A.I. : Behavior Tree vs. Decision Tree

Kukuh Basuki Rahmat 13515025
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13515025@std.itb.ac.id

***Abstract*—One of the characteristic of human is ability to reason and make decision. With decision making, a human may decide which action to do and which path to take in any situation. In designing artificial intelligence, it is the topmost goal that we create a replication of human behavior with the absolute likeness of a real human being. For that cause, algorithm designers has adapted Decision Tree (one application of discrete mathematics specifically 'tree' module) to create Behavior Tree. It is a sophisticated tree algorithm dan can make decision and react based on various condition and therefore able to execute complex behavior.**

***Keywords*—Artificial Intelligence, Behavior Tree, Decision Tree, Tree**

I. INTRODUCTION

One application of Tree in discrete mathematics is construction of Decision Tree. With decision tree, our decision for provided condition may be efficient and always involves thorough thinking. It is as if using an "if-then" statement in program but with complex branching so that the decision we take may always give out consistent result given the same "input".

While it is handy, it can only represent very simple AI. A Decision Tree is evaluated from top to bottom, from the root to the leaf once. Then, the tree execution is restarted and it evaluates and reads new condition. The problem arises when in one of the branches, both conditions are true therefore creating conflicts.

With these problem arising, a new algorithm using the tree structure is discovered and given the name Behavior Tree. We can see that by the name it is very closely related to artificial intelligence. Many game developers have tried and succeeded in making complex AI with these Behavior Tree using many kind of game engine such as Unity Engine, Unreal Engine, and many more.

However, before delving deep into how these behavior tree works, how is it better suited for AI than decision tree, it is compulsory to talk about what is a 'tree' beforehand. Which we will talk in the next chapter

II. DISCRETE MATHEMATICS : TREE

Tree is a form of graph that doesn't consist of any direction. A Tree connect two vertices of a graph by exactly one path. It needs to be connected, so that no vertices gets left out without a path connecting to them. If any graph structure contains a circuit or we can get back to one vertice by following the path that it is connected to, then it is not a tree. Or we can say that a tree is a graph that is acyclic connected. A group of tree which is in disjoint union may be called a forest.

Here, we will talk about the terms that is common in a tree structure, but first see that figure 1 gives example of tree.

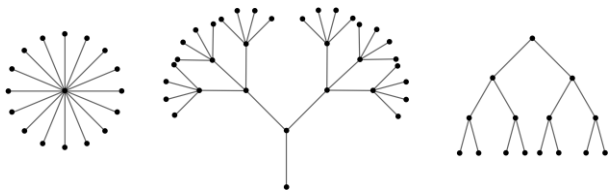

Figure 1 - Examples of Trees
jeremykun.files.wordpress.com

As we can see it consists of vertexes and paths, all connected and no cylic connected vertexes.

To understand more of the tree structure we need to know the common terms that may be found in a tree that is used or agreed by people that discovered the tree structure.

Root

The node of origin of a tree. Usually the root is depicted as a single node at the topmost of the tree. But it may be depicted in the bottom of the tree too, as we can see in the second picture in figure 1, the root is at the bottom of the tree. Though whatever the case, there must only be one root which is the "origin" of the tree.

Parent & Child

We cannot define a parent node without a child node. In

short, when a two node is connected, the node that is closer to the root node is called the parent node and the other node is called a child node. A parent may have more than one child while a child can only have one parent. If each parent node only have n child then the tree is called an n-ary tree or a tree with n arity.

Branch

A branch is a part of the tree that connects a node with the nodes directly below it (assuming the root is at the top) nodes that have branch are called branch node. A n-ary tree or a tree with arity of n may have n branch for each node or not at all (leaf node).

Siblings

Siblings is a group of node that is connected by the same parent node. It must not be connected for if it is connected then a cycle is found in the tree and it is not a tree anymore. It should be in the same level. More about levels will be discussed below.

Descendant

A node can be called the descendant of other node only if we can track the other node by recursively tracking the parent of the parent. If the other node is found in the track, we can say that the child node is the descendant of the other node.

Ancestor

Similar like de descendant node, if we can track two node recursively by referring to the child of the child and found the specific node, we can say that the parent node is the ancestor to the node.

Leaf

A leaf is a node that has no child at all. Usually at the greatest level. In tree traversing algorithm, this node is usually modified or evaluated the first. It is also called external node or the outermost node.

Height

The height if the tree is the number of branch from the root or origin node to the furthest leaf. It also represent how many levels a tree may have.

Path

A path is the sequence of nodes along the branches of a tree. For example a path from node A to B is the sequence of branch that may connect node A to node B.

Subtree

A subtree represent the descendant of a node. We can cut a subtree form a tree and produce a new tree that still follows the rule of tree.

