행위 범주에 기초한 실시간 에이전트 구조
An Architecture of Realtime Agent Based on Behavior Categorization

김하빈[a] 김인철[b]

[a,b]경기대학교 정보과학부
경기도 수원시 장안구 이의동 산 94-6
Tel: +82-31-243-9670, E-mail:{talkable[a], kic[b]}@kyonggi.ac.kr

요약

행위범주를 이용한 구조에서 가장 중요한 특징 두 가지는 첫째 행위의 계층을 주종이며 정적인 계층으로 구분하지 않고 행위 범주별 구분을 하여 복잡한 환경에 유연하게 대처할 수 있다. 둘째, 모든 행위를 객체화 하여 처리한다. 객체화 된 행위는 스스로의 문제점을 감시하고 처리하거나 보고하여 전체 구조의 간편화를 가져오게 된다. 본 논문에서는 행위를 범주 구분을 위하여 필요한 행위 설계 방식을 제시하고 행위 객체를 위한 구성요소를 소개한다.

Keywords:

Realtime agent; Agent architecture

1. 서론

고전적인 에이전트에 대한 연구는 비교적 정적이고 단순한 환경에서 행해졌으며 이러한 환경에 적합한 에이전트의 구조로 제시된 범용의 에이전트 구조들은 지속적으로 그 복잡성이 증가하고 있는 에이전트 환경에 유연하게 적용하기에는 미흡하다. 또한 행위의 복잡성이 증가함에 따라 단 방향으로만 진행되고 제한적이며 단순한 기능 구조만으로 구성된 기존 구조를 이용한 설계는 행위 별 구현의 복잡성을 증가시키게 된다. 따라서 이러한 문제를 해결 하기 위한 현대적 에이전트 환경에 유연하게 동작하는 에이전트 행위 기반 구조의 제시가 요구 되고 있다. 이러한 복잡성 높은 실시간 환경의 예로 게임 환경을 들 수 있다. 최근의 게임은 3차원 환경에서 복잡한 물리 법칙이 적용되어 직접 제어해 주어야 할 부분이 많아졌으며 빠르게 환경이 변화 한다. 최근에는 더욱 사실적인 에이전트의 연구를 위해 이러한 게임환경을 이용하는 예가 늘고 있다.

2. 관련연구

범용의 에이전트의 구조는 숙고형 구조나 반응형 구조, 이 두 가지를 결합한 혼합형 구조로 나누어 볼 수 있다. 이 혼합형 구조에는 복잡한 추론과정 없이 환경변화에 바로 반응할 수 있는 반응형 모듈과 실세계에 대한 기호모델을 바탕으로 계획을 세우거나 의사결정을 내리는 숙고형 모듈을 함께 내포하며, 숙고형 모듈에 비해 반응형 모듈에 더 우선권(precedence)을 부여하여 중요한 환경변화에 빠른 반응을 제공할 수 있도록 하였다. 1987년 Georgeff와 Lansky에 의해 개발된 PRS(Procedural Reasoning System)[2]은 대표적인 BDI 에이전트 구조이다. 이 구조는 내부에 기호로 명시된 믿음(belief), 소망(desire), 그리고 의도(intention)들의 집합과 계획 라이브러리(plan library)를 포함하고 있다. PRS는 실행 중인 지식영역들에 대해 명시된 바대로 문맥이 지켜지고 있는 지를 수시로 점검하여 문맥이 만족되지 않으면 곧바로 해당 지식영역의 실행을 중단함으로써 환경변화에 신속히 반응할 수 있다.

계층형 에이전트 구조는 크게 횡적 계층화와 종적 계층화를 하는 두 가지 형태로 분류된다. 횡적 계층화의 대표적인 예로 TouringMachine[3]을 들 수 있는데 1992년 Ferguson이 발표한 TouringMachine의 구조는 환경과 직접 맞닿는 인지부(perception)와

