한국컴퓨터정보학회 동계학술대회 논문집 제24권 제1호 (2016. 1)

Unreal Engine4의 Behavior Tree를 이용한 게임 AI 설계 및 구현

배성진*, 강명주**
*청강문화산업대학교 게임콘텐츠스쿨
e-mail: baehun92@naver.com*, mjkkang@ck.ac.kr**

Design And Development of Game AI Using Unreal Engine 4 Behavior Tree

Sung-Jin Bae*, Myung-Ju Kang**
*School of Game, Chungkang College of Cultural Industries

● 요 약 ●

본 논문에서는 언리얼 엔진4의 Behavior Tree(행동 트리)를 이용하여 NPC의 다양한 상태와 움직임을 가진 어드벤처 게임 AI를 설계 및 개발하였고, 그 효율성을 분석하였다. Behavior Tree는 상태와 행동을 계층적으로 나누어 AI의 행동을 결정하는 알고리즘으로 FSM(Finite State Machine, 유한상태기계)과 비교하여 유지보수와 행동 규칙 검증의 어려움을 해결하는 데 장점이 있음을 확인하였다.

키워드: 언리얼 엔진4(Unreal Engine4), Behavior Tree, FSM

I. Introduction

최근 어드벤처 게임 AI는 플레이어에게 더 큰 흥미를 주기 위해 많은 행동패턴을 요구한다[1]. 많은 행동패턴을 표현하기 위해 AI 상태가 많아지고 AI 상태가 많아질수록 AI 알고리즘의 유지보수 및 행동 규칙의 검증이 어려워진다. Behavior Tree는 이러한 어려움을 해소할 수 있는 AI 설계 방법 중의 하나이다. Behavior Tree가 적용된 게임으로는 대표적으로 헤일로2가 있고 현재 개발 중인 게임 AI의 대부분이 Behavior Tree로 제작되는 추세이다[2]. 본 논문에서는 다양한 상태와 행동 패턴이 있는 AI를 Behavior Tree를 적용하여 설계 및 구현하였다. 본 논문의 구성은 2장에서 AI 알고리즘 소개 3장에서는 언리얼 엔진4에서 제공하는 Behavior Tree의 특징을 알아보고 4장에서 FSM과 언리얼 엔진4에서 제공하는 Behavior Tree를 이용해 AI를 설계 및 구현하여 비교하였고 5장에서는 결론을 맺는다.

II. Preliminaries

1. Related works

AI 알고리즘 중 간결한 알고리즘은 FSM이다. FSM은 게임 AI의 상태를 결정하는 방법 중 하나로 상태와 상태전이를 통해 게임 AI의 행동과 상태를 명료하게 나타낼 수 있다. 그러나 게임 AI의 상태가 많은 경우 게임 AI의 상태에 해당하는 모든 상태전이를 구현해야 한다. 즉, FSM은 게임 AI의 상태가 많아질수록 많은 규칙으로 복잡도가 증가해 행동 규칙의 검증과 유지보수에 어려움이 있다[3].

위와 같은 문제점으로 인해 HFSM(Hierarchical Finite State Machines, 계층적 유한상태기계)을 사용한다. HFSM는 세분화 되는 레이어를 통해 상태수의 증가 문제를 부분적으로 해결하는 설계 기법이다[4]. 하지만 FSM보다 상태수의 추가 삭제에 따른 유지관리의 문제를 감소시켜줄 뿐 해결해 주지 못한다.

따라서, 본 논문에서는 이러한 문제점을 해결하기 위하여 Behavior Tree를 이용한 AI 설계 및 구현 방법을 제안한다.

III. Behavior Tree

언리얼 엔진4에서 제공하는 Behavior Tree의 특징은 다음과 같다.

1. 이벤트 주도형

언리얼의 엔진4 Behavior Tree는 기존 Behavior Tree와 다르게 이벤트 주도형 Behavior Tree로 모든 프레임마다 게임 AI의 상태를 지속해서 검사 하지 않는다. 이벤트 주도형은 AI의 상태 변화를 알려주는 이벤트가 발생하면 동작을 하는 것으로 이 이벤트는 블랙보드에서 발생이 된다. 블랙보드는 게임 AI의 정보가 저장되는 공간으로 그림 1 과 같이 AI의 상태가 변경이 되면 Behavior Tree에 이벤트를 보내 Behavior Tree가 동작을 하게 된다[5].