Visiting

Visiting means checking the value of a node when we gains control of the node. In AI this could mean checking values whether if enemy is present and other factors depending on the relevant game. Further action is then taken based on the values we check on visiting.

Traversing

Traversing means evaluating each and every node by specific order. It is usually evaluated from the left most to the rightmost and done recursively. To traverse means that every node is visited in order.

Levels

In a tree a node is grouped horizontally and labelled levels to the group. The root node is labelled level 0 and the child of the root node is labelled as level one and onward.

Keys

A keys is the value represented in each node, sometimes identical, a keys is based on which search operation is carried out for a node.

Forest

A Forest is a group of tree that is not connected or disjoint.

Below is figure 2 which explains some of the tree terms we discussed in this chapters.

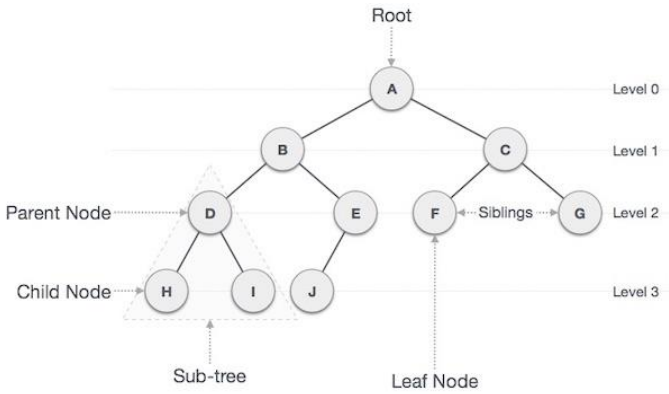

Figure 2 – Terms in a Tree
https://www.tutorialspoint.com

III. DECISION TREE VS. BEHAVIOR TREE

The most striking difference of behavior tree and decision tree is how each of them is evaluated. These evaluation may affect very big to how AI reacts to situation.

AI Using Decision tree when not carefully created can create confusion in program and therefore making the AI

somewhat sloppy. This confusion may be created if a situation is conflicting each other.

A Decision Tree is evaluated from the root to leaf, every frame or every n frame, the tree is refreshed and the AI do the corresponding action. It is composed in nodes that propose a question that can be answered by no, yes, or maybe. The question then answered by the corresponding branch that each answer for "yes" condition, "maybe" condition, and "no" condition. Therefore, 3 node must be branched from the question node.

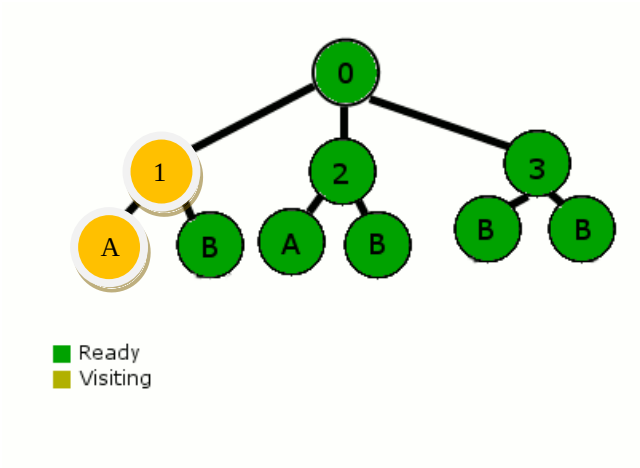

Figure 3 - Decision Tree Evaluation gamedev.stackexchange.com

In figure 3 we can see that the node 0 is the question node that must be answered by yes, no, and maybe. The node 1, 2, and 3 is the nde that can only be answered with yes/no or comparing value (i.e. greather than, less than). While the node A, B, and C is the execution node. If the condition in the above nodes all apply, we do the instruction stored in the node A, B, or C.

A Decision Tree is evaluated from the root to leaf, every frame or every n frame, the tree is refreshed and the AI do the corresponding action. It is composed in nodes that propose a question that can be answered by no, yes, or maybe. The question then answered by the corresponding branch that each answer for "yes" condition, "maybe" condition, and "no" condition. Therefore, 3 node must be branched from the question node.

Evaluation in behavior tree is evaluated differently. Behavior Tree does not reset every time an evaluation is made like decision tree does. If the leaf node has been evaluated, a behavior tree then traverse to the next sibling nodes and do so until all child node is evaluated. If all condition is met, then the behavior tree calls the corresponding behavior best suited to interact with the environment.

After a node has called a behavior to be executed, the node is set to "running" and in the next reset the "running" value is retained and the tree will know what state the character was in.