행동부(action), 그리고 하나의 제어 틀(control framework)속에 포함되어 있는 세 개의 제어 계층들로 구성되어 있다. 각 제어 계층은 에이전트의 행동을 결정하기 위해 독립적으로 병행 수행되는 하나의 프로세스이며, 제어 틀은 이들 계층들간의 중재 역할을 수행한다. 세 계층들은 모두 하나의 제어 틀(control framework) 내에 포함되어 있으면서 메시지 교환을 통해 서로 교신할 수 있다. 이러한 제어 틀의 목적은 계층간의 중재 역할을 하는 것이며 특히 제어 규칙(control rule)들을 이용하여 서로 다른 계층들에서 제안하는 동작들간의 충돌을 처리하는 것이다. Muller에 의해 제안된 InteRRaP[7]은 대표적인 종적으로 계층화 된 계층구조이며, 연속적으로 놓여진 각 계층은 하위 계층보다 더 높은 추상화를 나타낸다. 이 시스템 행위패턴들의 실행은 에이전트의 궁극적인 목표 달성을 위해 상위 계층들인 행위-기반 제어부와 협동 제어부로 하여금 단일 에이전트 계획 또는 협동계획을 생성하도록 요청할 수 있다. 이와 같은 과정을 거쳐 최종적으로 다시 실세계 인터페이스부에 의해 기본 동작들을 실행하거나 메시지들을 전송하게 된다. 앞서 열거한 에이전트 구조들은 나름대로 더욱 복잡성 높은 문제를 처리하기 위하여 흐름이나 계층별로 분리하여 각 모듈의 설계를 경량화 하며, 행위 결정부의 설계에 대한 방법을 제시하고 있다. 특히 InteRRaP과 TouringMachine과 같은 계층 구조에서는 에이전트 행위 제어 모듈 내부의 계층화 방법을 사용하여 더욱 복잡한 행위 구현에 대해 적극적인 해법을 제시하고 있다. 하지만 과거에 이러한 행위구조는 비교적 단순하며 정적인 환경에서 실험이 이루어져 제한적인 계층으로 인하여 지속적으로 복잡성이 증가하는 에이전트 환경에 대안을 적극적으로 제시하지 못하였다.

3. 에이전트 환경

하드웨어가 발전하며 에이전트의 활용 범위가 넓어짐에 따라 에이전트 환경은 변화하고 있다. 변화하는 에이전트의 환경이 가지는 특징은 다음과 같다.

-실시간 : 급격하게 변하는 환경에서는 에이전트가 사고하고 행동을 취할 때까지 기다려 주지 않는다. 때문에 실시간 에이전트는 어느 시간에 행동을 수행할 것인지 결정 할 수 있어야 하며 숙고가 필요한 판단과 반응적 판단의 혼합형 구조가 요구된다.

-다중 에이전트 : 환경이 다른 에이전트들에 의해 변화할 수 있다. 따라서 에이전트로 하여금 행위의 진행에 대한 예측을 더욱 어렵게 만든다.

-환경의 복잡성 : 에이전트 환경이 더욱 사실적으로 바뀜에 따라 에이전트 행위 또한 증가하였다. 다음과 같은 환경적 특성이 환경의 복잡성을 야기한다.

3차원 환경: 실수 수준의 3차원 물체로 이루어진 환경

사실적 환경: 물리법칙과 잡음이 적용되는 사실적인 환경

제한적 인지능력: 환경정보에 대한 정보 습득의 제한

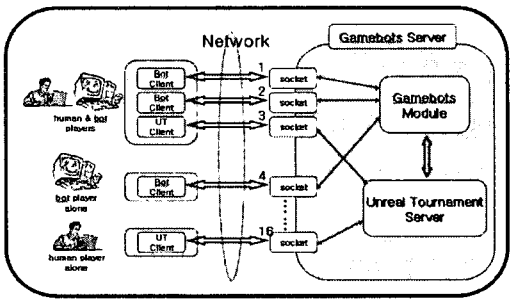
실시간: 동적 행위 진행으로 인한 환경의 지속적인 변화

다중에이전트: 환경을 변화 시킬 수 있는 요소

메시지통신: 동기, 비동기 기반의 메시지를 통한 에이전트간 통신으로 인한 정보의 다양성

행위의 다양화: 에이전트가 행할 수 있는 행위의 유형 증가

이러한 환경의 특징으로 인해 기존에 소개되었던 에이전트 구조만으로는 에이전트 구조의 충분한 지침이 되지 못하였다. 분산적이며 실시간으로 동작하는 환경에 대한 모든 고려 사항을 상위 계층의 행위 결정부에서 구현하는 것은 어려운 일이다. 때문에 이러한 환경에서 효율적이고 범용적으로 사용 할 수 있는 행위 기반 구조가 요구되고 있다.



[그림 1] GameBots 시스템

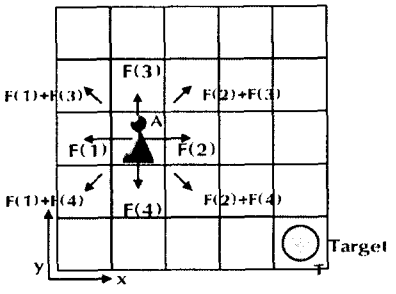
[그림 1]은 본 논문에서 실험 환경으로 사용하는 GameBots 시스템의 구조이다. GameBots은 견고한 3D환경을 제공하는 Epic사의 Unreal Tournament 게임 서버와 연동하며 보트 클라이언트들은 GameBots를 통하여 게임 환경에 접근 하며 인간 플레이어와

