실감형 Networked Virtual Environment 의 사실성 증진를 위한 Non Player Character 의 지능적 제어 프레임워크*

전경구[1],성미영[2],이상락[3]
인천대학교 멀티미디어시스템공학과[1], 컴퓨터공학과[2 3]
{kjun[1],mysung[2],srlee[3]}@incheon.ac.kr

Intelligent Control Framework for Non Player Characters of Immersive Networked Virtual Environment

Kyungkoo Jun[1], Meeyoung Sung[2], Sangrak Lee[3]
Dept. of Multimedia System Engineering[1], Dept. of Computer Engineering [2 3]
University of Incheon

요 약

본 논문에서는 실감형 Networked Virtual Environment (NVE)의 사실성 증진을 위한 Non Player Character (NPC)의 지능적 제어 프레임워크를 제안한다. 이 프레임워크는 반응의 다양성, 실시간성 그리고 NPC의 능동성 면에서 기존 게임에서 사용되는 NPC 구현 기법과 차이가 있다. 기존 NPC 제어구조의 경우, 휴먼 사용자의 행동에 따른 NPC의 반응이 일정 스크립트나 규칙에 따르기 때문에 정형적이며, 또한 NPC의 반응시간에 대한 실시간성을 고려하지 않고 있다. 또한 NPC는 휴먼 사용자의 액션에 반응하는 종속적이고 수동적인 역할만을 담당한다.

제안하는 프레임워크에서는 NPC는 각자의 취향을 가지고 있어 다양한 반응과 행동양식을 보일 수 있으며, NPC의 행동 결정 시간에 어느 정도 실시간성을 부여할 수 있으며, 또한 NPC의 역할이 수동적 형태에서 벗어나 능동적으로 계획하여 행동을 실행할 수 있다. 프레임워크의 구현을 위해 SWI-Prolog의 Rule based 추론엔진과 유전자 알고리즘을 사용하였다.

Keyword : NVE, NPC, Rule-based Inference, Genetic Algorithm

1. 서 론

컴퓨터 게임과 가상현실 (Virtual Reality) 분야의 그래픽 기술은 수 년간 놀라운 발전을 이루어 왔으나, 컴퓨터에 의해 조정되는 Non Player Character (NPC) 들의 행동제어 기술은 아직 더 많은 연구가 필요하다[1].

게임이나 가상현실에서 NPC가 필요한 이유는 다음과 같다. 첫 번째는 사람들이 조연 역할을 원하지 않기 때문이다. 이러한 조연 역할은, 예를 들어, 정문의 보초나 길거리 행인들 같은 것이다. 두 번째로는 싱글 플레이 게임일 경우, 사용자에 버금가는 지능을 가진 캐릭터가 존재하는 것이 게임의 오락성과 가상현실의 실재성을 향상 시킬 수 있기 때문이다. 세 번째 이유는 NPC는 스토리가 있는 게임의 경우에 게임을 진행시키는 역할을 한다. 이러한 역할은 어드벤쳐 게임이나 롤 플레잉 게임에서 두드러진다.

게임 사용자들을 위주로 한 설문조사 결과, 사용자들은 상대 캐릭터가 좀 더 지능적이며 또한 예상치 못한 행동을 보여주기를 원한다는 결과를 얻었다[2]. 즉 게임 사용자들은 좀 더 실제 같은 캐릭터들과 상대하기를 원하는 것이다. 하지만 현재 NPC들은 Finite State Machine (FSM)이나 스크립트 언어로 구현된 일정 루틴에 의해

* 본 연구는 산업자원부 (MOCIE)의 지원으로 인천정보산업진흥원 (IITPA)을 통하여 수행되었음.

통제되므로 복잡도나 유연성 면에서 기대에 못 미치는 경우가 많다.

NPC들을 제어하는 기술로는 Rule based 추론엔진, 퍼지 논리, State Machine, Decision Tree, Evolutionary Computing, 그리고 신경망 등이 있다.

Rule based 추론엔진은 환경의 현재 상태를 나타내는 Fact들과 IF-THEN 형태의 조건과 해당행동으로 표현되는 Rule들에 기반하여 동작한 한다. 하지만 추론 엔진의 연동에 따른 성능 문제가 있다.

