http://dx.doi.org/10.14400/JDC.2016.14.8.415

게임 엔진 행동 트리 제안

이면재
백석대학교 정보통신학부

A Proposal on Game Engine Behavior Tree

Myoun-Jae Lee
Division of Information Communication, BaekSeok University

요 약 행동 트리는 인공지능의 행동을 표현하는 트리로 FSM(Finite State Machine)보다 상태 전이가 용이하고 행동의 진행을 쉽게 알 수 있는 특징을 갖고 있다. 때문에 최근 FSM보다 널리 쓰이고 있는 추세이다. 본 논문은 이러한 배경에서 게임 엔진의 행동 트리의 장단점을 분석하고 이를 바탕으로 개선된 행동 트리를 제안하기 위한 것이다. 이를 위해 본 논문에서는 첫째, 유니티 엔진과 언리얼 엔진의 행동 트리 구조와 노드들의 역할을 먼저 살펴본다. 둘째, 살펴본 행동 트리의 구조와 노드들을 바탕으로 장점과 단점을 논한다. 셋째, 이 행동 트리들의 단점인 트리의 깊이와 실행 노드 검색 시간을 개선한 행동 트리를 제안한다. 본 논문은 추후 행동 트리를 사용해 게임 개발을 하려는 개발자들에게 도움을 줄 수 있다.

주제어 : 행동 트리, 게임 엔진, FSM, 인공지능, 유니티 엔진

Abstract A behavior tree is to express the behavior of artificial intelligence. The behavior tree has a characteristic that is easy to change state transitions than FSM(Finite State Machine), see the progress of the action. For these reasons, the behavior tree is widely used in more than FSM. This paper is to analyze the advantages and disadvantages on behavior trees of game engines, proposes the improved behavior tree based on analyzed them. To achieve this, in this paper, first, examines the role of node and the behavior tree structure of the unity engine, unreal engine. Second, discusses the advantages and disadvantages based on it. Third, proposes the behavior tree to improve the disadvantages of behavior tree of unity engine and unreal engine, depth of behavior tree and search time required to select the execution node. This paper can help developers using the tree to develop the game.

Key Words : Behavior Tree, Game Engine, FSM, Artificial Intelligence, Unity Engine

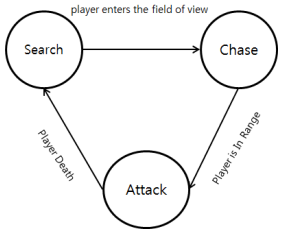
1. 서론

플레이어와 캐릭터가 격투를 하는 게임이든, 다른 세계를 구하며 모험을 하는 게임이든 게임에는 한 가지 공통점이 있다. 그것은 바로 '인공지능'이라고 하는 기능이다. 인공지능은 게임 내에서 플레이어를 제외한 스스로

* This research is supported by 2016 Baekseok University Fund.
Received 28 June 2016, Revised 29 July 2016
Accepted 20 August 2016, Published 28 August 2016
Corresponding Author: MyounJae Lee
(Division of Information Communication, BaekSeok University)
Email: davidlee@bu.ac.kr

ISSN: 1738-1916

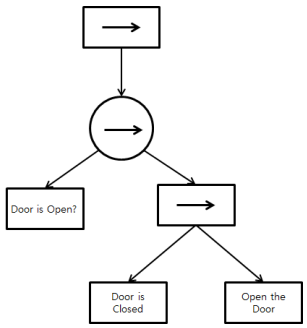
자율성을 가진 캐릭터들의 지능을 제작하는 방법이다. RPG의 경우 스토리 진행에 도움을 주는 게임내의 캐릭터들이나 초보자들을 안내해 주는 NPC, 격투게임에서는 직접 플레이어의 상대가 되는 캐릭터들이 이에 포함된다. 인공지능 설계는 규칙 기반 시스템(RBS), Planning, 유한 상태 기계(FSM), 동물의 행동패턴[2] 등 다양한 방법이 사용 된다[1]. 인공지능은 군중 시뮬레이션[3], 컴퓨터 통신[4], 생활환경[5], 지능형 캐릭터 구현[6,7], 지능형 교육 시스템[8], 지능형 침입탐지 시스템[9],얼굴인식[10], 상태 기계[17], 군중 시뮬레이션[18], 네트워크[19]등에 사용된다. 이중 FSM은 현재 가장 많이 사용되는 기법으로, 유한한 여러 개의 상태(행동)들을 정의하고, 상태들 사이의 인과 관계를 그래프로 표현한다. 상태의 구성과 전이에 대한 이해가 쉽지만, 상태와 전이가 많아질수록 관리가 어려워진다는 문제점이 있다[2]. [Fig. 1]은 FSM 예를 보여준다. Search, Chase, Attack 상태로 구성되어 있으며 상태 전환 조건이 화살표 위에 표시되어 있다.



