인간과 똑같은 캐릭터 만들기: 개방형 게임 세계용 데이터 구동방식 AI 프레임워크

Creating All Humans: A Data-Driven AI Framework for Open Game Worlds

존 크래주스키(John Krajewski)
가마수트라 등록일(2009. 2. 4)

http://www.gamasutra.com/view/feature/1862/creating_all_humans_a_datadriven_.php

게임 세계를 생동감이 느껴지는 캐릭터로 가득 채우는 것은 어려운 일이다. 그리고 게임플레이 내용이 덜 제한적이며 플레이어가 어디라도 갈 수 있고 무엇이라도 선택해서 경험할 수 있는 개방형 게임 세계일 경우 이는 더욱 힘들다. 플레이어가 가는 곳 어디에서라도 반응을 일으키고 흥미로운 스토리를 만들어내려면 게임 엔진이 유연해야 한다. 특히 이러한 게임의 AI는 선형 게임의 AI와는 요구사항이 다르므로, 기존 게임의 AI 기술을 사용할 경우 다른 아키텍처 측면을 강조하는 접근방식을 취해야 한다.

본 기사에서는 판데믹 스튜디오의 개방형 세계 타이틀인 <디스트로이 올 휴먼스 2>에 쓰인 데이터 구동방식 AI 아키텍처를 다루고자 한다. 캐릭터들이 취하는 데이터 기반 행동을 담는 프레임워크와, 이러한 행동이 일어나고 종합되며 최적화되는 방식을 설명할 것이다.

모래상자 게임과 같은 개방형 세계 게임은 플레이어들에게 원하는 것은 무엇이건 할 수 있으며, 개발자들이 제시한 세계의 틀 안에서 원하는 게임을 만들어낼 자유를 준다. 이들의 게임플레이는 비선형적이며, 몰입도가 매우 크다. 그러나 게임 개발자 측면에서 보면 플레이어들이 마주칠 시나리오를 제어하고 제한하며, 규정하기가 힘들다.

AI 코드는 어떤 사태가 벌어져도 대응할 수 있는 유연한 기반 위에 세워져야 한다. 또한 선형 타이틀보다 훨씬 광범위한 게임플레이를 처리할 수 있어야 하고 예상치 못한 상황에 대응할 수 있어야 한다. 요컨대 AI는 행동의 깊이보다는 넓이에 초점을 맞춰야 한다. 즉 다양한 행동을 만들어내고 가급적 쉽게 캐릭터에 행동을 적용할 수 있도록 아키텍처 능력을 향상시켜야 한다.

이에 대한 해결책으로는, 우선 데이터 구동방식으로 행동을 일으키는 것이 있다. 행동은 코드 변경 없이 이루어지고 종합되며 공유식 구성요소와 같이 재사용되며, 특화된 버전에 맞게 치환될 수 있어야 한다. 개발자가 타이머의 지속시간이나 적의 공격성 같은 행동내용 설정뿐 아니라 행동의 근본적인 구조도 조정할 수 있다면 이상적이다. 예를 들어 주어진

임무를 수행하려면 어떤 걸음을 걸어야 하는가, 그 걸음을 걸으려면 어떻게 해야 하는가? 같은 것이 행동의 근본적인 구조인 셈이다. 캐릭터의 행동을 다양한 상황에 맞추고, 다시 구현할 수 있게 하며, 행동을 빨리 일으키고 최적화하는 것은 게임을 보다 생동감 넘치게 하는 효율적인 방식이다.

행동의 기초: 주 임무와 하위임무 수행

<디스트로이 올 휴먼스 2>의 행동 체계의 기본은 계층적 유한상태 기계(hierarchical finite state machine: HFSM)이다. HFSM에서 행위자의 현 상태는 다양한 개념의 추상성 정도에 따라 정해진다. 각 계층 레벨의 상태는 하위 계층 레벨의 상태를 가능성 있게 사용하여 임무를 분해해 더욱 작은 문제로 나눈다. 예를 들어보면 적공격(*attackenemies*)은 추상성이 높은 개념이다. 그리고 그 아래에 있는 덜 추상적인 무기발사(*fireweapon*)는 적공격을 위한 수단의 일부가 되는 것이다. 이러한 HFSM 구조는 캐릭터 행동의 틀을 짜기 위해 게임 AI에 쓰이는 일반적인 수단이다. 또한 평면형 유한상태 기계(Flat FSM)에 대해 여러가지 즉각적인 이득이 있다(HFSM의 현재 정보는 참고문헌을 참조하라).

