The 33rd Annual Conference of the Japanese Society for Artificial Intelligence, 2019

A New Character Decision-Making System by combining Behavior Tree and State Machine

Youichiro Miyake[*1]

[*1] SQUARE ENIX CO., LTD.

Abstract: We have developed a new decision-making system that combines behavior trees and state machines into a single system. The system has both flexibility of behavior tree and strict control of state machines to give a scalability to development of a character AI. The new decision-making system, we call the AI Graph, extends the node formalism to enable sharing nodes between FSMs and Behavior Trees, provides advanced techniques for code reuse using trays which organize code reuse and behavior blackboards, and also provides many features for integrating with detailed low-level character behavior

1. Overview

Originally, behavior Tree and state machine are independent technologies. They each have different good and bad points. The intent behind Behavior Trees[Isla 01,02.5a,5b] is to make a series of character behaviors whereas the intent behind Finite State Machines is to make a stable cycle of character actions. A new system, AI Graph, can have multilayers of both state machine and behavior tree. Any node of any layer can have a state machine or behavior tree as a lower layer (Figure 1).

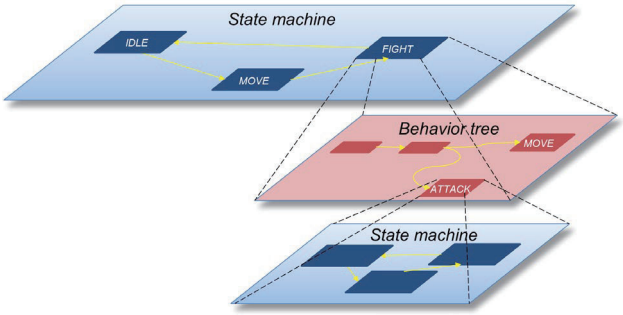

Figure 1. A concept of AI Graph

2. AI Graph Structure and Operation Principle

2.1 Operation principle

An AI Graph is a node based graph system to make a hierarchical structure with a GUI-based node graph tool during game development (Figure 2). AI program execute data made by the AI Graph tool. This is a data-driven system.

Level designers can make a multi-layered decision-making for each character by using a visual node graph tool called the AI Graph. For example, for the first step, a level designer makes a top layer state machine with several states by setting and connecting state machine nodes. Then the level designer can make a new state machine as a sub-state of one or more of the top-level states, or the designer can also make a new behavior tree inside any of the top-level states. Furthermore, the level designer can then make new state machines or behavior trees inside each subsequent sub-state. In this way, the level designer can make a hierarchical structure of state machines and behavior trees by simply editing nodes on the tool.

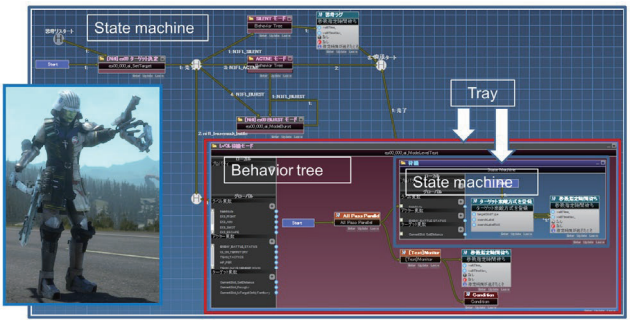

Figure 2. AI Graph tool image

Each layer of the AI Graph also has a blackboard system by which the designer can register variables used in the game. By connecting the blackboards of separate nodes, the different layers can share and use these variables.

2.2 Runtime execution process

An AI Graph user can generate data to be executed by the AI program. AI program execution process is :

(1) A decision making process executes a top layer. Go to (2).
(2) A decision making process executes a start node in the next layer.
a) When a node is primitive, after the node's process finishes, the next node is executed. Go to (2-a)
b) When a node is composite, the process executes a state machine or behavior tree in the node. Go to (2).

So the process continues executing nodes until it cannot go to a deeper layer. It then returns to a higher layer after finishing a lower layer.

When a state machine' transition happens in an upper layer, the state currently executing lower layers must be finished. In this case, after all processing of lower layers has finished, the transition occurs.

3. AI Graph Tool

By using the AI Graph tool, a user can make a state machine or behavior tree for each layer (Figure 3). To make the next layer, a user can select one node and make a state machine or behavior tree in it. In this way AI Graph makes a hierarchical nested structure. As requirements for a character increase in game development, and a developer wants to make character's decision making graph more precisely, the hierarchical nested structure allows developers to make as many layers as they want.

