여러 가지가 있을 수 있으나, 가장 기본적인 전략은 먼저 목표 달성을 위해 필요한 전체 작업을 적절한 세부단위로 나눈 다음, 그것들을 에이전트들에게 효과적으로 분배하는 것이다. 따라서 일반적으로 멀티 에이전트 협력문제에 대한 접근은 작업분할과 작업분산에 대한 연구로 집중된다[1, 8, 24].

작업분할의 경우, 경우에 따라 시스템 설계자에 의해 사전에 작업분할이 이루어지고 그것을

* 정 회 원 : 경기대학교 전자계산학과 교수
kic@kyonggi.ac.kr
[2006/02/01 투고 - 2006/02/17 심사 - 2006/03/15 심사완료]
☆ 본 연구는 2004학년도 경기대학교 학술연구비(일반연구과제) 지원에 의하여 수행되었음

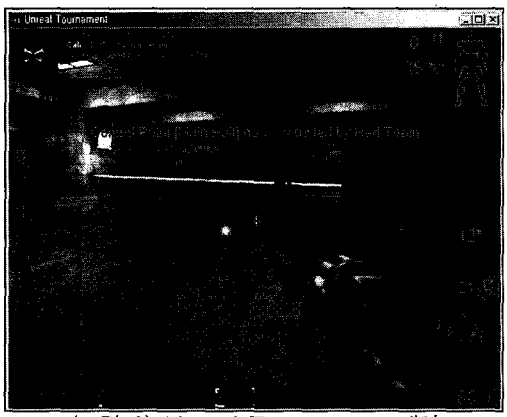
에이전트 구현단계에서 미리 프로그램화할 수 있으며, 또는 계층적 계획(hierarchical planning)과 같은 방법을 통해 에이전트들 스스로 동적으로 작업분할을 할 수도 있고, AND/OR 그래프와 같은 문제표현법에 이미 논리적인 작업분할이 내재되어 있을 수도 있다. 한편 작업분산을 위한 대표적인 방법으로는 시장 메커니즘(Market Mechanism), 계약 망(Contract Net), 멀티 에이전트 계획(Multiagent Planning), 조직 구조(Organizational Structure)[8], 이들을 혼합한 혼합방법 등이 존재한다[6]. 이중에서 특히 조직 구조는 조직론(organization theory)에 기초를 둔 방법으로서, 구성원 에이전트 각각에게 적합한 고유의 역할을 부여하는 것을 근간으로 하고 있다[7, 9, 22]. 이와 같이 각 구성원 에이전트에게 맞는 적절한 나름의 역할을 부여함으로써 서로 불필요한 중복 행위나 상호작용을 피하고 효과적으로 협력할 수 있음을 알 수 있다. 지금까지 멀티 에이전트 분야에서 시도되어 온 역할배정 방식은 주로 정적 역할배정방식으로서, 각 에이전트에 대한 역할배정이 실행 이전에 시스템 설계자에 의해 이루어지고, 실행 중에는 역할변경이 이루어지지 않는 방식이다. 이 방식이 성공하기 위해서는 설계자가 사전에 필요한 모든 역할들과 역할 수행자를 파악할 수 있어야 한다. 하지만 동적으로 빠르게 변화하는 환경에서는 실행에 필요한 모든 역할과 역할 배정을 사전에 결정하기 매우 어렵다. 따라서 정적 역할 배정 방식은 빠르게 변화하는 실시간 멀티 에이전트 환경에서는 그 효과를 기대하기 어렵다.

본 논문에서는 대표적인 멀티 에이전트 컴퓨터 게임 환경인 Unreal Tournament(UT) 환경에서 지능형 캐릭터 에이전트들의 효율적인 협력과 팀워크를 위하여, 메시지 기반의 통신 프로토콜을 설계하고, 이것을 기초로 에이전트들 간의 동적 역할배정 방법을 구현하였다.

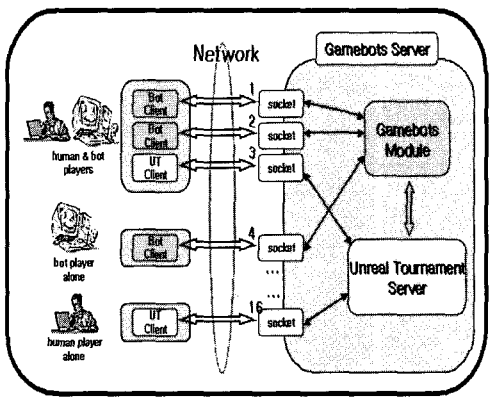
## 2. 멀티 에이전트 컴퓨터 게임 환경

그림 1의 Unreal Tournament 게임은 대표적인 3D 일인칭 액션 게임이다. 이 게임은 온라인 상용게임이면서도 대부분의 소스가 공개되어 있다. UT게임은 견고한 3D 시뮬레이션 엔진, 다양한 게임 유형과 맵을 제공하고 있으며, 게임의 확장성을 위해 전용 스크립트 언어인 Unreal Script도 제공하고 있다. Gamebots 시스템[5]은 지능형 에트워크를 통해 NPC를 제어하는 원격의 보트 클라이언트(bot client)들과 Gamebots 서버로 구성된다.