[Fig. 1] FSM

이를 개선하고 좀 더 정교한 인공지능을 만들기 위해 개발된 것이 행동 트리(Behavior Tree)이다. FSM과 유사한 형태로 표현되고 구현과 이해가 쉬운 FSM의 장점을 그대로 가지고 있다. 또한, 상태와 상태 사이의 관계가 많아질수록 관리가 힘든 FSM의 단점이 보완되어서 보다 복잡한 인공지능을 가진 캐릭터를 구현하기에 적합하다[11,12,13]. 이러한 이유로 현재 이용이 증가하는 추세이다. [Fig. 2]는 행동 트리의 예를 보여준다.

이러한 행동 트리는 게임 플레이어의 상태와 행동을 제어하려는 RPG(Role Playing Game), RTS(Real Strategy Game), FPS(First Person Shooting)장르이면 모두 사용 가능한 특징을 갖고 있다. 단 객체의 움직임이나 제어 논리가 크게 사용되지 않을 퍼즐 게임의 경우 행동 트리를 사용할 필요가 없을 것으로 판단된다.



[Fig. 2] Behavior Tree

본 논문은 상용 게임 엔진에서 사용하고 있는 행동 트리의 구성 요소와 해당 행동 트리의 장단점을 비교 분석하고. 이들의 장점을 조합한 행동 트리를 제안하는 것을 목표로 하고 있다.

논문의 구성은 다음과 같다. 2장에서는 행동 트리의 구성 요소에 대해 설명하고 행동 트리가 사용된 유니티 게임 엔진과 언리얼 게임 엔진의 장점과 단점을 논한다. 3장에서는 첫째, 2장에서 논했던 단점들을 개선한 행동 트리를 제안한다. 둘째, 제안된 행동 트리를 유니티 게임 엔진의 행동 트리와 언리얼 게임 엔진의 행동트리와 논리적으로 비교 분석한다. 4장에서는 결론 및 추후 연구방향에 대해 기술한다.

2. 행동 트리

본 장에서는 유니티 엔진과 언리얼 엔진의 행동 트리에 대해 장점과 단점을 기술한다. 이를 위해 첫째, 유니티 엔진과 언리얼 엔진에서의 행동 트리에 대한 구성 요소를 설명한다. 둘째, 특정 캐릭터의 AI를 유니티 엔진과 언리얼 엔진에서 제공하는 행동 트리로 제작하여 각 엔진의 행동 트리의 장점과 단점에 대해 기술한다.

2.1 행동 트리 구성 요소

행동 트리는 캐릭터 행동에 대한 AI를 트리로 정의하는 방법으로 캐릭터 행동에 대한 조건과 인과 관계, 우선순위 등에 해당하는 노드들을 연결함으로써 완성된다.

<Table 1>은 유니티 엔진과 언리얼 엔진의 행동 트리에서 제공하는 노드들을 보여준다. 유니티 엔진과 언리얼 엔진을 비교하는 이유는 행동 트리를 제공하는 게임 엔진중에서 가장 상용화된 게임 엔진이고, 이 엔진들

에서 실지 개발할 때 행동 트리를 많이 사용하기 때문이다. 또한, 구글 엔진을 통해 게임 엔진과 행동 트리를 검색해 보았지만 이 두 엔진을 제외하고 추가적으로 검색되는 게임 엔진이 없었기 때문이다.

액션 노드, 데코레이터 노드, 시퀀스 노드, 실렉터 노드의 역할은 비슷하지만 서비스 노드와, 우선순위 셀렉터 노드, 패럴럴 노드의 구성이 다름을 알 수 있다.

