ISSN 2224-087X. Електроніка та інформаційні технології. 2018. Випуск 9. С. 113–119
Electronics and information technologies. 2018. Issue 9. P. 113–119

УДК 519.172.1

CREATING AI FOR GAMES WITH UNREAL ENGINE 4

V. Kushnir, B. Koman

Ivan Franko National University of Lviv,
50 Drahomanova St., Lviv, Ukraine, 79005
vasylll95@gmail.com

Game AI in Unreal Engine 4 based on decision tree and called Behavior tree. The advantage of developing AI is the wide usage of this method in game industry for building an AI bots. It helps to build not a simple AI but a big model that helps us to build more interesting game. Nevertheless, this method of developing AI has a disadvantage which make hard to build a big system if you are only on a start to build AI with this method. In this paper I will show game engine called Unreal Engine 4 and how artificial intelligence can be developed. Also in this article will be shown a good start using this method for building great and large AI instantly.

Key words: Unreal Engine 4, Behavior tree, decision tree, blackboard, selector, root, loop.

Introduction [1]. Behavior tree is a graphical, modeling language and a good approach for developing big AI systems for games in a short period a time. It can be easily used to develop AI for different tasks. This method was developed by R.G. Dromey with first publication of some keyideasin 2001[1]. Early publications on this work used the terms "genetic software engineering" and "genetic design" to describe the application of behavior trees [1]. The reason for originally using the word genetic was because sets of genes, sets of jigsaw puzzle pieces and sets of requirements represented as behavior trees all appeared to share several key properties:

- they contained enough information as a set to allow them to be composed – with behavior trees this allows a system to be built out of its requirements,
- the order in which the pieces were put together was not important – with requirements this aids coping with complexity,
- when all the members of the set were put together the resulting integrated entity exhibited a set of important properties.

For behavior trees important emergent properties include: the integrated behavior of the system implied by the requirements and the coherent behavior of each component referred to in the requirements.

Use of the term genetic came from eighteenth century by thinker Giambattista Vico, who said, "To understand something, and not merely be able to describe it, or analyse it into its component parts, is to understand how it came into being – its genesis, its growth true understanding is always genetic". Despite these legitimate genetic parallels it was felt that this emphasis led to confusion with the concept of genetic algorithms. As a result, the term behavior engineering [1] was introduced to describe the processes that exploit behavior trees to construct systems. The term "behavior engineering" has previously been used in a specialized area

of Artificial Intelligence - robotics research. The present use embraces a much broader rigorous formalization and integration of large sets of behavioral and compositional requirements needed to model large-scale systems [1].

Since the behavior tree notation was originally conceived a number of people from the DCCS (Dependable Complex Computer-based Systems Group – a joint University of Queensland, Griffith University research group) have made important contributions to the evolution and refinement of the notation and to the use of behavior trees. Members of this group include: David Carrington, Rob Colvin, Geoff Dromey, Lars Grunske, Ian Hayes, Diana Kirk, Peter Lindsay, Toby Myers, Dan Powell, John Seagrott, Cameron Smith, Larry Wen, Nisansala Yatapanage, Kirsten Winter, Saad Zafar, Forest Zheng. Company Epic games uses this method of building ai and integrate it in their engine called Unreal Engine 4 [1].

The paper consists of several sections: Section 2 and 3 overviews original Behavior Tree and Behavior Tree in Unreal Engine 4. Section 4 describes how to use Behavior tree in Unreal Engine 4. In the last section was made a conclusion.

Original Behavior Tree [2,3]. BT is graphically represented as a tree in which the nodes are classified as root, control flow nodes, or execution nodes (tasks). For each pair of connected nodes the outgoing node is called parent and the incoming node is called child (Fig. 1). The root has no parents and exactly one child, the control flow nodes have one parent and at least one child, and the execution nodes have one parent and no children. Graphically, the children of a control flow node are placed below it, ordered from left to right (Fig. 1).

The execution of a BT starts from the root which sends ticks with a certain frequency to its child. A tick is an enabling signal that allows the execution of a child [2]. When the execution of a node in the BT is allowed, it returns to the parent a status running if its execution has not finished yet, success if it has achieved its goal, or failure otherwise.

A control flow node is used to control the subtasks of which it is composed. A control flow node may be either a selector (fallback) node or a sequence node. They run each of their subtasks in turn. When a subtask is completed and returns its status (success or failure), the control flow node decides whether to execute the next subtask or not. Fall back nodes are used to find and execute the first child that does not fail. A fallback node will return immediately with a status code of success or running when one of its children returns success. The children are ticked in order of importance, from left to right. Sequence nodes are used to find and execute the first child that has noty et succeeded [3]. A sequence node will return immediately with a status code of failure or running when one of its children returns failure. The children are ticked in order, from left to right. In order to apply control theory tools to the analysis of Behavior Trees, BT can be defined as three-tuple [2,3].