퍼지 논리에 기반한 NPC 제어 시스템은 불분명한 정보가 주어진 상황에서도 결정을 내릴 수 있다. 기존의 0과 1의 이진 형태의 정보를 사용하는 대신 사람이 사물을 인지하는 방식과 가까운 방식을 사용한다. 예를 들어, 덥다, 조금 덥다 식의 불분명한 정보에 기초하여 결정을 내린다. 하지만 퍼지 논리의 적용이 쉽지 않다는 단점이 있다.

State Machine은 NPC 제어구조를 구현하는데 가장 많이 사용되어 왔으며 가장 구현하기 쉬운 방식이다. 특히 FSM은 NPC가 가질 수 있는 상태의 개수를 유한 개로 한정하며, 각 상태에는 취할 수 있는 행동이 명시되어 있고, 그 결과 상태간 전이가 발생한다. FSM을 사용할 경우, 사용자가 NPC의 행동을 쉽게 예측할 수 있다는 단점이 있다.

Decision tree는 복잡한 형태의 IF-THEN 문을 트리 모양으로 구성한 것으로, 루트 노드에서 시작하여 종단 노드 방향으로 이동하면서 결정을 내리는 과정이다. 종단 노드는 해당 행동을 가지고 있다. Decision tree는 구현하는 데 있어서 메모리 요구량이 많다는 단점을 가지고 있다.

Evolutionary Computing은 생물학에서 모티브를 얻은 것으로, 생명체가 번식하고 진화하는 과정을 모델링 한 것이다. 특히 유전자 알고리즘으로 대표되는 기법은 적자생존 기법을 사용하여, 세대가 지속될수록 점점 더 진화되어 최적화된 형태를 띄게 된다. 하지만 알고리즘의 특성상 초기 warming-up 시간이 필요하다는 단점이 있다.

이외에도 NPC의 학습기능을 가능토록 하는 신경망이 있으며, 또한 사용자와 NPC간 보다 자연스러운 상호작용을 위한 자연언어 처리기술이 있다.

이러한 기존 NPC 제어구조는 나름대로 장단점이 있지만, 이들 제어구조들은 일반적으로 독립적으로 사용되어 왔다. 따라서 본 논문에서는 기존 제어구조들을 혼합하여 기존 NPC들의 행동한계를 극복하는 새로운 NPC 제어 프레임워크를 제안한다.

본 논문에서 제안하는 NPC 제어를 위한 프레임 워크는 Rule based 추론엔진과 유전자 알고리즘을 사용하여 NPC가 보다 다양하게 반응하는 동시에 계획을 세워 능동적으로 행동을 할 수 있게 한다. 본 논문은 2장에서 관련 연구에 대해 소개하고, 3장에서는 제안하는 프레임워크 구조에 대해 설명하며, 4장에서는 구현부분을 다룬다. 그리고 마지막으로 5장에서 결론을 맺는다.

2. 관 련 연 구

Soar[3]에서는 NPC가 스스로의 취향에 따라 선택하는 목표지향적 제어구조를 가지고 있다. 즉, NPC가 복수의 목표 중 하나를 선택해야 되는 상황에서 자신의 취향에 기반하여 결정을 내리게 된다. 또 다른 Soar의 특징은 사용자의 행동을 예측한다는 것이다. 예를 들어 방으로 들어간 사용자를 추적하는 대신 사용자가 방에서 나올 때까지 방에서 기다리는, 예측에 기반한 행동을 할 수 있다. 실제로 Soar는 Quake 게임의 Death match 버전을 구현하는데 사용되었다. Soar는 스스로 계획하는 능동적 형태의 NPC를 구현하는 데는 적당치 않다는 단점이 있다.

H-CogAff[4]에서는 NPC의 행동결정이 보다 실시간성을 가지게 하는데 중점을 두고 있다. 이러한 실시간성 제공을 위해 H-CogAff는 두 가지 특징을 가지고 있는데 하나는 계층형 구조이고, 다른 하나는 중간 지점형 계획이다. 하지만 이 구조에서는 NPC의 행동이 다양하지 못해서 사용자가 쉽게 행동을 예측할 수 있다는 단점이 있다.

Excalibur 프로젝트[5]는 실시간성을 가진 범용형 에이전트 구조를 제안한다. 여기에서 계획은 계속적으로 외부상황에 따라 수정되고, 또한 계획 생성 후에도 주기적으로 실시간성이나 외부 제약상황을 반영하기 위해 변경된다. 이 구조는 범용성에 중점을 두어, 특정 도메인을 위한 NPC 구현에는 적합치 않다.

