논문 2006-43CI-2-5

게임NPC지능 개발을 위한 부하분산과 그룹 행동을 지원하는 유연한 플랫폼 구조

(Flexible Development Architecture for Game NPC Intelligence to Support Load Sharing and Group Behavior)

임 차 섭*, 김 태 용**

(Cha-Seop Im and Tae Yong Kim)

요 약

최근 컴퓨터 게임은 점점 복잡해지며 게임 이용자들은 컴퓨터에 의해 행동하는 NPC들이 보다 사실적이며 세련되길 원하기 때문에 게임NPC 개발자들은 인공지능 측면에서 보다 많은 노력을 기울일 필요가 있다. 이에 따라, 게임 NPC 지능 개발을 위한 플랫폼은 보다 사실적이며 반응적이고 쉬운 NPC 개발을 위해 실시간, 독립성, 유연성, 그룹 행동을 비롯한 다양한 인공지능을 지원해야 한다. 본 논문에서는 이전 플랫폼들의 문제점들을 알아보고, 해결하기 위한 게임 NPC 지능 개발 플랫폼의 구조를 제안한다. 제안하는 플랫폼은 4개의 모듈로 구성되며, 부하분산을 통해 기존 플랫폼들보다 높은 성능을 보여주며, 각 모듈을 통해 다양한 인공지능 기법 지원, 효율적인 그룹 행동, 다양한 게임 환경에서 독립적인 NPC 개발과 같은 장점들을 가진다.

Abstract

As computer games become more complex and consumers demand more sophisticated computer controlled NPCs, developers are required to place a greater emphasis on the artificial intelligence aspects for their games. The platform for game NPC Intelligence Development should support real-time, independence, flexibility, group behavior, and various A.I to NPC that are reactive, realistic and easy to develop. This paper presents an architecture to satisfy these criteria for the platform of game NPC intelligence development. The proposed platform shows the higher performance than existing platform through the load sharing, and it also has some advantages which are supporting the various AI techniques, efficient group behavior, and independence to develop NPC intelligence.

Keywords : NPC, 지능 개발 플랫폼(Intelligence Development Platform), 그룹행동(Group Behavior)

I. 서 론

오늘날 컴퓨터 게임에서 NPC(Non Player Characters)들은 게임 이용자와 복잡한 상호작용을 하며 게임 이용자들은 보다 사실적이며 합리적인 NPC의 행동을 기대한다. 이런 NPC들은 단순한 행동 패턴만을 가진 유닛을 비롯해 전략, 전술을 구사하거나 게임 이용자와의 협동과 같은 다양한 역할을 수행하게 된다[1]. 특히, 다수의 NPC들을 사용하는 최근 게임들에 있어 NPC들의 사실적인 행동을 위해서는 개별 NPC들의 행동뿐만 아니라, NPC들의 그룹을 형성하여 지능을 수행할 필요가 있다[2]. 예를 들어, 축구, 농구등 팀을 이루는 스포츠 게임이나, 많은 수의 NPC가 필요한 MMORPG, 전략게임들에 있어 개별적인 NPC행동뿐만 아니라, 그룹행동을 위한 지능을 설계하는 것이 더욱 사실적으로 보이며 효율적이다. 표 1.은 게임 장르별로 전략, 전술적인 상대, 파트너, 유닛, 지원NPC처럼 NPC의 다양한 역할과 이를 구현하기 위한 스크립팅, 유한 상태 머신,

* 학생회원, ** 정회원, 중앙대학교 영상공학과
(Department of Image Engineering The GSAIM, Chung-Ang University)
※ 본 연구는 중소기업청의 산학연 공동기술개발 컨소시엄사업과 서울시 산학연 협력사업의 지원으로 수행되었음.
접수일자: 2005년11월28일, 수정완료일: 2006년3월3일

표 1. 게임 장르별 지능 NPC의 역할과 유용한 인공지능 기법
Table 1. Different roles and useful AI techniques of intelligence NPCs with each game genre.

장르	지능 NPC의 역할	유용한 인공지능 기법
일인칭 슈팅	전술적 적NPC, 파트너	스크립팅, 유한 상태 머신, 퍼지 논리
롤플레잉	전술적 적NPC, 파트너, 지원 캐릭터	유한 상태 머신, 스크립팅, 퍼지 논리
어드벤처	전술적 적NPC, 파트너, 지원 캐릭터	스크립팅, 유한 상태 머신, 퍼지 논리
전략	전략적 적NPC, 유닛(Units)	유한 상태 머신, 스크립팅, 신경망, 유전자 알고리즘
갓(God) 게임	유닛(Units)	신경망, 유전자 알고리즘
스포츠	전략적 상대, 전술적 적NPC, 유닛(Units)	신경망, 유전자 알고리즘, 유한 상태 머신

퍼지 논리, 신경망, 유전자 알고리즘과 같은 유용한 인공지능 기법을 보여준다[1][3].

일반적으로 게임NPC 지능은 독립적인 개발 플랫폼 없이 게임 환경에 종속적으로 개발되기 때문에, 게임 환경과의 독립성 보장과 병렬적인 개발이 어려우며, 다른 게임 환경에서의 지능 NPC의 재사용이 힘든 단점들을 가진다. 또한, 게임NPC 지능 개발을 위해서는 인공지능을 표현할 수 있는 기법 개발이 선행되어야 하기 때문에, 효율적으로 게임NPC 지능을 개발하기 어렵게 된다. 이러한 문제점들을 해결하기 위해 게임 NPC 지능 개발 플랫폼은 기본적으로 독립적인 게임 환경과 반응하며 사실적인 게임 NPC를 개발하기 쉬운 환경을 제공해야 한다[4].

본 논문에서는 기존 플랫폼 연구에서의 문제점들을 알아보고 이를 해결할 수 있는 플랫폼의 구조를 제안한다. 또한, 제안하는 플랫폼 구조를 바탕으로 플랫폼을 구현하여 기존 플랫폼들과의 성능을 비교, 평가한다.

II. 관련 연구

표2에서 보여주듯 최근의 상업적 게임들은 규칙 기반 시스템, 유한 상태 머신, 결정 트리뿐만 아니라, 보다 사실적인 NPC의 행동을 개발하기 위해 퍼지 논리, 신경망, 유전자 알고리즘와 같은 학술적이며 진보적인 인공지능 기법들을 사용하고 있음을 알 수 있다.

표 2. 상업적 게임에서의 인공지능 기법
Table 2. Summary of AI techniques in Commercial Games(NN : Neural Network, A-Life : Artificial Life, FuSM : Fuzzy Logic State Machine, GA: Genetic Algorithm, RBS: Rule-Based System).

게임	장르	인공지능
Age of Empire	전략	FSM
Baldur's Gate	롤플레잉	RBS
Battlecruise:3000AD	전략	NN, Fuzzy logic
Black & White	God	Decision Tree, NN
Creatures	God	GA, NN
Close Combat 2	전략	FuSM
Dirt Track Racing	액션	NN
Fx Fighter	파이팅	RBS
Return Fire II	액션	GA
Sigma	전략	GA
The sims	God	A-Life, FuSM
S.W.A.T 2	전략	Fuzzy logic
Ultima Online	롤플레잉	A-Life
Unreal	일인칭 슈팅	A-Life
Virtual Fighter 2	파이팅	RBS
Half-Life	일인칭 슈팅	A-Life, FSM

다양한 장르의 게임들은 다양한 형태의 입출력을 필요로 한다. 아케이드 스타일의 게임들은 매우 제한적인 입력을 가지는 반면, 실시간 전략 스타일의 게임들은 지형 탐색, 대형 유지, 명령 수행, 적 공격에 대한 반응 등을 위해 다양한 형태의 입력을 가지게 된다. 출력의 경우, 이산적인 형태의 명령, 연속적 명령, 하나의 NPC에 대한 명령이나 여러 NPC의 행동을 위한 명령등 여러가지 형태로 나타날 수 있다.

