Journal of Intelligent & Fuzzy Systems 37 (2019) 6071–6079
DOI:10.3233/JIFS-179190
IOS Press

# Behavior tree design of intelligent behavior of non-player character (NPC) based on Unity3D

Xianwen Zhu*
*International College, Huanghuai University, Henan, Zhumadian, China*

**Abstract**. At this stage, the rapid development of computer technology and information technology in China has provided favorable conditions for the development of the game. In order to pursue the game experience, how to use artificial intelligence in the game has become a new research hotspot. Therefore, the current situation of artificial intelligence used in games was investigated, and the principles of Unity3D game engine were studied; then the intelligent behavior model for NPC was established by using the behavior tree as the basic algorithm, and the AI architecture of the agent in the game was designed; moreover, combined with the above analysis, the behavior tree model based on Q learning algorithm was calculated, and the application of Unity3D in the game was completed; finally, a game model was developed in combination with Unity3D game engine. The results show that the behavior tree based on Unity3D game engine can realize NPC's intelligent behavior simply and efficiently, and the system can run at a good speed, which has theoretical guidance for the follow-up research of game artificial intelligence and simulation training.

Keywords: Unity3D, intelligent behavior, behavior tree, non-player character (NPC)

## 1. Introduction

In recent years, the domestic and international game industry has shown a rapid development trend, and the technical level of the game has also been greatly improved [1]. In the current game, there are more realistic pictures, more realistic and realistic game sounds, and more and more natural ways of interacting. In addition, artificial intelligence (AI) in the game has become an important aspect of game design [2]. The intelligent interaction between players in the game can greatly enhance the user's experience and improve the playability of the game. As the core technology to improve the gameplay and selling points of many commercial games, game artificial intelligence provides players with a way to interact with NPCs in the game and raise the gaming experience to a higher level [3]. Therefore, the research and application of game artificial intelligence technology has high commercial value. However, the richer the amount of information, the more troublesome it is to maintain the knowledge of NPC. The behavior tree evolved from a hierarchical finite state machine. It combines easy-to-understand logic blocks that can be used to create a variety of complex game behaviors for NPCs. At present, the relevant re-search data on artificial intelligence behavior is still quite rare. This research has important practical significance and can lay a solid foundation for future intelligent behavior research.

In this paper, the principle of Unity3D game engine is studied. The NPC intelligent behavior model based on behavior tree basic algorithm is constructed. The AI architecture of Agent in game is

*Corresponding author. Xianwen Zhu, International College, Huanghuai University, Henan, Zhumadian, 463000, China. E-mail: zhu.x.w@163.com.

designed by the logic visibility of behavior tree. The path-finding algorithm is then invoked through the framework's behavior module to calculate the optimal path between the starting point and the end point, implementing the mobile NPC, and updating the NPCs at each game frame. Finally, the intelligent NPC game features based on Unity3D were written, and the operation efficiency of the behavior tree was monitored and analyzed in the game.

The algorithm proposed in this paper has certain advantages. First of all, the NPC intelligent behavior model constructed in this paper combines easy-to-understand logic blocks, which can be used to create various complex game behaviors, and the operation process is simple. This paper designs only six NPCs in the AI architecture of the Agent in the game, so the system operates. The main consumption is the rendering of the scene model, thus ensuring the stability of the system operation.

In this paper, the behavior tree algorithm is used to study the intelligent behavior of NPC. Based on the analysis of game intelligence research progress, the behavior tree intelligent behavior style based on Unity3D is proposed. At the same time, according to the behavior tree theory, the behavior tree model based on Q learning algorithm is calculated. According to the above learning algorithm, the intelligent NPC game characteristics based on Unity3D are written, and the operation efficiency of the behavior tree is monitored and analyzed in the game. The results show that Unity3D-based NPC can effectively achieve the target's intelligent behavior under the control of the behavior tree algorithm.