우리는 구현한 HFSM의 각 상태를 행동이라고 부르며, HFSM은 시스템의 기본 아키텍처 유닛을 이루고 있다. 캐릭터가 게임에서 할 수 있는 모든 것들은 HFSM에 의해 행동을 다양한 방식으로 종합하여 이루어진다. 하나의 행동은 여러 행동을 일으킬 수 있는데, 실행되는 하위 행동은 상위 임무의 구체적인 일부이다. 각 임무를 여러 부분으로 나누면 행동 유닛을 다른 행동에서 재사용하고, 특별한 경우에는 무시하며 구조를 동적으로 바꾸는 등의 일을 하는데 노력을 크게 절약할 수 있다. 또한 더 나아가서 시스템을 직관적이고 변경하기 쉽게 만드는 데 더 많은 시간을 투자할 수 있다.

하위 임무 시작 임무를 작은 부분으로 나누는 데는 다양한 방법이 있으며 그 중 어떤 방법이 가장 좋은지는 임무의 유형에 따라 다르다. 임무가 특정한 요구조건이나, 연속적인 단계 수행이나, 목록에 나온 여러 행위를 무작위적으로 수행하거나 기타 다른 것을 필요로 하는가? 우리는 다양한 방식으로 하위 행동을 시작함으로써, 하나의 행동을 여러 조각으로 나누는 다양한 수단을 구현하게 되었다.

하위 임무에 우선순위 부여 하위 행동을 시작하는 최초의, 그리고 가장 일반적인 방식은 행동에 우선순위를 부여하고, 이를 목록화하는 것이다. 우선순위에 따라 시작된 행동은 즉시 조합되어(메모리가 할당되어 상위 행동의 하위 행동으로 추가된다.) 특수한 보류 모드를 설정하게 된다(그림 1 참조).

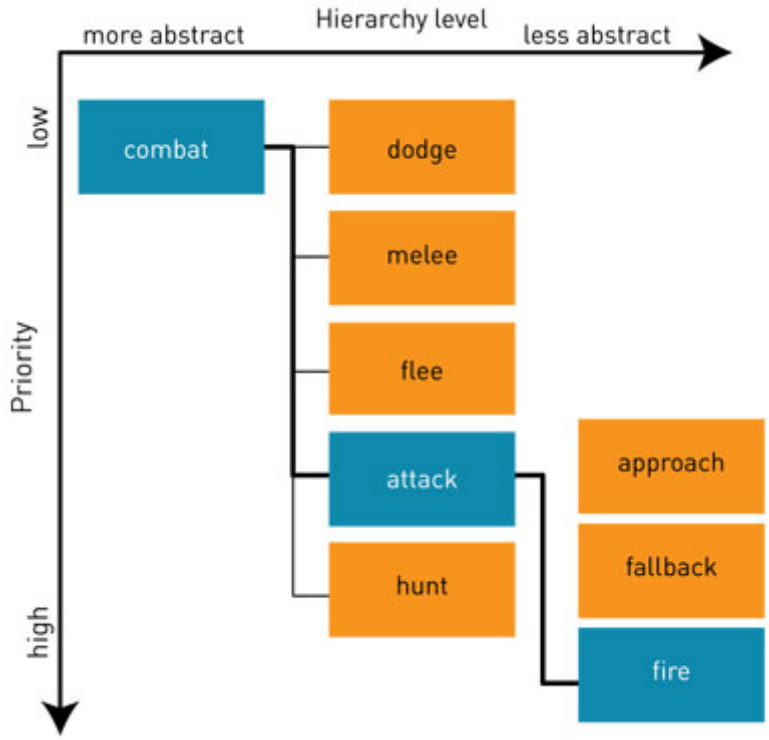

그림 1 전투 행동은 우선 순위가 설정된 하위 행동을 시작한다. 이로 인해 더 많은 우선 순위가 설정된 하위 행동이 일어나게 된다. 오렌지새은 보류된 행동을,청색은 활발한 행동을 나타낸다.