에이전트가 동일한 환경을 공유한다. 이때 에이전트가 인지할 수 있는 정보는 인간 플레이어의 인지 범위를 초과하지 않아 제한적 인지능력 특성요소를 보인다. 또한 복잡한 3차원 환경과 물리 법칙의 적용하여 3차원 환경특성과 사실적 환경 특성요소를 만족하며 1/10초 단위로 인지하며, 수행하여야 하는 실시간 요소를 포함하고 있다. 멀티에이전트 요소로는 최대 15개의 서로 협업하거나 경쟁할 수 있는 다른 게임 클라이언트들이 존재하고 이들은 각기 환경을 끊임없이 변화시키며 전투나 충돌을 통해 직접적으로 다른 플레이어에 영향을 주기도 하여 서로간의 행위 예측을 더욱 어렵게 하고 있다. 각 게임 클라이언트는 이동하고, 전투하고, 대화하는 20가지 이상의 기본 행위를 가지고 있으며 각 행위는 단발성 행위이거나, 한번 실행한 행위가 오랫동안 지속될 수 있는 특성을 지니고 있다. 특정한 행위는 그 행위가 진행되는 동안 다른 행위를 발생 시킬 경우 먼저 진행되고 있던 행위가 중단되지 않을 수 있는 등 다양한 특성의 행위들이 존재한다.

4. 범주형 계층 구조

계층의 분류에 있어서 과연 주종적 관계를 가지는 계층화가 가장 중요한가? 범주형 구조는 이러한 물음에서 시작한다. 범주구조가 기존 에이전트 구조와의 가장 큰 차이점은 에이전트 계층이 가지는 논리적 처리 단계의 분류가 아닌 행위의 유형별 분류에 중점을 두었다는 것이다. 에이전트 환경에 따라 행위 유형은 다른 이유로 범주형 계층구조는 기존에 사용되어오던 에이전트 구조와는 달리 계층에 포함 되어야 하는 행위의 범위를 정의하지 않는다.

4.1 2차원 격자상 목표추적 로봇



[그림 2] 2차원 격자 목표 추적 로봇

[그림 2]와 같은 2차원 격자 평면상에서 목표를 추적하는 로봇을 생각해 보자. 이 로봇은 동서와 남북으로 움직일 수 있는 별도의 제어를 받는 두 개의 모터를 가지고 있다. 또한 이를 조합하여 동남, 동북, 서남, 서북의 여덟 방향으로 움직일 수 있다.

이 문제를 종속적인 계층방법 이용하여 해결하면 동서남북으로 이동하는 행위를 각각 f(1), f(2), f(3), f(4)라 할 때 이 행위를 조합하여 사용하는 네 가지 행위와 하위 계층의 행위를 그대로 전달하는 행위, 모두 4개의 행위를 가지는 계층을 구성 할 수 있다. 따라서 각 계층이 가지는 행위는 다음과 같다.

```
Layer_Action {
  F(1), F(2), F(3), F(4)
}
Layer_Action_Set {
  F(a){f(1)}, F(b){f(2)}, F(c){f(3)}, F(d){f(4)},
  F(e){f(1)f(3)}, F(f){f(1)f(4)},
  F(g){f(2)f(3)}, F(h){f(2)f(4)}
}
```

이 행위계층을 구현하면 로봇 A가 목표 T를 추적하기 위한 제어는 다음과 같다.

```
Movement_Rule{
  A.x < T.x & A.y < T.y → F(f) : 동북으로 이동
  A.x < T.x & A.y > T.y → F(e) : 동남으로 이동
  A.x < T.x & A.y = T.y → F(a) : 동으로 이동
  A.x > T.x & A.y < T.y → F(g) : 서로 이동
  A.x > T.x & A.y > T.y → F(h) : 남북으로 이동
  A.x > T.x & A.y = T.y → F(b) : 서로 이동
  A.x = T.x & A.y < T.y → F(c) : 북으로 이동
  A.x = T.x & A.y > T.y → F(d) : 남으로 이동
  A.x = T.x & A.y = T.y → nil : 이동하지 않음
}
```

다른 방법으로 이를 행위를 범주별로 구분하였다. 동과 서로 이동을 한 범주로, 역시 남과 북으로의 이동 또한 한 범주로 나눈다.

```
Category X_Array_Movement {f(1), f(2)}
Category Y_Array_Movement {f(3), f(4)}
```

그리고 목표추적을 위한 범주별 제어는 각각 다음과 같이 표현이 가능하다.

```
X_Array_Movement_Rule{
  A.x < T.x → f(1) : 동으로 이동
  A.x > T.x → f(2) : 서로 이동
  A.x = T.x → nil : 이동하지 않음
}
Y_Array_Movement_Rule{
  A.y < T.y → f(3) : 북으로 이동
  A.y > T.y → f(4) : 남으로 이동
  A.y = T.y → nil : 이동하지 않음
```

}

종속적 계층을 사용한 경우 행위의 분류를 개념적 상하 관계로 정의하였고 한 계층에서는 한 개의 행위만 선택하기 때문에 행위 결정이 상대적으로 복잡하다. 반면 범주형 계층은 동시선택 가능한 행위단위로 범주 분류하여 동등한 수준의 행위의 동시 선택이 가능하며 각 범주별 제어를 별도로 해 주고 있다 이 예에서 볼 수 있듯이 두 방법 가운데 범주를 이용한 방법과 같이 별도의 제어를 해 주는 것이 더욱 간단하게 처리되는 것은 명확하며 이러한 유형의 설계 방법을 선택할 수 있지만 기존의 범용 에이전트 구조에서 직접 제시하고 있지는 않다.