Being-in-the-world[6]는 머드 게임 환경에서의 에이전트 구조이다. 이것은 에이전트가 자율적으로 환경 내에서 생존해 가면서 동시에 성장하도록 하는데 초점을 두고 있다. 에이전트 구조는 철학자 이름으로 명명한 2계층으로 되어 있다. 데카르트 레이어는 추론을 담당하며, 추론 결과에 따른 실행은 하이데거 레이어가 담당한다. 이 구조는 NPC 반응의 다양성을 구현하는 데 한계가 있다.

3. 실감형 NVE를 위한 NPC 구조 제안

여기서는 구현하고자 하는 실감형 NVE내에서 NPC가 가져야 할 기본적인 행동양식에 대한 요구사항을 정리하고, 이러한 요구사항을 만족시키기 위한 NPC 구조를 제안한다.

3-1 요 구 사 항

NPC의 모델링 대상은 관람 공간 안을 자율적으로 돌아다니면서 관람하는 캐릭터들이다. 이러한 캐릭터들은 다음과 같은 동작을 하면서 실감형 NVE의 사실성을 증가시킨다.

- 사용자 캐릭터와 충돌 시 반응한다. 반응은 캐릭터의 기질에 따라 공손히 사과하기도 하고, 또는 화를 내기도 한다.

- 캐릭터들이 많이 모여 있는 곳에 가려는 성질을 가지고 있다. 이것은 사람들이 관람할 때 많은 사람들이 모여있는 곳에 모이고자 하는 성질을 반영한다.

- 피로에 대한 생리적인 반응을 보여서, 일정시간이상 관람을 한 이후에는 휴식공간을 찾아 휴식을 취하거나, 음식물을 섭취한다.

- 가족이나 단체를 구성하는 NPC들은 서로 간에 상호작용하면서 관람한다.

- NPC들은 개별적인 기질을 가지고 있어서 관람태도가 다르다. 예를 들어 계획적인 동선을 가지고 관람하는 캐릭터가 있는가 하면, 무작위로 돌아다니는 캐릭터가 있다. 특히 아동을 나타내는 NPC들은 아동 나름의 특성을 반영한다.

3-2 실감형 NVE의 구조

본 논문에서 제안하는 NPC의 제어프레임워크는 인지, 추론 및 계획, 선택, 그리고 행동의 순차적인 구조를 갖는다. 인지단계에서는 환경에 대한 현재 상태 정보를 받아들여 자신의 Knowledge base에 축적한다. 추론 및 계획 단계에서는 Rule과 인지단계에서 받아들인 정보를 이용하여 상황에 대한 추론과 목표 달성을 위한 계획을 세우게 된다.. 선택단계에서는 복수 개의 계획이 주어진 상황에서 취해야 할 행동을 결정하게 된다. 행동단계에서는 최종적으로 선택단계에서 결정된 행동을 실천으로 옮겨 외부상황에 영향을 끼치게 된다. 이러한 행동과 다른 NPC의 행동, 그리고 게임 사용자의 상호작용에 의해 상태에 변화가 생기고, 다시 NPC는 인지단계를 통해 이러한 변화를 받아들여, 위에서 기술한 단계를 반복함으로써 환경과 상호작용하는 NPC가 만들어지게 된다.

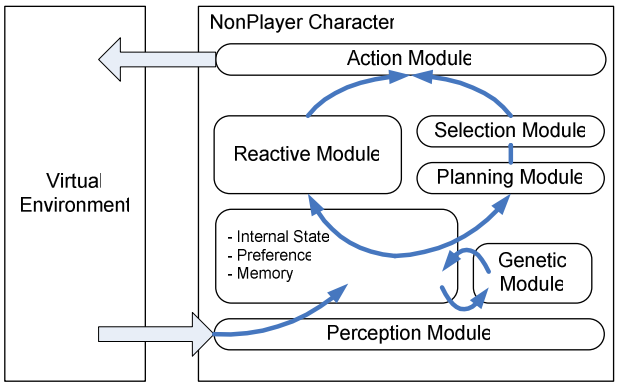

그림 1. 제안하는 NPC 의 지능구조

논문에서 제안하는 NPC의 지능구조는 그림 1과 같다. NPC 지능은 크게 NPC 내부 저장소와 모듈들로 구성된다. 내부 저장소는 현재 NPC의

상태, NPC의 취향 데이터, 그리고 Memory라 명명된 Knowledge base가 있다. 모듈들은 각각의 역할에 따라 명명된 6개의 모듈이 상호작용한다. 모듈들의 역할에 대한 설명은 다음과 같다.