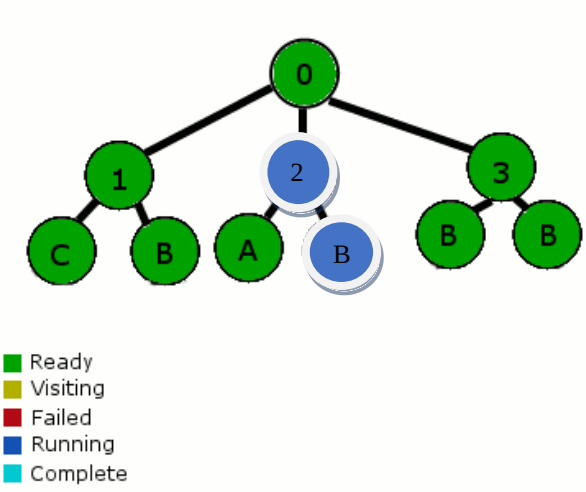

Figure 4 - Behavior Tree Evaluation gamedev.stackexchange.com

In order for sophisticated AI to be accomplished, the child note in behavior tree must be ordered based in their priority. In AI the priority is usually set to the requirements of condition that state the AI to be "alive" such as Health Point and find cover from enemy followed by attack enemy or load gun.

Other than "alive" condition it can also be the big goal of the game (AI for one character and AI for many character chasing a goal or strategy AI is different) such as destroying the home base of enemy. One example is enemy home base is at low HP level simultaneously with the character HP. The character would sacrifice to destroy the enemy home base. The orderings of priority is crucial to make the best AI possible.

One case of when behavior tree is more desirable then decision tree is when a creature A is in low HP so it needs to recover and return to home base. In the middle of the way, creature A meets creature B which is the enemy and creature B is at lower HP than creature A.

With decision tree, since usually vital value such as health point is at the priority evaluation, creature A will still be in low HP and in every tree refresh, creature A will still be heading to home base.

With behavior tree however, the "low HP" behavior will retain in the next tree refresh and after checking the priority node it will still be true. After evaluating the siblings or the less priority node, the tree will detect that enemy is nearby and may be able to make other

evaluations such as if enemy HP is lower, attack enemy using ranged attacks while running, else hide and take another path to home base. With the conditions provided creature A will attack creature B and scoring points while still able to heal in home base. A more player-like approach to creature behavior is able to be written.

More detailed discussion about behavior tree and how to apply them will follow in the next chapter.

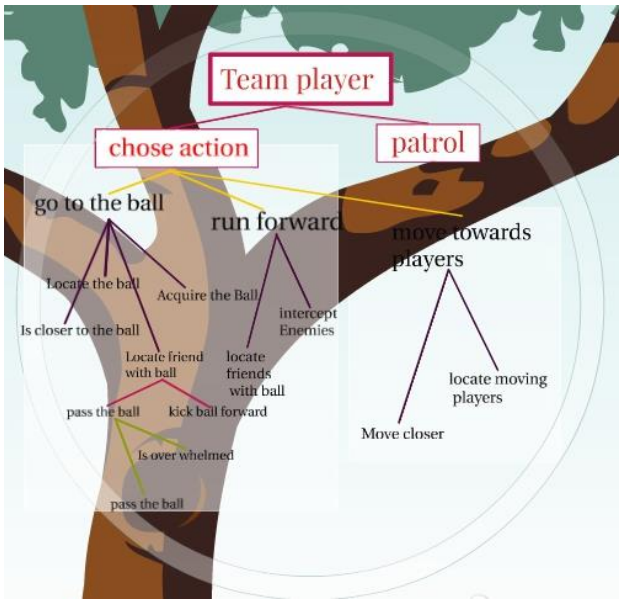

Figure 5 - Ball Control Behavior Tree
https://www.prezi.com, Remi Dubois

IV. Behavior Tree in AI

A. Basics

By definition, a Behavior Tree is a tree of hierarchical nodes that control flow of decision making of an AI entity. The leaves are the actual commands that control actions that may be executed by an AI entity. The branches are utility nodes that control the AI walk down the tree to reach the sequence of command best suited to the situation

The tree can get complex with nodes calling other sub-tree to perform particular behavior. We can start creating the tree from the simplest behavior and growing to a very complex and life-like behavior.

B. Traversal

As have been stated before, an efficient behavior tree retains previous evaluation result to the next evaluation, so that the command picked to execute may represent the best possible action. This is also done to prepare if the tree may get very deep which is evaluated every frame may get the processor working heavy load.

C. Flow

A behavior tree is composed of many types of nodes which all have the common core functionality which is to return statuses. The common status return is success, failure, and running. Success and Failure informs the parent that the action in the node is successfully done or not. While the running means that the operation is not yet determined. After the next tree refresh, the value can be changed depending on the situation.

D. Node Type

Node types in behavior tree consist of three types which is Composite, Decorator, and Leaf.

