Research on the Application of Artificial Intelligence in Games

Jiachen Zhang, Huihuang Li, Yi Teng*, Ruilin Zhang, Qiang Chen and Guoming Chen
School of Computer Science, Guangdong University of Education, China
*Corresponding anthour: tengyi@gdei.edu.cn

2022 9th International Conference on Digital Home (ICDH) | 978-1-6654-5478-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDH57206.2022.00039

***Abstract*—With the advent of the information age, artificial intelligence technology (AI) is widely used in various fields, and more and more applications are used in game development. At present, through artificial intelligence technology, the non-player character (NPC) has a certain degree of self-learning and self-optimization ability to improve the playability of the game, which has become a hot spot in the application research of video games. In this paper, a game artificial intelligence system is constructed through finite state machine, fuzzy state machine, artificial neural network and genetic algorithm, and the NPC that meets the requirements of the game is obtained. This paper aims to study the application of artificial intelligence technology in game development, which has certain reference significance for artificial intelligence technology in games application in the future.**

Keywords-artificial intelligence; finite state machine; fuzzy state machine; artificial neural network; games

I. INTRODUCTION

In 2021, the 19th Asian Games in Hangzhou officially announced 6 e-sports events, and e-sports officially became a sports event in our country. Under the trend of globalization, China has carried out unprecedented economic and cultural exchanges with other countries in the world. Concepts such as "e-sports" and "video games" have taken on a new look in the hearts of the Chinese people. They have changed from the "teenage mental drugs" that everyone criticized to a formal sports competition.

With the rapid development of the Internet and computer technology, people's requirements for video games are getting higher and higher. To meet the experience needs of players and increase the playability of the game, artificial intelligence technology (AI) is applied in the development of video games [1]. In-depth exploration of the application of artificial intelligence in games development has certain economic and social benefits, and the application of artificial intelligence in games can provide a practical basis for the development of artificial intelligence, and artificial intelligence can also provide a theoretical basis for the application of artificial intelligence in games [2-3].

The development potential of artificial intelligence technology in the game industry is limitless, and highly intelligent non-player characters (NPC) can bring players a more interesting game experience. Based on the use of the C# programming language in the Unity3D engine and combined with the visual studio integrated development environment to develop a shooting game with artificial intelligence, this paper focuses on the discussion and application of the AI mechanism of the NPC: finite state machine (FSM), fuzzy state machine (FuSM) and artificial neural network (ANN), and finally get intelligent NPC characters. This paper aims to study the application of AI artificial intelligence technology in game development and contribute to the development of artificial intelligence in the game industry.

II. RELATED TECHNOLOGIES

A. Finite state machine

Finite state machine means that there are limited "states" in the machine, and the game object can only have one of these states at any time, and switch states according to the information data input from the outside world [4]. The characteristics of the finite state machine are that the programming is simple and easy to debug, the calculation requirements are low, and the state switching of the finite state machine can be directly observed, which is direct and flexible.

Finite state machines have different implementations depending on the number of states. If the number of states is small, you can simply use a series of "if...else..." judgment statements or switch statements to judge and switch states; but if the number of states is large, many "if...else..." or switch statements will make the code bloated and not easy to debug. This paper mainly solves the above problems and realizes the finite state machine of NPC by using the "state pattern" in the design pattern.

B. Fuzzy state machine

Fuzzy state machine is also a kind of state machine, but it is a variant of finite state machine based on fuzzy logic, which is a superset of traditional logic that extends the concept of "partial truth" [5].

Unlike finite state machine, fuzzy state machine is not general, that is, fuzzy logic can have multiple states at the same time [6]. But like finite state machine, fuzzy state machine keeps track of a series of possible states, calculating an "activation level" (membership) in each state to decide which given states the game object should be in. Under the control of the fuzzy state machine, the behavior of the NPC is not determined by one state, but by a series of states that are currently activated.

C. Artificial neural network

Artificial neural network is a mathematical model that imitates the structure and function of biological neural network [7-8]. Artificial neural network is composed of

978-1-6654-5478-0/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDH57206.2022.00039

input layer, hidden layer, and output layer. Each layer has several neuron nodes, and each neuron node represents a specific output function (excitation function). Each neuron node in each layer is connected to all nerve nodes in the next layer, and the connection of each two neuron nodes represents the weighted value of the connection signal, which is called the weight, equivalent to human memory. Each layer except the input layer also has a bias value. The input layer data, the weight, bias value, and excitation function jointly determine the output layer data.

Although artificial neural networks have the advantages of autonomous learning, associative storage, and finding optimal solutions, the "non-linear" nature of artificial neural networks makes it difficult to control the computational process in game development. Artificial neural network is "nonconstant qualitative", that is, it has the nature of self-learning and seeking the optimal solution autonomously, which makes the output value of a neural network difficult to predict [9-11]. These two elements are important to keep in mind during game development.