## 2. Analysis of game intelligence behavior

### 2.1. Unity3D

Unity3D is a game engine developed by Unity Technologies, and the operation interface is shown in Fig. 1. It combines speed and interactivity with one powerful rendering engine, and it has a highly optimized graphics rendering pipeline for DirectX and OpenGL, and allows developers to develop 2D or 3D games efficiently and visually, so it's a fully integrated game engine. In addition, Unity3D game engine can be very convenient to release the developed games to various platforms, such as iPhone, Android and other platforms [10]. Developers can publish their work as a web game with the Unity Web Player plug-in. The player needs to download

Fig. 1. Unity3D user interface.

this very small plug-in to be able to play Unity page edition game easily. The Unity3D game engine provides a great deal of Shaders for developers through the use of the vegetation systems Unitree [11]. These Shaders fully meet the needs of developers, so that it can also run a broad and lush vegetation landscape even under the low-end hardware devices. Open terrain editor, the menu is from left to right successively: elevation and decrease of terrain, accurate mapping of terrain height, smooth terrain, drawing of topography, drawing of trees, drawing of grass, small objects, and terrain parameters setting. In addition, it supports all major file formats and can work with other applications [12]. It also uses PhysX's physics engine, which enables developers to easily implement various physical effects. The Unity game engine offers a highly sophisticated lighting rendering system for soft shadows and baking, and its shaders incorporate characteristics of easy use, flexibility, and high performance [13].

Currently, the latest version of the Unity3D game engine is Unity4.0. It can release the works to more than ten current popular platforms, such as MAC, Windows, and Android. In addition, with the purchase of a more powerful genuine Unity3D game engine, it can be installed in 2 different systems, such as Windows and Mac after authorization. The official website is unity3d.com.

The Unity3D game engine conference is booming. The first Unity3D game engine conference was held in May 2010, and the venue was in Korea, which has led to a great deal of business investment and the sign of an engine booking agreement [14]. Asia, as an important strategic place for Unity3D game engine company, held a seminar in Shanghai in 2013. Soon, Unity3D game engine company announced the abolition of support for Flash support.

## 2.2. Behavior tree

Generally speaking, game AI is divided into two layers, decision making and behavior. The behavior layer can be understood as a middleware in the game. The upper layer is the decision logic, and the lower layer is the animation resource. The goal is to translate AI requests into animated playback requests [15]. A puppy's behavior may be defined as follows: walk, run, sleep, eat, drink, etc. These can be thought of as a series of actions, or logical actions. These actions form a collection called behavior pools. The behavior tree is a way of organizing these behaviors effectively [16].

The behavior tree is a data structure that improves and replaces finite state machines. The finite state machine is a set of states and a set of rules that transform these states according to conditions [17]. Finite state machine can be used to describe the intelligent behavior of NPC. However, when the status of the role needs to be handled gradually, the finite state machine will become very difficult to understand and maintain because of the explosive growth of scale.

Here's an example of the behavior of a soldier in a game to introduce the structure of a behavior tree. Figure 2 shows the behavior tree of a soldier.

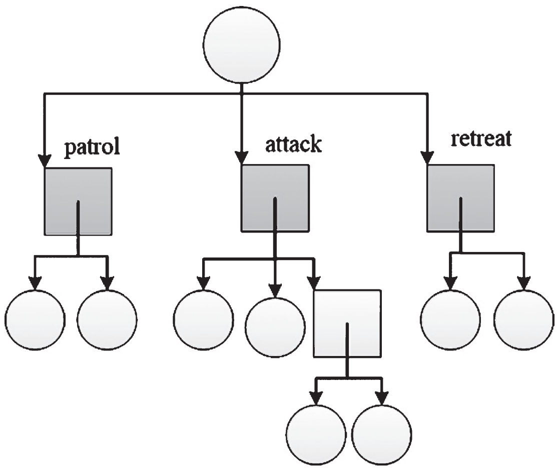


Fig. 2. Structure of soldier's behavior tree.

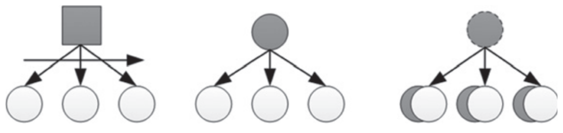


Fig. 3. Three combinations of nodes.