〈Table 1〉 Unity Engine[11]

Node	Symbol	Function
Action		Basic node
Decorator		define whether or not a branch in the true
Sequence		ticks its children sequentially until one of them returns FAILURE, RUNNING or ERROR, If all children return the success state, the sequence also returns SUCCESS
Selector		ticks its children sequentially until one of them returns SUCCESS, RUNNING or ERROR. If all children return the failure state, the priority also returns FAILURE
Priority selector		decide the highest priority node
Parallel		Child nodes are executed at the same time

〈Table 2〉 Unreal Engine[14]

Node	Symbol	Function
Task		Basic node
Decorator		define whether or not a branch in the true
Sequence		ticks its children sequentially until one of them returns FAILURE, RUNNING or ERROR, If all children return the success state, the sequence also returns SUCCESS
Selector		ticks its children sequentially until one of them returns SUCCESS, RUNNING or ERROR. If all children return the failure state, the priority also returns FAILURE.
Service		Running on a fixed frequency Update Blackboard

두 엔진의 행동 트리에서 보이는 가장 큰 차이점은 노드의 우선순위를 결정하는 우선순위 셀렉터 노드가 유니티 엔진에서는 제공되지만 언리얼 엔진에서는 제공되지 않는다는 점이다.

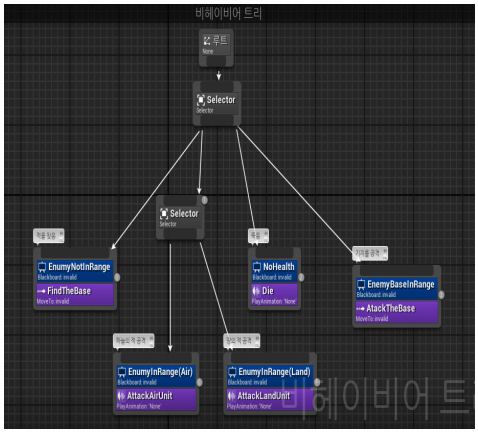
2.2 행동 트리 비교 분석

두 엔진의 행동 트리가 갖고 있는 공통된 가장 큰 장점은 한눈에 알아보기 쉽다는 것이다. 트리 형태로 표현되기 때문에 하나의 행동이 이루어지기 위한 조건과 다음 행동이 무엇인지를 쉽게 알아볼 수 있는 장점이 있다.

본 절에서는 두 가지 엔진으로 만든 동일한 행동에 대한 행동 트리를 예시로 장점과 단점에 대해 기술한다.

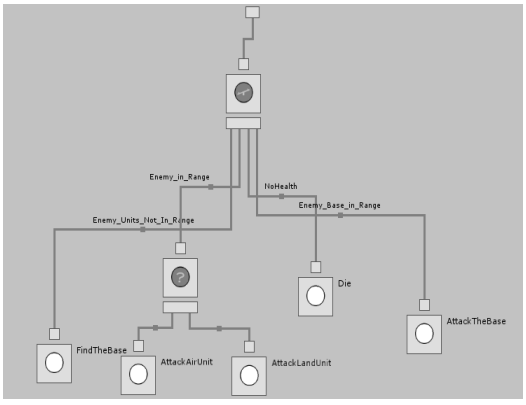
본 논문에서 사용한 행동 AI는 캐릭터 현재 위치를 중심으로 일정 범위 내에 기지가 없는 경우 기지를 찾기 위해 이동 전진하고. 전진 도중에 적(enemy)을 만나는 경우 적을 공격하고, 체력이 50 미만일 경우 아이템을 사용하고, 그렇지 않을 경우에는 상대의 기지를 찾는 것을 계속한다. 이때 기지를 발견하는 경우 기지를 공격하는 것을 행동으로 설정한다.