Gamebots 서버모듈은 원격의 보트 클라이언트들에게 휴먼 플레이어와 동일한 센서정보를 제공하고 역으로 보트 클라이언트의 행동명령을 받아 실



〈그림 1〉 Unreal Tournament 게임

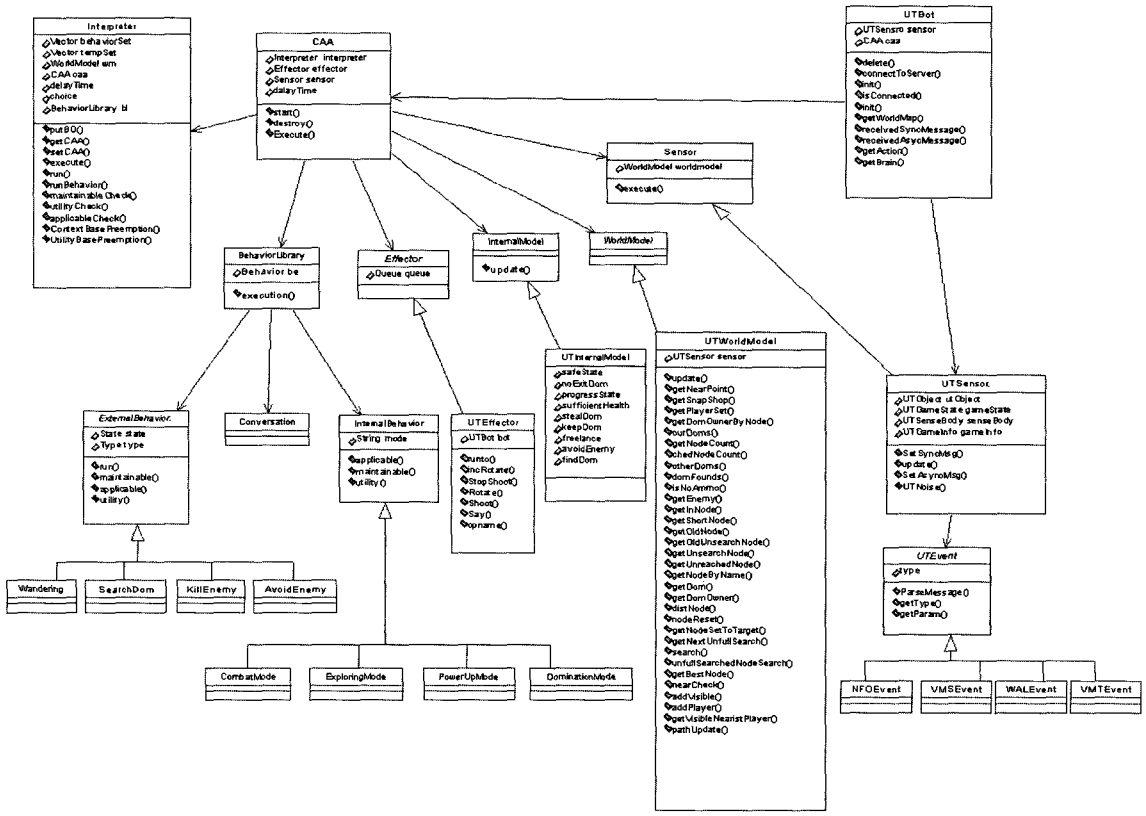


〈그림 2〉 Gamebots 시스템

행하는 역할을 한다.

Gamebots 서버가 원격의 에이전트에게 전달하는 메시지들은 크게 초기화 메시지(initialization message), 동기 메시지(synchronous message), 비동기 메시지들(asynchronous message)로 구성된다. 각 에이전트의 연결 초기 단계에서는 초기화 메시지의 교환을 통해 서버 측에서는 에이전트에게 게임 타입과 게임의 승리조건 등을 알려주며, 에이전트는 자신이 제어하는 캐릭터의 이름, 소속 팀 등을 서버 측에게 알려준다. 일단 연결이 되고 나면, 서버는 동기 메시지 모드와 비 동기 메시지 모드를 옮겨 다니며 캐릭터들의 지각정보(sensory information)를 해당 에이전트들에게 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 시각정보와 자신의 위치 및 신체정보, 그리고 게임 상황 정보(팀별/개인별 득점)를 묶어 전달하는 것이다. 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 진입했을 때 관련 정보를 전달해준다. 한편, 원격의 에이전트로부터 Gamebots 서버에 전달되는 명령 메시지 유형에는 RUNTO, TURNTO, JUMP, SHOOT, STOP, CHANGEWEAPON 그리고 MESSAGE 등이 있다.

Gamebots시스템에서 에이전트들은 팀 전용 채널 또는 글로벌 채널의 MESSAGE 명령을 이용함으로써 팀 동료들 혹은 상대팀을 포함한 모든 에이

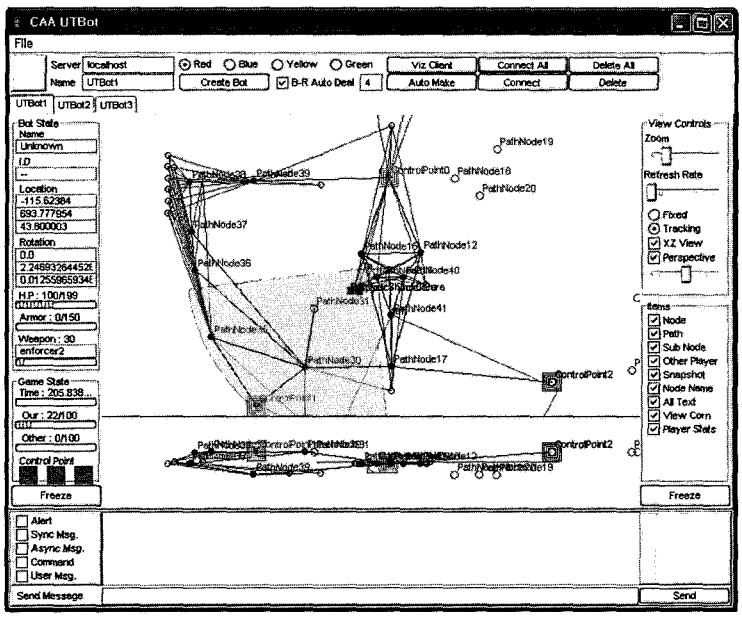



〈그림 3〉 에이전트의 내부 구조

전트들에게 메시지 전달이 가능하다. 하지만 Gamebots 시스템이 제공하는 통신 메시지는 단지 일정한 길이의 임의 문자열로서, 현재로서는 에이전트들 간의 지식공유와 조정을 위한 어떤 통신 프로토콜도 제공하고 있지 않다. 그뿐만 아니라, 현재의 Gamebots 시스템은 에이전트들의 팀워크[13, 14]를 위한 어떤 명시적인 메커니즘도 제공하고 있지 않다. 본 연구에서는 Gamebots 시스템의 팀 전용 메시지 기능을 이용하여, 에이전트들의 효율적인 팀워크를 위한 통신 프로토콜과 동적 역할 배정 메커니즘을 구현한다.

