*Journal of Multimedia Information System VOL. 7, NO. 4, December 2020 (pp. 263-268): ISSN 2383-7632 (Online)*
http://doi.org/10.33851/JMIS.2020.7.4.263

# Behavior Tree-based Scenario Development Technology to Induce Various Experiences of VR content

Jinseok Seo[1*], Ungyeon Yang[2]

## Abstract

This paper introduces an event modeling and simulation system using behavior trees. The system aims to overcome the limitations of existing fixed, simple scenario-based training content, and to extend the behavior of objects to enable various experience deployments. To achieve this goal, we made specific tasks of behavior trees can change according to users' reaction and developed an adaptive simulation module that can analyze and execute behavior trees that changes at runtime. In order to validate our approach, we applied the adaptive behavior tree simulation to the scenarios in our virtual reality simulation-based fire training system we have been developing and demonstrated the implementation results.

**Key Words**: Virtual reality, Experience diversification, Behavior tree, Adaptive simulation.

## I. INTRODUCTION

As the virtual reality market grows, the quantitative demand for content increases rapidly and the demand for scalability is also faced. In order to cope with this demand, the productivity of content has become very important. However, there is a limit to improving content productivity with a development method that relies on computer programming languages such as C++ or C#.

Content productivity problems are more prominent in virtual reality simulation-based training systems. The main purpose of the training system is to maximize the educational goals by providing users with a variety of experiences. In order to ensure that all users with different training proficiency and abilities have the opportunities to explore a variety of virtual activities, virtual training scenarios that can handle numerous cases are required.

We have been developing a fire training system using virtual reality simulation and trying to solve the problems mentioned above in this system for several years. We are using behavior trees to design the behaviors of the virtual objects that make up our system. Behavior tree is very useful in designing the behavior models of the objects required by our planner. However, as we said earlier, more and more complex behavior trees are needed to handle countless experience situation.

Therefore, in order to tackle the problem, we started this research. In our research, we made some changes to behavior tree to meet our requirements. We defined four types of new behavior tree nodes, which can be dynamically changed according to the training situation at runtime and implemented an adaptive behavior tree simulation module in which these special tasks can operate. And in order to confirm whether the adaptive simulation we devised is effective, we applied it to the virtual reality simulation-based training system we were developing.

This paper is organized as follows. In Section 2, we describe of some case studies related with our research. Section 3 explains some of the problems we resolved in the process of implementing our adaptive simulation module. Section 4 introduces the four types of special behavior tree nodes and modeling methods of them. Section 5

**Manuscript received December 04, 2020; Revised December 21, 2020; Accepted December 28, 2020. (ID No. JMIS-20M-12-035)**
Corresponding Author (*): Jinseok Seo, 176 Eomgwangno, Busanjin-gu, Busan 47340, Korea, +82-51-890-2712, jsseo@deu.ac.kr.
[1]Game Engineering Major, Dong-eui University, Busan, Korea, jsseo@deu.ac.kr
[2]Creative Content Research Division, Electronics and Telecommunications Research Institute, Daejon, Korea, uyyang@etri.re.kr

demonstrates an example scenario that apply our approach. Finally, we will make concluding remarks in Section 6.

## II. RELATED WORKS

In order to increase the productivity of contents, the virtual reality and game industries have included various standardized authoring support tools in their production pipeline. Typical examples are Unity Engine's Mecanim and Unreal Engine's Blueprint. Although there are tools that require some knowledge of computer programming now, it is expected to become a tool that even beginner can easily access in a near future.

Unity's Mecanim uses a tool like the Hierarchical State Machine (HSM) to create the effect of transitioning animation according to changes in the state of an object [1]. In addition to this, Unity recently added a visual scripting tool called Bolt [2] as a standard asset. Using Bolt, it is possible to implement not only animation but also the entire game logic using only visual scripting.

Unreal's Blueprint [3] is a visual scripting language that greatly expands data flow diagram (DFD) and is a tool that helps you program the behavior of objects. Unreal also includes Behavior Tree as a visual tool for AI of game agents [4]. The study in [5] also showed the possibility of an agent capable of reinforcement learning using Environment Query System (EQS) [6] and visual tools provided by Unreal.

As in this research, the problem of adaptively simulating objects in virtual environments is similar to planning problems in artificial intelligence. This is because adaptively simulating requires finding or replacing actions to accomplish a particular purpose. SHOP2(Simple Hierarchical Ordered Planner 2) [7] is a representative example and a system based on Hierarchical Task Network (HTN). As an example of planning [8] is a procedural scenario creation technique using HTN in lifesaving training games can generate variety of dynamic scenarios in unpredictable environments.