기존 연구에서 인공지능 테스트 연구[2][3]와 멀티 에이전트 연구[5]를 위한 환경으로서 여러 가지 플랫폼이 제안되었으며, 이와 같은 플랫폼들은 여러 장점들을 가지지만 아래와 같은 여러 가지 문제점들을 해결하지 못한다.

Quakebot[6]는 일인칭 슈팅 게임인 Quake 2게임 이용자와 맞서서 행동 하도록 설계된 에이전트로 Soar[3] 환경에서 개발되었으며, 지식과 행동을 정의하기 위해 동적 계층 작업 분할 기법을 사용하여 예상, 학습할 수 있

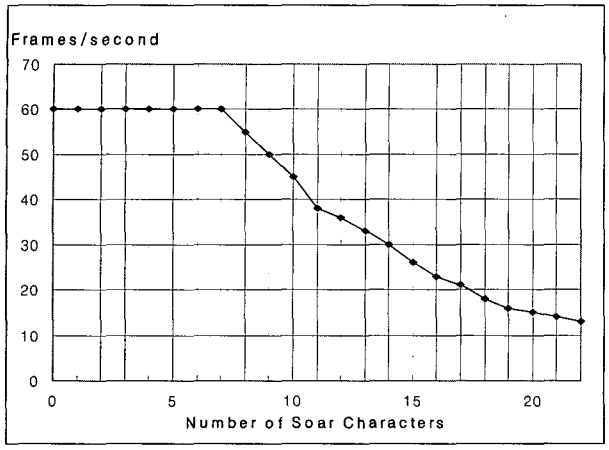

그림 1. Soar 캐릭터 수에 따른 게임 객체 갱신율
Fig. 1. Game update rate as number of Soar characters.

는 능력을 가진다[7]. 하지만, Quakebot 에이전트는 현재까지 싱글 에이전트만을사용할 수 있는 제한을 가진다. 이처럼, Soar는 게임서버에서 많은 수의 규칙이 정의된 복잡한 NPC를 동작시킬 경우 게임 서버의 부하를 증가시키기 때문에 멀티 NPC 지원이 어려우며, 이를 기반으로 하는 그룹 행동 또한 지원하지 못한다. 그림 1의 그래프는 Soar와 언리얼 토너먼트(UT) 게임 서버간 연결에서 성능을 보여준다. 각 인공지능 캐릭터는 최소한의 처리만을 하는 간단한 에이전트로써, 초기에는 일정 수준이상의 fps(Frames Per Second)를 유지하지만, 인공지능 캐릭터의 수가 늘어남에 따라 성능이 떨어지며, 20개 이상의 캐릭터를 동작시킬 경우, 게임 서버의 성능이 10fps 이하로 정상적인 게임 플레이를 할 수 없게 된다. 또한, Soar는 규칙 기반 시스템을 기본으로 하는 전문가 시스템이기 때문에 수식 표현을 비롯한 퍼지 논리, 신경망, 유전자 알고리즘과 같은 다양한 인공지능 기법을 제공하지 못한다.

Gamebots[5]는 멀티 에이전트 시스템으로써, 여러 에이전트를 이용할 수 있으며, 다양한 게임 모드를 지원하지만, UT만을 게임 서버로 지원할 뿐 게임 환경과 독립성은 지원하지 못한다. 또한, gamebots는 gamemodule을 통해 FPS(First Person Shooting)게임인 UT와 에이전트간의 프로토콜만을 정의하기 때문에, 입출력 형태가 다른 장르의 게임에서 다양한 인공지능 기법을 이용한 다양한 역할의 지능 NPC를 개발하기 어려우며, 게임 서버를 통한 그룹행동만을 지원할 뿐, 플랫폼이 게임 서버와 독립적으로 그룹행동을 지원하지는 못하기 때문에 그룹행동의 재사용성이 떨어진다.

FEAR[2]는 다양한 인공지능 모듈들과 게임 NPC DLL을 개발하기 위한 유연한 구조를 제공한다. 하지만, NPC DLL 호출 루틴은 게임 서버의 서브 루틴으로 게임 서버와 NPC DLL간 높은결합도를 보인다. 이와 같은 방식의 NPC DLL은 게임 서버의 자원을 선점하여 게임 서버의 부하를 증가시키며, 특히 게임 NPC의 처리시간이 게임 서버가 업데이트하는 주기보다 긴 경우, 게임 서버는 실시간성을 지원하지 못한다. 또한, 높은 결합도로 인해 FEAR는 게임 환경과의 독립성을 제공하기 어려운 구조를 가지며, 플랫폼을 통한 독립적인 그룹행동 개발에도 어려움을 가진다.

Ⅲ. 유연한 구조를 위한 요구사항

다양한 게임 장르에서의 게임 NPC의 다양한 역할들을 정의 할 수 있고, 이전 연구들에 있어 문제점으로 지적된 점들을 해결하기 위해서는 다음과 같은 요구사항을 만족해야 한다.

- 실시간성 : NPC가 게임과 상호작용하기 위해서 플랫폼은 반드시 실시간성을 보장해야 한다. 이것은 플랫폼이 게임 서버의 성능을 저하 시키지 않고, 게임 서버의 자원을 선점하지 않아야함을 뜻한다. 특히, 사실적인 NPC 개발을 위해 사용되는 신경망 이나 유전자 알고리즘 기법은 많은 CPU-time을 요구한다. 이런 경우에 있어서도 플랫폼은 게임 환경의 실시간성을 보장해야 한다.
- 독립성: 사실적인 NPC와 다양한 게임 환경을 지원하기 위해 플랫폼은 게임환경과의 독립성을 지원해야 한다. 독립성은 사실성을 위해 게임 이용자가 이용할 수 있는 정보 수준의 제한을 두게 하며, 게임 NPC 지능을 게임 환경과 독립적으로 개발 할 수 있게 한다. 이처럼, 게임환경과 플랫폼간의 낮은 결합도를 통해 지능 개발자는 게임 NPC의 재사용율을 높이며, 게임 환경에 상호작용하기 위한 노력을 줄일 수 있다.
- 그룹 행동 : 플랫폼은 그룹 행동을 쉽고 간단하게 개발할 수 있는 수단과 게임 환경과 효율적인 상호작용 방법을 제공해야 한다. 게임 서버의 팀플레이 지원과 독립적으로, 플랫폼은 게임 NPC들간의 그룹 행동을 지원할 수 있어야 한다. 이것은 NPC가 게임 이용자들이 상호 협력하는 것처럼 보다 사실적으로 보이게 한다. 또한, 그룹행동 수행 시 발생하는 게임 서버의 부하를 줄일 수 있어야 한다.
- 다양한 인공지능 기법 : 플랫폼은 다양하고 사실적인 NPC를 위해 규칙 기반 시스템, 유한 상태 머신, 퍼