A Composite node is a node that can have one or more children. They process their children node by certain traversing rules and the returns success or fail or running to the parent. The most commonly used composite is the sequence which evaluates each child in sequence, returning failure if any children evaluation returns fail and returning success if all children returns success.

A Decorator node can only have one child and transform the return status of its child whether to terminate child, repeat child, or many other uses depending on the node in respect. A common used example is the inverter which negates the return status of the child node.

The leaf node is the most powerful node since it contains the actions to be executed. This is also the node at the highest level of the tree. The common example is the walk leaf node. The walk node would make the character walk to a specific location. The leaf node can be very expressive when combined with composite and decorator node. A leaf node may also call another behavior tree passing the current tree data as a parameter for the called behavior tree.

E. Composite Nodes

Sequence: A sequence node evaluate every child from the determined order and inform the child return status to the parent. Fail if any child fails, Success if all success, and Running if not all child are finished evaluating

Selector: Selector is the “OR” version if sequence “AND”. A selector node returns success if any child success and returns fails if all child fails. A selector provide us options that the character should try before deciding that it is unable to be done.

Random: The random composite is just a sequence composite or selector composite but evaluated in random order to create more randomness in AI and unpredictability, just like a living being would be.

F. Decorator Nodes

Inverter: Inverter nodes negates the leaf node

evaluation much like a “NOT” gate in logic gate. They are used mostly in conditional test.

Succeeder: A suceeder node automatically return success to the parent node. This node is useful in case when processing a branch where failure is expected or anticipated but the processing of a sequence of its siblings is still desired. A Succeeder combined with an inverter may create the decorator nodes Failer.

Repeater: A repeater decorator repeat its child node and often used at the base of a tree. It is done so to make a child run continuously. Another application is to execute actions in its children a set number before returning result to its parents.

Repeat Until Fail: The repeat until success decorator repeat its child until the child encounters a failure status return. The repeat until fail decorator then will return success to its parent.

Not just the discussed decorator, we may create our own definition of decorator to suit our needs of AI creation.

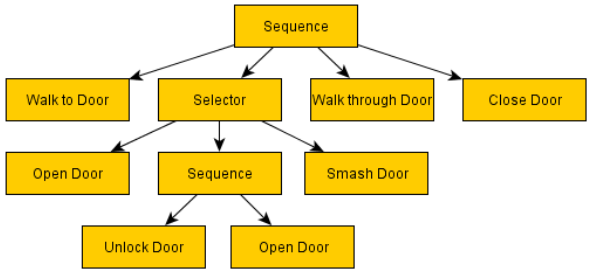

Figure 6 - Simple Open Door Behavior Tree
www.gamasutra.com, Chris Simpson

V. CONCLUSION

Behavior Tree is a powerful and complex tool to represent more human-like and life-like behavior. However a complex algorithm may need thorough thinking to create. Decision Tree on the other side is simple and easy to think of, like the usual if-then. So which tree is better used to create an AI depends on the problem to solve. If we want to make complex Player verses Player-like experience it is more suitable to use complex behavior tree. If we just want to make simple AI and more conservative on memory and processing resource, a Decision Tree based AI would suffice just fine.

REFERENCES

[1] Byte56 [Michael House]. “Decision Tree vs Behavior Tree” *Stack Exchange*,gamedev.stackexchange.com/questions/51693/decision-tree-vs-behavior-tree. Accessed 3 Dec. 2016. 23.37.

[2] Chris Simpson. “Behavior Trees for AI: How they work” *GAMASUTRA*, 17 Sept. 2009, gamasutra.com/blogs/ChrisSimpson/20140717/221339/Behavior_trees_for_AI_How_they_work.php. Accessed 4 Dec. 2016. 24.03.

[3] “Data Structure and Algorithms - Tree” *Tutorials Point*, https://www.tutorialspoint.com/data_structures_algorithms/tree_data_structure.htm. Accessed 5 Dec. 2016. 18.51.

[4] Rémi Dubois. “AI – Decision Tree & Behavior Tree” *Prezi*, https://prezi.com/ez06l7qeja6s/copy-of-ai-decision-tree-behaviour-tree/. Accessed 5 Dec. 2016. 19.27.

[5] Jeremy. Kun. “Trees – A Primer”. *Jeremy Kun*. https://jeremykun.com/2012/09/16/trees-a-primer/. Accessed 7 Dec 22.09.

[6] Maxim Likhachev. “Intelligence I: Basic Decision-Making Mechanisms” *Carnegie Mellon University*, http://www.cs.cmu.edu/~maxim/classes/CIS15466_Fall11/lectures/intelligenceI_cis15466.pdf. Accessed 9 Dec. 12.21.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 8 Desember 2016

Kukuh Basuki Rahmat 13515025