Because behavior trees are very natural to apply computational operations to nodes that make up themselves, they are well suited as a means for adaptive simulation. In [9], they introduce the parameterization to behavior tree and an authoring tool for AI agents. An example of applying operations to behavior tree's nodes can be found in [10]. In this research, the genetic algorithm is used to optimize the nodes of behavior tree.

## III. ADAPTIVE BEHAVIOR TREE SIMULATION MODULE

Each time we update the fire training system we have been developing, there are more and more object behaviors that we need to implement newly. In addition, it was found that the behavior tree needs to be continuously updated whenever training is performed for users. To update a behavior tree, you need to implement a custom action task to invoke some action method on an object and add the resulting task to the behavior tree. For this reason, we implemented an adaptive behavior tree simulation module that can dynamically change the action method we want without modifying behavior trees or implementing new custom tasks.

The simulation module we implemented acts as a component of a game object in the Unity engine [12]. We utilized Opsive's Behavior Designer [11] for the basic simulation and authoring of behavior tree, which is very suitable for our research. Each adaptive task implemented in this research inherits from the base classes provided by Opsive's Behavior Designer.

The core of this simulation module is a generic action task called "Method Invoker" that allows us to use the desired Unity GameObject's action method as a parameter instead of implementing a new custom action task. When you put a Method Invoker task in a behavior tree, you need to set the Unity GameObject and the component that owns the method that this task will execute, the method name to execute, and the parameters required for the method. All of these processes are done in the Inspector window of Unity, and for this, we implemented "Method Finder Drawer" by customizing the Object Drawer provided by Obsive.

A Method Invoker task also works as a scene-level component of a Unity GameObject, but we have implemented serialization functionality so that it can be saved as an asset that is a project-level component. This serialized task can be referenced by a specific node in behavior trees, or it can replace a specific node at runtime. By implementing this way, we enabled adaptive simulation in which some nodes of behavior trees can change at runtime.

In addition to Method Invoker, we had to implement a few additional custom tasks for adaptive behavior tree simulation. They include Set Variable, ChangeableAction, D-Sequence, D-Selection, etc. Details of each task are postponed to the next chapter.

## IV. MODELING ADAPTIVE BEHAVIOR TREE

In this chapter, we will explain the process of modeling object behavior using four types of behavior tree nodes we defined in this research, which are created using the Method Invoker task mentioned above. These four tasks allow specific nodes in behavior trees to dynamically change in their own different ways.

### 4.1. Parameterized Task

When you put a Method Invoker task in a behavior tree, you are prompted for the method name you want to execute and parameters the method needs. The parameters set in this task are created as shared variable objects of the corresponding game object. Therefore, this parameter value can be accessed from other nodes or other behavior trees at runtime.

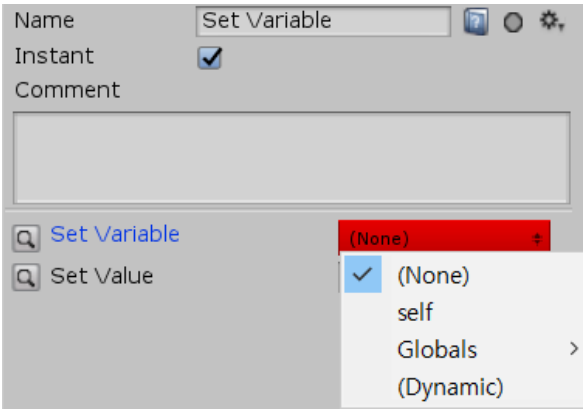


Fig. 1. Set Variable task that can change the argument value of Parameterized task at runtime.

For the external reference of the parameters in the Method Invoker task, we implemented a custom behavior tree node called "Set Variable" (Fig.1). The Set Variable task changes the value of a shared variable of the GameObject with a behavior tree component that want to change the value of a parameter, and the changed value is passed back to the Method Invoker task of that behavior tree component.

### 4.2. Substitute Task

Substitute task can be replaced at runtime with another node that includes its subtree. Fortunately, the Opsive's Behavior Designer provides an External Behavior Tree functionality that can store a behavior tree node including its subtree as an external asset.

By actively utilizing this External Behavior Tree, we implemented three types of substitute tasks. They are ChangeableAction task and two composite tasks, which are D-Sequence (changeable sequence) (Fig. 2) and D-Selector (changeable selector).