Composite nodes are the essence of the behavior tree. The searching of behavior nodes is realized by combining nodes. The logical direction of the whole behavior tree can be seen from the composite nodes. The visibility of logic is one of the characteristics of behavior trees [18]. In a standard behavior tree, the behavior tree consists of a combination of decision logic trend, execution patterns, and behavior nodes. The behavior node contains several execution states: success, fail, and running. A composite node that determines the direction of the logic usually contains multiple child nodes. These sub nodes can be behavioral nodes and composite nodes. The state of composite node is determined by the states of sub nodes and the combinatorial logic of itself. The commonly used combination nodes include the selector node, sequence node and the parallel node.

The selector node: one of the sub nodes is selected to execute it. Sequence nodes: the execution of all sub nodes is performed in turn. The current sub node returns the "finished" state before executing the next sub node. Parallel nodes: all sub nodes execute concurrently. The three combinations of nodes are shown in Fig. 3.

The composite node is to actually control the way nodes execute. The sub nodes can be either a composite node or a leaf node. The execution of the composite node is actually the execution of the combinational logic the composite node. It can also extend other combinations of nodes, such as loop nodes. The difference between a composite node and a behavior node is that the composite node is only responsible for the combinational logic of the behavior tree, and it does not involve the details of the game. A composite node library is defined during the development of a general behavior tree.

## 2.3. Design of AI architecture for agents in game

An agent is a behavior entity that resides in an environment and can execute actions autonomously and flexibly to meet design goals. An agent is often semantically described as an abstract functional system that resembles a computer program [19]. In this respect, agents can also be called abstract agents. It does not exist objectively, which is unlike other units in the real world, such as computer systems, biological systems, or organizations. There are also some definitions that emphasize their autonomy, and hence

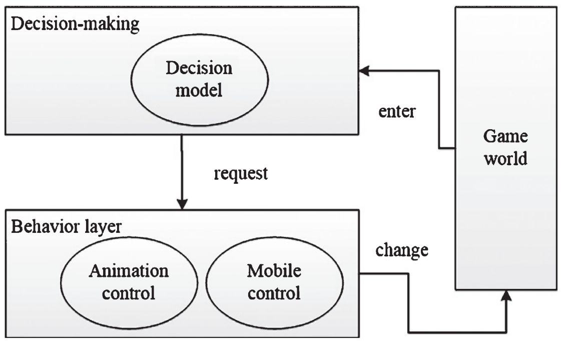


Fig. 4. AI architecture agent.

are called autonomous agents. Some people think that the essence of AI is that its behavior is goal oriented, so there is also the term "rational intelligence".

In the game, the AI should pay attention to is how to make NPC can be like a human or animal as "perception", "thinking", "action", and let the game's role look like a real human or animal response. Figure 4 describes the AI architecture design of an agent as a whole [20]. Individual soldier AI is divided into two modules: one is decision module and the other is behavior module. The decision module is responsible for choosing the right behavior. This is done using finite state machine technology for behavior choice. The input is the information about the world obtained by the individual soldier's perception system, such as whether to see the NPC, whether to hear the sound, the information of the surrounding players and so on. The output is the specific behavior, such as self-discovery and hidden fortifications, the shooting in the hidden point, squatting to avoid player fire, running to replace the hidden point, the hit rate of controllable bullets, and notifying the rest of the teammates to move to NPC to prepare for combat after finding NPC. The behavior module mainly includes two modules: animation control and motion control. The animation control of the behavior module provides API to facilitate the call of the upper layer, without considering the realization of the bottom layer, especially the complex animation jump, animation integration, animation playback, speed control and so on. After receiving the request for routing in the upper layer, the mobile control module of the behavior module will call the pathfinding algorithm to calculate the optimal path between the starting point and end point, implement mobile NPC, and update the location of the NPC at each game frame.

## 3. Behavior tree model based on Q learning algorithm

### 3.1. Q learning algorithm

Here, Q learning is used to solve the learning problem in an uncertain environment, that is, to determine the best action that can be achieved by learning by using the perceived ability of the environment. Q learning creates and maintains a table, and it maintains a state, takes the evaluation value of an action, gives the agent a function that associates the state with the reward, and then returns the reward value to the evaluation value. In this way, the evaluation value is gradually improved and optimized by continuously obtaining a reward signal in the environment.