4.1 행위범주 구분

범주가 포함해야 하는 행위의 범위가 구조에 제한되어 있지 않으므로 범주형 구조를 사용하는데 있어 가장 선행되어야 할 것은 에이전트 환경에 맞는 행위의 범주를 정의하는 것이다. 행위 범주는 행위의 종속적 계층관계와 병렬적 계층관계 두 가지 모두를 기준으로 하게 되는데 이는 다음과 같다.

종속적 계층관계

개념적으로 종속관계를 가지는 행위들은 다른 행위 범주에 속한다. 이는 고전적인 계층형 구조에서 각 계층에 포함될 행위의 성격을 나누는 기준과 같다.

병렬적 계층관계

동등한 수준을 가지지만 병렬성이 보장되는 행위는 다른 계층으로 구분된다. 뛰는 행위와 말하는 행위를 함께 할 수 있는 것이 간단한 예가 될 수 있다. 또한 2차원 격자상 목표추적 예에서처럼 같은 수준의 행위이지만 서로 다른 제어를 받는 행위 또한 다른 계층으로 분류가 가능하다.

이러한 규칙으로 나누어진 행위범주는 결국 같은 범주에 속하는 행위는 논리적이거나 물리적으로 동시에 진행할 수 없는 행위들로 나누어 지게 된다. 다른 계층형 구조와 같이 각 행위범주에서 선택되는 행위는 한 개의 행위이다.

4.3 행위제어부

메시지 통신에 기반한 에이전트 환경은 논리적으로는 병렬적으로 진행되는 행위지만 그 행위를 동시에 발생시키는 것은 불가능한 경우가 많다. 때문에 범주형 계층방법을 사용하기 위해서는 한 결정주기에 발생되거나 진행중인 행위결정 결과들을 병렬처리 할 수 있는 행위제어부가 요구된다. 행위제어부는 에이전트 환경의 제약에서 오는 행위 실행한계를 조절한다. 이 과정에서 현재 처리되어야 할 행위들 가운데 하나를 선택하여 해당 행위실행을 하게 된다. 이러한 선택과정만으로는 기존의 한 결정주기당 한 행위를 발생시키는 것과 같지만 범주형 계층구조를 위해서 행위제어부는 행위결정주기와 독립된 주기를 가지게 된다. 행위실행 주기는 에이전트 환경에서 허락하는 최소한의 행위 주기와 맞추어져 행위결정부에서 발생된 행위를 최대한 실행하게 된다. 이러한 과정에서 행위 결정부에서 발생한 행위는 다른 범주에서 발생된 높은 우선순위를 가지는 행위에 의해 행위실행이 연기될 수 있고 때로는 행위를 실행할 의미가 없어지는 시간까지 지체되기도 한다. 때문에 행위를 발생할 때 행위 실행에 대한 우선권에 대한 숙고가 필요하며 본 논문에서는 행위의 객체화로 이를 해결하고 있다.

4.4 행위객체

범주형 구조에서는 행위 계층의 자유화로 인해 행위 발생시 증가할 수 있는 복잡성을 행위의 객체화로 극복한다. 행위 객체는 행위를 실행하는 기본적인 기능과 행위실행 상황을 감시하고 보고하는 부분, 그리고 행위 선택에 필요한 속성을 포함하고 있다. 행위객체의 행위실행은 환경에 직접 작용할 수 있는 행동을 수행하거나 다른 범주의 행위 객체를 발생하는 두 가지로 분류할 수 있다. 행위객체는 행위의 실행과 성공적 수행이 보장되지 않는 복잡한 에이전트 환경에 적응하기 위해 행위실행 가능여부에 대한 평가와 실행상황 감시를 위한 기능을 포함하여야 한다. 행위객체는 행위가 선택되려 할 때 행위의 실행가능 여부에 대한 평가를 할 수 있어야 한다. 행위객체는 실행되면 그 순간부터 행위가 종료될 때까지 범주 내에서 다른 행위로 교체되거나 중단되기 전까지 지속적으로 실행진행을 하게 된다. 실시간으로 진행되는 환경에서는 행위가 선택될 때 가능한 행위일지라도 행위가 진행되는 시간에 더 이상 진행이 불가능해 질 수 있다. 이를 위해 행위 객체는 지속적으로 행위실행상황에 대한 감시를 하게 된다. 행위객체에 의해 발생된 행위실행이 성공적으로 수행완료 되었을 때와 행위가 중단되었을 경우 행위를 발생시킨 행위결정부나 다른 행위객체에 보고되게 된다. 행위객체의 실행평가는 환경에 대한 직접적인 평가와 행위객체의 실행에 의해 발생된 다른