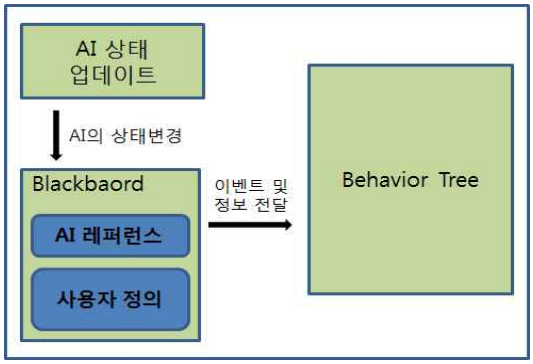

Fig. 1. Transform the event of blackboard

2. 노드의 구성

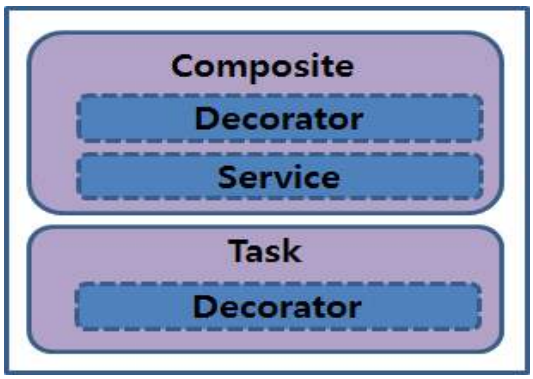

Fig. 2. The node structure of Behavior Tree

그림 2는 언리얼 엔진4 Behavior Tree의 노드 구성으로 이 노드들을 이용하여 계층적인 구조를 가진다. 언리얼 엔진4 Behavior Tree의 노드는 Composit, Task 노드가 있고 이를 도와주는 Decorator, Service가 있다. Composit 노드는 연결된 자식노드의 실행 규칙을 정하는 노드로 Decorator, Service가 적용된다. Task 노드는 게임 AI의 동작을 실행하는 노드로 Behavior Tree의 말단에 존재하고 Decorator가 적용된다.

Decorator는 노드의 실행 여부를 결정하며. Service는 블랙 보드의 검사 및 업데이트에 사용된다[5].

IV. The Proposed Scheme

본 논문에서는 언리얼 엔진4의 Behavior Tree를 적용하여 게임 AI를 설계하고 구현하였다.

1. 게임 소개

본 논문에서 구현할 게임은 어드벤처 게임으로 플레이어 남성 캐릭터와 게임 AI 여성 캐릭터가 나와 주변 사물 혹은 캐릭터와 상호작용을 하며 퍼즐을 풀어나가는 게임이다. 게임 AI는 플레이어에게 더 큰 흥미를 주기위해 많은 상태와 상호작용을 가지게 된다.

2. FSM 설계

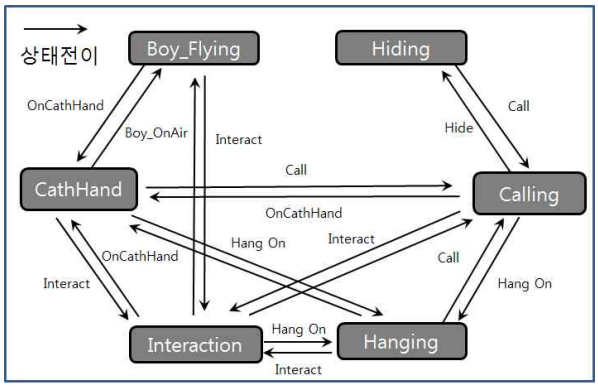

Fig. 3. FSM for the game

FSM을 이용하여 AI를 설계는 그림 3과 같이 6개의 상태를 이용하여 행동을 결정을 하고 있다. 각각의 상태는 상태전이를 이용하여 다른 상태로 이동을 하고 만약 상태를 추가 또는 제거하게 된다면 수정한 상태와 관련된 모든 상태전이를 관리해 주어야 한다.

3. Behavior Tree 설계

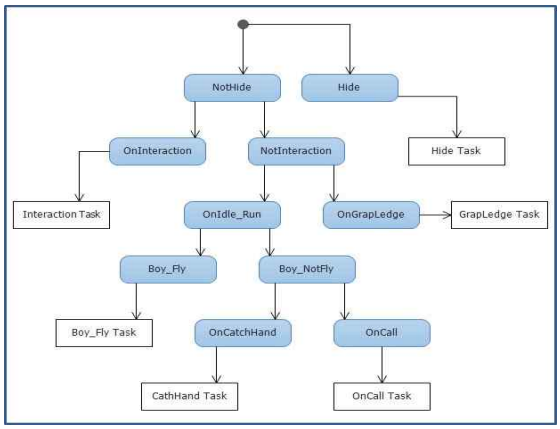

Fig. 4. AI design using Behavior Tree

그림 4는 그림 3의 FSM을 Behavior Tree로 설계한 것이다. Behavior Tree는 노드로 이루어져 있고 이 노드들은 AI 상태를 사용하여 동작한다. 각 노드가 사용하는 AI 상태의 우선순위에 따라 노드의 우선순위가 정해진다. 우선순위가 높은 노드는 우선순위가 낮은 노드들에 비해 상위 노드에 존재 한다. 우선순위가 낮은 노드가 동작 중에 우선순위가 높은 노드의 상태의 변화가 일어나면 우선순위가 낮은 노드의 동작을 멈추고 우선순위가 높은 노드의 동작을 한다. 또한 AI 상태가 추가 또는 제거시에는 수정된 AI 상태를 사용하는 노드를 우선순위에 맞게 추가, 제거하여 재배치를 한다.