The Substitute tasks are replaced with predetermined external behavior trees when the conditions set in the ChangeableCondition Field of them are satisfied. In Changeable Condition, the value of a shared variable (described above) can be used as a condition and complex conditions can be set by combining multiple conditions.

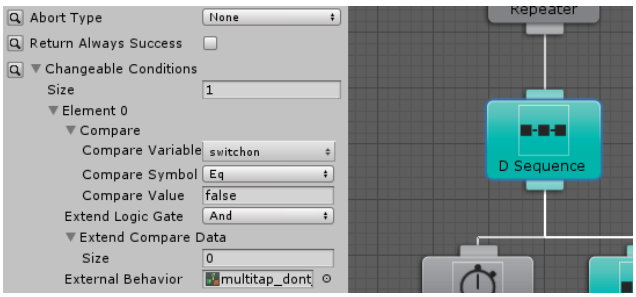


Fig. 2. D-Sequence task, one of the Substitute tasks, can set various conditions.

### 4.3. Stochastic Task

There are three types of custom tasks for stochastic execution in our behavior tree. The first one is Random Selector. Random Selector task is a composite node and is used to select one of the multiple scenarios by randomly selecting and executing one of the child tasks. The second one, Random Probability task, is one of the Condition tasks and returns Success with random probability. This task is used to select whether to run the scenario of its child node. The third, Set Shared Float Random task, is an Action task that modifies a float type variable value to a random value. Of course, in this task, you have to set the maximum and minimum values.

### 4.4. Mutated Task

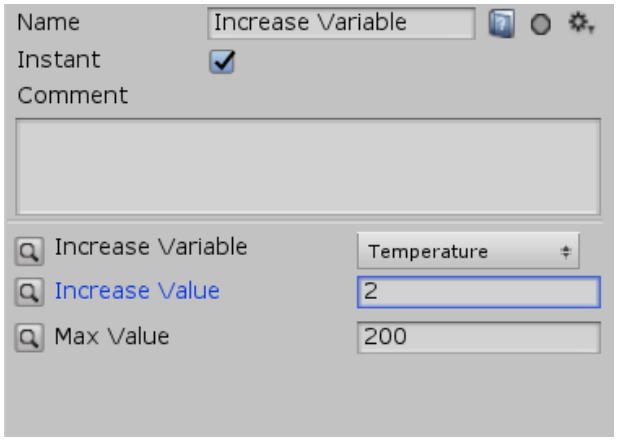


Fig. 3. Increase Variable task that increases the temperature by 2 degrees every second.

There are two types of mutated tasks; Increase Variable and Decrease Variable. Increase Variable (Fig. 3) task makes the value of a specific variable increase at regular intervals at running time. In the inspector window for this task, you must set an incremental step

and a maximum value. The Decrease Variable task does exactly the opposite.

## V. SIMULATION RESULTS

We tested for evaluation by modeling 55 adaptive task nodes in 10 virtual objects, which includes an electric fan, an induction range, a refrigerator, etc. By adding 55 adaptive tasks in this way, it was possible to create training scenarios for an infinite number of cases mathematically, although the difference is not large. Fig. 4 and 5 show how the behavior tree of an induction range in the kitchen has been changed.

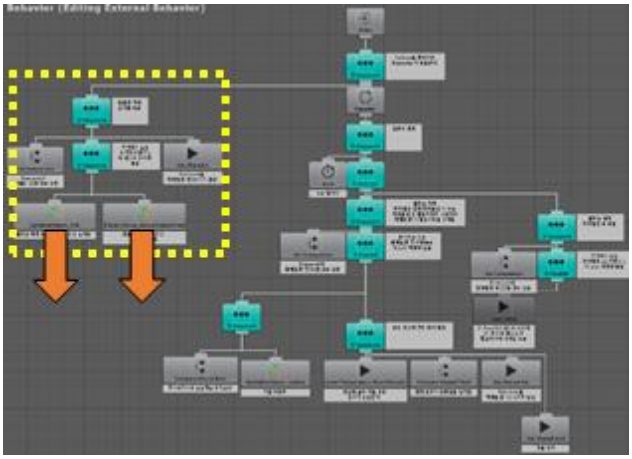

Fig. 4. Induction Range's initial behavior tree. The changes in the dotted box section can be seen in Fig. 5.