The AI Graph system has a real-time debug system that connects to and communicates with the game's run-time. Active nodes are high-lighted on the decision graph tool as they are executed. During development, this makes finding any problems in the decision graph much easier. AI Graph maintains scalability, variation, and diversity in character AI design through the course of development because of its data-driven approach. In this article, we will explain the AI Graph structure, operation principle, and examples from FINAL FANTASY XV.

AI Graph tool is used to make a character's decision making based on behavior trees and state machines. It has three regions (Figure 4). The center of the screen is a field to build a state machine and behavior tree graph by connecting nodes. The left vertically long window shows variables and nodes which are already made and can be re-used. The right vertically long window shows properties for customizing a node, and is called the property window. A node can be connected with another node by an arc. In a state machine, a node denotes a state and, an arc indicates transition of the state. In a behavior tree, a node denotes a behavior or operator of behavior tree and an arc is used to express behavior tree structure. A tray is used to enclose a state machine or a behavior tree. This enables a user to move one entire state machine or behavior tree by moving the tray, and it is also easy to see the layered architecture through the tray hierarchy.

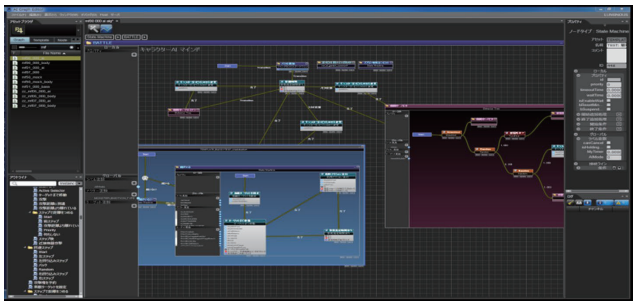

Figure 3. AI Graph Tool screen shot

3.1 Implementation Techniques of AI Graph Node.

In AI Graph, all nodes are re-used. For example, a node that can be used in a state machine can also be used in a behavior tree. But ordinarily the execution method of state machines and behavior trees are different. To make it possible for an AI node to be executed in both state machine and behavior tree, each AI Graph node has four components (Figure 4):

1. Start process (when a node is called)
2. Update process (while a node is executed)
3. Finalizing process (when a node is terminated)
4. A condition at terminate

For both behavior tree and state machine, the start process, the finalizing process and the update process are necessary to begin to execute, finalize and execute a node. The difference between them is what causes stopping a node. For a behavior tree, a node

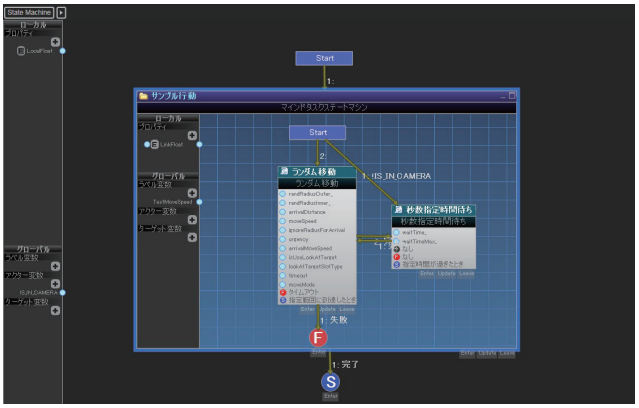

Figure 4. AI Graph Tool screen shot

terminates itself by judging an internal terminate condition while a state machine node is terminated by an external transition condition. Thus if a node has these four components, it can be executed in both behavior trees and state machines.

3.2 Data and Override

An AI Graph can be saved as an asset file. If an AI Graph is very fundamental for a character, it is repeatedly called and used. But the AI Graph is required to partially change the pattern, because it should be adjusted to each character. For example, when a state machine is saved as an asset file, a user will change a state of the state machine to make a more precise behavior tree or state machine in the state. A function to change a node is called an "override", much like C++.