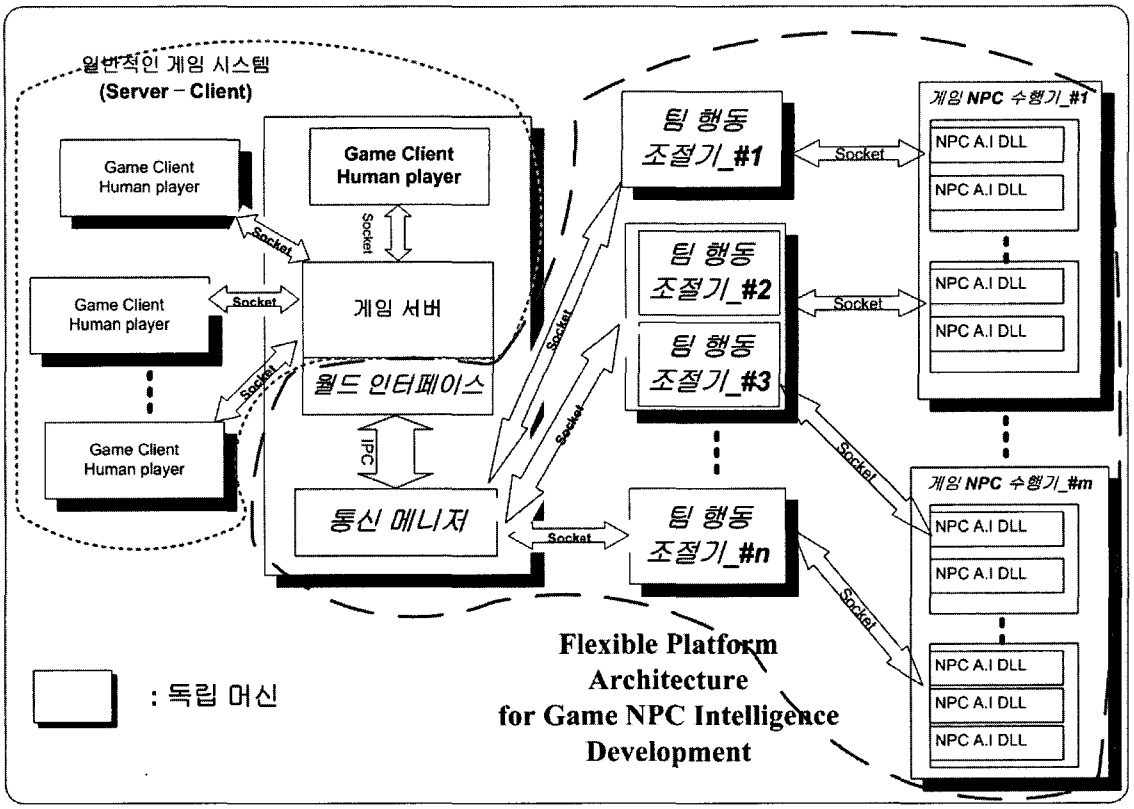

그림 2. FPAGNID(Flexible Platform Architecture for Game NPC Intelligence Development) 전체 구조
Fig. 2. Architecture of FPAGNID.

지 논리, 신경망, 유전자 알고리즘등과 같은 다양한 인공지능 모듈들과 이를 효율적으로 구성할 수 있는 환경을 지원해야 한다. 이러한 모듈들은 게임 NPC 지능 개발을 보다 쉽게 해주는 수단을 제공한다.

Ⅳ. 게임 NPC 지능 개발을 위한 유연한 플랫폼 구조

본 절에서는 게임 NPC 지능 개발의 효율성을 높여주는 FPAGNID(Flexible Platform Architecture for Game NPC Intelligence Development)의 모듈 구성과 3절에서 언급한 요구사항들을 만족시키는 방법에 대해 설명한다.

1. 모듈의 구성

그림 2는 FPGNID의 전체 구조를 보여주며, 제안하는 플랫폼 구조 FPAGNID는 월드 인터페이스, 통신 메니저, 팀 지능 조절기, 게임 NPC 지능 수행기로 이루어진다. 각 모듈은 기능별로 독립적으로 구성되며, 각 모듈은 수행 결과를 이웃 모듈에 전달함으로써 게임 환경에서 게임 NPC의 지능 행동을 수행하게 된다.

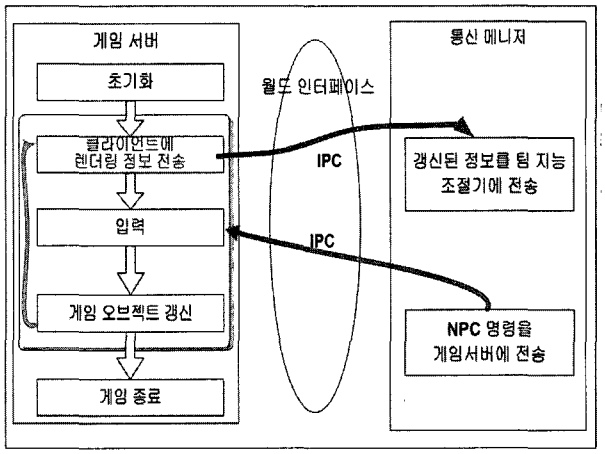

그림 3. 월드 인터페이스의 역할
Fig. 3. Role of World Interface.

가. 월드 인터페이스

월드 인터페이스는 NPC의 행동을 게임 서버에 적용시키고, 게임 환경정보를 게임NPC에 전달하기 위해 상호간 데이터 형식을 전환하는 역할을 한다. 그림3 에서 보듯이 일반적으로 게임 서버는 종료되기 전까지 렌더링 정보를 클라이언트들에 보내며, 입력을 처리한 후 게임 객체의 정보 갱신을 반복하게 된다. 월드 인터페

표 3. FPAGNID의 입출력 데이터 구조 예. a)적 인지 정보, b)NPC 명령 정보

Table 3. Example of FPAGNID's I/O data architecture. a) Enemy sensory information, b)command of a NPC

a)

적 인지 정보	값 속성(자료형)
name	npc_id(int), name(string)
health	npc_id(int), health(int)
velocity	npc_id(int), x(float), y(float), z(float)
weapon	weapon_id(int)
distance	npc_id(int), x(float), y(float), z(float)

b)

명령이름	값 속성(자료형)
moveto	x(float), y(float), z(float)
turnto	pitch(float),yaw(float), roll(float)
use_item	item_id(int)
shoot	count(int)
change_weapon	weapon_id(int)

이스는 클라이언트에 보내는 렌더링 정보에 상응하는 정보를 게임NPC에 적용시키기 위한 데이터 형식으로 전환하여 통신 메니저에 전달하며, 통신 메니저를 통해 전달 받은 게임 NPC의 명령 정보를 게임 서버에 종속적인 입력형태로 데이터를 전환하게 된다. 데이터 형식의 상호교환을 위해 FPAGNID에서 사용하는 데이터 구조는 데이터 헤더와 데이터 값으로 구성된다. 데이터 헤더는 NPC의 ID와 데이터를 송신하는 모듈의 ID를 포함하며 데이터 값은 표 3처럼 명령이름과 값의 속성을 쌍으로 하는 목록으로 이루어진다. 이것은 NPC의 명령뿐만 아니라, 게임 서버에 의해 전송된 게임 환경 정보도 포함하며 필요에 따라 명령을 추가할 수 있기 때문에 입출력이 다른 다양한 게임 환경을 지원하기 유리한 구조를 지닌다. 게임 환경에 따라 적합한 전환 규칙을 월드인터페이스에 정의함으로써 게임 NPC는 게임 환경과 독립적으로 설계, 구현할 수 있게 된다. 또한, 월드 인터페이스는 IPC(Inter Process Communication) 기법을 통해 통신 메니저와 통신하므로, 소켓 보다 낮은 부하를 발생시키며 빠른 처리에 용이하다.

나. 통신 메니저

통신 메니저는 월드 인터페이스가 포함된 여러 게임 서버와 다수의 팀 지능 조절기의 연결을 관리하여 게임 서버와 팀 지능 조절기에서 전달받은 데이터를 적합한 게임 서버와 팀 지능 조절기에 전달하게 된다. 이를 위해 통신 메니저는 다음과 같은 과정을 통해 각 NPC별 맵핑 테이블을 생성하고, 이용한다.