Q (s, a) is the reward evaluation value for an action pair, that is, the cumulative return of action a in the s state. When making a decision, the optimal policy under the s state is calculated. It can consider the Q value obtained by executing the action a in the s status, that is, Q (s, a). The agent determines the execution action a in the s status. Then it is found that the instant reward is r and it goes into status s, and then a new iteration of the Q value is needed.

$$Q(s, a) = (1 - a)Q(s, a) + \alpha[r + \gamma \max Q(s', a')] \tag{1}$$

Where: Q (s, a) is the evaluation value of an action pair in the current state; $Q(s', a')$ is the evaluation value for the subsequent state action pair; r is instantaneous rewards for performing a actions for s states; $\alpha$ is the learning rate, that is, the extent to which the new Q value improves the current Q value; $\gamma$ is a conversion factor that determines the proportion of future rewards and current awards.

The choice of actions can be predefined in terms of a strategy such as greed, $\varepsilon$ greed, and softmax. Every time the greedy policy chooses the best evaluation value to take actions, that is, to select the best principle first. But in the initial stage of learning, because the data accumulated before are relatively small, the Q value cannot correctly reflect the reward value. If it takes the action with the highest Q value directly in all actions, it may result in the total execution along only one path, ignoring the existence of other better solutions. The $\varepsilon$ greed advocates that the exploration is not just greed. This exploratory and greedy strategy can be balanced by introducing a random probability to select other possible actions. The $\varepsilon$ greedy strategy is used here for action selection.

### *3.2. Algorithm flow*

The input to the algorithm is a behavior tree. Firstly, it is necessary to analyze the tree and find the deepest sequence nodes that will act as actions in reinforcement learning, so as to establish corresponding Q values in the Q values table. The main table will be partitioned into sub tables based on actions. The state of the highest Q value is placed in the corresponding condition node of the behavior tree as the Q conditional node instead of the original conditional node. Secondly, the sub nodes of the node are reordered according to the maximum Q value, so that the whole behavior tree is reconstructed and optimized.

The Q learning algorithm is performed in the preprocessing phase. The algorithm generates empirical data for decision-making. The following data will determine which action the smart unit will perform in its current state. This causes subsequent Q conditional nodes to substitute for conditional nodes. The Q conditional node is a simple lookup table that contains all the high available (high Q values) states. The corresponding actions can be determined by these states.

The generated Q value table can easily be partitioned into sub tables. The actions of the action tree correspond to the Q value sub table of the corresponding action. The sub table is sorted from high to low depending on the Q value. The status of the highest Q value of the corresponding percentage is then singled out according to the parameter. Figure 5 shows the process. The table is divided into sub tables based on actions and placed in the Q conditional nodes instead of the conditional nodes. The subsequent algorithm filters the corresponding state of the highest Q value with only a certain percentage of the parameters.

### *3.3. Implementation of Unity3D technology*

A complete Unity3d program is composed of several scenes. Each scene contains many game objects. Each object can have a number of components, in which the script inherits from the MonoBehavior components for initialization, updates and other operations. What can be seen in the scene is presented and controlled by the camera.

Scene is the basic unit of a Unity3D program. Any Unity3D program is composed of several scenes, and the program jumps in the scenes through the scripts. The scene is organized in the form of a scene map. A scene map is actually a tree structure in which each node is a game object. There is a parent-child relationship between the objects. When the parent object