Perception 모듈은 외부 가상 환경으로부터 정보를 감지하고 내부 저장소 내의 Memory에 Fact형태로 저장한다. 이 모듈의 가장 큰 역할은 외부 정보를 가공하여 Fact 형태로 만드는 것이다. 이렇게 만들어진 Fact는 뒤에 설명할 Reactive 모듈과 Planning 모듈이 사용하는 Rule-based 추론 엔진에 의해 사용된다. Perception 모듈은 또한 선택적으로 외부 상황을 받아 들일 수 있는 구조로 되어있다. 이것은 NPC의 감지능력 범위 조절과 과다한 외부 입력으로 인한 성능 감소를 방지하는데 사용된다. 이 기능에 대해서는 뒤에서 자세히 설명한다.

Reactive 모듈은 외부 환경 자극에 반사적으로 반응하도록 한다. 인간이 외부 자극에 별다른 추론이나 사유 없이 반응할 수 있는 것처럼, Reactive 모듈 역시 그러한 방식으로 반응한다. 특히 Reactive 모듈이 동작할 때 외부 자극뿐만 아니라 NPC의 내부상태를 저장하고 있는 Internal State 또한 참조한다. 이는 NPC의 상태에 따른 반응의 다양성을 만들어 내는 것을 가능하게 한다. 또한 이것은 NPC 내부 상태가 이전 자극에 의해 바뀔 수 있으므로, 다음 행동에 이전 자극의 영향을 반영시킬 수 있다는 특징이 있다. Reactive 모듈을 사용하는 가장 큰 장점은 NPC가 외부 자극에 상당히 빠르게 반응할 수 있다는 것이다.

Planning 모듈은 수동적 반응만을 할 수 있는 Reactive 모듈의 단점을 보완하여, 스스로 계획하고 행동할 수 있는 NPC를 만들어 낸다. 이 모듈은 외부 자극에 대한 반응으로 목표를 설정하고, 순차적으로 그 목표를 이루기 위한 계획을 세울 수 있다. 즉 목표란 NPC가 도달하고자 내부 상태이고, NPC는 목표가 주어졌을 때 그 목표를 성취할 수 있는 계획을 스스로 세우고 실천한다.

Selection 모듈은 Planning 모듈의 수행 결과, 하나의 목표에 대해 복수 개의 수행 계획이 주어졌을 때 적절한 계획을 선택한다. 하나의 목표에 대해 여러 개의 계획이 가능해지는 상황은 그림 2와 같이 Goal Hierarchy라는 트리 형태로 그려질 수 있다.

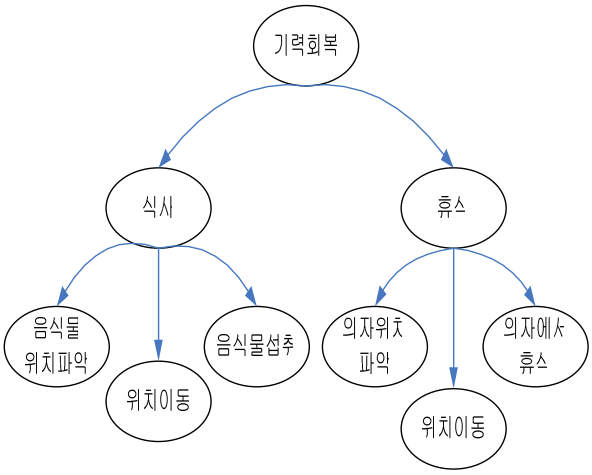

그림 2. Goal Hierarchy

이 트리에서 루트 노드인 “기력회복”은 NPC가 도달하고자 하는 목표를 나타내고, 중간 노드들인 “식사”와 “휴식”은 목표를 이루기 위한 두 개의 별개 계획들이다. 다시 종단노드들은 좌측에서 우측으로 진행되는 순차적인 구체적 행동들이다. 예를 들어, 기력회복을 위한 식사 계획은 그림 2에서와 같이, 우선 음식물의 위치를 파악하고, 근처로 이동한 후, 해당 음식물을 섭취함으로써 이루어 질 수 있다. 휴식이라는 계획을 선택했을 경우에도 비슷한 방식으로 진행된다.