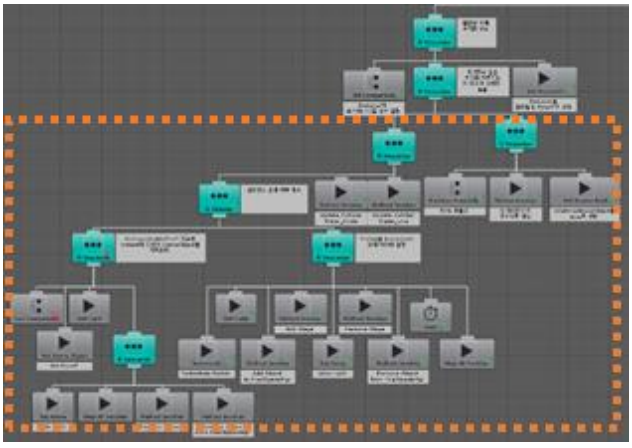

Fig. 5. Behavior tree that has changed at runtime. The subtree in the dashed box in Fig. 5 has been changed to the subtree in this figure.

## VI. CONCLUSION

In this paper, we introduced an adaptive training simulation module that can alleviate the productivity problem in virtual reality contents. We have added four new task types to behavior tree, which is most commonly used for modeling AI for objects in virtual reality systems. These newly added tasks implemented to change dynamically according to the user's skill level or reaction. As a result of applying such an adaptive simulation to the virtual reality simulation-based fire training system we were developing, it was possible to operate various training scenarios without designing new behavior trees or changing some part of them.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2019R1F1A1041854) and Electronics and Telecommunications Research Institute (ETRI) grant funded by ICT R&D program of MSIT/IITP[2019-0-01347, Developing of Realistic Fire Training Content Technology to Help Simulate Fire Sites and Improve Command Capabilities]. If you intend to utilize the contents of this report, you must disclose that the research was funded by Electronics and Telecommunications Research Institute (ETRI).

### REFERENCES

[1] Unity Technologies, "Animation System Overview," Nov. 2020; https://docs.unity3d.com/Manual/AnimationOverview.html

[2] Unity Technologies, "Unity engine visual scripting," Nov. 2020; http://unity.com/products/unity-visual-scripting

[3] Epic Games, "Introduction to Blueprints," Nov. 2020; https://docs.unrealengine.com/en-US/Engine/Blueprints/GettingStarted/index.html

[4] Epic Games, "Behavior Trees," Nov. 2020; https://docs.unrealengine.com/en-US/Engine/ArtificialIntelligence/BehaviorTrees/index.html

[5] R. Boyd, "Implementing Reinforcement Learning in Unreal Engine 4 with Blueprint," Diss. University Honors College, Middle Tennessee State University, 2017.

[6] Epic Games, "Environment Query System," Nov. 2020; https://docs.unrealengine.com/en-US/Engine/ArtificialIntelligence/EQS/index.html

[7] D. Nau, T.-C. Au, et al., "SHOP2: An HTN planning system," *Journal of artificial intelligence research*, vol. 20, pp. 379-404, 2003

[8] A. Liapiss, G. N. Yannakakis, and J. Togelius, "Adapting models of visual aesthetics for personalized content creation," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 3, pp. 213-228, 2012.

[9] A. Shoulson et al., "Parameterizing behavior trees," in

*Proc. of International conference on motion in games*, Springer, Berlin, Heidelberg, 2011.

[10] C-U. Lim, R. Baumgarten, and S. Colton. "Evolving behaviour trees for the commercial game DEFCON," in *Proc. of European conference on the applications of evolutionary computation*, Springer, Berlin, Heidelberg, 2010.

[11] OPSIVE, 2020; https://opsive.com/solutions/ai-solution/.

[12] Unity, 2020; https://unity.com.

## Authors

**Jinseok Seo** received his BS degree from Konkuk University, Korea, in 1998 and an MS and a PhD degree in the Department of Computer Science and Engineering from Pohang University of Science and Technology (POSTECH), Korea, in 2000 and 2005, respectively. In 2005, he joined the Department of Game Engineering at Dong-eui University, Korea where he is currently a professor.

His research interests include virtual reality, augmented reality, and game AI algorithms.

**Ungyeon Yang** received his BS degree in computer science and engineering from Chungnam National University, Daejeon, Rep. of Korea, in 1997. He received his MS and PhD degrees from Pohang University of Science and Technology (POSTECH), Rep. of Korea, in 2000 and 2003, respectively. Since 2003, he has been a principal researcher with Electronics and Telecommunications Research Institute (ETRI).

His research interests include wearable display, information visualization, 3D user interfaces, human factors, haptics and multimodal user interaction in the field of virtual/mixed reality and ergonomics.