[Fig. 3]은 언급된 행동에 대한 언리얼 엔진의 행동 트리를 보여준다. 첫 번째 레벨에 루트 노드가 있으며 두 번째 레벨에 셀렉터 노드가 있다. 셀렉터 노드는 적이 일정 영역에 없으면 기지를 공격하는 EnemyNotInRange 리프 노드와 기지가 일정한 영역에 있으면 기지를 공격하는 EnemyBaseInRange 리프 노드, 공격 방법을 결정하는 셀렉터 노드로 구성되어 있다. 이 셀렉터 노드는 적이 Air Unit으로 공격할 일정 범위에 있으면 공격하는 EnemyInRange(Air) 리프 노드와 적이 Land Unit으로 공격할 일정 범위에 있으면 공격하는 EnemyInRange(Land) 리프 노드로 구성되어 있다. 언리얼 엔진에서는 상황에 맞는 행동을 선택하기 위해 데코레이터 노드를 사용하는데, 이 노드의 조건을 매번 왼쪽노드부터 순차적으로 검사해야 한다. 예를 들어 두번째 레벨의 셀렉터 노드의 경우 EnemyNotInRange 노드의 조건과 셀렉터 노드의 조건, EnemyBaseInRange 노드의 조건을 순차적으로 검색한다. 따라서, 순차적인 검색 노드의 수에 따라 실행될 노드를 결정하는 시간이 길어질수 있다. 현재 행동 트리의 깊이는 4로 구성되어 있다.



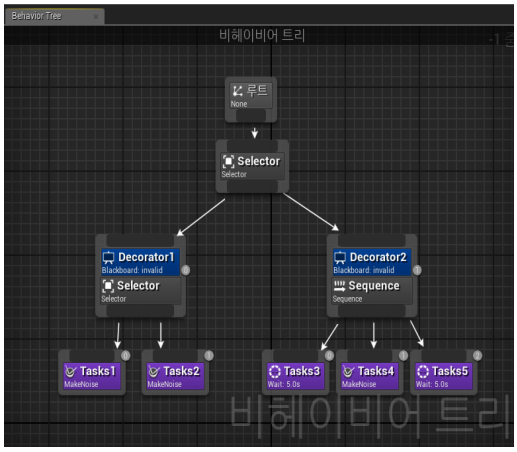
[Fig. 3] Unreal Engine Behavior Tree

[Fig. 4]는 유니티 엔진의 행동 트리를 보여준다. 유니티 엔진에서는 언리얼 엔진과 다르게 우선순위 셀렉터 노드를 제공하는데 이 노드를 사용함으로써, 모든 노드들을 순회하지 않아도 상황에 맞는 노드가 바로 실행되도록 해준다. 우선순위 셀렉터 노드는 Enemy_Unit_Not_In_Range, Enemy_in_Range, NotHealth, Enemy_Base_In_Range 노드에 우선순위를 정할 수 있다. 행동 트리의 깊이도 3이 되어 언리얼 엔진보다 행동 트리의 깊이가 작아서 리프 노드를 실행하는 시간이 언리얼 엔진에서 보다 짧아질수 있다.

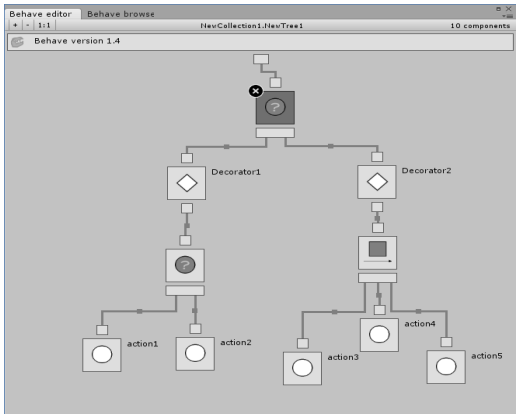


[Fig. 4] Unity Engine Behavior Tree[11]

[Fig. 5]와 [Fig. 6]의 행동 트리는 언리얼 엔진 행동 트리의 장점을 나타내기 위해 임의로 작성된 행동 트리이다. 두 행동 트리 모두 노드 구성이 동일한데, 차이점은 언리얼 엔진의 행동 트리의 경우 데코레이터 노드가 조건을 적용해야하는 노드에 연결되어 있는데 비해, 유니티 엔진의 행동 트리는 그렇지 않다는 것이다. 이는 많은 조건을 검사하게 될 경우 어떤 조건이 필요한지를 보다 직관적으로 볼 수 있게 해준다. 또한, 조건이 많아져서 추가해야할 노드가 많아지는 경우 트리의 깊이가 깊어지는 유니티 엔진에 비해 조건을 붙혀서 같은 레벨에 나열하는 언리얼 엔진은 트리의 깊이가 짧아진다는 장점이 있다.