4. AI 실행 결과

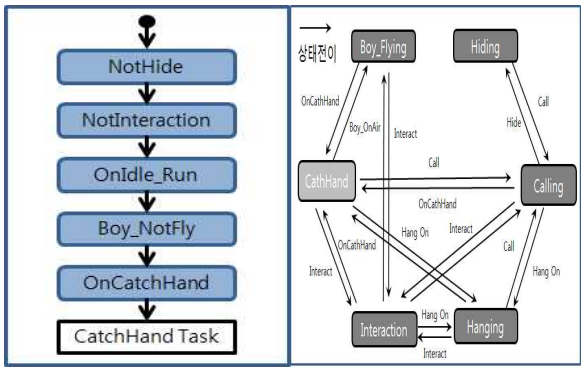

Fig. 5 Behavior Tree, CatchHand state of FSM

그림 5는 AI가 CatchHand 상태의 Behavior Tree와 FSM의 AI 알고리즘을 나타낸다. 그림 5의 Behavior Tree는 AI 알고리즘의 로직이 그대로 드러나 현재 실행되는 AI의 행동과 행동을 결정하기까지 검사한 AI 상태를 명확하게 알 수 있다. 반면 그림 5의 FSM은 현재 AI 상태만을 확인 할 수 있다.

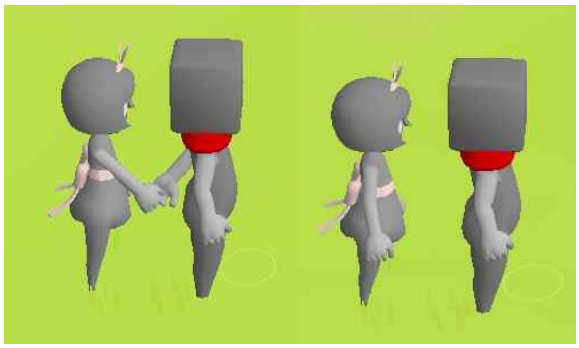

Fig. 6. CatchHand state(left) and Idle state(right)

그림 6은 그림 5의 AI 알고리즘을 이용하여 AI의 실행 화면으로 CatchHnad상태와 Idle상태의 행동을 보여준다.

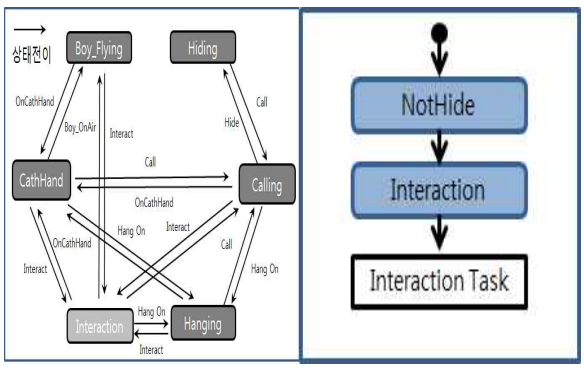

Fig. 7. Behavior Tree, Interaction state of FSM

그림 7은 AI가 Interaction 상태의 Behavior Tree와 FSM의 AI 알고리즘을 나타낸다. 그림 5과 마찬가지로 Behavior Tree는 AI 알고리즘이 그대로 드러나는 반면 FSM는 현재 AI 상태만을 확인 할 수 있다.

Fig. 8. Interaction state(left) and Idle state(right)

그림 8은 그림 7의 AI알고리즘을 이용하여 AI의 실행 화면으로 Interactrion상태와 Idle상태의 행동을 보여준다.

V. Conclusions

본 논문에서는 Behavior Tree와 FSM을 이용한 AI설계 및 제작을 통해 Behavior Tree와 FSM을 비교분석하였다. FSM은 AI 상태가 많은 경우 상태에 해당하는 모든 상태전이를 구현해야 하기 때문에 복잡도가 증가하고 유지보수와 행동 규칙의 검증에 어려움이 있는 반면, Behavior Tree는 계층적으로 상태와 행동을 나누어 복잡도의 증가를 해결하고 AI 알고리즘의 로직이 그대로 들어나 유지보수와 행동 규칙의 검증의 어려움이 해결되는 장점을 있음을 알 수 있었다.

References

[1] K.D. Kwon, I.C. Kim ”Artificial Intelligence Computer Game". Review of Korean Society for Internet Information 8(4), 2007.12, 67-74 (8 pages)

[2] "[NDC 2009] AI Development using Behavior Tree“, http://www.slideshare.net/yonghakim900/2009-ndc?qid=9881c675-a795-46bc-a62c-7b6eea3be9af&v=qf1&b=&from_search=4)

[3] Hyungil Kim, Hyunnim Yoon "The Control of Character's Behavior by Using FSN_Based Probability Estimation in Games. JOURNAL OF KOREA MULTIMEDIA SOCIETY, 8(9), 1269-1281. 2005.9

[4] Eunjung Lee, Gyu-Wan Kim. "Design and Implementation of Mobile App Authoring Tool Using HFSM". Journal of Korean Institute of Information Technology, 13(3), 163-173.2015.3

[5] Unreal Engine docs (https://docs.unrealengine.com)