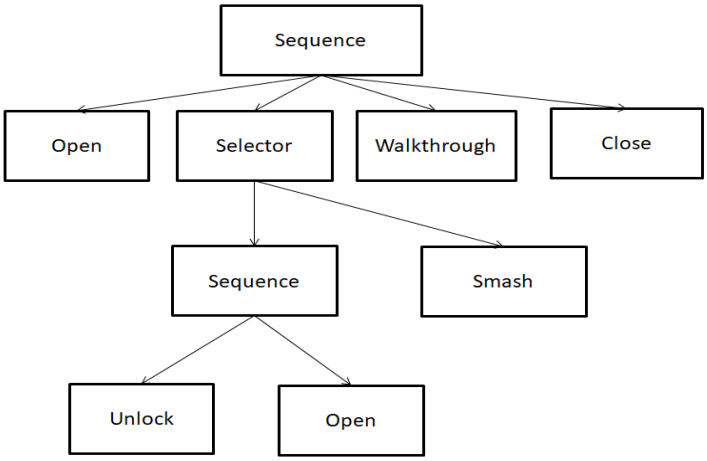

Fig. 1. Abstract model of Behavior tree.

Behavior tree in Unreal Engine 4 [4]. Behavior tree in Unreal Engine 4 is more difficult than his original. It is a combination of two assets:

1. Blackboard, which uses as AI memory and stores data as a dictionary,
2. Behavior Tree, which is the AI head. It makes a decision that causes another decision.

For developing Behavior tree there are main nodes [4]:

Composite	These are the nodes that define the root of a branch and the base rules for how that branch is executed
Task	These are the leaves of the Behavior Tree, the nodes that "do" things and don't have an output connection.
Decorator	Also known as conditionals. These attach to another node and make decisions on whether or not a branch in the tree, or even a single node, can be executed.
Service	These attach to Composite nodes, and will execute at their defined frequency as long as their branch is being executed. These are often used to make checks and to update the Blackboard. These take the place of traditional Parallel nodes in other Behavior Tree systems.
Root	The Root node is unique in the Behavior Tree and is the starting point for the Behavior Tree. It can only have one connection, and you cannot attach Decorators or Services to it. The Root Node has no properties of its own, but selecting it will show the Behavior Tree properties in the Details Panel, where you can set the Blackboard Asset of the Behavior Tree.

Composite Nodes define the root of a branch and the base rules for how that branch is executed.

Composite nodes consists of nodes:

- Selector.Nodes execute their children from left to right, and will stop executing its children when one of their children succeeds. If a Selector's child succeeds, the Selector succeeds. If all the Selector's children fail, the Selector fails.
- Sequence Nodes execute their children from left to right, and will stop executing its children when one of their children fails. If a child fails, then the Sequence fails. If all the Sequence's children succeed, then the Sequence succeeds.
- The Simple Parallel node allows a single main task node to be executed along side of a full tree. When the main task finishes, the setting in Finish Mode dictates if the node should finish immediately, aborting the secondary tree, or if it should delay for the secondary tree to finish.

Tasks are nodes that "do" things, like move an AI, or adjust Blackboard values.

Decorator, also known as conditionals in other Behavior Tree systems, are attached to a Composite or a Task node and define whether or not a branch in the tree, or even a single node, can be executed.

There are several common decorators that used more often:

- The Blackboard node will check to see if a value is set on the given Blackboard Key.Decorator node in shows how to connect data from Blackboard into Behavior tree and this decorator checks if the value from blackboard is set.
- Conditional Loop. Loop runs while the condition in it is true.
- Does path exist decorator checks if bot can use MoveTo task.
- Loop decorator helps us to execute some part of tree several times. Unlike conditional decorator this loop use not a condition but the number of cycles loop body must be executed.

Services attach to Composite nodes, and will execute at their defined frequency as long as their branch is being executed. These are often used to make checks and to update the Blackboard. These take the place of traditional Parallel nodes in other Behavior Tree systems.

Root is a starting point for our Behavior Tree.

Implementation in Unreal Engine 4 [4]. For AI implementation in Unreal engine 4 was chosen to make a bot which will fight with another bots and when there will be only one he will come to home.

Make a fire event which will draw a line and hit the enemy as illustrated on Fig. 2.