[Fig. 5] Unreal Engine Behavior Tree

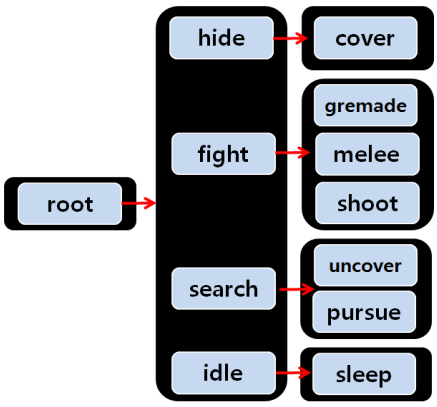


[Fig. 6] Unity Engine Behavior Tree

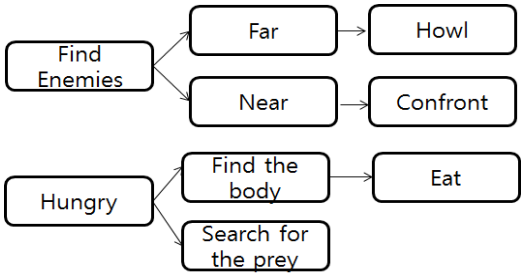
[Fig. 7]의 행동 트리는 HALO2 게임 캐릭터에 사용된 행동 트리이다. 다양한 행동들이 들어갔음에도, 행동 트리의 사용으로 행동이 어떻게 진행되는지 한눈에 알아보

기 쉽다.

[Fig. 8]은 듀랑고 게임의 동물 AI에 대한 행동 트리를 보여준다. 각 동물의 상태에 따라 행동이 달라지기 때문에 각 동물들에 대한 상태 유지와 생성에 많은 자원이 필요하다는 문제가 생길 수 있다. 이 문제점을 개선하기 위해 동물들을 무리로 묶는 무리 AI를 만들었다. 무리 AI는 어미 AI가 무리의 행동을 결정하고 야생에서 발생할 수 있는 다양한 상황에 따라 특정 행동을 어미가 선택하면 이를 무리를 구성하는 다른 동물들이 따라 할 수 있도록 행동 트리가 구성되었다[16].



[Fig. 7] HALO 2 Behavior Tree[15]



[Fig. 8] Behavior Tree Durango

3. 행동 트리 제안

본 장에서는 유니티 엔진과 언리얼 엔진에서의 행동 트리의 단점들을 보완할 수 있는 행동 트리를 제안하고 트리의 깊이와 실행 노드를 결정하는데 소요되는 검색 시간을 중심으로 비교 분석한다.

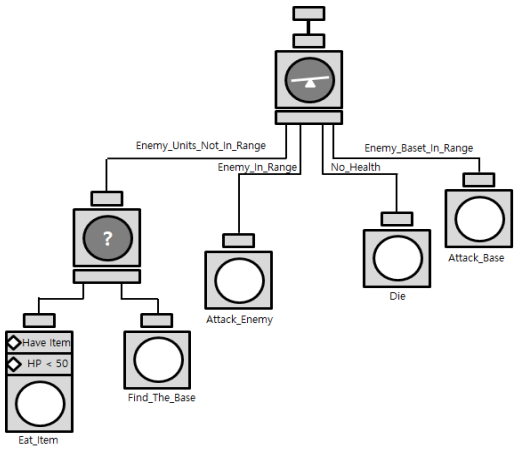
제안된 행동 트리의 비교 분석을 위해 실지 실행시간과 메모리 양을 비교하려고 하였으나 행동 트리를 스크립트 언어를 이용하여 자동적으로 생성하는 방법이 제공되지 않아서 논리적으로 그림을 통해 트리의 깊이와 실행 노드 결정시간을 비교한다. 트리의 깊이가 길어질수록 리프 노드까지 도달하는 시간이 오래 소요되고 실행 노드를 결정하기 위한 검색 시간이 커질수록 최종 행동을 실행하는데 소요되는 시간이 늦어질수 있기 때문에 이 2가지 요소를 중심으로 비교한다.