따라서 Selection 모듈은 하나의 목표에 대해 Goal Hierarchy가 생성되었을 경우, 목표를 이루기 위한 중간 노드들로 나타내진 계획들 중 어느 것을 선택할 지를 결정한다.

Selection 모듈이 사용하는 결정의 기준은 NPC의 내부 저장소에 있는 Preference이다. Preference는 NPC의 개별적 특성을 반영하고 있어서 복수 개의 선택 가능한 계획들에 대해 선택할 때 정량적 기준을 제공한다.

Genetic 모듈은 NPC의 Preference를 조절하는 역할을 한다. NPC의 최종 목표는 관람환경 내에서 일정 기력을 유지하면서 가능한 많은 곳을 관람하는 것이 목적이다. 이러한 목적을 이루는

방식은 NPC의 Preference에 따라 여러 가지로 나타날 수 있다. Genetic 모듈은 유전자 알고리즘을 사용하여 각 NPC의 Preference별로 목표 성취 정도를 평가하여 Preference의 적합도를 계산한 후, Preference를 조절한다.

제안하는 프레임워크가 시간 제약을 포함할 수 있도록 하기 위해, Reactive 모듈과 Planning 모듈을 사용한다. 앞서 설명한 H-CogAff의 이중 계층 구조와 마찬가지로 즉각적인 반응이 필요한 부분에서는 Reactive 모듈을 사용하고, 복잡한 계획이 필요한 경우 Planning 모듈을 거치도록 하여 시간 제약을 만족시킨다. 또한 실제 구현에 있어서 정해진 시간까지만 Planning 모듈을 수행하는 기술이 필요하며, 이것은 향후 연구의 주제이기도 하다.

4. 구 현

본 논문에서 제안하는 프레임 워크의 구현은 기능상 NPC 내부 저장소 구현과 각 모듈 구현으로 나뉘어진다. 실제 구현은 SWI-Prolog를 사용하여 구현되었다.

내부 저장소 구현은 내부 상태 변수, Preference, 그리고 메모리 세 부분으로 구성된다. 우선 내부 상태 변수들은 NPC의 상태 정보를 지속시간 정보와 함께 가지고 있는 Tuple 형태로 구현된다. 지속시간 정보는 해당 상태가 유효한 시간을 가지고 있다. 예를 들어, NPC가 행복한 상태일 경우, 그러한 상태가 지속되는 시간정보가 같이 기록된다. 이 시간 정보는 시간경과와 함께 감소하여, 0으로 되었을 경우, NPC는 더 이상 행복한 상태를 갖지 않는다.

메모리는 NPC가 외부 환경에 대해 인지하고 있는 정보를 나타내며, SWI-Prolog의 Knowledge base의 Fact로 구현된다. 이러한 메모리 정보 각각 마다 지속시간 정보를 가지고 있어, 외부 환경의 temporal 정보를 보다 쉽게 처리할 수 있다. 예를 들어, 외부 환경에서 소리가 단발적으로 발생하는 경우 이 사실은 아주 짧은 시간 동안만 지속되어야 하기 때문이다.

Preference는 NPC의 취향을 담고 있는 벡터로써, 각 위치 별로 해당 취향을 나타내는 1에서 10까지의 숫자를 저장하고 있다. 큰 숫자일수록 해당 취향이 강함을 나타낸다. Preference 벡터의 사용은 Selection 모듈의 구현 설명에서 자세히 다룬다.

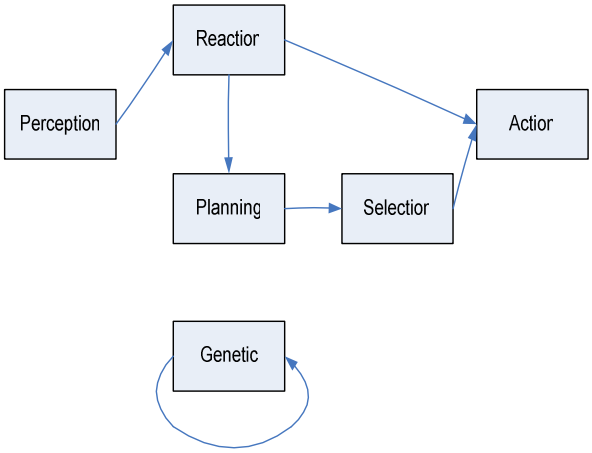

그림 3. NPC 모듈간의 연계구조

모듈들은 그림 3과 같이, 두 그룹으로 나뉘어, 순차적으로 연관되어 실행되는 Perception, Reaction, Planning, Selection, 그리고 Action 모듈들과 독립적으로 실행되는 Genetic 모듈로 구분된다. 순차적으로 연결된 모듈들의 경우, 각 단계 모듈의 출력은 다음 단계 모듈의 입력이 된다.