어떤 행동이 보류 모드에 들어가게 되면, 갱신되지 않고, 자체 설정에 따라 스스로 활성화를 결정할 때까지 대기하게 된다. 활성화되면 이 행동은 하위 행동을 시작한다. 활성화된 행동에만 하위 행동이 있으며 보류된 행동은 하위 행동이 시작될 때까지 대기하는 것이다.

상위 행동 하에서는 단 하나의 활성화된 행동만이 실행되는 것이 규칙이다. 그러나 이로 인해 다음과 같은 문제도 발생된다. 다양한 행동을 실행할 수 있을 때는 어떻게 해야 하는가? 그러므로 우선순위를 정해 어떤 하위 행동이 더욱 중요한지를 정해야 한다. 우선순위에 따라 하위 행동을 시작할 때 이들의 우선순위는 상위 행동에 추가된 순서대로 정해지는 것이다. 먼저 목록에 올라간 행동일수록 우선순위가 높다(참고문헌의 Isla를 참조).

이러한 방법을 사용하면 숫자로 우선순위를 정할 때 생기는, 우선순위가 점점 확대되어 다른 것들을 압도하는 우선순위 변형과 같은 문제를 피할 수 있다. 여기서는 우선순위 정의를 상황에 맞추어, 형제로 시작되는 행동의 작은 부분집합에만 연관되도록 하는 것이다.

위의 예에서, 무기발사가 공격의 하위 행동일 뿐 아니라 전투의 하위 행동이기도 한 것처럼 행동의 계층구조와 각 행동에 딸린 하위 행동을 알 수 있다. 현재 보류된 행동(회피)이 필요하다고 판단될 경우(NPC가 자신을 향해 사격이 가해졌음을 포착했을 때) 현재 활성화된 형제 행동인 공격을 중단하고 보류 상태에 되돌린다. 그럼으로써 공격의 모든 하위 행동이 삭제되는 것이다. 다른 형제 행동이 실행되지 않으면 회피가 시작된다.

이러한 우선순위 적용 수단은 대부분의 경우 유효하지만 가끔씩 단순한 선형 배열만으로는 임무의 중요성을 설명할 수 없는 경우가 있다. 우선순위가 높더라도 중요하지는 않은

임무가 현재 실시하는 중요한 임무를 중단해서는 안 되는 경우 중단 가능이라는 기능을 구현할 수 있다. 이는 현재 활성화된 행동의 우선순위를 높여 그 실행의 특정 부분에서 중단되지 않도록 하는 것이다.

이를 통해 단순한 선형 배열보다 훨씬 복잡한 방식으로 우선순위 설정을 할 수 있다. 예를 들어 그림 1에서 난투는 회피보다 높은 우선순위를 갖고 있지만 난투 실행 도중 회피를 시작해야 한다고 결정되는 경우라도 일단 난투의 애니메이션을 다 마쳐야만 한다면 난투 애니메이션 장면이 회피 동작으로 인해 끊기지 않도록 할 수 있는 것이다.

연속 및 무작위 하위 행동. 또 다른 하위 행동 시작 방식은 연속 및 무작위이다. 연속적으로 시작되는 행동은 목록에 나온 순서대로 실행된다. 맨 처음 행동을 실행할 수 있다면 맨 마지막 행동을 종료할 때까지 목록에 나온 행동들이 연속적으로 실행된다. 맨 마지막 행동이 종료되면 상위 행동도 완료된다. 하위 행동 그룹을 무작위로 골라 시작하는 경우 단 하나의 하위 행동 그룹이 선택되어 실행되며, 상위 행동은 하위 행동이 종료됨으로써 완료된다.