1. 게임 서버는 통신 메니저와 연결시 통신 메니저에 게임 서버 ID 전송.
2. 통신 메니저는 연결된 게임 서버 목록에 새로운 게임 서버 ID 저장 및 팀 지능 조절기에 전달.
3. 팀 지능 조절기는 게임 NPC 수행기에 전송.
4. 게임 NPC 수행기는 NPC 등록시 NPC가 동작할 게임 서버 ID와 NPC ID를 데이터 헤더에 포함 시킨 후 팀 지능 조절기에 전달.
5. 팀 지능 조절기는 데이터 헤더에 팀 지능 조절기의 ID를 추가한 후, 통신 메니저에 전송.
6. 통신 메니저는 맵핑 테이블에 <NPC ID, 팀 지능 조절기 ID, 게임 서버 ID> 정보 추가

다. 팀 지능 조절기

FPAGNID에서 효율적인 그룹행동을 지원하기 위해 팀 지능 조절기가 사용된다. 기본적으로 팀 지능 조절기는 NPC 정보와 게임 환경 정보를 관리 및 전송하는 역할을 하며, 팀 단위의 행동을 결정한다. 같은 팀 지능 조절기에 연결된 NPC들은 하나의 팀을 이루게 되며, 게임 서버에 의해 갱신된 각 NPC의 상태 정보와 각 NPC가 인지한 정보는 팀 지능 조절기에 의해 관리되고, 팀 NPC 들과 공유된다. 또한, 팀 지능 조절기는 게임 NPC 수행기에 의해 갱신된 각 NPC의 명령정보를 통신 메니저에 전달하게 된다.

효율적인 그룹행동을 수행하기 위해 그림 4처럼 그룹행동을 위한 명령을 계층적으로 정의한다. 그룹행동을 위한 명령 체계는 세 개의 층으로 이루어지며, 최상위 레벨은 그룹의 목표를 의미한다. 그룹은 여러 목표를 가질 수 있으며 하나의 목표는 다양한 전략으로 이루어지고, 전략을 수행하기 위해 NPC는 각자 여러 행동을 수행하게 된다. 이와 같은 명령체계를 이용해 팀 지능 조절기는 다음과 같은 과정을 통해 그룹 행동을 수행한다. 그림 4에서 그룹 행동은 세 가지로 구성되며 적을 포획하기 위해 팀 지능 조절기는 여러 전략 중 대형 유지 전략을 택하고, 팀 NPC들에게 이를 전송하게 된다. 팀의 전략을 전달 받은 NPC들은 이를 수행하기 위해 미리 정의된 지능에 따라 자율적으로 행동을 결정하고, 결정된 행동을 팀 지능 조절기에 전달하게 된다. 이 때, 필요에 따라 NPC는 팀 지능 조절기에 팀 NPC

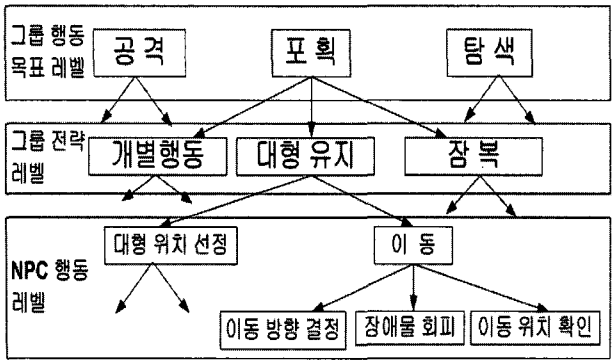

그림 4. 그룹 행동을 위한 계층적 명령 체계의 일부
Fig. 4. A portion of a sample command hierarchy for a group behavior.

의 상태 정보를 질의하고, 팀 지능 조절기는 이에 응답한다.

라. 게임 NPC 수행기

FPAGNID는 NPC DLL을 통해 NPC의 행동을 정의하며, NPC는 게임 NPC 수행기를 통해 다양한 인공지능 기법을 이용할 수 있다. 게임 NPC 수행기는 다수의 NPC DLL을 실행하고, 그 결과인 NPC의 명령정보를 팀 지능 조절기에 전달하기 위해 멀티 쓰레드 기법을 이용한다. 멀티 쓰레드 기법을 통해 게임 NPC 수행기는 다수의 NPC DLL을 병렬적으로 실행시킨다.

게임 NPC 수행기는 게임 환경과 그룹행동 전략에 동적으로 반응하기 위해 팀 지능 조절기에서 전달받은 게임 환경 정보나 그룹행동을 위한 명령정보 혹은 팀 정보를 해당 NPC에 전달하여 게임 NPC DLL이 이를 이용할 수 있게 한다. 또한, NPC의 행동을 결정, 적용하기 위해 게임NPC 수행기는 동기적 메시지와 비동기적 메시지를 팀 지능 조절기 혹은 게임서버에 전송하고, 팀 지능 조절기 와 게임서버로부터 전송 받는다. 게임 서버는 일정 주기마다 게임 객체들의 상태를 갱신하며, 게임NPC 수행기는 갱신된 NPC 상태정보를 동기적으로 전송 받으며, NPC 명령정보를 게임 서버와 동기적으로 전송해야 하는 반면, 게임 환경 정보 질의나 팀 정보 질의와 같은 메시지는 게임서버와 비동기적으로 전송하고, 게임 환경 정보나 팀 정보를 비동기적으로 전송 받는다.

2. 유연한 구조

4개의 모듈로 구성된 FPAGNID는 독립적인 머신에서 동작 가능한 유연한 구조를 가진다. 유연한 구조를 통해 그룹 행동 조절기와 게임NPC 실행기는 게임 서버

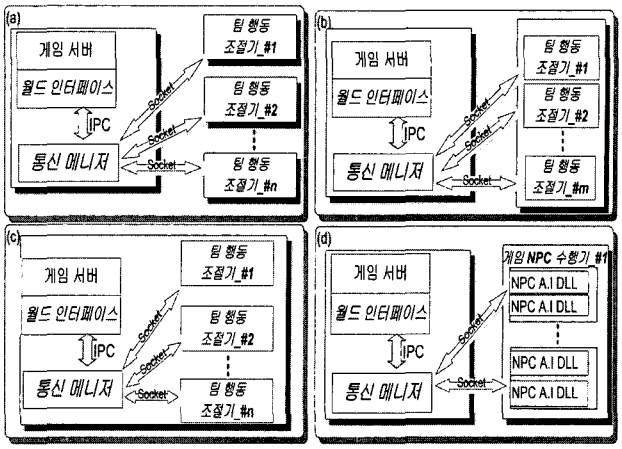

그림 5. 유연한 구조를 통한 다양한 연결
Fig. 5. various connection via flexible architecture.

와 독립된 머신에서 동작시킬 수 있으며, 필요에 따라 하나이상의 모듈을 연결하거나 같은 머신에서 동작시킬 수 있다. 예를 들어, 복잡한 여러 팀들을 효율적으로 수행하기 위해 그림 5의 (a)와 같이 여러 머신에 분산하여 팀 지능 조절기를 동작시킬 수 있으며, 단순한 팀들일 경우엔 (b)처럼 하나의 머신에서 여러 개의 팀 지능 조절기를 수행시키거나 (c)처럼 게임 서버와 같은 머신에서 동작 시킬 수 있으며, 그룹 행동이 필요하지 않는 경우 (d)와 같이 통신 메니저와 게임NPC 수행기를 바로 연결 할 수 있다.