Perception 모듈에서는 외부 환경 정보들을 모아서 내부 저장소 메모리에 Knowledge base의 Fact로 변환하여 입력한다. Fact들은 단순한 문자열로 SWI-Prolog의 문법을 따른다.

Perception 모듈은 플래그를 사용하여 NPC 내부 상태에 따라 외부 환경 정보가 메모리로 유입되는 것을 차단한다. 예를 들어, NPC가 청각 상실이나 수면 등으로 외부 상황을 인지할 수 없는 상황을 구현할 수 있다.

Reaction 모듈은 내부 메모리내의 정보와 NPC 내부 상태를 이용하여 Rule-based 추론엔진을 구동한다. 이 모듈은 하나의 외부 상태에 대해 하나의 반응만이 매칭되는 구조로 구현되어 실시간성을 보장하도록 한다. 예를 들어, 다른 캐릭터와 충돌을 회피하는 행위 같은 반사적으로 일어나는 행동 등을 Reaction 모듈이 제어한다.

Planning 모듈은 Reaction 모듈의 완료 후에 선택적으로 실행된다. 외부 환경 변화가 반사적 반응만을 필요로 하지 않고, NPC가 좀 더 복잡한 행동을 하는 것을 요구할 경우 실행된다. Planning 모듈은 그림 2의 Goal Hierarchy를 그림 4의 SWI-Prolog를 사용하여 구현한다.

```
goal(refresh) :-
        state(health-low).

plan(have_food) :-
        goal(refresh).

plan(get_rest) :-
        goal(refresh).

action(find_food) :-
        plan(have_food).

action(move_to_food) :-
        plan(have_food),
        finished(find_food).

action(eat_food) -
        plan(have_food),
        finished(move_to_food).
```

그림 4 Goal Hierarchy 의 구현

그림 4에서는 세부계획들이 순차적으로 수행되도록 하기 위해, 전 단계 세부 계획들의 수행여부가 현 단계 계획의 실행조건으로 추가되는 형태로 Goal Hierarchy를 구현하였다.

	능동성	수동성
Preference	7	7
“have_food”	8	2
“get_rest”	2	8

그림 5 Preference 벡터와 계획성향 벡터

Selection 모듈은 Planning 모듈의 실행결과 복수 개 계획이 가능할 경우에 어느 계획을 실행할 것인지 결정한다. 그림 4의 경우 “have_food”와 “get_rest”의 두 개 계획이 선택되었다. 이 때, Selection 모듈은 Preference 값을 이용하여 실행할 계획을 선정한다. Preference 벡터 내의 각 값은 NPC 취향에 대한 가중치를 가지고 있고, 각 계획은 성향을 가지고 있다. 예를 들어, 그림 5와 같이 Preference로써, 능동성과 수동성이 있다고 하자. 그리고 NPC의 Preference 벡터가 (7, 3)으로 표현되고, 7은 능동성, 3은 수동성을 나타낸다고 하자. 이 경우, NPC는 능동적인 취향을 가진 캐릭터이다. 그림 4의 “have_food” 계획은 “get_rest”계획에 비해 보다 능동적이므로 성향벡터로써 (8, 2)를 갖고, “get_rest”는 반대로 (2, 8)을 갖는다고 하자. 이 때, Selection 모듈은 Preference 벡터와 계획의 성향 벡터를 곱해서 나온 결과값의 크기로 실행계획을 선정한다. 위의 경우, “have_food” 계획이 62 (= 7*7 + 2*3)이고, “get_rest” 계획이 38 (= 7*2 + 3*8) 이므로, NPC의 능동적 성격이 반영되어 “have_food” 계획이 선정된다.