하위 행동 논 블로킹. 상위 행동 아래의 어느 행동이 활성화되어 실행 중인 경우에도 다른 행동을 논 블로킹 방식으로 시작하면 다른 행동도 동시에 활성화할 수 있다. 이런 행동들은 우선순위 목록에 있지 않은 것이다. 이런 행동은 다른 행동과 동시에 취해 임무를 달성해야 하는 경우 유용하다. 예를 들면 기동하면서 사격을 해야 한다거나, 특정 조건 하에서 목소리를 내야 할 경우 또는 여러 효과를 활성화 또는 비활성화시켜야 하는 경우 등이다. 일반적으로 하위 행동 논 블로킹 방식으로 실행되는 행동은 동시에 실행될 수 있는 형제 행동을 간섭하지 말아야 한다. 논 블로킹 방식 하위 행동은 오직 상위 행동이 비활성화될 경우에만 중단될 수 있으며, 형제 행동에 영향을 받지 않기 때문이다.

이러한 수단을 다양하게 조합하여 데이터 구조를 작성함으로써, 단일 행동으로 특징지을 수 없는 어렵고 다양한 레벨에서 결정을 내리고 임무를 처리할 수 있다.

새 퍼즐 조각 만들기

이 관점에서 볼 때 HFSM을 구성하는 기본 유닛의 특징은 행동이라고 볼 수 있다. 따라서 데이터 구동방식의 행동을 만들면 행동의 변형값과 설정은 물론 수행하는 행위 구조 역시 나타낼 수 있다(그림 2를 참조).

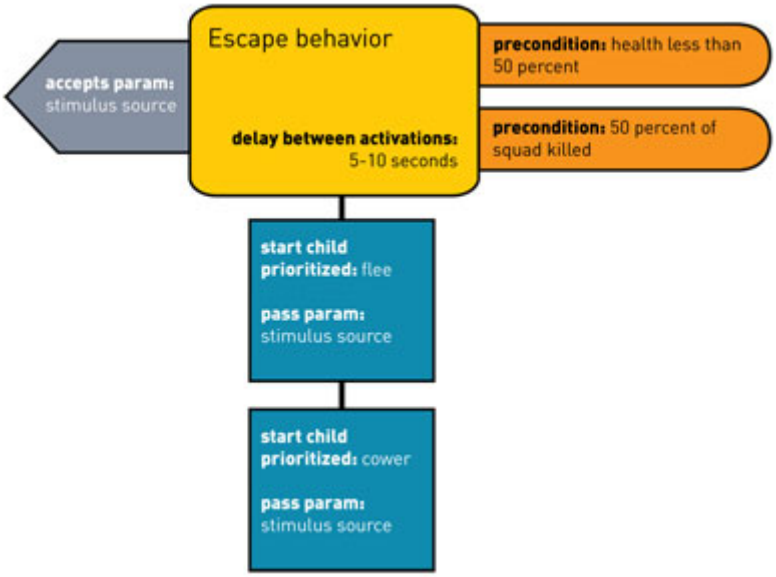

그림 2 이 행동 유닛의 전제 조건 중 하나라도 만족될 경우 행동은 활성화되어 두 하위 행동에게 자극 원본 매개변수를 보낸다. 두 하위 행동 간에 우선순위가 정해지면, 높은 우선순위를 가진 하위 행동인 도주가 실행 가능한 대로 실행된다. 그리고 도주가 실패할 경우에만 위축이 활성화된다.

각 행동은 자급형이다. 행동을 활성화시키는 상황, 더 이상 실행할 수 없는 상황, 중지시키는 조건, 활성화, 비활성화, 갱신하는 조건 등 행동에 대한 모든 것은 행동 자체 내에 이미 정해져 있다. 그러나 가장 중요한 것은 어떤 하위 행동을 시작할 것인지 정해져 있다는 것이다. 이러한 각 행동 특성은 행동마다 하나씩 있는 .behavior 파일 내에 설정되어 있다. 이 파일은 다른 모든 게임 데이터와 함께 분석되고 읽혀진다.

보류 상태로 만들어진 행동의 경우 이후 이것의 임무는 언제 활성화시킬 것인지를 정하는 것이다. 여기에는 활성화에 필요한 요구조건이 포함되어 있기 때문이다. 이러한 요구조건은 전제조건 목표 목록을 행동에 추가함으로써 정해진다. 전제조건은 게임 세계의 조건에 비추어 참인지 거짓인지를 판별할 수 있는 규칙의 모음이다. 게임 이벤트, 캐릭터의 체력, 분대장의 명령 등은 모두 특정 행동을 활성화할 수 있는 조건이며, 전제조건은 이 모든 것을 묻도록 설정되어 있다.