## 3. 에이전트의 내부구조

본 연구에서는 Gambots 시스템 환경에서 자율적으로 동작하는 지능형 캐릭터 에이전트의 제어를 위해 범용 에이전트 구조의 하나인 CAA (Context-sensitive Agent Architecture)를 채용하였다. 본 연구를 위해 개발된 CAA는 에이전트의 모든 부분을 Java 언어로 구현할 수 있는 행위기반의 에이전트 구조이며, 복잡한 행위들을 효과적으로 표현하고 수행할 수 있는 기능을 제공한다. 이를 위해 환경 변화에 따라 실행 중이던 행위를 즉각적으로 중단하거나 다른 행위로 전환할 수 있는 높은 상황-민감성(context- sensitivity)을 제공한다. CAA는 크게 행위 라이브러리(behavior library), 월드 모델(world model), 내부 모델(internal model), 센서(sensor)와 실행기(effector), 그리고 인터프리터(interpreter) 등으로 구성된다. 그림 3은 CAA를 포함한 지능형 캐릭터 에이전트의 내부 구성요소들에 대한 클래스 다이어그램(class diagram)을 나타낸 것이다. CAA의 행위 라이브러리(behavior library)는 실행 가능한 모든 행위 객체들을 저장, 관리한다. 각 행위는 행위 상태와 행위 유형을 가진다. 하나의 객체로 표현되는 각 행위는 실행 상태에 따라 생성(create), 대기(waiting), 수행(executing), 중지(interrupt), 복귀(resume), 종료(finish) 등의 서로 다른 상태들을 가진다. 행위의 유형들로는 내부 행위(internal behavior)와 외부 행위(external behavior), 그리고 대화 행위(conversation)가 있다. 내부




〈그림 4〉 에이전트의 사용자 인터페이스

행위는 주로 에이전트 내부의 의사결정을 의미하며, 실행을 통해 내부 모델(internal model)에 저장된 현재의 행위모드를 갱신하게 된다. 이에 반해, 외부 행위는 실제 실행기(effector)를 통해 외부 환경을 변화시키는 행위를 말한다. 또, 대화 행위는 다른 에이전트에게 정보를 제공하거나 서비스를 요청하는 등의 목적으로 행해지는 메시지 송신 행위를 말한다. CAA의 월드 모델은 외부 환경의 상태를, 반면에 내부 모델은 에이전트의 내부 상태 즉, 행위모드를 표현한다. CAA의 인터프리터(interpreter)는 에이전트의 행위에 관한 총체적인 제어를 담당한다.

본 연구에서는 CAA를 기초로 Gamebots 시스템 환경에 동작하는 지능형 캐릭터 에이전트인 CAA UTBot를 구현하였다. 하나의 CAA UTBot는 UT 게임 플레이에 필요한 상태정보들을 담고 있는 월드모델과, 게임전략(strategy)들을 표현하는 내부행위들, 그리고 기본 기술(skill)들을 나타내는 다수의 외부행위들을 포함하고 있다. 월드모델에는 에이전트의 이름과 식별자, 팀 번호, 팀 구성원의 수, 최대 팀 점수, 게임서버 주소 등과 같은 정적인 정보와 에이전트의 위치와 방향, 건강정도, 현재 무기, 맵 정보 등과 같은 동적인 정보들이 저장된다. CAA UTBot의 내부모델에는 에이전트의 현재 행위모드와 몇 가지 인자들이 함께 저장된다. UTBot은 Explore, Dominate, Collect, Died, and Healed 등 다섯 가지 서로 다른 행위모드를 가질 수 있으며, 시작 위치(starting position)와 목표대상(target object) 등의 인자들을 추가로 가질 수 있다. CAA UTBot는 현재 Explore, Attack_Point, Defend_Point, Collect_Powerup, Collect_Weapon, Collect_Armor, Chase, Attack, Retreat, MoveTo. 등 다수의 외부행위들을 가지고 있다. 또한, CAA UTBot는 그림 4와 같은 그래픽 사용자 인터페이스를 제공하는데, 사용자는 이것을 통해 에이전트의 내부 정보와 행위 선택 과정 등을 추적함으로써 에이전트의 지능 행위들을 감시하고 분석할 수 있다.

## 4. 에이전트 통신 프로토콜

기존의 많은 에이전트 통신 언어 및 통신 프로토콜들이 Speech Act Theory에 기초를 두고 개발되었다. Speech Act Theory는 인간의 자연언어를 요청(request), 제안(suggest), 결정(commitment), 응답(reply) 등과 같은 행위(action)들로 보는 이론이다. 이 이론에 따르면 하나의 발화 행위(speech act)는 다음과 같은 세 가지 다른 요소(aspect)를 갖는다고 한다. (1) Locution : 발화자에 의한 물리적인 음성 전달 (2) Illocution : 발화자가 의도하는 의미(meaning) (3) Perlocution : 청취자에 의해 일어나는 행동 등이다. 그리고 Speech Act Theory는 발화자 의도의 강도를 promise, report, convince, insist, tell, request, demand 등과 같은 몇 개의 Performative들로 나타낼 수 있다. 이러한 Speech Act Theory에 기초를 둔 에이전트 통신 프로토콜에 관한 몇몇 연구들이 있는데, 대표적인 것들이 DARPA프로젝트에 의해 개발된 KQML (Knowledge Query and Manipulation Language)과 에이전트 기술 표준화 기구인 FIPA에 의해 개발된 ACL(Agent Communication Language) 등이 있다. 하지만 이들 언어는 본 연구의 대상영역인 실시간 컴퓨터 게임과 같은 영역에 적용하기에는 필요 이상의 많은 Performative들과 긴 메시지 길이 등 적합지 않은 부분이 많다. 따라서 본 연구에서는 Speech Act Theory에 기초 둔 통신 프로토콜을 설계함으로써 기존의 KQML이나 ACL 등과 같은 범용성과 확장성을 가지면서도, 실시간 인터렉티브 컴퓨터 게임 환경의 제약성을 충분히 만족할 수 있는 통신 프로토콜을 설계한다. 또한 이 통신 프로토콜 위에서 구현될 에이전트 간 역할배정 등 협력을 위한 상호작용을 충분히 지원할 수 있는 기능을 함께 고려하여 설계하였다. 본 연구에서 제안하

는 에이전트 통신 프로토콜의 근간을 이루는 메시지 형태는 다음과 같다.