각 기능별로 나누어진 모듈은 결합도를 낮게 유지하기 때문에, 모든 기능을 하나의 모듈에 정의한 Gamebots[2]의 Gamemodule 에 비해 많은 장점을 가진다. 낮은 결합도는 모듈의 재사용성을 높이며, 유지, 보수를 용이하게 한다. 즉, 다양한 게임에 적용하기 위해서 단지 월드인터페이스만을 수정하거나, 새로운 팀을 위해 팀 지능 조절기를 추가하는등 다른 모듈과 상관없이 필요 모듈에 관한 작업만을 수행할 수 있다. 또한, 결합도가 낮은 유연한 구조는 실시간성, 독립성, 그룹행동 및 다양한 인공지능 기법들을 지원함에 있어 여러가지 장점을 지닌다.

가. 실시간

실시간을 유지하기 위해 플랫폼에서 게임 NPC들을 실행시킬 때 발생하는 부하를 분산시키는 것은 중요하다. Laird[8]는 더 많은 NPC를 실행시키기 위해서는 다른 머신에 부하를 분산시켜야 함을 지적한다. FPAGNID는 각 모듈이 독립적인 머신에서 동작 가능하여 부하를 분산하고, 자원을 선점하지 않는 멀티 쓰레드 기법을 통해 실시간성을 보장한다.

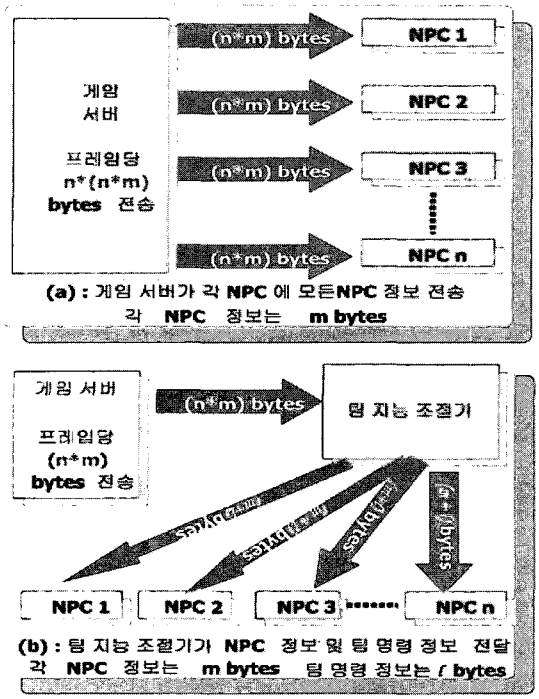

그림 6. 그룹 행동에 필요한 네트워크 패킷 비교 (a) 일반적인 플랫폼, (b) FPAGNID

Fig. 6. Required packet size for group behavior in general platform and FPAGNID. (a) general platform, (b) FPAGNID

그룹행동을 결정하기 위해서 각 NPC는 그룹으로 정의된 NPC들의 정보들이 필요하며, 이것은 일반적으로 게임서버에 추가적인 부하를 주게 된다. 실시간성을 만족하면서 그룹 행동을 지원하기 위해서는 그룹 행동을 위한 추가적 부하를 낮게 유지할 필요가 있다. FPAGNID는 팀 지능 조절기를 통해 게임 서버에 추가적인 부하를 주지 않으면서 각 NPC에 팀 명령과 NPC 정보를 전달할 수 있다. 그림 6의 (a)는 일반적인 플랫폼에서 그룹행동을 위해 게임 서버가 각 NPC에 n개의 팀 NPC들의 상태정보 m bytes를 전달할 때 필요한 네트워크 대역폭은 $n\times(n\times m)bytes$임을 나타내며, 이러한 추가적인 부하는 NPC의 수가 늘어남에 따라 지수적으로 증가하는 반면, (b)에서 처럼 FPAGNID에서 필요한 전체 네트워크 대역폭은 $((n\times m)+(n\times(m+l)))bytes$ 임을 나타내며, 이것은 NPC의 수 n이 늘어남에 따라 선형적으로 증가함을 알 수 있다. 일반적으로 명령 정보의 크기는 NPC의 상태정보 크기보다 작기 때문에 각 NPC에 모든 NPC의 상태 정보를 전달하는 방식보다 네트워크 대역폭을 효율적으로 사용한다.

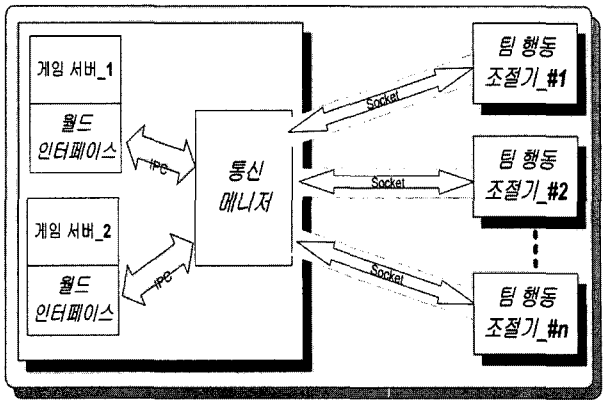

그림 7. 통신 메니저와 월드 인터페이스를 이용한 다수의 게임 서버 연결

Fig. 7. Connection with multi-game server using Communication Manager and World Interface.

나. 독립성

FPAGNID는 다양한 게임 환경과 사실적인 NPC를 지원하기 위해 월드 인터페이스와 통신 메니저를 통해 독립성을 제공한다. 게임 환경에 따라 월드 인터페이스를 정의해줌에 따라, 게임 환경과의 독립성을 유지할 수 있으며 게임 서버에 의해 게임 이용자에게 전송하는 데이터와 상응하는 데이터만을 이용하기 때문에 NPC는 게임 이용자와 같은 제한적인 정보만을 이용할 수 있으며, 이것은 사실적인 NPC 구현을 보장한다.

게임 환경과의 독립성을 통해 그림7과 같이 둘 이상의 게임과 연결이 가능하다. 게임 환경에 적합한 데이터로 전환하는 월드 인터페이스와 팀 지능 조절기와 여러 게임 환경과의 연결을 보장하는 통신 메니저를 통해 하나 이상의 게임 환경과 연결가능하기 때문에 하나의 NPC를 여러 장르의 게임 환경에서 테스트할 수 있는 장점을 가진다.

다. 그룹 행동

최근 게임이나 멀티 에이전트 시스템에 있어 그룹 행동은 점점 중요해지고 있다. 하지만, 그룹 행동은 다른 NPC와 의사소통하기 위한 부가적인 정보가 필요하기 때문에 게임 서버의 부하를 증가시킬 수 있다. 게임 서

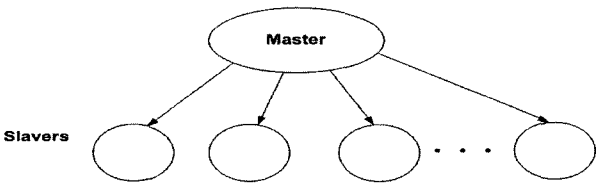

그림 8. Master-slave parallel GA의 구조

Fig. 8. Architecture of Master-slave parallel GA.