전제조건 정의 설정은 별도 파일에 저장되며 다른 행동을 위해 재사용이 가능하다. 그러므로 행동 설정 파일 내에 전제조건을 늘어놓기만 해도 새로운 행동에 쓰일 전제조건을 쉽게 선택할 수 있는 자료실을 만들 수 있다.

행동이 활성화되면 자료실은 대부분의 직접적인 사례의 경우 그 본 목적을 수행할 수 있다. 캐릭터의 자산(애니메이션, 음향, 효과)을 작동시키는 행동이나 코드측 변화 없이 더 많은 하위 행동을 시작하는 행동의 경우 뛰어난 유연성을 이용하여 새로운 행동에 신속히 살을 붙일 수 있다. 더 복잡한 행위의 경우 코드 지원형 행동을 사용한다.

코드 지원형 행동은 기준 행동에까지 쓸 수 있는 모든 설정을 갖추고 있으나 그 외의 것도 코드 내에서 만들어진 해당 모듈에 연결할 수 있다. 무기발사, 길 따라가기, 난투 등의 행동은 코드 측 조합 행동으로서, 너무 특정화되어 있어 일반화하거나 모든 행동에 사용하기는 힘든 행위를 한다. 예를 들어 무기발사의 경우에 쓰이는 '한 발씩 간격을 두고 사격'과 '연사로 사격' 같은 설정은 다른 대부분의 행동에는 쓸모가 없다. 이러한 코드

지원형 행동은 다른 행동과 마찬가지로 데이터 구조에 적합하며 HFSM의 주변에서 흔히 볼 수 있다. 또한 더욱 추상적인 행동의 하위 행동의 기준으로 쓰인다. 일반적으로 고등 레벨 행동의 역할은 대상과, 그 대상에 대해 실시해야 하는 임무를 분리하고 하위 행동을 시작하여 임무를 달성하는 것이다.

그러나 우리가 다루는 대상, 예를 들어 전투 중 캐릭터가 제거해야 하는 표적이나, 플레이어에게 피해를 입힌 마지막 사람 등은 게임이 실제로 실행될 때까지는 정의되지 않는다. 따라서 행동의 대상을 대조하고 결정을 내릴 방법이 필요하다. 매개변수 체계가 필요한 것이다.

대상과 행동을 연결

행동에 설정 목록을 결부하는 것에는 매우 큰 시스템 일반화가 따른다. 그러나 런타임 중 임의의 상황에 맞게 대상에 동적 조작을 가한다는 뜻은 아니다. 예를 들어 무기발사의 대상은 변할 수 있으므로, 그 명령을 작성하고 있을 때 발사할 대상을 알 수는 없다. 각 행동은 알 수 있어야 하고 게임 세계 속의 임의의 대상에 대해 런타임 도중에 결정짓게 된다.

<디스트로이 올 휴먼스 2>에서 사용한 방법은 하위 행동이 만들어질 때 상위 행동이 하위 행동에게 매개변수를 보내는 것이다. 그러면 이들 행동은 매개변수로서 대상을 문의하고 조작하거나 또는 더 하위 행동에 매개변수를 전달한다.

매개변수를 받아들여야 하는 행동은 매개변수를 둘 장소를 정의하게 된다. 그 장소는 상위 행동이 활성화하는 것으로 채워져야 한다. 여기서 장소 안의 변수는 다양한 방식으로 활용된다. 변수는 하위 행동으로 보낼 수도 있고, 이벤트 또는 메시지로 보낼 수도 있으며 코드 지원형 행동으로 보내지거나 코드 측에서 사용 가능할 수도 있다. 예를 들면 길 따라가기 행동에서는 최초 매개변수로 대상이 지나간 곳을 찾을 것이며, 난투에서는 최초 매개변수가 있는 곳으로 주먹을 휘두를 것이다.