( performative
:sender s-agent
:receiver {r-agent}
:content content-str
:reply-with id-1 :in-reply-to id-2
:reply-by timelimit )

그리고 메시지에 사용되는 Performative들의 종류와 의미는 다음과 같다.

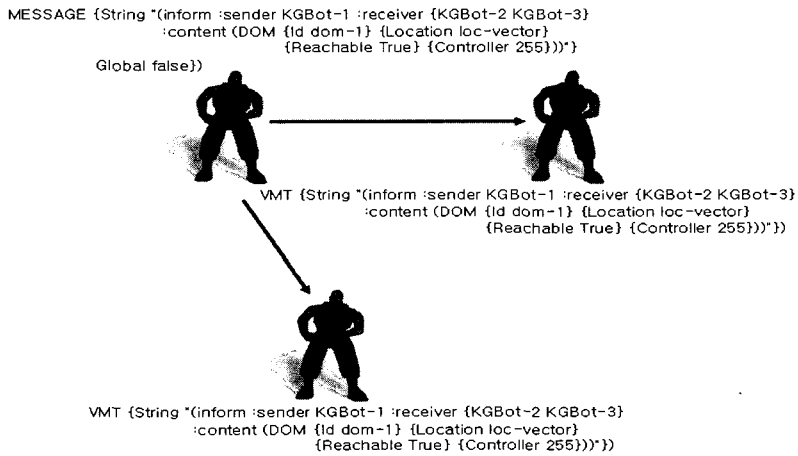
Query : 정보 요청
Inform : 정보 제공
Cfp : 상대에게 제안을 요청
Propose : 특정 행위 또는 작업을 수행토록 제안
Accept-proposal : 제안 받은 것을 수락
Reject-proposal : 제안 받은 것을 거절
Request : 특정 행위나 작업을 수행토록 요청
Agree : 요청받은 바를 수행하겠다는 응답
Cancel : 다른 에이전트에게 수행하라고 요청했던 작업을 중도에 취소하라는 요청

Query와 Inform 메시지의 구체적인 예는 다음과 같다.

(**query** :sender agent-1 :receiver {agent-2 } :content{(fact map $map) }:reply-with id-1)
(**inform** :sender agent-2 :receiver {agent-1 } :content {(fact map mapstr) } :in-reply-to id-1)

한편, Cfp, Propose, Accept-proposal, Reject-proposal 등의 구체적인 예들은 다음과 같다.

(**propose** :sender agent-1 :receiver {agent-2} : content {(do (agent agent-2) (achieve domi nate dom point-1) (achieve domi nate dompoint-2)) }:reply-with id-1 :reply-by time-1)
(**accept-proposal** : sender agent-2 : receiver {agent-1} :content {(do (agent-2) (achieve domi nate dompoint-1) ) } : in-reply-to id-1)
(**inform** : sender agent-2 : receiver {agent-1} : content {(done (do (agent agent-2)(achieve dominate dompoint-1) (achieve domi




〈그림 5〉 통신 프로토콜에 기초한 메시지 전송의 예

nate dompoint-2)))} : in-reply-to id-1)

(**cfp** : sender agent-1 : receiver {agent-2} : content{(do (agentagent-1)(achieve dominate $dompoint))}with-reply:id-1:reply-by time-1)

(**propose** :sender agent-2 :receiver {agent-1 } :content {(do (agent agent-1) (achieve dominate dompoint-1) (achieve dominate dompoint-2)) } :in-reply-to id-1)

Request, Agree, Cancel 등의 구체적인 예들은 다음과 같다.

(**request** :sender agent-1 :receiver {agent-2 } :content {(do (agent agent-2)(achieve explore)) } :in-reply-to id-1 :reply-by time-1)

(**agree** :sender agent-2 :receiver {agent-1 } :content {(do (agent agent-2)(achieve explore)) } :in-reply-to id-1)

(**cancel** :sender agent-1 :receiver {agent-2 } :content {(do (agent agent-2)(achieve explore)) } :in-reply-to id-1 :reply-by time-2)

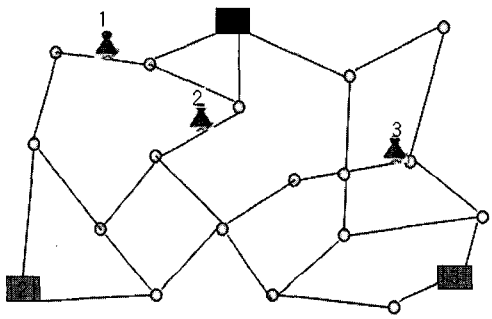
그리고 메시지 컨텐트 부분의 구문(syntax)은 아래와 같은 규칙을 따른다.

- {(fact map $map) }
- {(goal *) }
- {(achieve dominate pt-1)(achieve dominate pt-2) }
- {(do (agent agent-1)(achieve dominate pt-1)(achieve dominate pt-2)) }
- {(done (do ...)) }
- {(cancelled (do ...)) }
- {(failed (do ...)) }

한편, 현재 Gamebots 시스템에서 허용하는 NPC들 간의 팀 메시지 형식은 아래와 같다.

•팀 메시지 전송
(MESSAGE {String "xyz"} {Global False})
•팀 메시지 수신
(VMT {String "xyz"})

따라서 이러한 Gamebots 시스템의 메시지 틀에 맞추어 앞서 제안한 통신 프로토콜을 적용하면, 그림 5와 같은 팀 구성원간의 메시지 전송이 가능하다.