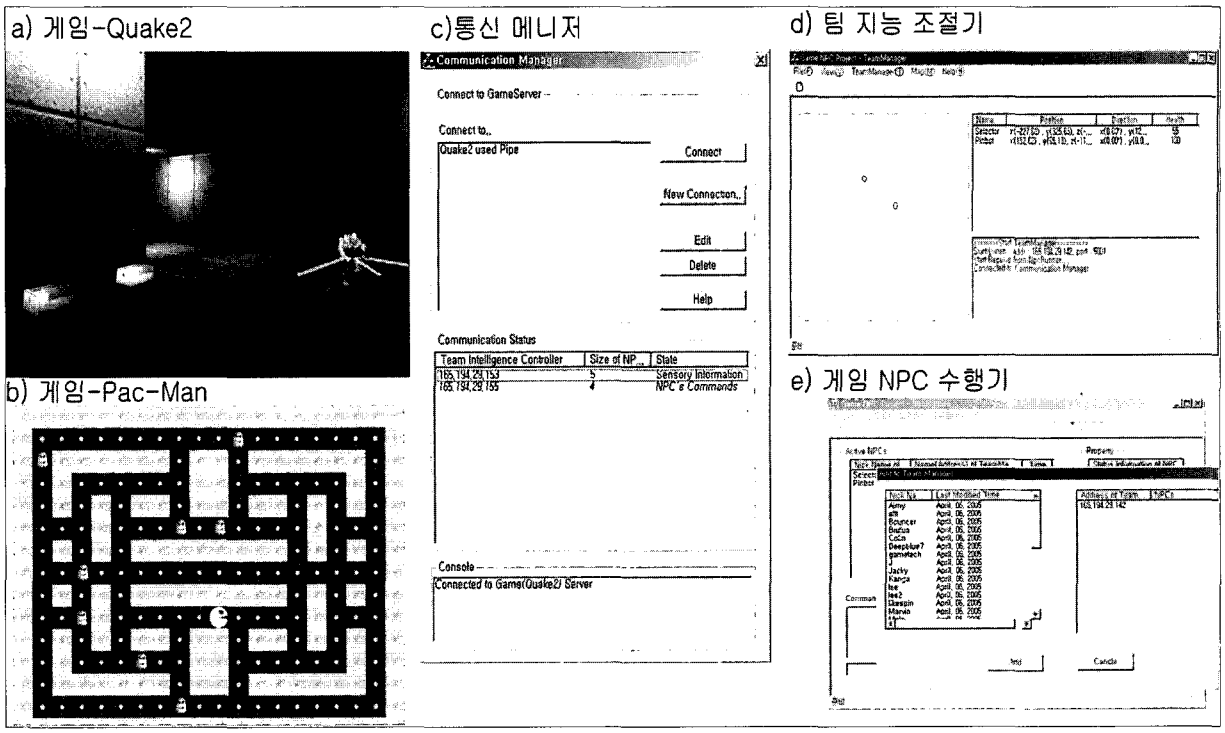

그림 9. 각 모듈과 실험에 사용된 게임들의 스크린 샷. a) Game-Quake2, b) Game-Pac-Man c) 통신 메니저, d) 팀 지능 조절기, e) 게임 NPC 수행기

Fig. 9. Screen-shots of FPAGNID's Modules and games used in experiment. a) Game-Quake2, b) Game-Pac-Man c) Com-munication Manager, d) Team Intelligence Controller, e) Game NPC Runner

버의 부하를 증가시키지 않으며 그룹행동을 지원하기 위해 FPAGNID는 팀 지능 조절기를 사용한다.

FPAGNID는 팀 지능 조절기를 통해 그룹 명령 전달 방식뿐만 아니라, 여러 방법으로 그룹 행동을 정의할 수 있는 유연한 구조를 보여준다. 팀 지능 조절기를 칠판 아키텍처의 칠판으로 사용하여 칠판 아키텍처로 이용할 수 있으며, Master-Slave GA[9] 구조로 이용할 수 있다. 예를 들어, 칠판 아키텍처처럼 팀 지능 조절기는 갱신된 중심 위치, 평균 속도, 리더 NPC의 상태 및 팀 NPC들의 포메이션 정보 등을 포함하는 NPC들의 상태 정보를 유지하고, 필요에 따라 게임 NPC 수행기는 팀 지능 조절기를 통해 팀의 정보를 이용하여 팀 행동을 결정할 수 있다.

그림 8은 Master-Slave 병렬 유전자 알고리즘을 위한 구조를 보여준다. Master는 개체군을 저장하며 유전자 알고리즘 연산자를 수행하고, 개체를 각 slave들에게 분산시킨다. Slave들은 개체의 적합도만을 수행한다. 일반적으로, 병렬 유전자 알고리즘 기법들은 실제로 높은 성능을 보장하며 구현하기 쉬운 장점을 가진다[9]. Master-Slave GA는 FPAGNID에서 팀 지능 조절기는 Master로, 게임 NPC 수행기는 Slave로 이용할 수 있기 때문에 팀 지능 조절기-게임 NPC 수행기 구조에 적용하기 적합하다.

라. 다양한 인공지능 기법

다양하고 사실적인 NPC를 쉽게 개발하기 위해 플랫폼은 다양한 인공지능 기법을 지원해야 한다. 게임 NPC 수행기는 게임 NPC의 지능이 정의된 NPC DLL들을 멀티 쓰레드 기법을 통해 수행시킬 뿐만 아니라, 다양한 인공지능 기법과 이를 이용할 수 있는 인터페이스 모듈, 유연한 구조를 통해 게임 NPC DLL을 개발할 수 있는 환경을 제공한다. 인공지능 기법 모듈은 결정 트리, 퍼지 논리, 규칙 기반 시스템, 유한 상태 머신, 유전자 알고리즘, 신경망을 포함한다. NPC는 지능을 결정하기 위해 인공지능 기법 모듈들을 동적으로 로딩할 수 있으며, NPC의 구조는 XML을 통해 계층적으로 정의할 수 있다.

V. 플랫폼 구현 및 평가

모든 모듈은 윈도우즈 환경에서 C++ 언어를 이용하여 구현하였다. 그림 9는 실험에 사용된 게임들-

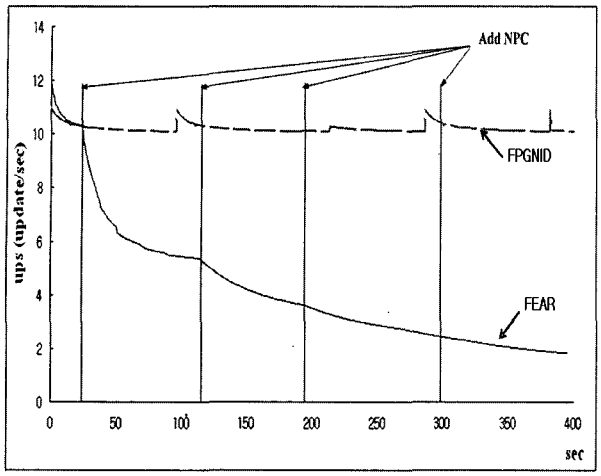

그림 10. 200ms를 소요하는 NPC 지능 처리시 FPGNID와 FEAR의 성능

Fig. 10. Performance of game server with FPGNID and FEAR for NPC that are required 200ms.

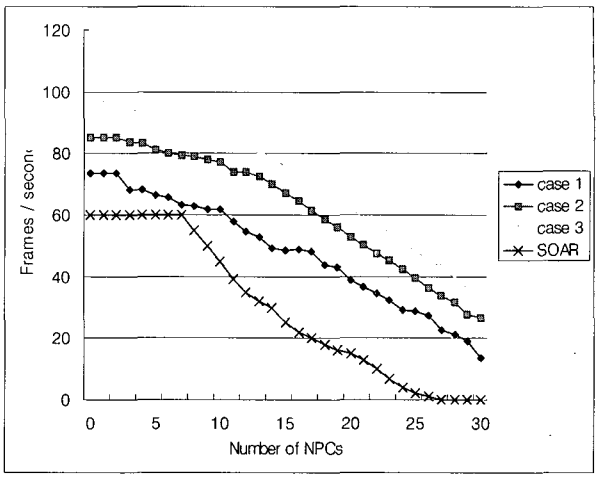

그림 11. 각 경우에서 NPC의 수가 증가함에 따른 초당 프레임 수

Fig. 11. FPS changes for various cases and number of NPCs.