이에 덧붙여 HFSM 상태는 원래 대상을 변이시킨다. 이 경우 상위 행동과 하위 행동을 거치면서 대상이 변이된다. 이 대상을 다른 행동에 연결시켜 그 행동의 대상으로 인지하게 하고 대상에 행동의 기능을 적용한다. 더욱 완고한 HFSM 구조에 스크립트와 유사한 상관관계를 융합시켜 조직을 희생하지 않고도 더 큰 유연성을 확보하는 것도 한 방법이다.

부분적으로 구현된 행동은 확고함과 추상성의 중간 정도 단계에 있다. 매개변수를 무시하는 능력으로 인해 행동을 더욱 다양하게 이용할 수 있으며 매개변수를 다양하게 이용하는 것은 이 방식에서 점점 흔한 일이 되어가고 있다. 매개변수의 표준적 사용방식을 이용하기 위해 코드 측에서 일부 표준 임무를 처리하는, 완전한 코드 지원형 행동은 아니지만 부분적으로 구현된 행동을 만들었다. 이들은 HFSM의 주변부에 속하지 않으며 행동을 설정하는 데이터가 입력될 때까지는 여전히 추상적인 보조 행동이다.

가장 흔하게 사용되는 부분적으로 구현된 행동은 범위테스트이다. 이는 존재하는 동안 계속 추적하고 저장하여 여러 결정을 내리는 데 활용하는 표적 매개변수를 받아들인다.

행동이 작성되면 활성화되기도 전에 매개변수가 통과되므로 매개변수를 통해 행동을 활성화할지를 결정할 수 있다. 범위테스트의 경우 통과한 표적(최초 매개변수)을 실험하고 이것이 두 번째 범위 내에 있을 경우 비활성화하는 추가 설정이 가능하다.

이러한 기능이 매우 유용하다는 것이 입증되었다. 행동은 길 따라가기나 난투 같은 데이터 정의형 행동과 코드 지원형 행동의 부모 역할을 하기 때문에 임무 수행을 중단하고 게임 세계 속 대상의 속성에 따라 반응하며 표적이 지나치게 멀리 있는 경우에는 실패사례를 제공한다. 범위테스트 행동을 겹친 후 적용해 다양한 시기에 다양한 대상에 적용하면 단순한 구성요소 몇 가지만으로도 동적 대상에 대한 복잡한 결정을 정의할 수 있다.

행동 보호

행동 보호가 의미하는 점은 바로 이것이다. 행동 보호가 활성화될 경우 최초 매개변수로서 행동을 보호하라는 지시를 내리게 된다. 그리고 우선순위가 정해진 행동을 시작하게 된다. 가장 우선순위가 높은 행동은 언제나 캐릭터가 보호받을 수 있는 거리 내에 머무르는 것으로서, 접근으로 표현된다. 접근은 표적이 너무 멀리 있을 경우 활성화되어, 이동체계에 가까이 가라는 지시를 전달하는 범위테스트 행동으로 구현된다(그림 3을 참조).