〈그림 6〉 Team Domination 게임

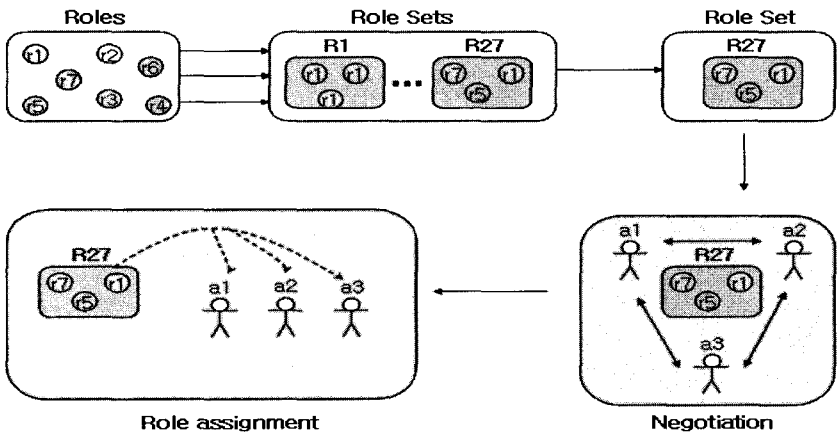
## 5. 동적 역할 배정

에이전트의 역할배정을 위한 기존의 방법들은 주로 정적인 방법들로서, 설계단계에서 각 에이전트가 맡아야 하는 역할을 미리 정의한 후, 실행단계에서는 단지 이와 같이 정의된 역할을 수행하는 방식들이다. 이러한 정적인 역할배정 방법은 실행 시 별도의 에이전트 간 협상[2, 23, 24]을 요구하지 않으므로 실시간 환경에 적용하기 적합하다. 하지만, 이 방법은 설계단계에서 실행 시에 발생할 모든 상황들을 미리 예측하고, 그에 따른 에이전트들의 역할을 모두 결정해야 한다는 문제점이 있다. 대부분의 실시간 인터렉티브 컴퓨터 게임들에서는 환경변화가 매우 빠르고 그 변화도 비결정적이어서 이러한 정적인 역할배정 방법을 적용하기에 어려움이 많다. 반면에, 에이전트들의 역할 배정과 관련된 모든

결정을 실행 시까지 미루었다가 에이전트간의 실시간 협상을 통해 결정하는 순수한 동적배정 방법을 생각해 볼 수 있다. 하지만, 이 방법도 역할배정을 위한 에이전트간의 많은 통신으로 인한 지연시간 때문에 실시간 환경에 효과적으로 적용하기 어려운 점이 있다. 따라서 본 연구에서는 기존의 정적 역할 배정의 장점인 실시간 적용성과 동적 역할 배정의 유연성을 결합한 새로운 동적 역할 배정 방식을 제안한다. 본 연구는 특히 그림 6과 같은 Team Domination 게임에서 각 팀 구성원 에이전트들 간의 역할배정 방식에 초점을 두고 있다. 하지만, 이러한 동적 역할배정 방식은 유사한 대부분의 팀별 인터렉티브 컴퓨터 게임들에도 적용될 수 있을 것이다.

Team Domination 게임은 2명 이상의 캐릭터들로 구성된 2~3개의 팀들이 서로 경쟁하며 복잡한 미지의 3차원 공간 안에 숨겨진 점령 목표지(domination point)들을 찾아내고 그것들을 오래 점령함으로써 팀 점수를 높여 승리하는 게임이다. (그림 6)은 Team Domination 게임의 공간을 2차원 그래프 형태로 간략화한 것이다. 그림에서 원으로 표시된 각 노드는 캐릭터 에이전트들의 공간 이동을 돕기 위해 맵 상에 마련된 이동점(way point)들을 나타내며, 3 개의 사각형 노드들은 숨겨진 점령 목표지(domination point)들을 나타낸다. 또, 그림에서 점령 목표노드의 색은 그 노드를 점령하고 있는 팀을 표시하며, 캐릭터 에이전트에 표시된 번호는 한 팀을 이루는 팀 구성원 에이전트들의 식별번호를 나타낸다. 지능형 캐릭터 에이전트들이 한 팀을 이루어 이와 같은 Team Domination 게임을 성공적으로 이끌어 가기 위해서는 미지의 공간을 돌아다니며 월드 맵을 작성하고 숨어있는 점령 목표지들을 찾아내는 Explorer의 역할과 탈환과 방어를 통해 찾아진 점령 목표지들을 지속적으로 점령하는 Dominator의 역할이 팀 구성원들에게 필수적으로 요구된다. 그러나 이러한 구성원들의 역할은 게임 내내 고정되어 있어서는 효과를 얻을 수 없고, 게임 상황에 맞추어 적합한 역할들로 변화될 수 있을 때 더욱 효과적이다.

본 연구에서는 팀워크를 위해 필요한 에이전트의 역할집합들은 설계단계에서 미리 정의해두지만, 이 역할집합에 명시된 각각의 역할을 어느 에이전트가 맡아 수행하는 것이 가장 적합한지는 실행단계에 가서 각 에이전트의 실제 실행상태에 따라 결정하는 동적 역할배정 방식을 제안한다. 이 방식은 역할배정에 필요한 에이전트 간의 협상을 최소화할 수 있어 실시간 컴퓨터 게임들에서 매우 효과적이다. 그림 7은 본 연구에서 제안하는 에이전트들의 역할배정 과정을 도




〈그림 7〉 역할 배정 과정

식화한 것이다. 그림에서 위쪽 과정은 설계단계에서 역할집합이 결정되는 과정을 나타내고, 아래쪽 과정은 실행단계에서 에이전트들 간의 협상에 의해 역할이 배정되는 과정을 나타내고 있다.

본 연구에서 제안하는 역할배정 방식을 정형적으로 정리하면 아래와 같다.