In this paper, the number of data required is the number of neuron nodes in the output layer, and the number of elements affecting the output data is the number of neuron nodes in the input layer.

D. Genetic algorithm

The genetic algorithm simulates Darwin's theory of biological evolution and takes the optimization technology of natural selection and biological genetic mechanism as the premise, and obtains the optimal solution by simulating the natural evolution process [12]. Genetic algorithm follows the principle of survival of the fittest in nature, and converts the problem-solving method into a method similar to the cross-mutation of chromosome genes according to mathematical knowledge. It is a kind of randomization algorithm that draws on natural selection and biological genetic mechanism.

III. NPC AI MECHANISM

A. Perception system

Patrol, attack, pickup, and other states in the NPC finite state machine all need to detect game objects in the environment, so before constructing the NPC finite state machine, it is necessary to construct the NPC perception system.

The NPC perception system only needs to implement the vision system [13]. Set the NPC's sight distance to 10 and the degree to 120°. The visual detection mechanism is to emit rays at the location of the NPC according to the viewing angle and the distance, detect the information of objects colliding with the rays, and use it as the input condition of finite state machine.

B. Finite state machine

Define seven states of NPC: patrol, attack, escape, get-away, hidden, pickup, and death. And define state priority: death > get-away > hidden > pickup > escape > attack > patrol.

NPC state behaviors and state switching conditions is shown in Fig. 1, as follows.

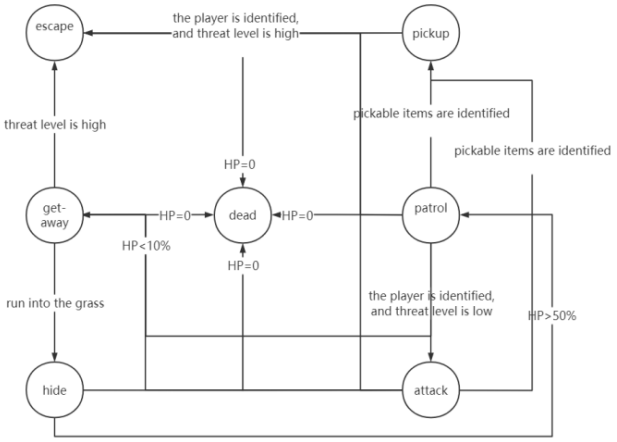

Figure 1. NPC state switching diagram.

1) Patrol state: If the player is not recognized within the NPC's vision range, it will patrol in a random direction; if the player is recognized and the player's threat level is low, it will switch from the patrol state to the attack state; if the player is recognized and the player's threat level is high, it will switch to the escape state; if an item that can be picked-up is identified, it will switch to the pickup state.

2) Attack state: Attack the player. If the player's threat level increases during the battle (players pick up high-level equipment or take medicine bottles), the NPC will switch to the escape state; if the pick-up item is identified during the attack, it will switch to the pickup state; if the attack is on the verge of death, it will switch to get-away state.

3) Get-away state: Find and go into the grass. If the player's threat level is greatly increased during the get-away state, it will switch to the escape state; if the player's threat level remains unchanged, enter the grass and switch to the hidden state.

4) Escape state: Find and go to the portal, reaching the portal means the escape is successful.

5) Pickup state: Detect and pick up the item at the location of the item.

6) Hidden state: Slowly recovers its own blood volume (HP) in the hidden state. When the HP level is higher than 50%, it will switch to the patrol state.

7) Dead state: When the NPC's HP is 0, it enters the dead state and destroys the NPC object.

Finite state machine runs the state behavior method in the current state object frame by frame. When NPC visual perception system obtains state switching condition, the finite state machine creates a new state object and runs the state behavior method in the new state object frame by frame.

The operating mechanism of the finite state machine is

as follows. Finite state machine runs the state behavior method in the current state object frame by frame. When NPC visual perception system obtains state switching condition, the finite state machine creates a new state object and runs the state behavior method in the new state object frame by frame.

C. Fuzzy state machine

Define the NPC's blood volume with fuzzy state: healthy, injured and dying.

Define the HP interval of healthy, injured, and dying states. The schematic diagram of NPC states at different HPs is shown in Fig. 2.

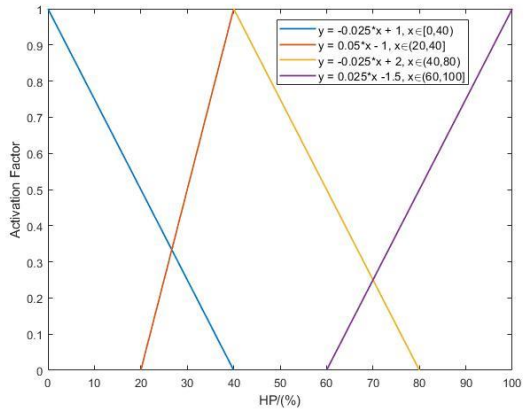

Figure 2. Schematic diagram of state activation factor with different HPs.