Planning 모듈과 Selection 모듈의 실시간성 보장을 위해 모듈 소스 코드 내에 소요시간 측정을 위한 부분들이 삽입되어 있다. 따라서 모듈 수행 시간 직전에 허용되는 최대 소요시간을 설정하고, 모듈은 수행 도중 지속적으로 허용 소요 시간의 소진 여부를 체크한다. 만약 허용 시간에 도달했을 경우, 그 시간까지의 결과에 기초하여 행동을 결정하게 된다. 실시간 보장을 위한 부분에 대해서는 향후 추가 연구가 필요한 부분이다.

Action 모듈은 Reactive 모듈이나 Selection 모듈에 의해 선정된 행동을 실제 NPC의 행동 루틴으로 변경하여, NPC가 수행할 수 있도록 하는 역할을 한다.

Genetic 모듈은 정형화된 NPC의 행동이 외에도 목적에 부합적이면서도 다양한 NPC의 행동을 보여줄 수 있게 한다. Genetic 모듈은 적용 도메인의 성격에 따라 적합도를 평가하는 방법이 달라지게 된다. 본 논문에서 적합도는 NPC가 가상공간내의 관람객이라는 가정하에 평가하도록

한다.

Genetic 모듈은 그림 5와 같은 NPC의 Preference 벡터를 유전자로 사용한다. 초기 단계에서 각 NPC는 각각 고유한 Preference 벡터를 가지며, 주기적으로 유전자 알고리즘을 이용하여 적합도 계산 결과에 따라 상위 랭크 된 적합도를 가진 Preference 벡터의 NPC들 만을 다음 수행 사이클에 적용하고, 그렇지 않은 NPC들은 이들 Preference 벡터들의 Crossover나 Mutation을 통해 새롭게 수정된 벡터로 초기화된다.

Genetic 모듈을 구현하는 데 있어, 적합도 평가는 NPC별 최종 목표를 달성하는 정도로 평가된다. 예를 들어, NPC의 목표가 최대한 기력을 보존하면서 여러 곳을 둘러보는 것이라면, 적합도는 유사 목표를 가진 모든 NPC의 기력의 합과 방문 지역 개수의 합 중에 해당 NPC가 차지하는 비율로 계산된다.

5. 결 론 및 향 후 연 구

본 논문에서는 실감형 NVE 의 몰입성을 증진시키기 위해서 자율적으로 동작하는 NP 를 지능적으로 제어하기 위한 구조를 제안하고 구현에 대해 설명하였다.

제안한 구조는 상태정보를 저장하는 내부 저장소와 정보를 처리하여 행동을 결정하는 모듈들로 구성되어 있다. 이 구조의 특징은 반사작용과 사유작용을 각각 다른 모듈에서 처리하여 실시간성을 지원할 수 있으며, NPC 마다 취향을 가지게 하여 그에 따른 행동을 하게 할 수 있으며, 또한 유전자 알고리즘에 의해 NPC 의 취향을 합목적적으로 변형시켜가면서 정형적인 NPC 행동패턴을 탈피하게 한다. 구현을 위해서는 SWI-Prolog 의 Rule-based Inference 을 사용하였다.

논문의 NPC 제어 구조는 향후 Hard-Real time 성격의 실시간성 지원에 관한 분야와 유전자 알고리즘의 보다 효율적인 적용을 위한 NPC 행동의 적합도 계산 분야에 관한 내용으로 연구가 진행될 예정이다. 또한 현재 개발중인 실감형 NVE[7]의 NPC 제어구조로 통합되고 있다.

참 고 문 헌

1. B. Namee and P. Chunningham, “A Proposal for an Agent Architecture for Proactive Persistent Non Player Characters”, in Proceedings of 12th Irish Conference on AI & Cognitive Science, 2001.
2. P. Sweetser, D. Johnson, et al, “Creating Engaging Artificial Characters for Games”, in Proceedings of the 2nd Int. Conference on Entertainment Computing, 2003.
3. A. Newell, “Unified Theories of Cognition”, Cambridge, Massachusetts. Harvard.
4. A. Sloman, “Varieties of Affect and the CogAff Architecture Scheme”, in Proceedings of Symp. on Emotion, Cognition, and Affective Computing, 2001.
5. A. Nareyek, “Intelligent Agents for Computer Games”, Computers and Games, LNCS 2063, 2003.
6. M. Depristo and R. Zubek, “Being-in-the-world”, in Proceedings of the 2001 AAAI Spring Symposium on AI and Interactive Entertainment, 2001.
7. http//marvel.incheon.ac.kr