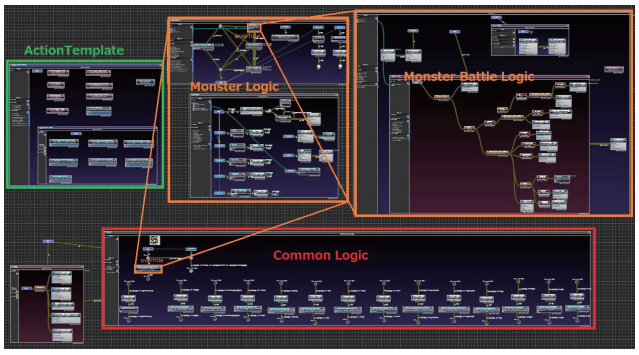

Figure 5. Overriding a monster's AI Graph

Furthermore, a monster's AI Graph can be created by overriding the graph repeatedly from common logic to monster battle logic (Figure 8). In this way, overriding methods makes AI Graph development easier and more effective.

3.3 Blackboard in AI Graph

Variables can be shared via blackboard with two types of blackboard (Figure 5). One is a local blackboard which belongs to a tray[Nii 86a,86b]. Variables of a local blackboard can be shared only in that local blackboard. The other is the global blackboard. Variables of the global blackboard can be shared with game and all characters' individual AIs. In the AI Graph tool, both blackboards are shown on the left side. And some variables are listed in them. To use variables of one local blackboard in a lower tray, the upper blackboard must be connected to the local blackboard of the lower layer included in the tray. Two connected blackboards can share variables.

These variables are used to describe the properties of a node and the transition conditions of a state machine, and so on. For example, the global variable "IS_IN_CAMERA" means whether an actor is in camera or not, and this variable can be used to describe a transition condition inside a state machine contained in a tray.

3.4 Parallel Thinking by AI Graph

For some situations, a character must think about two things at a time. AI Graph allows a character to have two concurrent thinking processes, and it is better to make two simple graphs rather than one big complex graph (Figure 6).

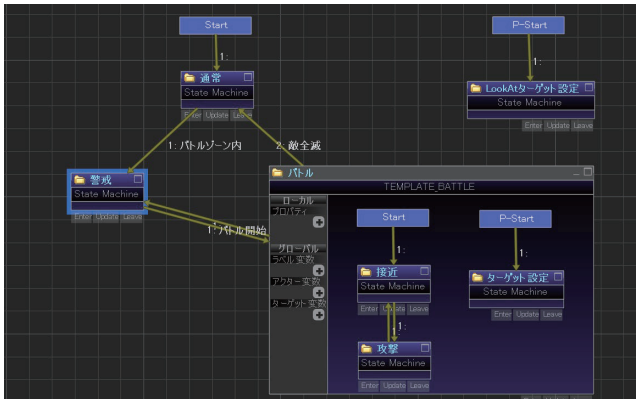

Figure 6. Parallel thinking

For example, one thinking process is a simple state machine to set a character behavior, and the other is a simple state to cause the character to look at a target that suddenly appears. The one state machine begins from a "START" node, and the other state machine begins from "PSTART" node.

The two state machines are executed concurrently. So the character can look around and search for a new target while it keeps attacking. Further, a behavior tree can execute two processes by a parallel node. For example, one behavior is to decide a target and the other is to approach and attack.

3.5 Interrupting the thinking process

It often happens that a character stops its thinking and must execute another specific action. An interrupt node interrupts a process of AI Graph when an interrupting condition is satisfied, and it executes the node linked to the interrupt node. For example, when a new game mission starts, monsters must rush to a player. After rushing into a player's position, they begin their original thinking process. In this case, two AI Graphs are prepared. One AI Graph includes an interrupt node (Figure 7). It causes the current tray to stop and the other Tray process to start when the transition condition connected to the interrupt node is satisfied. And after the tray process finishes, the process returns to the original process.

4. Use case of AI Graph in FINAL FANTASY XV

FINAL FANTASY XV is an RPG game in which a player travels in a large open world with three buddies while they fight with monsters and enemies in real-time (Figure 7). All characters have intelligence to make their decisions by themselves. Also for the player character, AI supports the player character's behaviors.

Figure 7. A player (left), a, monster, and a buddy character(right)

4.1 Body and Intelligence

AI Graph describes a character's intelligence and decision-making, but it does not describe physical body motion. It only gives an order of body motion via each node. And an AI Graph uses information required only for decision-making.

A character system consists of three layers: an AI layer, a body layer, and an animation layer. These three modules send messages to each other and share variables via blackboards.