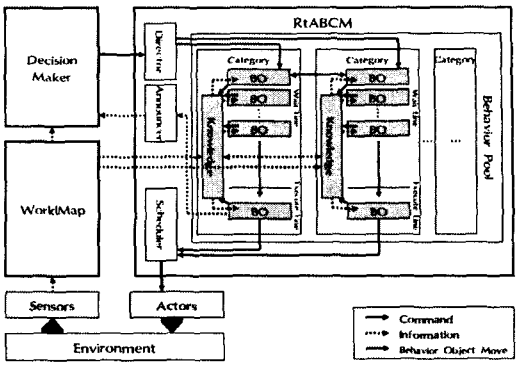
행위객체의 평가보고의 두 가지 요소에 의해 결정된다. 행위객체가 발생될 때 행위객체에는 그 중요도와 실행 속성이 지정되게 된다. 기본적으로 중요도에 의해 행위선택이 되며 추가적인 행위실행에 관련된 속성으로는 행위가 선택되고 실행되기까지의 대기시간, 대기 가능하지만 시간 내에 꼭 실행되어야 하는지의 여부 등을 포함한다. 행위객체는 스스로의 행동을 감시하며 그 보고를 할 수 있는 대화형 객체이다. 이의 구현을 위해 단방향으로 행위 진행만을 하는 형태에 비해 그 구현 비용이 증가하게 되지만 발생시키는 다른 범주의 행위에 대한 고려를 그 보고만으로 처리 할 수 있으므로 전체적인 복잡성은 감소하게 된다. 행위의 객체화는 더욱 복잡해지는 환경에 적응성을 높일 수 있다.

4.2 행위 결정

범주형 계층에서의 행위 결정은 낮은 수준계층의 한 행위를 선택하는 것이 아닌 다른 범주의 행위를 여러 개 발생 시킬 수 있고 2차원 격자상 목표추적 예와 같이 범주별로 분리한 행위 결정은 그 복잡성이 감소한다. 또한 행위의 실행가능 여부와 그 진행 상황에 대한 감시는 행위객체 스스로 수행하게 되므로 최상위 행위결정부는 그 행위의 평가보고만을 처리하는 것으로 행위진행의 신뢰성을 가질 수 있다.

5. 구현

본 논문에서는 앞서 제시한 범주형 계층구조를 가지는 RtABCM(Real-time Agent Behavior Control Module)을 사용하여 복잡한 실시간 에이전트 환경 요소를 가지는 KGBot으로 구현 하였다. KGBot은 GameBots 서버와 TCP통신을 사용하여 연결, Unreal Tournament에서 게임을 수행하는 에이전트이다. KGBot은 Domination 게임 형태에서 실험하였는데 이 게임 형태에서 가장 주된 목적은 지역을 점령하거나 점령한 지역을 지키는 것이며 이 과정에서 적과 전투하며 전투를 위한 아이템의 습득하고 지역을 탐색하는 행위가 필요하다.

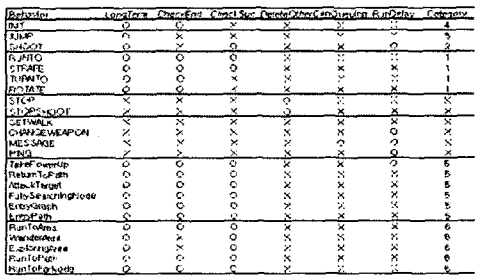


[그림 3] RtABCM 의 구조

[그림 3]은 RtABCM의 구조를 나타내고 있다. RtABCM은 행위객체(behavior object)를 담고 있는 행위 풀(behavior pool)과 행위객체의 제어를 위한 모듈로 구성된다. Director는 행위객체의 생성, 교체 그리고 삭제를 하며 Announcer는 의사 결정부에게 행위객체에서 발생하는 이벤트를 보고한다. 그리고 Scheduler는 최종적으로 선택된 행위 중 수행 가능한 행위를 Effecter에 전달하게 된다. WorldMap은 Sensors의 정보를 바탕으로 환경의 상황을 저장하고 있으며 Decision Maker에서는 WorldMap의 정보 가운데 단편적이고 명확한 정보를 바탕으로 전략 수준의 행위 결정만을 하게 되는 비교적 간단한 형태로 구현 되었으며 각 범주별 행위가 세부적인 상황의 처리에 대한 고려를 하고 있다.

5.1 행위분석

범주계층 구조를 설계하는 과정에서 가장 선행되어야 할 것은 에이전트 환경에서 제공하는 행동부터 이를 이용하여 어떠한 행위를 작성할 것인가를 인지하고 행위 범주별로 행위를 작성하는 것이다.