Quake2와 Pac-Man- 과 모듈들의 스크린 샷을 보여준다. 통신 메니저는 연결된 게임서버 정보와 팀 지능 조절기의 정보를 보여주며, 팀 지능 조절기는 팀 NPC의 인지정보를 바탕으로 한 맵 정보와 팀 NPC의 상태 정보를 보여준다. 게임 NPC 수행기는 사용가능한 NPC DLL 목록과 현재 동작 중인 NPC의 상태를 보여주며 FEAR[2]를 기본으로 구현되었다. 게임 NPC 수행기는 현재 파이썬을 이용한 스크립팅, XML을 이용한 규칙 기반 시스템, 퍼지 논리, 유한 상태기, 신경망, 유전자 알고리즘을 DLL을 통해 NPC DLL이 사용할 수 있도록 지원하고 있다. 본 실험에 사용된 게임 서버로 상업용 FPS 게임인 Quake 2와 Dxframeworks를 기반으로 한 Pac-Man이 사용되었다. 본 실험에 사용된 게임 NPC 지능들은 게임 NPC 수행기 환경에서 DLL로 구현되었다.

실시간 게임에 있어 시간 제약은 중요한 요구사항 중 하나이다[10]. 실시간을 보장하지 못할 경우, 게임 이용자는 원활한 게임을 진행할 수 없게 된다. 실시간을 보장하기 위해서 NPC는 게임 서버와 지속적으로 상호작용해야 하며, 게임 서버는 성능을 유지해야 한다.

일반적으로, Quake2 게임 서버는 초당 10번 게임 객체들의 정보를 갱신하고, 클라이언트에 전송한다. 즉, NPC를 포함한 게임 객체들의 정보를 업데이트는 100ms를 주기로 이루어진다. 그림 10은 한 사이클에 200ms를 소요하는 유전자 알고리즘을 이용하여 행동을 학습하는 NPC 지능을 처리하기 위해 FEAR와 FPGNID를 사용할 경우, 게임 서버의 성능을 보여준다. 게임 서버의 자원을 선점하여 게임 NPC의 지능을 갱신하는 FEAR의 경우에는 NPC의 수가 증가함에 따라 갱신율이 떨어져 실시간성을 보장하지 못하는 반면, 기능별 모듈을 통한 부하 분산과 멀티 쓰레드를 통한 기능 수행을 지원하는 FPAGNID의 경우 NPC의 수와 상관없이 게임 서버에서 게임 객체 정보 갱신율을 100ms로 유지하여 실시간을 보장함을 볼 수 있다.

플랫폼을 이루고 있는 모듈의 구성에 따른 부하분산의 결과를 알아보기 위해 각 경우에서 게임 서버의 성능을 측정하였다. 실험에 사용된 NPC는 규칙기반 시스템에 의해 자신의 체력에 따라 시야에 들어오는 NPC를 인식하고 공격하거나 회피하는 지능을 수행한다. 실험은 같은 환경에서 10번 반복 수행되었으며, 그림 11은 각 경우에서 NPC의 수가 증가함에 따른 평균 초당 프레임 수를 보여준다. case 1은 게임 환경과 FPAGNID의 모든 모듈이 같은 머신에서 동작하였을 경우의 결과를 나타내며, case 2는 게임 환경과 통신 메니저, 팀 지능 조절기를 같은 머신에서 동작 시키고, 게임 NPC 수행기만을 다른 머신에서 동작시켰을 경우의 결과를 보여준다. case 3은 게임 환경과 통신 메니저를 하나의 머신에서 동작시키고, 팀 지능 조절기와 게임NPC 수행기를 각각의 머신에서 동작시켰을 경우의 결과를 나타낸다. 실험 결과는 case 1 > case 2 > case 3 > soar[7] 순으로 높은 성능을 보여주었다. 모든 경우에 있어 기존 플랫폼인 soar보다 높은 성능을 보여주며, 각 모듈을 독립적인 머신에서 동작시켜 부하를 분산하였을 경우 더 높은 성능을 보임을 알 수 있다. 이것은 게임 서버가

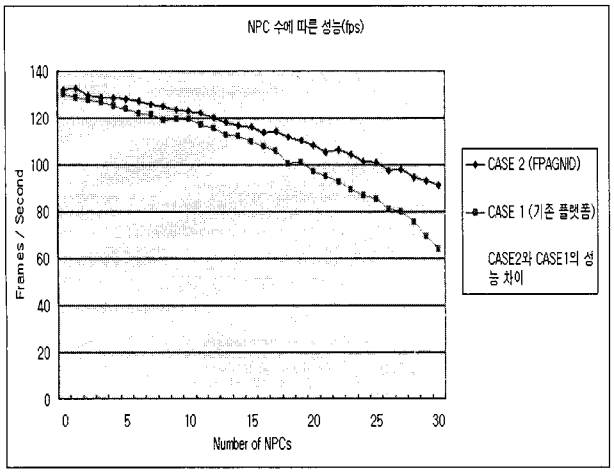

그림 12. 기존 플랫폼과 FPAGNID의 팀 지능 조절기를 통한 그룹 행동 시 게임 서버의 성능 차이

Fig. 12. Performance of game server with CASE 1(general paltform) and CASE 2(FPAGNID) included Team Intelligence Controller for group-behavior.

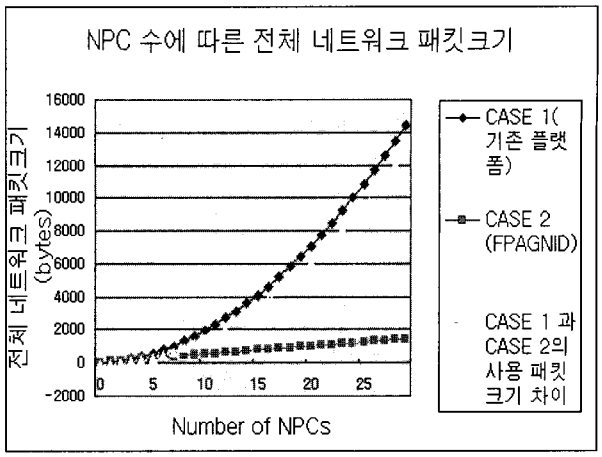

그림 13. 기존 플랫폼과 FPAGNID의 팀 지능 조절기를 통한 그룹 행동 시 네트워크 패킷 크기의 차이

Fig. 13. Required size of packets with CASE 1(general paltform) and CASE 2(FPAGNID) included Team Intelligence Controller for group-behavior.

기존 플랫폼에서 많은 지능 NPC를 수행하기 힘든 것과 달리 FPAGNID를 통해 보다 더 많은 지능 NPC를 수행할 수 있음을 의미한다.

팀 지능 조절기의 효율성을 알아보기 위해 Pac-Man 게임에서 그룹행동을 수행하는 NPC의 수를 증가시키며 팀 지능 조절기를 사용 여부에 따라 게임 서버의 성능과 한 프레임 당 사용하는 전체 네트워크 패킷 크기를 측정, 비교 하였다. 실험에 사용된 NPC는 unit-center reference 기법[11]을 이용한 line 포메이션[11]을 유지하며 플레이어의 위치로 이동하게 된다. 이를 위해 사용되는 패킷은 NPC ID를 포함하는 4바이트 헤더정보와 NPC 위치정보와 상태정보를 포함하는 12바이트를 합해 총 16바이트로 이루어지며, 그룹행동을 위해 팀 지능 조절기가 각 NPC에 전달하는 명령정보는 헤더 4바이트와 그룹 명령 4바이트와 중간 위치 정보 8바이트로 총 16바이트로 구성된다. 그룹행동을 위한 명령 패킷은 그림 6에 설명된 방법에 따라 게임서버와 팀 지능 조절기에 의해 생성되며 NPC에 전달된다.