본 연구에서 제안한 행동 트리는 2장에서 설명했던 두 엔진의 행동 트리의 장점을 조합하는 행동 트리이다. 즉 유니티 엔진의 우선순위 셀렉터 노드와, 언리얼 엔진에서 노드에 직접 붙어 조건을 비교하는 데코레이터 노드를 동시에 사용하는 행동 트리이다.

[Fig. 9]의 행동 트리는 상대의 기지를 찾아가는 캐릭터 AI를 표현하고 있다. 범위 내에 상대의 기지를 찾는데 기지가 없을 경우, 두 가지의 행동중 하나를 선택해서 수행한다. 조건을 검사해 아이템을 갖고 있고 체력이 50 미만일 경우 아이템을 사용하고, 그렇지 않을 경우에는 상대의 기지를 찾는다. 만약 적을 만나게 된다면 상대를 공격한다. 체력이 없어진다면 캐릭터는 죽고, 범위 내에서 적 기지를 발견했다면 적의 기지를 공격한다. [Fig. 9]에서와 같이 제안된 행동 트리는 언리얼 엔진의 장점인 데코레이터 노드와 유니티 엔진의 장점인 우선순위 셀렉터 노드를 조합하여 구성한다.

제안된 행동 트리는 유니티 엔진의 장점인 우선순위 셀렉터를 사용하기 때문에 언리얼 엔진의 행동 트리와 같이 매번 상황에 맞는 행동을 찾기 위해서 행동이 끝날 때마다 일일이 데코레이터 노드를 사용해 모든 조건을 검사하거나 트리를 처음부터 끝까지 모두 살펴볼 필요가 없다. 이는 노드를 탐색하는 수행시간을 줄여주는 장점이 있다. 즉 우선순위 실렉터를 사용하여 실행빈도가 가장 높은 노드부터 실행하기 때문에 매번 순차적으로 실행될 노드를 결정하는 언리얼 엔진에 비해 실행할 노드를 결정하는 검색 시간이 짧아질수 있다.

또한, 언리얼 엔진에서처럼 데코레이터 노드를 조건이 필요한 노드에 붙여서 사용한다. 이로 인해 트리의 깊이가 커지고 복잡해지는 단점을 보완할 수 있다. 결과적으로 언리얼 엔진에서는 행동 트리 깊이가 4([Fig .3])이었는데, 3으로 감소되었다.



[Fig. 9] The proposed Behavior Tree

4. 결론 및 추후 연구 방향

인공 지능을 사용하는 개체, 즉 캐릭터와 같이 움직이고 행동하는 개체의 행동 관계를 나타내는 행동 트리는 FSM에 비해 직관적이고 이해하기 쉬운 장점이 있다.

본 연구에서는 상용 엔진인 유니티 엔진과 언리얼 엔진의 행동 트리의 구성 요소를 살펴보고 이 구성 요소들을 중심으로 장단점을 비교 분석하고 이를 개선한 행동 트리를 제안하였다.

비교 결과, 언리얼 엔진은 노드에 직접 붙어 조건을 비교하는 데코레이터 노드로 인해 실행될 노드를 결정하는 시간이 짧아질수 있지만 유니티 엔진에서의 우선순위 셀럭터 노드가 제공되지 않기 때문에 매번 순차적으로 실행될 노드를 결정해야 하는 단점이 있었다. 이와 비교하여 유니티 엔진에서는 데코레이터 노드에 조건을 적용해서 실행할 노드가 붙어 있지 않기 때문에 언리얼 엔진에서보다 트리의 깊이가 깊어질 수 있지만 우선순위 셀렉터가 있어서 현재 상황에 적합한 실행 노드를 빠른 시간에 결정할 수 있는 장점을 갖고 있다.

본 논문에서는 이 두 엔진의 장점, 언리얼 엔진의 데코레이터 노드와 유니티 엔진의 우선순위 셀렉터 노드가 포함된 행동 트리를 제안하고 이에 대한 우수성을 보여주기 위한 연구이다.

이를 위해 본 연구에서는 캐릭터 AI가 동일한 상황에서 유니티 엔진과 언리얼 엔진에서의 행동 트리, 그리고 제안된 행동 트리를 트리의 깊이와 실행 노드 검색 시간 면에서 논리적으로 비교하였다.