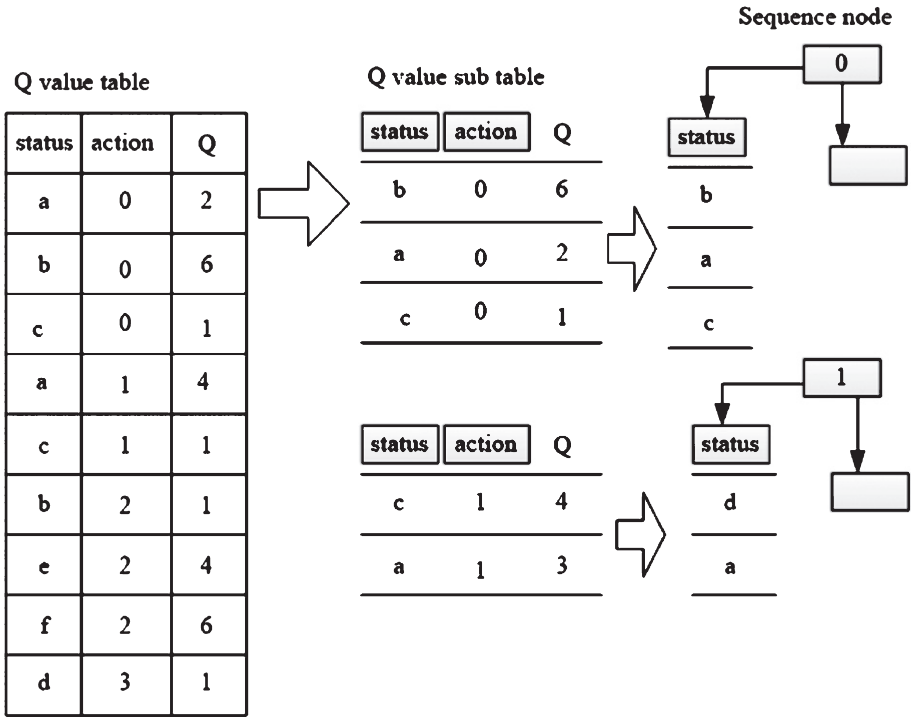


Fig. 5. Insert the Q conditional node into the behavior tree based on the Q value.

moves, rotates, and scales, the sub-level objects are transformed together, as shown in Fig. 6.

Game object: in the scene, the organization is formed in the form of a scene map. A scene map is actually a tree structure. The nodes in the tree hierarchy join the specific components to become objects that have practical functions such as rendering grids, cameras, light sources, and so on. Game objects can be packaged into.prefab format files for easy reuse.

Component can be mesh, light source, camera, particle system, physical collider, cloth, joint, audio, animation and the most important script.

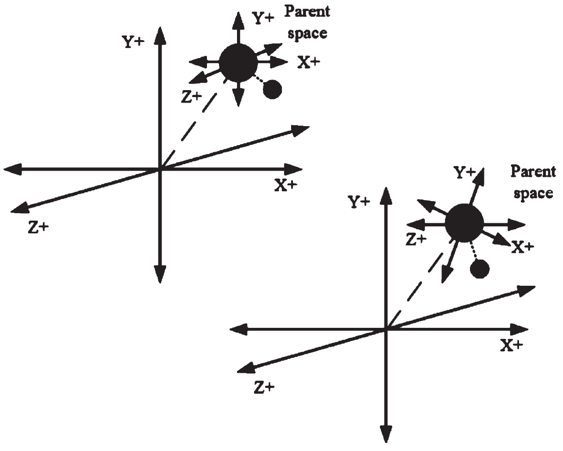


Fig. 6. Space movement.

## 4. Game design

### 4.1. Design of the game hierarchy

As mentioned earlier, each scene in Unity3D is organized in a tree hierarchy, and each game object can have several components. Therefore, the implementation of the whole game system must comply with this specification. In this specification, in order to meet the high cohesion and low coupling principle of software design, a hierarchical structure of the game is designed by using MVC model. The advantage of this is that the game structure is clear and simple. In addition, the three layers are isolated from each other and do not affect each other, which is convenient for later expanding of the functions, as shown in Fig. 7.

The game system is divided into three layers: view layer, control layer and model layer. The view layer is primarily for game objects that are added to the scene hierarchy. These objects are often rendered objects or cameras. The control layer is the management script component that is added to the main camera or the empty object. These scripts mainly aim at the control of the corresponding function module and communicate with other function modules, such as interface management class and network on-line management class. The model layer is the logical script component that is added to the render object. These scripts handle the game object logically.