AI Graph does not directly initiate animation data. AI Graph send a message to the animation layer via a body layer which consists of a state machine. Especially for shooting and damage behavior, AI Graph calls the special control nodes prepared in a body layer.

This three-layered architecture separates the roles to control a character between intelligence and physical body. And it also avoids increasing the size of an AI Graph (Figure 8).

A body layer represents a character's body as a node of a state machine. For example, a character's body state is expressed as running, jumping or climbing a ladder.

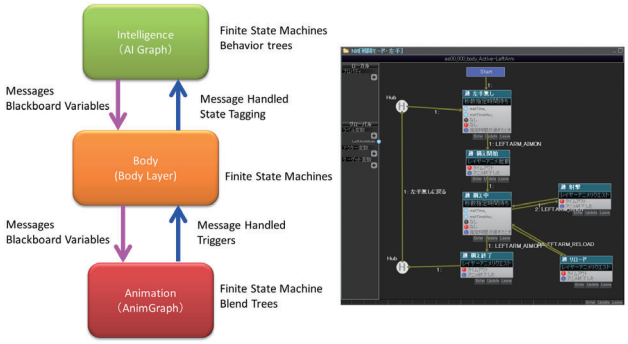

Figure 8. 3 layered system of character

4.2 AI development in game

For AI development, fast iteration is one of the most important features to keep AI improving until the end of development. As such a user should be able to reload an AI Graph without compiling when they want to make a change. In AI Graph Editor, an AI Graph can be compiled in the Editor independently from other systems' code. This an example of a data-driven system.

There are two debug windows (Figure 9). While a game program runs, an AI Graph keeps a connection with the program. This is called the visual node debugger. In this debugger, the active node currently being executed is highlighted in green. This enables a user to trace the active node in real-time.

The other debug window is in a game window. The window displays detailed logs which are generated from a character's AI Graph and AI Graph variables.

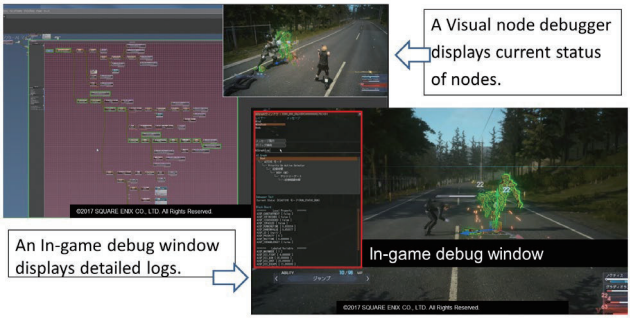

Figure 9 Visual node debugger (left) and In-game debug window (right)

5. Conclusion

As a game environment and game rules becomes more complex, a character is required to behave more smoothly and intelligently. When our project of next-gen AI development began, we realized and felt that improvement of our AI logic tool was required. After many discussions, an idea to combine both state machine and behavior tree was accepted and agreed to by the team. It allows a user to use both techniques in a nested hierarchical node structure to make a stable and flexible AI logic. We called this tool the AI Graph Editor. In this article, the basic principles and applied examples of AI Graph, and how these technologies are used in FINAL FANTASY XV which was released in 2016, are explained.

References

[Isla 01] D. Isla, R. Burke, M. Downie, B. Blumberg.: A Layered Brain Architecture for Synthetic Creatures. In Proceedings of IJCAI (2001).

[Isla 02] Damian Isla, Bruce Blumberg: Blackboard Architectures, AI Game Programming Wisdom, Vol.1, 7.1, pp.333-344 (2002).

[Isla 05a] Damian Isla: Managing Complexity in the Halo2 AI,Game Developer's Conference Proceedings. (2005).
http://www.gamasutra.com/view/feature/130663/gdc_2005_proceeding_handling_.php

[Isla 05b] Damian Isla.: Dude, where's my Warthog? From Pathfinding to General Spatial Competence", AIIDE(2005).
http://naimadgames.com/publications.html

[Nii 86a] H. Penny Nii: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures, AI Magazine, Vol.7 Num.2, pp38-53(1986).
http://www.aaai.org/ojs/index.php/aimagazine/article/view/537

[Nii 86b] H. Penny Nii: Blackboard Application Systems, Blackboard Systems and a Knowledge Engineering Perspective, AI Magazine, Vol.7 Num.3, pp82-107(1986). .