$A= \{a_1, a_2, .., a_n\}$ : n명의 에이전트들로 구성된 팀

$R= \{r_1, r_2, ..., r_k\}$ : k개의 서로 다른 역할들

$t_i$ : $(R_{i-1}, s) \rightarrow R_i$ : 역할집합 변이함수(role-set transition function)

($R_{i-1}$ : 이전의 역할 집합,
s : 환경상태,
$R_i$ : 새로운 역할 집합)

$m_i$ : $A \rightarrow R_i$ : 역할 배정 함수(role assignment function)

$F_i= <A, R_i, m_i>$ : 팀 포메이션(team formation)

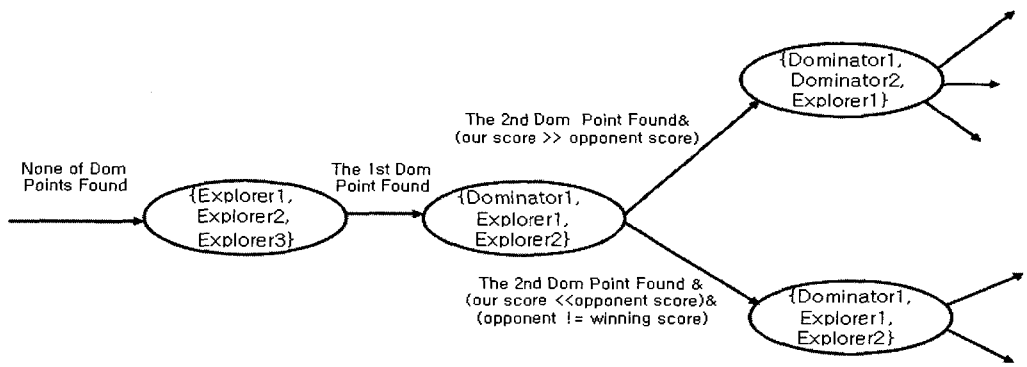
(A : 에이전트 집합, $R_i$ : 역할 집합, $m_i$ : 역할 배정 함수)

이때 역할집합 변이함수 $t$는 시스템 설계자에 의해 미리 정의되고, 역할 배정 함수 $m_i$은 에이전트 간의 실시간 협상에 의해 결정된다.

게임 상황에 따른 역할집합의 변이는 그림 8과 같은 역할집합 변이도(Role-set Transition Diagram)로 표현할 수 있다. 역할 집합 변이도에서 각 노드는 필요한 역할들의 집합을 나타내고, 각 링크는 역할 집합의 변화를 나타낸다. 그림 8의 역할 집합 변이도는 Domination 게임을 위한 전형적인 한 가지 예를 나타내고 있다.

이 역할 집합 변이도는 게임 초반 점령 목표지(domination point)들을 하나도 발견하지 못한 때에는 오직 Explorer 역할들만을 요구하나, 점령 목표지들을 발견함에 따라 그곳을 점령하고 지키는 Dominator 역할들을 요구하는 역할집합 변이를 보여주고 있다.

이와 같이 역할 집합은 역할 집합 변이도에 따라 자동적으로 결정되나, 실질적인 역할배정은 에이전트들 간의 메시지 통신을 통한 협상[25]에 의해 결정한다. 본 연구에서는 인터렉티브 컴퓨터 게임의 실시간 제약을 고려하여 에이전트들 간에 많은 메시지 교환을 요구하는 긴 협상 방법 대신 빠른 결정을 도출할 수 있는 경매(auction)를 적용한다. 경매는 대표적인 경제기반의 멀티 에이전트 협상 방법으로서, 각 에이전트는 자신의 이익(profit)에 따라 입찰(bid)을 제시하고, 이중에서 가장 높은 입찰가를 제시한 에이전트가 낙찰자로 결정되는 방식이다[19]. 본 연구에서는 역할 집합에서 요구하는 특정 역할을 맡을 에이전트를 결정하기 위해 이와 같은 경매




〈그림 8〉 역할 집합 변이도

를 적용한다. 역할 배정을 위한 경매를 위해서는 그 역할에 대한 각 에이전트의 비용함수(cost function)와 수입함수(revenue function)를 정의하여야 한다. 일반적으로 역할 비용함수 C는 다음과 같이 정의할 수 있다. C: $R_i \rightarrow R^+$. 이때 $R_i$ 는 역할 집합을, $R^+$는 양의 실수집합을 각각 나타낸다. 본 연구에서는 Team Domination 게임을 위한 가능한 역할들로 Explorer와 Dominator 들만을 고려하였다. Dominator 역할에 대한 각 에이전트의 비용함수로는 정해진 점령 목표지(domination point)까지의 도달거리를 사용하였고, Explorer 역할에 대한 비용함수로는 현재까지 작성된 월드 맵 경계선(frontier)까지의 도달거리를 사용하였다. 한편, 수입함수 R은 다음과 같이 정의할 수 있다. R: $R_i \rightarrow R^+$. 이때 역시 $R_i$ 는 역할 집합을, $R^+$는 양의 실수집합을 각각 나타낸다. 본 연구에서는 Dominator [1]역할에 대한 각 에이전트의 수입함수로는 목표지 점령을 통해 얻게 될 취득 예상 점수(score)를 사용하였고, Explorer 역할에 대한 수입함수로는 새로 발견할 노드와 에지의 수를 사용하였다.

역할 r에 대한 한 에이전트의 예상 이익은 그 역할을 수행함으로써 얻게 될 수입(revenue)과 소요될 비용(cost)의 차(difference)로 계산할 수 있다. 이렇게 역할 r에 대한 예상 이익이 계산되면, 각 에이전트는 자신의 예상 이익만큼을 입찰가로 제시한다. 따라서 결과적으로 해당 역할을 수행함으로써 가장 큰 이익을 얻을 수 있는 에이전트가 그 역할을 위한 낙찰자로 배정된다. 이와 같은 경매방식을 통해 역할집합에 포함된 역할들을 차례대로 적합한 에이전트에게 할당한다. 한편, 경매를 진행하기 위해서는 경매인(auctioneer) 에이전트가 있어야 하는데, 본 연구에서는 별도의 경매인 에이전트를 두지 않고 팀 구성원 에이전트 중의 하나에 경매인 역할을 겸하도록 하였다. 팀 구성원 에이전트들과 경매인 에이전트들 사이의 공모(announce)와 입찰(bid), 그리고 낙찰(clearing) 메시지 교환은 앞서 설계한 에이전트 통신 프로토콜에 따라 이루어진다.