[그림 4] 행위 분석의 예

[그림 4]는 KGBot의 행위 분석에 사용한 예이다. 이 과정에서 GameBots서버에 직접적인 효과를 주는 행위, 다시 말해 가장 낮은 수준의 범주계층이라 할 수 있는 행위의 범주를 분석하고 이를 바탕으로 논리적으로 진행 가능한 행위 범주와 범주에 포함될

행위들을 작성 하였다. 행위 범주 분류의 예로 서버에 직접 영향을 줄 수 있는 행위 중 움직임에 관한 범주를 들 수 있는데 이 범주에는 RunTo (지점으로 이동), Strafe (시선을 다른곳으로 하며 이동), TurnTo (지점으로 회전), Rotate(지정각도만큼 회전)의 행위가 속한다. 이 행위들은 동시에 발생할 수 없으며 같은 범주의 다른 행위진행 중 발생할 경우 이전 행위는 정지되게 된다. 같은 수준의 행위이면서 이와 다른 범주의 행위는 Shoot(무기 발사), Message(다른 플레이어와 통신)등이 있는데 이 행위는 움직임에 관한 범주와는 무관하게 발생 시킬 수 있다 특히 Message의 경우 1/10초 단위로 행위 가능한 다른 행위와는 다르게 대략 1/3초 단위로 발생 할 수 있는 특성을 지니고 있다. 이는 행위 범주의 객체가 지니는 속성으로 기억하고 있으며 행위를 실행하는 Scheduler는 이를 참조하여 행위제어를 한다. 행위 분석 과정에서 범주 분류를 위하여 다음의 사항을 기준으로 하였다.

-행위 진행의 시간

-행위 실행 주기

-행위 재실행 대기 여부

기본 행위를 사용하는 상위 계층에 해당하는 행위범주는 환경에서 오는 제약을 기준으로 하기보다 논리적 기준으로 범주 분류를 하게 되는데 이를 분류하는 기준은 다음과 같이 하였다.

-행위간 간섭

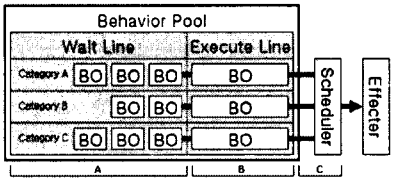
-행위 유형

-행위의 종속적 계층

행위유형과 종속적 계층을 기준으로 분류하게 되면 자연히 행위간 간섭이 없는 행위들로 범주화 된다. 다른 행위를 진행하기 위하여 먼저 진행중인 행위를 중지하여야 하는 경우이다. 이 특성은 행위 범주 분류에 있어 가장 중요한 사항이다.

5.2 행위객체

RtABCM은 실시간이며 복잡한 판단이 요구되는 상황에 대하여 행위기반 구조로 유연하게 동작하는 병렬적 행위객체를 설계하고 필요한 구성 요소를 제시하여 그를 위한 해결책을 제시하고 있다. 모든 행위는 객체로 표현되며, 행위객체는 자신의 범주, 중요도, 다른 행위와의 연관성, 행위 제약을 표시하는 속성을 가지고 있다. 해당 행위를 진행하는 동안 상태를 평가하고 스스로 교정하거나 보고 할 수 있는 함수를 포함한다. 실행 시간 동안 작동중인 행위객체는 기존의 계층 구조에서 한 계층의 역할을 담당하게 되므로 이외의 행위 계층이 가져야 하는 속성들을 도메인에 맞도록 추가 할 수 있다. 행위풀은 행위객체를 저장한다. 저장된 행위는 객체화 되어 행위가 선택된 순간부터 실행되거나 다른 모듈에 의해서 삭제 될 때까지 행위 범주별로 저장 되는데, 각 범주별로 행위풀에서 선택된 행위들은 대기열에 위치하게 되며, 그 구동이 scheduler에 의해 허가 받게 되면 행위는 실행열로 이동한다.



[그림 4] 행위풀의 행위처리

[그림 4]는 행위 풀에서 행위객체가 선택 되는 세 단계를 나타내고 있다. 각 행위객체는 범주별 대기열과 실행열을 가지게 된다. 대기열은 둘 이상의 행위 객체가 쌓일 수 있고, 특별한 경우가 아닐 때에는 먼저 대기열로 진입한 행위객체가 먼저 선택되어 실행열로 옮겨진다. 각 카테고리 별로 실행열에는 하나의 실행중인 객체만 있을 수 있고 실행열에 위치한 행위객체는 두 가지 행위 중 하나를 하게 되는데 첫 번째는 스케쥴러로 직접 행위 요청을 하거나, 다른 범주의 행위를 발생 시키게 된다. 행위가 다른 행위를 발생시키게 될 경우 실행열에 직접 발생 시키게 되는데 행위객체에 의해 발생된 행위는 같은 등급을 가졌을 경우 특별한 경우를 제외하고 대기열에서 직접 선택된 행위보다 우선순위가 떨어지게 된다. 스케쥴러는 매 실행주기마다 실행열의 행위를 평가하여 Effecter로 전달하게 된다.