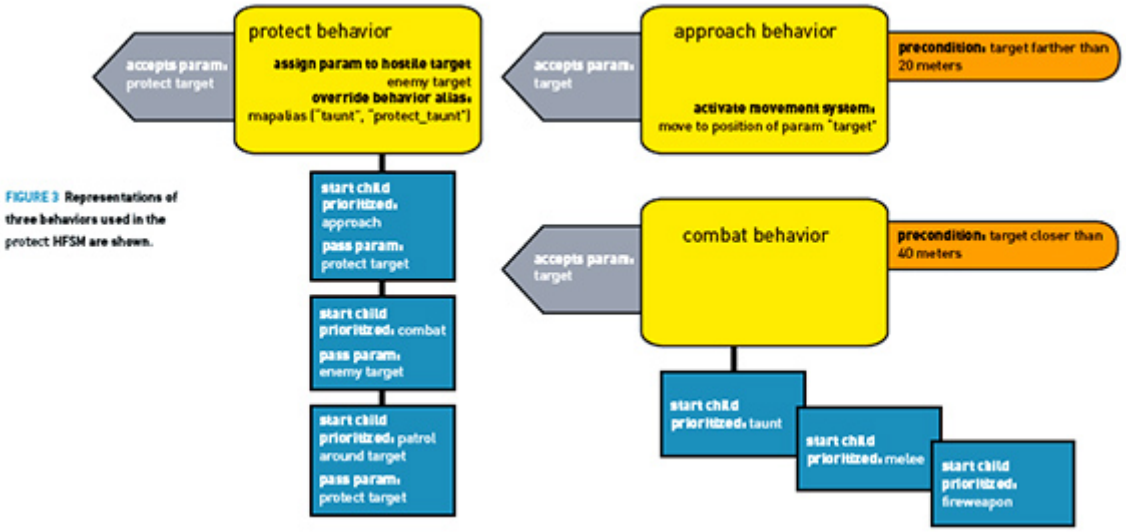

FIGURE 3 Representations of three behaviors used in the protect HFSM are shown.

두 번째로 보호받고 있는 캐릭터는 근처의 모든 적과 싸워야 한다. 캐릭터에서 실행되는 표적 선택 모듈에 질문함으로써 적을 탐지한다. 이는 행동 체계와는 별도로 실행되는 모듈이다. 또 다른 범위테스트 행동인 이 요구조건을 유지하기 위해 전투가 시작되며 적이 너무 가까이 있을 경우 하위 행동을 활성화하고 시작하여 표적과 교전하게 된다.

전제조건이 충족되지 못해 접근도 전투도 시작할 수 없는 경우 기본적으로 방황이 활성화된다. 이것은 가장 낮은 우선순위를 지닌 행동이며 어떤 전제조건도 없기 때문이다.

그러면 캐릭터는 보호받는 행위자 주변을 무작위로 순찰하게 된다.

런타임 시 매개변수를 받아들이는 행동을 만듦으로써 두 가지 다른 임무를 실행하는 동시에 다양한 객체, 즉 보호받아야 하는 인물과 위협을 가하는 적들을 움직이는 구조를 완전히 데이터 내에서 정의할 수 있다.

행동의 공유 및 재사용

캐릭터의 행위를 계층별로 정의할 때 얻는 큰 장점은 계층에서 쉽고 직관적으로 행동을 재사용하고, 대체하고, 제거할 수 있다는 것이다.

가장 단순한 것은 제외하더라도 이 시스템의 모든 행동은 .behavior 설정 파일 내에 그 설정이 정의되어 있다. 그러나 한 행동이 하위 행동을 만들어낼 경우 파일이름을 바로 사용하여 시작하지는 않는다. 대신 별칭을 사용한다. 행동을 시작하는 모든 캐릭터는 별칭 목록을 정의하여, 함수를 사용, 별칭에 해당하는 .behavior 파일을 찾는다. 한 레이어로 행동 참조를 추상화함으로써, 함수 정의 방식을 바꿈으로써 행동 구성요소를 사용자 정의하고 재사용할 수 있다.

HFSM 대신 평면형 FSM을 사용하여 캐릭터를 조종할 경우 단점은 기존 행동의 한 가지 측면을 바꿈으로써 신속하게 행동을 바꿀 수 없다는 데 있다. FSM을 사용하면서 이 문제를 해결하려면 전체 상태 기계를 다시 만들고 저장해야 한다.

우리는 계층 내의 특정 레이어에 있는 행동을 교체하는 기능을 추가함으로써 이 문제를 피할 수 있었다. 주어진 캐릭터의 행동 별칭 함수를 바꾸면 계층 내의 행동 위치에 상관없이 별칭을 참조할 때 만들어진 실제 행동을 교체할 수 있다. 이 특징은 닌자에게 특별한 난투 능력이나 무기발사 능력을 부여할 때와 같이, 널리 사용되는 행동을 사용자 정의할 때 일반적으로 적용되었다(그림 4를 참조).