〈표 1〉 최고점수 100일 때, 동적 역할 배정 팀과 임의 역할 배정 팀과의 득점 비교

| 맵 / 시도 | Map 1 (# of Nodes=30) | Map 2 (# of Nodes=90) | Map 3 (# of Nodes=120) |
|---|---|---|---|
| 1회 | 100 : 58 | 100 : 24 | 100 : 18 |
| 2회 | 100 : 46 | 100 : 34 | 100 : 22 |
| 3회 | 100 : 48 | 100 : 30 | 100 : 12 |
| 4회 | 100 : 65 | 100 : 38 | 100 : 30 |
| 5회 | 100 : 57 | 100 : 35 | 100 : 28 |

〈표 2〉 최고점수 200일 때, 동적 역할 배정 팀과 임의 역할 배정 팀과의 득점 비교

| 맵 / 시도 | Map 1 (# of Nodes=30) | Map 2 (# of Nodes=90) | Map 3 (# of Nodes=120) |
|---|---|---|---|
| 1회 | 200 : 109 | 200 : 55 | 200 : 23 |
| 2회 | 200 : 113 | 200 : 47 | 200 : 29 |
| 3회 | 200 : 99 | 200 : 51 | 200 : 41 |
| 4회 | 200 : 130 | 200 : 62 | 200 : 41 |
| 5회 | 200 : 101 | 200 : 58 | 200 : 62 |

## 6. 실 험

앞서 제시한 역할 배정 방식의 효과를 알아보기 위해, 협상에 의해 동적으로 역할 배정이 이루지는 에이전트 팀과 그렇지 않은 팀 간의 Team Domination 게임을 수행하는 실험들을 수행하였다. 각 팀은 3명의 동질 구성원(homogeneous member)들로 이루어지며, 각 구성원 에이전트는 본 연구를 위해 개발된 지능형 캐릭터 에이전트인 CAA UTBot를 이용하였다. 동적 역할 배정 팀은 구성원 에이전트들 간의 통신 메시지 교환을 통해 각 에이전트에게 Dominator의 역할과 Explorer의 역할을 게임 상황에 따라 동적으로 배정 가능하도록 구현하였고, 반면에 임의 역할 배정 팀은 팀 구성원들 간에 역할

배정을 위한 특별한 협상과정을 두지 않음으로써 구성원 각자의 판단에 따라 역할을 임의로 결정하도록 구현하였다. 따라서 동적 역할 배정 팀은 구성원들의 역할에 대한 팀 차원의 조정(coordination)이 게임 도중에 이루어지는 반면, 임의 역할 배정 팀은 이러한 팀 차원의 조정이 전혀 없다는 것이 차이점이다. 한편, 정적 역할 배정 방식은 팀 성능이 너무 낮아 본 연구의 실험에 포함시키지 않았다. 정적 역할 배정 방식이 특별히 성능이 낮은 이유는 Team Domination 게임 특성상 처음부터 Dominator의 역할을 부여받은 에이전트들은 Explorer 에이전트들이 점령 목표지들을 발견해낼 때까지는 아무런 도움도 주지 못한 채 기다리고 있어야만 하기 때문이다. 실험을 위해 노드 수가 각각 30개, 90개, 120개 등 복잡도가 다른 3개의 맵들을 이용하였고, 각 맵 당 5회씩 두 팀 간에 승부가 결정될 때까지 게임을 진행하여 최종적으로 각 팀이 취득한 점수들을 비교하였다. 표 1과 표 2는 각각 최고점수가 100, 200 일 때, 두 팀 간의 최종 취득 점수를 비교한 실험 결과를 나타낸다.

표 1과 표 2를 살펴보면, 예외 없이 모든 경우에 동적 역할 배정 팀이 임의 역할 배정 팀에 대해 큰 점수 차이로 승리를 거두었다는 것을 알 수 있다. 또 한 가지 주목할 점은 게임의 한계점수, 즉 최고점수가 100인 표 1의 실험결과에 비해 최고점수가 200인 표 2의 실험결과에서 동적 역할 배정 팀의 우수성이 더 높게 나타났다는 점이다. 이것은 게임이 더 오래 지속될수록 역할 배정이 팀 점수에 미치는 영향이 더 커진다는 것을 의미한다. 또한, 표 1과 표 2를 통해 공통적으로 발견할 수 있는 것은 게임에 사용된 맵이 커지고 복잡할수록 두 팀 간의 점수 차이는 더 벌어진다는 것이다. 따라서 게임 환경이 복잡하고 광역화할수록 역할 배정이 에이전트 팀 성능에 더 큰 영향을 미친다는 것을 알 수 있다. 실험에서 발견된 또 다른 흥미로운 현상 중의 하나는 점령 목표지들에 대한 발견이 빨리 이루어질수록 동적 역할 배정의 효과가 더 크게 나타고, 점령 목표지 발견이 지연될수록 효과가 적게 나타난다는 것이다. 따라서 본 연구의 실험대상인 Team Domination 게임에서는 Explorer 역할을 맡은 각 에이전트들의 공간 탐색 능력이 역할 배정의 기회와 효과에 큰 영향을 미친다는 것을 알 수 있었다.