### 4.2. NPC design

In this game, some basic attributes of NPC are defined, such as health value, moving speed, type and so on. It has n road point attributes as the current starting point. In the MoveTo function, NPC advances to the sub nodes of the current road point. When it is

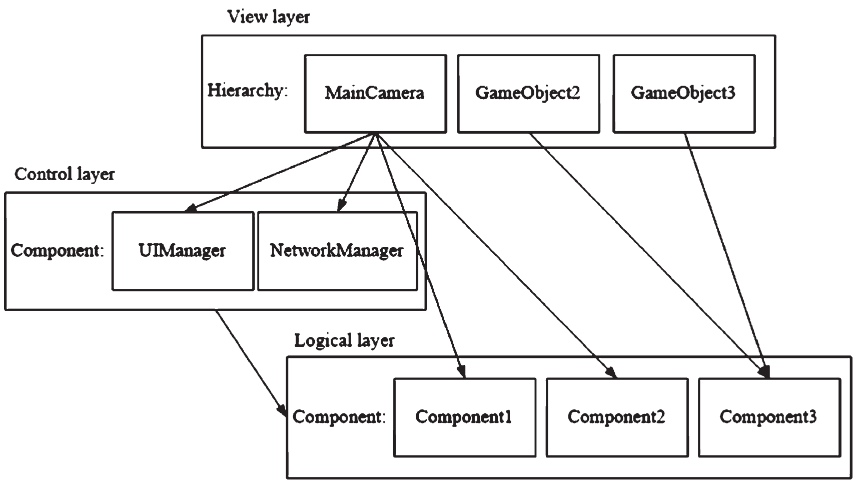


Fig. 7. Game hierarchy diagram.

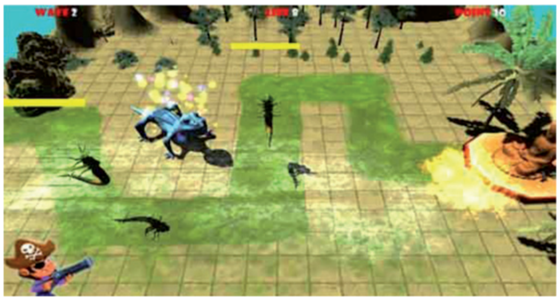


Fig. 8. Model assignment.

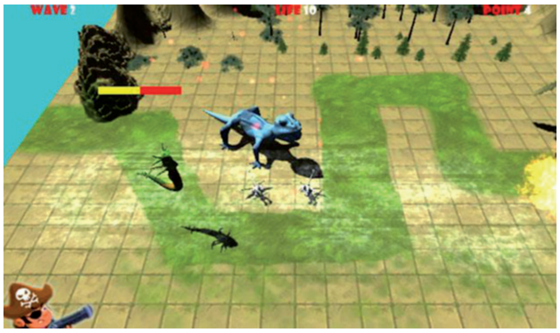


Fig. 9. Life bar effect.

close to the sub road point, it will moves to the next road ahead. It should be noted that there is no calculation of Y axis in the calculation of the distance between NPC and sub road point. Because the NPC in the air is expected to fly above the point of the road, it is considered to be the point of arrival. When NPC goes to the last road point, that is, to reach our base, it will destroy itself and make the base reduce a bit of life value. The model placed in the preset is put in the channel, so as to make the script to the model and complete the NPC settings. The running is shown in Fig. 8.

When NPC is attacked, how much damage it has done is unknown. In order to be able to indicate its remaining health values, it is necessary to create a life bar for it. In the material, a plane model with a map is prepared. The map is made up of simple yellow and red. The plane model UV is used. Yellow represents the remaining value of life, and red indicates the loss of life, as shown in Fig. 9.

The Ini function is responsible for initialization, which is mainly used to get the UV of the life bar model. In the UpdateLife function, a ratio based on the current life value and the maximum life value is calculated, and then the Pad function around is

Table 1
Rendering statistics

| Name | Parameter |
|---|---|
| FPS | 221 |
| Draw Calls | 607 |
| Tris | 412.5k |
| Verts | 574.2k |
| Used Textures | 46–16.6MB |
| Render Textures | 0–0B |
| Vran Usage | 3.7MB–50.0MB |
| Vob Total | 401–29.6MB |