비교 결과, 트리 깊이 비교에서 제안 행동 트리는 언리얼 엔진에서의 트리 깊이보다 우수하였고, 실행 노드 검색시간 비교에서는 유니티 엔진에서의 실행 노드 검색 시간보다 우수함을 보였다.

추후에는 각각의 단점들을 보완할 수 있는 방법뿐만 아니라, 보다 좋은 기능을 가진 노드들을 제안하여 게임 엔진의 행동 트리에 이바지될 수 있게 하는 것이 이 연구의 최종 목표이다.

ACKNOWLEDGEMENT

"This research is supported by 2016 Baekseok University Fund."

REFERENCES

[1] Lee ManJae, "Artfical Intelligence in Game", Journal of KIPS, Vol.9, No.3, pp.69-76, 2002.

[2] MuounJae Lee, "Implementation of NPC Artificial Intelligence Using Agonistic Behavior of Animals", Journal of Digital Convergence, Vol.12, No.1, 2014.

[3] Yong-Gwan Kim, "Study on the Principle of Crowd Simulation Applying Artificial Intelligence", The Korean Journal of Animation, Vol.5, No.3, 2009.

[4] YoungIm Cho, "A Study on Application of Artificial Intelligence in Computer Networks", PYONGTAEK REVIEW, Vol.11, 1998.

[5] Mee Kyung Nam, "A Study on Present Condition of Development and Market of Artificial Intelligence Robot In and Out of Country", Journal of KSDC, Vol.16, No.2, 2010.

[6] Byeong Heon Cho, Sung Hoon Jung, et.al, "Artificial Intelligence : An Implementation of Intelligent Game Characters using Neural Networks", The KIPS Transactions : Part B. Vol.11, No.7, 2004.

[7] Myun Sub Lee, Byeong Heon Cho, et.al, "An Intelligent Characters for Fighting Action Games

Using Genetic Algorithms", The KIPS Transactions : Part B, Vol.12, No.3, 2005.
[8] Yong Beom Kim, Yung Sik Kim, "Artificial Intelligence : Development of a Adaptive Knowledge Base Object Model for Intelligent Tutoring System", The KIPS Transactions : Part B, Vol.14, No3. 2006.
[9] Hyeon-Uk Le, Ji-Hun Kim, Hyunchul Ahn, "An Integrated Model based on Genetic Algorithms for Implementing Cost-Effective Intelligent Intrusion Detection Systems", Journal of Kiiss, Vol.18, No.1, 2012.
[10] Kyungul Bae, "Implementation and Design of Artificial Intelligence Face Recognition in Distributed Environment", Journal of Kiiss, Vol.10, No.1, 2004.6.
[11] Aung Situ, Cliford Peters, "Unity game AI programming to implement a variety of artificial intelligence technology to Unity", Acon press, 2015.
[12] MyounJae Lee, Implementation of NPC Artificial Intelligence Using Agonistic Behavior of Animals, Journal of Digital Policy Society, Vol.15, No.1, pp.555-561, 2014.
[13] MyounJae Lee, "An Artificial Intelligence Evaluation on FSM-Based Game NPC, Journal of KCGS, vol.14, No.5, pp.127-136, 2014.
[14] https://docs.unrealengine.com/latest/KOR/Engine/AI/ BehaviorTrees/NodeReference/index.html
[15] http://www.slideshare.net/yonghakim900/2009-ndc
[16] http://betanews.heraldcorp.com/article/631183
[17] JinSeok Seo, "HSM(Hierarchical State Machine) based LOD AI for Computer Games", Journal of Digital Policy Society, Vol.14, No.2, pp.143-169, 2013.
[18] SunJun Yoo, "A Study on Types and Limitations of Control Systems in Computer Game Artificial Intelligence", Journal of Digital Policy Society, Vol.7, No.1, pp.35-40, 2006.
[19] KI-Dom Lee, "A Exploration of Neural Network Development Methodologies", Journal of Digital Policy Society, Vol.9, No.4, pp.91-101, 2011.

이 면 재(Lee, MyounJae)

- 1994년 2월 : 홍익대학교 이학석사
- 2006년 8월 : 홍익대학교 이학박사
- 2009년 3월 ~ 현재 : 백석대학교 정보통신학부 교수
- 관심분야 : 기능성 게임, 게임 제작
- E-Mail : davidlee@bu.ac.kr