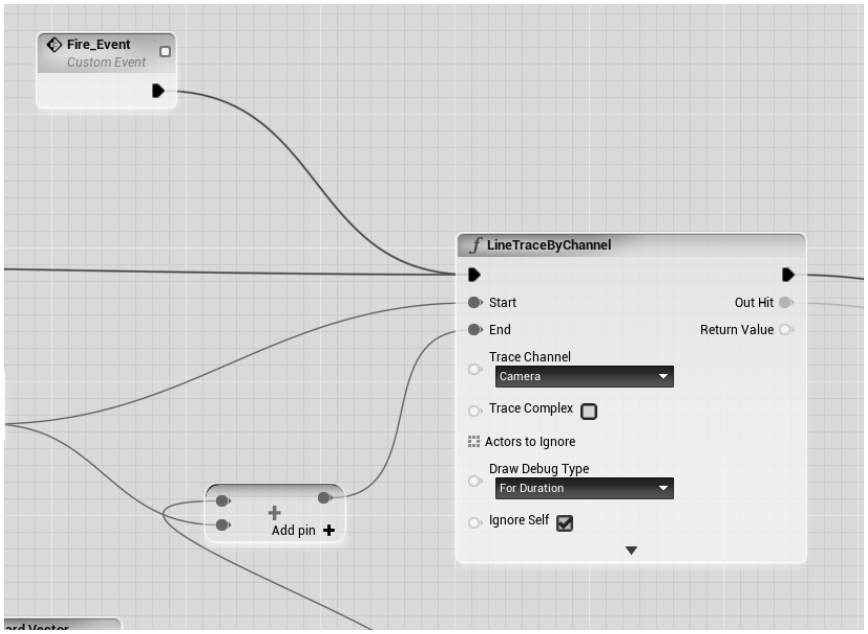

Fig. 2. Three bots and a box as home location.

Also was used AI Perception component to find enemy and set value function to send enemy object reference to the blackboard used by behavior tree (Fig. 3).

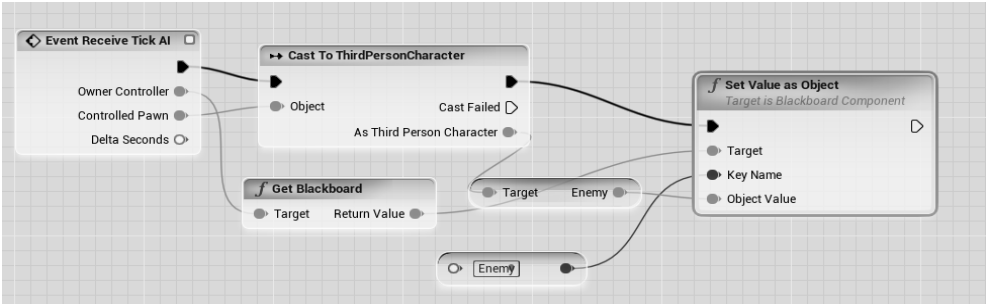

Fig. 3. Setting enemy variable to the Blackboard if bot find one otherwise – none.

On Fig. 4 shown already developed Behavior tree. Tree starts from the Root node and move to selector that has connected: loop decorator (with infinite parameter, which means that was used infinite loop), home location service(service that checks out home location position and if it stays unchangeable) and find enemy service(service that helps bot to find a new enemies using bot sight and hearing perception). Enemy reference object and home location stores in blackboard as shown on Fig. 5.