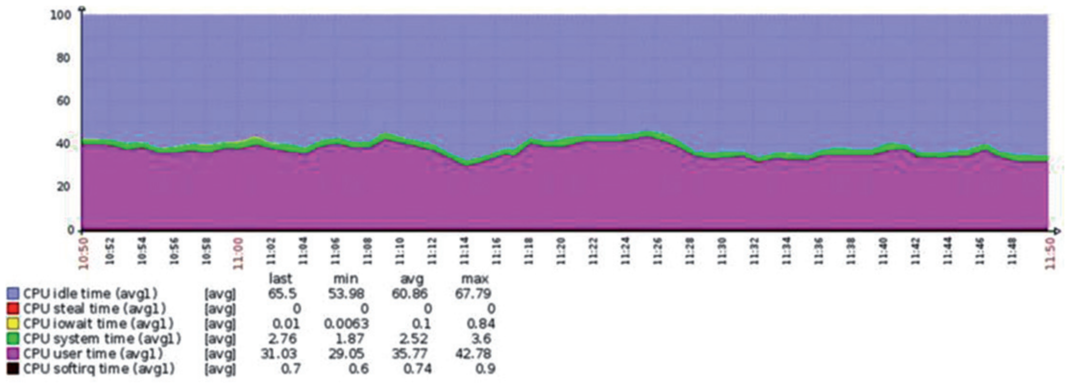
called to move the UV position. The lower the value of life, the less yellow it displays, and the more red it displays. The SetPosition function is responsible for setting the location of the life bar. Because the bar of life always moves with the corresponding NPC and appears above it, the function sets an offset parameter that moves the life bar up a certain distance.

Then, it is needed to assign the LifeBar.cs script to LifeBar.tbx as the script component and make it as a prefab file. In the Inspector window of NPC, the prefab of the life bar is associated with it. At the same time, it is necessary to add the settings bar in the Enemy.cs script, move with NPC, and update the life bar status in real time, so as to reduce the life bar when uses run the game, as shown in Fig. 9.

## *4.3. Operation efficiency of behavior tree*

The efficiency of artificial intelligence script is an important index to show the success of artificial intelligence design. The behavior tree has the advantage of simple logic. There are only 6 NPCs in this system, so the main consumption of system operation is the rendering of scene model. The system runs stably. Through the statistical function of Unity3 D, the finished program is rendered and analyzed. The analysis results are shown in Table 1.

From this table, it can be seen that the frame rate of game view updating and rendering is 221 FPS, and it can be run in real time. There are 607 objects in the scene that are plotted. In addition to the 6 characters NPC, the rest are models in the scene. The texture Used Textures is 46–16.6MB, which indicates that when the frame is drawn, the number of textures is 46, and the memory size is 16.6MB. The graphics fidelity is good and the screen itself carries 3.7MB of memory. The current video memory (memory) uses the approximate boundary, and the video card has 0.86GB memory. The number of unique grids uploaded to the graphics card is 401.



| | | last | min | avg | max |
|---|---|---|---|---|---|
| 🟦 CPU idle time (avg1) | [avg] | 65.5 | 53.98 | 60.86 | 67.79 |
| 🟥 CPU steal time (avg1) | [avg] | 0 | 0 | 0 | 0 |
| 🟨 CPU iowait time (avg1) | [avg] | 0.01 | 0.0063 | 0.1 | 0.84 |
| 🟩 CPU system time (avg1) | [avg] | 2.76 | 1.87 | 2.52 | 3.6 |
| 🟪 CPU user time (avg1) | [avg] | 31.03 | 29.05 | 35.77 | 42.78 |
| ⬛ CPU softirq time (avg1) | [avg] | 0.7 | 0.6 | 0.74 | 0.9 |

Fig. 10. CPU profile.

Figure 10 shows how the CPU is used when the program is running. The waveform in the diagram represents the time occupancy of the action tree script when the program is run. There are a few peaks in this part, which indicates that the smart script runs steadily and it meets the design requirements. In summary, the diagram shows that the behavior tree script is stable when the program is running. The behavior tree script consumes very little system computing resources and can meet the requirements of real-time operation.