그림 12의 그래프는 그림 6의 팀 지능 조절기를 사용하지 않는 일반적인 경우를 보여주는 CASE 1과 FPAGNID에서 팀 지능 조절기를 통해 부하 분산을 지원하는 경우를 보여주는 CASE 2 두 가지 경우에 대한 실험 결과와 두 경우의 성능 차이를 보여준다. CASE 1의 경우엔 각 NPC는 게임서버에서 모든 NPC의 정보를 전달받은 후, NPC의 중간점을 계산하여 자신의 위치를 결정하는 반면, CASE 2에서는 팀 지능 조절기가 게임 서버로부터 팀 NPC들의 정보를 전달 받은 후, 각 NPC에 그룹 명령과 NPC들의 중간 위치정보를 전달하고, 각 NPC는 그룹 명령과 중간 위치 정보를 바탕으로 자신의 위치로 이동하게 된다.

게임 서버가 모든 NPC에 대해 팀 NPC의 상태정보를 전송하는 CASE 1에서는 NPC의 수가 증가함에 따라 게임 서버가 처리해야 하는 부하가 증가하기 때문에 급격히 성능이 떨어지는 반면, 팀 지능 조절기 사용시 게임 서버는 기존과 같은 정보만을 처리하고, 효율적으로 그룹 행동 명령을 각 NPC에 전송하는 CASE 2에서는 게임 NPC 수에 따른 게임 서버의 성능 감소폭이 CASE 1보다 작음을 보여준다. 또한, 게임 서버의 부하가 NPC 수 n에 지수적으로 증가하는 CASE 1과 선형적으로 증가하는 CASE 2에서의 성능 차이는 NPC의 수 n^2에 비례하여 증가함을 알 수 있다. 이것은 FPAGNID의 팀 지능 조절기를 통해 기존보다 많은 수의 팀NPC의 그룹행동을 정의할 수 있음을 의미한다.

네트워크를 이용한 게임 지능NPC 개발에 있어 사용하는 네트워크 패킷의 크기는 더 많은 그룹 행동 명령을 전달할 수 있기 때문에 중요한 의미를 가진다. 그림 13은 팀 지능 조절기 여부에 따른 플랫폼을 구성하는 전체 네트워크의 패킷 크기와 그 차이를 보여준다. CASE 1은 그룹행동을 위해 게임서버가 모든 NPC의 정보를 각 NPC에 전달하여 총 네트워크 패킷 크기는 팀을 구성하는 NPC의 수 증가에 따라 급격히 증가하는 반면, CASE 2는 팀 지능 조절기를 통해 모든 NPC 정보를 전달하는 대신 그룹행동 명령을 전달하기 때문에

NPC의 수에 선형적으로 증가한다. 초기 NPC의 수가 3개 미만일 경우엔 CASE 2가 더 많은 네트워크 패킷을 사용하지만, 3개 이상일 경우엔 그 차이가 급격히 늘어남을 알 수 있다.

각 실험의 결과를 통해 모듈을 통한 부하분산과 그룹 행동을 위해 팀 지능 조절기의 사용이 게임 서버의 성능과 네트워크 자원 사용에 있어 기존 플랫폼보다 효율적임을 알 수 있다. 또한, 하나의 게임 환경을 지원하는 Gamebots[5]와 달리 Quake2 와 Pac-Man 게임 환경에서의 실험을 통해 다양한 게임 환경을 지원함을 보여주었다.

VI. 결론 및 향후 연구

본 논문에서 게임 NPC 지능 개발 플랫폼 구조를 위한 요구사항을 분석하고, 이를 만족하는 플랫폼 FPAGNID를 제안하였다. FPAGNID는 게임 환경과 독립적으로 게임 NPC DLL을 개발할 수 있으며, 이를 통해 다양한 게임 환경을 지원하고, 실시간을 만족하기 위해 부하를 분산하며 게임 서버의 자원을 선점하지 않는다. 또한, 게임NPC 수행기를 통해 다양한 인공지능 기법을 지원하며, 팀 지능 조절기는 스포츠 게임과 같은 팀 행동을 필요로 하는 게임에서 기존 플랫폼들 보다 효율적으로 그룹 행동을 지원한다. 마지막으로, FPAGNID는 각 모듈의 부하분산을 통해 기존 플랫폼보다 높은 성능을 보여주었다. 이러한 게임 NPC 지능 개발 플랫폼을 통해 보다 사실적인 게임 NPC에 대한 연구와 다양한 게임 환경에 표준적으로 적용될 수 있는 월드 인터페이스에 관한 연구가 필요하다.

참 고 문 헌

[1] Brian Schwab, "AI GAME ENGINE PROGRAMMING", Charles River Media, 2004.

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

[3] Rosenbloom, P., Laird, J.E., and Newell, A. "The Soar Papers: Readings on Integrated Intelligence", MIT Press, 1993.

[4] Van Lent, M. C. and Laird, J. E., "Developing an Artificial Intelligence Engine", In Proceedings of the Game Developers'Conference, San Jose, Calif. Forthcoming, 1999.

[5] Adobbati, R.,"GameBots: A 3D virtual world test bed for multiagent research", In Proceedings of the Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS, 2001.

[6] Laird, J.E. and van Lent, M. "Human-level AI's Killer Application: Interactive Computer Games.", AAAI Fall Symposium Technical Report, North Falmouth, Massachusetts, pp. 80-97, 2000.

[7] Laird, J. E. "It Knows What You're Going to Do: Adding Anticipation to a Quakebot.", AAAI 2000 Spring Symposium Series: Artificial Intelligence and Interactive Entertainment, AAAI Technical Report SS00 -02, 2000.

[8] Laird, J. E., Assanie, M., Bachelor, B., Benninghoff, N.,Enam, S., Jones, B., Kerfoot, A., Lauver, C., Magerko, B.,Sheiman, J., Stokes, D., and Wallace, S. "A Test Bed for Developing Intelligent Synthetic Characters.", AAAI 2002 Spring Symposium Series: Artificial Intelligence and Interactive Entertainment, 2002.

[9] Cantu-Paz E, "A survey of parallel genetic algorithms, Calculateurs Paralleles, Paris: Hermes", 10,2, pp. 141-171, 1998.

[10] J. Faerber, "Traffic Modelling for Fast Action Network Games", ACM Multimedia Tools and Applications, Vol. 23, Issue 1, pp. 31- 46, 2004.

[11] Tucher Balch, Ronald C.arkin, "Behavior-Based Formation Control for Multirobot Teams", IEEE Transaction on Robotics and Automation, vol. 14, no. 6, pp. 926-939, 1998.

저 자 소 개

임 차 섭(학생회원)

2004년 중앙대학교 컴퓨터공학과 학사 졸업.

2006년 중앙대학교 첨단영상대학원 영상공학과 석사 졸업.

<주관심분야 : 게임 인공지능, 지능 플랫폼>

김 태 용(정회원)

1986년 한양대학교 전기공학과 학사 졸업.

1988년 한양대학교 전자통신공학과 석사 졸업.

1998년 2월 항공과대학교 컴퓨터공학과 박사 졸업.

2003년 9월~현재 중앙대학교 첨단 영상대학원 교수

<주관심분야 : 컴퓨터 비젼, 컴퓨터 보안, 영상통신, 게임>