범주 속성

범주 속성에 대한 값들은 범주 분류의 중요한 지침이 될 수 있다. KGBot은 아래의 범주 속성을 정의하였다.

-Series: 행위 범주의 종류를 나타낸다. 다른 시리즈의 범주는 서로 다른 실행주기를 가지게 되어 한 실행주기에 같이 실행 될 수 있다. 일반적으로 각 범주의 행위는 다른 시리즈를 가지게 된다.

-Rerun: 행위가 한번의 실행만으로 끝나는지, 지속적인 행위 실행 반복을 하게 되는지를 나타낸다

일반적으로 다른 행위를 발생시키는 행위가 이에 해당된다.

-ReRunDelay: 속한 행위가 한번 행위를 한 후 반복 실행 될 때까지의 제한 시간을 표시한다.

-ExecuteTime: 해당 범주의 어떠한 행위가 실행된 시간을 기억한다.

-ExecuteType: 행위가 직접 Effecter에 영향을 주는지, 또는 다른 행위를 발생시키는지를 표시한다.

-CanQueuing: 행위가 대기열에 누적될 수 있는지를 나타낸다.

-Term: 행위가 발생하여 실행 될 때 오랜 시간 동안 진행 되는지를 나타낸다. 행위 진행 시간이 0인 행위는 실행 되는 순간 실행열에서 삭제 된다.

행위 속성

-Category: 행위가 속한 범주를 나타낸다.

-Execute(): 행위의 실행에 필요한 구현을 포함 한다. 행위의 실행은 다른 행위의 발생,

-CanRunning(): 행위가 현재 실행 가능한지 평가한다. 현재 불가능한 행위라 판명되면 해당 행위는 삭제된다.

-MaxWaitTime: 대기열에서 실행열로 옮겨지기 전까지 대기 가능한 시간을 나타낸다. 이 시간을 초과하면 행위객체는 대기열에서 삭제된다.

-Rating: 행위의 중요도를 나타낸다. 구현시 직접 지정해 줄 수 있지만, 평가 함수를 이용하여 설정 할 수 있다.

-MaxWaitExecuteTime: 실행열에서 대기할 수 있는 최대 시간을 나타낸다. 이 시간을 초과하면 행위객체는 Necessarily를 참조하여 삭제되거나 최우선 실행되게 된다.

-Necessarily: 반드시 실행 되어야 하는 행위이다. 이 속성을 가진 행위는 대기제한시간까지 대기할 수 는 있지만 대기 제한시간을 넘기게 되면 최우선순위로 실행 된다.

-IsEndRunning(): 행위가 실행이 끝났는지를 평가한다. 이 때 결과값은 정상종료, 혹은 비정상 종료일 경우 어떠한 이유인지에 대한 구체적 제시를 하게 된다. 행위가 종료되었다면 실행열의 행위는 삭제 된다.

-ExecuteLimit: 행위의 실행 진행시간의 한계를 표시한다. 이 시간을 넘긴 진행중인 행위는 실패한 것으로 간주, 중단한다.

5.3 RtABCM 구성 요소

병렬적이고 유동적이며 대화형 행위객체의 구동을 위해 RtABCM은 다음과 같은 구성요소를 포함하고 있다.

-Director

Director는 선택된 행위를 객체형태로 변환, 행위 대기 열에 추가 시키는 역할을 한다. 매번 행위객체를 추가 할 때 director는 이미 대기 열에 위치한 행위객체와 새로이 선택된 행위객체를 비교하게 되는데, 그 관계를 고려하여 행위객체의 추가, 교체, 삭제행위가 발생하게 된다. 기본적으로, 동일한 범주의 행위객체, 즉 동시에 수행 할 수 없는 행위객체는 동시에 행위 풀에 저장 될 수 없다.

-Announcer

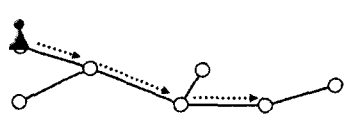
Announcer는 행위객체의 이벤트를 상위 행위 결정부에 전달하는 역할을 한다. 상위 행위 결정부에서 직접적으로 발생하지 않은 행위객체의 이벤트는 RtABCM내부의 해당 행위객체를 발생시킨 행위객체에게만 전달 되므로 최상위 행위 결정부는 스스로 발생시킨 행위객체의 이벤트 처리만 고려하는 것이 가능하다.

-Scheduler

대기 열에 있던 행위 객체들을 등급, 시간 제어 속성들을 고려하여 선택. 실행 열로 옮기게 된다. 계층별로 한 시간 단위 당 한 개의 행위만을 수행하게 되지만, 행위가 실행된 후 완료 되기 전까지 감시 할 수 있도록 기억하게 된다. 실행 열에 있는 행위 객체들은 환경에서 제공하는 행동 시간 단위, 행위별 재실행 시간을 고려하여 수행하게 된다. 행위의 수행은 직접적인 환경에의 작용뿐 아니라 새로운 하부 계층 행위객체를 만들어 내기도 한다.