## 5. Conclusions

At present, with the development of computer game industry, game AI has become one of the most important research fields of computer games. It has important research value to enrich the fidelity, reliability and playability of computer games. In this article, the intelligent behavior of NPC was focused on; then based on the analysis of the research progress of game intelligence, the intelligent behavior style of behavior tree based on Unity3D was proposed; moreover, in combination with the above analysis, the Unity3D game engine, behavior tree and game intelligent AI architecture were studied, and the implementation of Unity3D in game was obtained; at the same time, according to the theory of behavior tree, the behavior tree model based on Q learning algorithm was calculated, and the algorithm flow was summarized; in addition, according to the above learning algorithm, the intelligent NPC game characters based on Unity3D were written, and the operation efficiency of the behavior tree was monitored and analyzed in the game. The results show that the Unity3D based NPC can effectively achieve the intelligent behavior of the target under the control of the behavior tree algorithm. In addition, the system resource is less, which can effectively meet the requirements.

## References

[1] M.J. Lee, An artificial intelligence evaluation on FSM-based game NPC, *Journal of Korea Game Society* **14**(5) (2014), 127–136.

[2] C. Li, R. Sun and Y. Dai, Intelligent exhibition platform of Chinese ancient farming virtual scene based on Unity3D, *Transactions of the Chinese Society of Agricultural Engineering* **33**(1) (2017), 308–314.

[3] I. Sagredo-Olivenza, Supporting the construction of a GUI component for specifying the behavior of non-player characters in unity, *International Journal of Creative Interfaces & Computer Graphics* **6**(1) (2015), 38–55.

[4] H. Zhong, X. Chen and W.Q. Shu, Research and development on 3D simulation training evaluation system of substation based on unity3D, *Applied Mechanics & Materials* **727-728** (2015), 987–990.

[5] K. Jiang, F.Y. Zheng and H.K. Teng, The design of three-dimensional computer room monitoring system based on unity3D, *Applied Mechanics & Materials* **602-605** (2014), 1838–1841.

[6] J.H. Bae and A.H. Kim, Design and development of Unity3D Game Engine-based smart SNG (Social Network Game), *International Journal of Multimedia & Ubiquitous Engineering* **9**(8) (2014), 261–266.

[7] D. Lu, Testing the effects of memory structures and recall on non playable characters in Unity3D. California State University Northridge, 2013.

[8] Pineda P. Villalonga, Control de NPC en un ARPG en Unity3D. Disseny De Videojocs, 2013.

[9] L.X. Ji and J.H. Ma, Research on the behavior of intelligent role in computer games based on behavior tree, *Applied Mechanics & Materials* **509** (2014), 165–169.

[10] H. Xiong, Y. Zhu and F. Zhang, Formal specification and verification of IEC 61850 IED interoperability based on behavior tree, *Automation of Electric Power Systems* **37**(24) (2013), 66–71.

[11] T. Kaul, T. Meyer and W. Sextro, Formulation of reliability-related objective functions for design of intelligent mechatronic systems, *Proceedings of the Institution of*

*Mechanical Engineers, Part O: Journal of Risk and Reliability* **231**(4) (2017), 390–399.

[12] H. Yongtao, W. Zheng and L. Diming, The research of product intelligent design and properties based on behavior-flow time-series transmission nets, *International Journal of Digital Content Technology and its Applications* **7**(9) (2013), 246.

[13] T.L. Du, T. Carmichael and T. Govender, In vitro: *In vivo*, and in silico evaluation of the bioresponsive behavior of an intelligent intraocular implant, *Pharm Res* **31**(3) (2014), 607–634.

[14] R.A. Herrmann, General Intelligent Design (GID) and Mindom Behavior, 2015.

[15] M. Brandstätter, NPC - Non Player Character, 2014.

[16] D.M. Buede, P.J. Sticha and E.T. Axelrad, Conversational Non-Player Characters for Virtual Training, 2016, pp. 389–399.

[17] Rasim, A.Z.R. Langi and Munir, A survey on adaptive engine technology for serious games, **1708**(1) (2016), 371–384.

[18] R.D. Saputro, Implementasi Artificial Intelligence sederhana terhadap Non Player Character pada Horror Game 3D “SESAT” menggunakan Unity 3D,bahasa C# dan JavaScript berbasis Windows, 2015.

[19] S. Feng and A.H. Tan, Towards autonomous behavior learning of non-player characters in games, *Expert Systems with Applications* **56**(C) (2016), 89–99.

[20] M.J. Lee, Implementation of NPC artificial intelligence using agonistic behavior of animals, **12**(1) (2014).