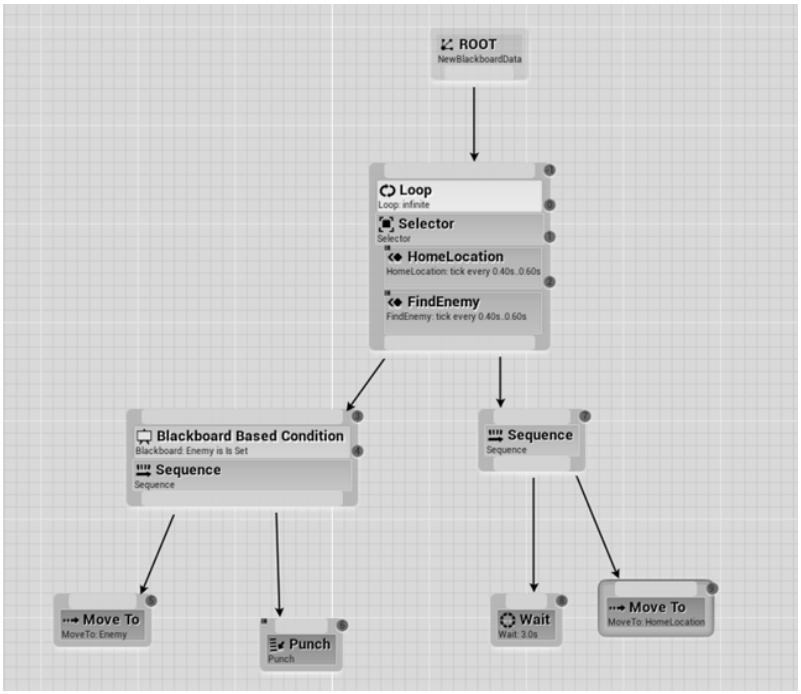

Fig. 4. Behavior tree of our bot.

Fig. 5. Blackboard where bot saves data he needed.

As a result bots finds their enemies and call fire event which create a line a put damage to the traced bot. After fighting if bot can't find another enemy, bot goes to his home location and next time when he finds one more enemy, he will move to him and will try to kill.

Conclusion. In the paper, was shown AI development for games using Behavior tree on Unreal engine 4. Also in article was shown development interface of Behavior tree. Figures show us that we can easily build a Behavior tree for different tasks. So the main advantages of these development are: 1) Simplify on development process by using visual development. 2) Fast bot and event reaction of a bot from AI. 3) Using virtual perceptions like: sight and hearing that makes our bot more humanable about interaction in game environment. 4) Unreal Engine 4 already defined services, tasks and decorators that helps us to build AI with more interesting mind. I think that in future Behavior tree will be used more widely except game industry and it will get more improvements which will cause a great results.

REFERENCES

1. *Colledanchise M*. How Behavior Trees Modularize Hybrid Control Systems and Generalize Sequential Behavior Compositions, the Subsumption Architecture, and Decision Trees. / M. Colledanchise, P. Ögren. – Fort Collins, 2016. – 505 c.
2. *Colledanchise M*. Behavior Trees in Robotics and AI: An Introduction / M. Colledanchise, P. Ögren., 2017. – 198 c.
3. *Ögren P.* Increasing Modularity of UAV Control Systems using Computer Game BTs / Petter Ögren., 2012. – 16 c.
4. Unreal Engine 4 Documentation [Електронний ресурс] // Epic Games. – 2018. – Режим доступу до ресурсу: https://docs.unrealengine.com.

Стаття: надійшла до редакції 15.04.2018,
доопрацьована 23.04.2018,
прийнята до друку 25.04.2018.

СТВОРЕННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ІГОР ЗА ДОПОМОГОЮ UNREAL ENGINE 4

В. Кушнір, Б. Коман

Львівський національний університет імені Івана Франка,
вул. Драгоманова 50, Львів, Україна 79005
vasylll95@gmail.com

Ігровий штучний інтелект в Unreal Engine 4 ґрунтується на деревах рішень, які називають Деревами поведінки. Перевага розробки полягає в широкому використанні цієї технології в ігровій індустрії економіці, фінансах. Ця розробка допомагає збудувати складну систему, яка додасть цікаві речі в нашу гру, дає змогу створити бота, який би автоматично виконував певні дії за нас. Однак цей метод має декілька недоліків, які не дають змоги збудувати складну систему, якщо ви тільки почали працювати з цією технологією. Також вона не є надто гнучкою порівняно з нейронними мережами, алгоритмами класифікації чи кластеризації.

Описано використання цієї технології для створення штучного інтелекту, схарактеризовано його використання в різних сферах діяльності, наведено їхню структуру, відображено візуальний інтерфейс для побудови такого дерева й зазначено відмінності між класичними поведінковими деревами та поведінковими деревами в Unreal Engine 4, використання їх у побуті та застосування для складних операцій. Також описано збереження даних у поведінкових деревах, маніпуляцію з даними, передавання даних між поведінковими деревами та спеціальною візуальною мовою програмування Блюпрінт, взаємодію Блюпрінтів один з одним, щоб отримати доречний контролер для керування ігровим ботом, взаємодію відчуттів ігрового бота з поведінковим деревом як на вхід, так і на отримання ботом певних завдань, які він повинен виконати. У кінці відображено побудовану систему керування ігровим ботом з використанням візуальної мови програмування Блюпрінт, систему прийняття рішень з застосуванням поведінкового дерева, а також взаємодію між ними.

Ключові слова: Unreal Engine 4, дерева поведінки, дерева рішень, дошка, селектор, корінь, цикл, послідовність.