5.4 행위 실행 예

[그림 5]는 KGBot의 행위 진행 예이다.



[그림 5] KGBot 행위 진행 예 1

그림에서 에이전트가 진행중인 행위를 다음과 같이 중요한 사항만을 표기하였다.

Strategy Category

-Domination_A_point{rating: 5, order: DecisionMaker}

Movement Plan Category

-RunToPath{rating: 5, order: Domination_A_Point}

Movement Category

-RunToXY{rating: 5, order: from RunToPath}

PowerUp Category

-TakePowerUp{rating: 9, order: from DecisionMaker}

에이전트는 지역점령, 아이템습득 행위를 실행 중이며 지역 점령 행위는 경로로 이동행위를 발생시키고 이는 또한 지점으로 이동하는 행위를 발생시키고 있다. 다른 범주는 병렬적 진행이 가능하므로 논리적으로 모든 행위는 동시에 진행되고 있다. 아이템 습득 행위도 진행되고 있지만 현재 취득 할 수 있는 아이템이 없으므로 별다른 행위를 취하고 있지 않다. 다음 결정주기에 행위결정부에서 다시 행위를 발생하지 않아도 취소되거나 완료되지 않는 이상 행위의 진행은 계속 된다.

[그림 6] KGBot 행위 진행 예 2

행위 진행 중 취득 가능한 아이템이 발견 되어 아이템 습득행위는 움직임 범주에 새로운 행위의 발생을 시도한다.

Strategy Category

-Domination_A_point{rating: 5, order: DecisionMaker}

Movement Plan Category

-RunToPath{rating: 5, order: Domination_A_Point}

Movement Category

~~-RunToXY{rating: 5, order: from RunToPath}~~

-RunToXY{rating: 9, order: from TakePowerUp}

PowerUp Category

-TakePowerUp{rating: 9, order: from DecisionMaker}

한 행위 범주에서 한 행위만 실행 될 수 있으므로 이중 높은 중요도를 가지는 아이템취득 행위에서 발생된 지점으로 이동 행위만이 선택되어 실행 되게 된다. 아이템 습득 행위가 종료 되기까지 지역 점령 행위는 지속적으로 행위를 시도하게 되고 아이템 습득 행위가 종료되면 즉시 행위를 계속 하게 된다. 이 과정에서 최상위 행위 결정부에서는 어떠한 고려도 하지 않으며 이미 발생시킨 행위들간의 경쟁에 의해 두 행위의 선택이 행해지게 된다. KGBot에서는 이러한 방법을 통하여 아이템의 습득 이외의 지역을 점령하는 도중 필요한 다른 행위들 즉, 적과의 전투와 같은 행위를 지역점령 행위에 논리적인 영향을 주지 않고 병렬적으로 진행 할 수 있었다.

6. 결론

본 논문에서는 기존 계층형 구조와 구분되는 범주를 이용한 에이전트 구조를 제시하고 이를 RtABCM으로 구현, KGBot에서 실험하였다. 동적 계층이라 표현 될 수 있는 범주를 사용한 방법으로 복잡한 실시간 환경에의 적응성을 보였다. 범주형 구조를 사용할 경우 행위를 객체단위로 설계하여 복잡한 행위에 대한 강건성을 높이고, 유연한 행위 구현으로 상위 행위 결정부의 설계에 자유로운 병렬적 행위 진행을 선택 할 수 있었다. 또한 범주형 구조를 구체적으로 구현한 RtABCM으로 사용한 현대적인 복잡성 높은 실시간 환경에서 범용적으로 사용 할 수 있는 에이전트 구조를 제시하였다.

참고문헌

[1] Adobbati, R., Marshall, A.N., Scholer, A., Tejada, S., Kaminka, G.A., Schaffer, S., Sollitto, C. (2001). GameBots: A 3D virtual world test bed for multiagent research. In Proceedings of the second International Workshop on Infrastructure for agents, MAS, and Scable MAS.

[2] Ferguson, I. A. (1992). TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents. PhD thesis, Clare Hall, University of Cambridge, UK.

[3] Georgeff, M. P. and Lansky, A. L. (1987) Reactive reasoning and planning. In Proceedings of AAAI-97, pages 677-682,

[4] Jose M. Vidal, Paul Buhler. (2002). Teaching Multiagent Systems using RoboCup and Biter. Interactive Multimedia Electronic Journal of Computer-Enhanced Learning, 4(2).

[5] Klaus Dorer. (1999). Extended Behavior Networks for the magmaFreiburg Team. RoboCup-99 Team Descriptions, Simulation League, Team magmaFriburg, pages 79-83.

[6] Muller, J. P., Pischel, M., and Thiel, M. (1994). A pragmatic approach to modelling autonomous interacting systems. In Wooldridge, M. and Jennings, N. R., editors, Proceedings of the 1994 Workshop on Agent Theories, Architectures, and Languages, pages 226-240, Amsterdam, The Netherlands.