## 7. 결 론

본 논문에서는 멀티 에이전트 게임 환경에서 지능형 캐릭터 에이전트들의 효율적인 협력과 팀워크를 위한 메시지 기반의 통신 프로토콜과 동적 역할 배정 방법을 제시하였다. 본 논문에서 제시한 통신 프로토콜은 범용성과 확장성을 가지면서도, 인터렉티브 컴퓨터 게임 환경의 실시간 제약을 만족할 수 있으며, 에이전트들 간의 동적 역할배정을 위한 협상을 지원할 수 있도록 설계되었다. 또한 본 연구에서 제시한 동적 역할 배정 방식은 설계단계 혹은 실행단계에서 역할 배정에 필요한 모든 사항을 결정하는 기존의 정적 역할 배정 방식이나 동적 역할 배정 방식과는 다른 새로운 역할 배정 방식이다. 이 역할 배정 방식에 따르면, 설계단계에서는 팀워크를 위해 필요한 역할집합들을 미리 결정하고, 실행단계에 가서 이 역할집합에 명시된 각각의 역할을 어느 에이전트가 맡아 수행할 것인가를 결정한다. 이 방법은 역할배정에 필요한 실행단계의 에이전트 간 협상을 최소화할 수 있는 방법으로서, 실시간 멀티 에이전트 게임 환경에 적합한 방법이다. 본 본문에서는 실험을 통해 새로운 동적 역할 배정 방식의 우수성을 보였다. 계획하고 있는 향후 연구로는, 역할 배정에 필요한 모든 결정 사항을 실행단계까지 미루는 순수 동적 역할 배정 방식과의 비교 실험을 통해 본 연구의 방식이 시간적인 면에서 어느 정도 효율성이 높은지를 분석하는 연구와 새로운 응용 영역으로 이 역할 배정 방식을 확장하는 연구가 있다.

## 참 고 문 헌


[1] Anthony Chavez, Alexandros Moukas, Pattie Maes, "A Multi-agent System for Distributed Resource Allocation," Proceedings of the First International Conference on Autonomous Agents (Agents'97), 1997.

[2] Dajun Zeng Katia P. Sycara, "Generic Negotiator," Preliminary Report, 1994.

[3] Frank Dignum, "Dialogue in Team Formation : A Formal Approach," Foundations and Applications of Collective Agent Based Systems (CABS), ESSLLI99 workshop, 1999.

[4] F.M.T. Brazier, P.A.T. van Eck, J. Treur, "Modeling Competitive Co-operation of Agents in a Compositional Multi-Agent Framework," Knowledge Acquisition, Modeling and Management, 1997.

[5] Gal A. Kaminka, et al, "GameBots : A Flexible Test Bed for MultiAgent Team Research," Communications of ACM, Vol.45 , No.1, pp.43-45, 2002.

[6] Gehard Weiss, Multiagent Systems, MIT Press, 1999.

[7] Luiz Chaimowicz, Mario F. M. Campos, Vijay Kumar, "Dynamic Role Assignment for Cooperative Robots," 2002.

[8] J. E. Doran, S. Franklin, N. R. Jennings T. J. Norman, "On Cooperation in Multi-Agent Systems," Dept. of Computer The Knowledge Engineering Review, 1997.

[9] Mehdi Dastani, Virginia Dignum, Frank Dignum, "Role-Assignment in Open Agent Societies." 2000.

[10] Michael S. Miller, Richard A. Volz, "Training for Teamwork"" 2001.

[11] Mihaela Oprea, "An Adaptive Negotiation Model for Agent-Based Electronic Commerce," 2000.

[12] Milind Tambe, "Teamwork in Real-World Dynamic Environments," ICMAS-96, pp. 361-368, 1996.

[13] Milind Tambe, "Towards Flexible Teamwork," Journal of Artificial Intelligence Research, 1997.

[14] Milind Tambe, Jafar Adibi, Yaser Al-Onaizan, Ali Erdem Gal A. Kaminka, Stacy C. Marsella, Ion Muslea, "Building Agent Teams Using an Explicit Teamwork Model and Learning," Artificial Intelligence, 1998.

[15] Milind Tambe, Wei-Min Shen, Maja Mataric, Dani Goldberg, Pragnesh Jay Modi, Zhun Qiu, Behnam Salemi, "Teamwork in Cyberspace: Using TEAMCORE to Make Agents Team- Ready", 2000.

[16] Onn M. Shehory, Katia Sycara, Somesh Jha, "Multi-Agent Coordination through Coalition Formation," 1998.

[17] Onn Shehory, Sarit Kraus, "Task Allocation via Coalition Formation among Autonomous Agents," Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95), 1995.

[18] Paolo Giorgini, Manuel Kolp, "Multi-Agent Architectures as Organizational Structures," 2001.

[19] Peter R. Wurman, Michael P. Wellman, William E. Walsh, "The Michigan Internet AuctionBot: A Configurable Auction Server for Human and Software Agents," Proceedings of the 2nd International Conference on Autonomous Agents (Agents '98), 1998.

[20] Peter Stone, Layered Learning in Multiagent Systems, MIT Press, 2000.

[21] P. Busetta, L. Serafini, et al, "Extending

Multi- Agent Cooperation by Overhearing," Lecture Notes in Computer Science, 2001.

[22] Ranjit Nair, Milind Tambe, Stacy Marsella, "Role Allocation and Reallocation in Multiagent Teams: Towards A Practical Analysis," 2003.

[23] R.Schwartz and S.Kraus, "Negotiation on Data Allocation in Multi-agent Environments," Proceedings of AAAI-97, 1997.

[24] Sarit Kraus, "Negotiation and Cooperation in Multi-Agent Environments," Artificial Intelligence journal, Special Issue On Economic Principles of Multi-Agent System, Vol. 94, No.1-2, pp. 79-98, 1997.

[25] Van Parunak, "Characterizing Multi-Agent Negotiation," Proceedings of IWMAS'98, 1998.

[26] William E. Walsh, Michael P. Wellman, "Modeling Supply Chain Formation in Multiagent Systems," Agent Mediated Electronic Commerce (IJCAI Workshop), 1999.

[27] William van der Sterren, "Squad Tactics : Team AI and Emergent Maneuvers," In AI Game Programming Wisdom, Charles River Media Inc, 2002.

## ◐ 저 자 소 개 ◑

**김 인 철 (Kim, Incheol)**
1985년 서울대학교 수학과 졸업(학사)
1987년 서울대학교 대학원 계산통계학과 졸업(석사)
1995년 서울대학교 대학원 계산통계학과 졸업(박사)
1996년~현재 경기대학교 전자계산학과 교수
관심분야 : 인공지능, 지능형 에이전트, 계획 및 학습
E-mail : kic@kyonggi.ac.kr