일반적인 전투 행동 지도
```
MapAlias("Combat", "pedestrian_combat.behavior")
MapAlias("Melee", "pedestrian_melee.behavior")
MapAlias("PathFollow", "pedestrian_pathfollow.behavior")
MapAlias("Patrol", "pedestrian_patrol.behavior")
MapAlias("Protect", "pedestrian_protect.behavior")
MapAlias("Approach", "pedestrian_approach.behavior")...
```

닌자 행동 지도
```
INCLUDE("Common Combat Behaviors Map")
OverrideAlias("Melee", "ninja_melee_claw")
OverrideAlias("FireWeapon", "ninja_throw_shuriken")
OverrideAlias("Flee", NONE)
```

그림 4 일반적인 행동 지도와 난투 수단 및 사용 무기가 다르고, 후퇴를 모르는 닌자의 특별 행동 지도 목록

별칭은 대개 행위자에 직접 연관된 파일 내에 정의되어 있다. 그러나 우리는 행동 데이터 구조 내 어디서라면 별칭이 재정의되게 함으로서 유연성을 늘렸다. 예를 들면 이전에 언급한 행동 보호의 경우 HFSM 내에서 하위 행동으로 사용되는 두 가지 접근 행동이 있는데, 하나는 보호해야 할 캐릭터를 따라가는 것이고 다른 하나는 전투에서 싸워야 할 적에게 접근하는 것이다.

이 두 가지 행동은 개념적으로는 목표에 접근하는 것 같으나 그 방식은 달라야 한다. 예를 들어 전투 시의 접근 방법은 옆걸음을 사용하는 등 한층 더 공격적인 방식이고, 보호 시의 접근 방법은 지켜야 할 표적으로 뛰어가기만 해도 된다.

행동이 포함된 별칭 설정을 무시함으로써 별도의 분기에서 이러한 사용자 정의를 할 수 있다. 이러한 설정이 행동 내에 있는 경우 새로운 함수를 사용하지 않고 자신 또는 더욱 하위 행동을 활성화시키는 하위 행동을 촉발시킨다. 예를 들어 전투 분기 내의 접근을 무시하고 더욱 공격적인 버전을 사용할 수 있는 한편 그에 반대되는 분기가 별도로 이를 무시하여 덜 공격적인 버전을 사용할 수도 있다. 하위 행동이 별칭을 활성화할 때면 언제라도 HFSM 내의 각 상위 행동을 통해 걸러내어 새로운 별칭 함수를 점검해 할당된 것을 찾아낸다.

행동을 사용자 정의하는 또 다른 간단한 방법은 능력을 추가하여 사용자 정의를 할 뿐만 아니라 상위 행동 이하의 모든 행동을 제거하는 것이다. 이는 행동 별칭을 특별한 “none” 키워드에 함수 연결하여, 마주칠 경우 행동을 시작하지 않게 함으로서 가능하다. 이는 수류탄을 던지지도, 피하지도 않는 다양한 적을 만들어낼 때 매우 유용하다.

집단용 AI

모래상자식 플레이 기능을 갖춘 게임의 경우 AI는 게임 세계 내에서 상호작용을 할 수 있는 각양각색의 흥미로운 캐릭터들을 제시해야 한다. 또한 게임 세계의 크기가 너무 거대해져 하드 코드로 처리하기 어려울 정도가 되어서도 안 된다. 가끔씩 행동 설정을 노출시키는 것도 모자라 임무 구조와 하위 임무까지 노출시켜야 할 경우가 있다. 그 경우 강력하고 편리한 방법을 사용해야 한다.

<디시트로이 올 휴먼스2>에서 AI 아키텍처를 감안할 때 우리가 했던 선택은 이러한 특색을 증진시키는 것이었다. 그 기능은 융통성 있는 퍼즐조각 같은 시스템으로, 일반적인 방식으로 노출되어 있다. 하드 코드로만 처리된 행동과 스크립트 정의형 행동 사이에서 테크니컬 디자이너들이 관리할 여지를 남겨둔다.

실제로 판데믹 오스트레일리아의 패키지 같은 복잡한 시스템은 다양한 추상 레벨에서 각 개별 조각이 들어맞는 식으로 노출된다. 그 결과 이러한 조각을 재활용하고 확장하여 활용도를 높이고 가상 세계에 더욱 다양한 가상 인생이 살아갈 수 있도록 할 수 있다.

*[**편집자 주**: 이 기사는 커뮤니티의 가치로 간주되므로 가마수트라의 편집자가 독자적으로 배포한 것입니다. 이 기사의 배포는 Intel's Visual Computing microsite의 플랫폼이자 벤더 독리형 부분인 인텔을 통해 가능했습니다.]*