As Fig.2 shows, when HP is at [20%, 40%], NPC is in a healthy and injured state at the same time; when HP is at [60%, 80%], NPC is in injured and dying state at the same time, but membership of state is different.

The operating mechanism of the fuzzy state machine is as follows. Fuzzy state machine runs the state behavior methods for all active state objects in the current state frame by frame. When NPC visual perception system obtains state switching condition, the fuzzy state machine runs the state behavior methods for all active state objects in the current state frame by frame.

Define NPC's blood volume fuzzy state machine: NPC will judge whether it needs the medicine bottle of restore or get-away according to its own blood volume.

D. Artificial neural networks

In complex game scenarios, when the NPC is in a get-away or escape state, an optimal escape route is needed to calculate. Schematic diagram of the artificial neural network is shown in Fig. 3.

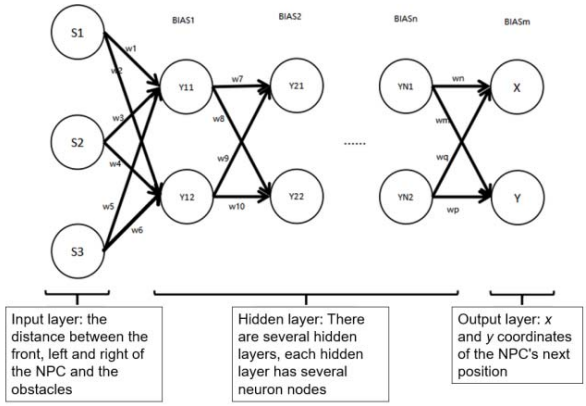

Figure 3. Schematic diagram of the artificial neural network.

Data meaning diagram of the neural network input layer and output layer is shown in Fig.4. Input layer data S1, S2, and S3 represent the distance between the NPC and the obstacles in the current three directions (the front, left, and right directions). Output layer data of the x-coordinate and y-coordinate are identified as the next movement position of the NPC.

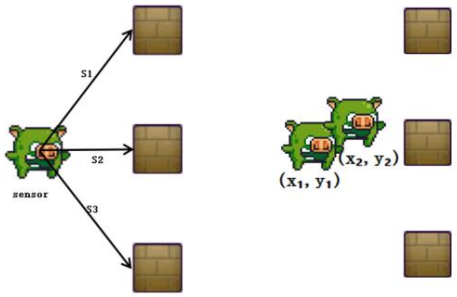

Figure 4. Data meaning diagram of the neural network (a)input layer and (b)output layer.

Schematic diagram of artificial neural network combined with genetic algorithm is shown in Fig.5. First, initialize the NPC population, and then add an artificial neural network to each NPC individual in the NPC population. According to the distance between the NPC and the obstacle as the input layer data, run the artificial neural network to obtain the output layer data of the x-coordinate and y-coordinate that identifies the next movement position. Combined with the genetic algorithm, the fitness function is designed according to the distance between the NPC and the portal and the overall distance walked by the NPC to obtain the fitness of the neural network. Finally, it is sorted according to the fitness of each NPC individual, and the best performance is selected. Several NPC individuals are cross-bred, and the gene mutation is introduced to obtain the next generation of NPC population, and this cycle is repeated until the artificial neural network of the best escape route is obtained.

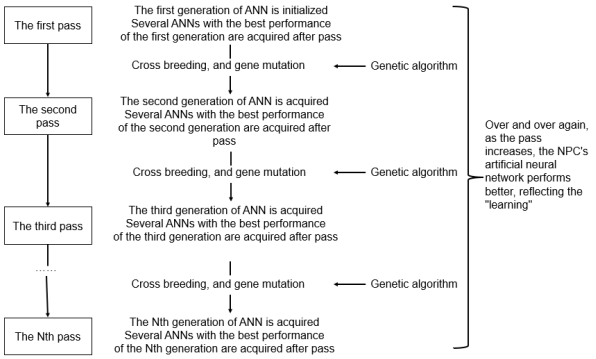

Figure 5. Schematic diagram of artificial neural network combined with genetic algorithm.

IV. Implementation Points and Tests

This paper uses the C# programming language in the Unity3D engine and combines the Visual Studio integrated development environment to develop a shooting games with artificial intelligence, mainly including resource implementation, menu scenes, standard mode scenes (including maps, players, game items, NPC), FSM scene (including NPC perception system, NPC finite state machine), FuSM scene (NPC blood volume fuzzy state machine), ANN scene (through artificial neural network and genetic algorithm to achieve the best escape route in the escape state).

A. NPC perception system

Create a scene object manager: Create an empty object and add an ItemManager.cs script to it. In this script, define an array for each game object to hold all the game objects in the scene so that the NPC sense system can acquire and sense the game object.

Create an Aspect class for the perception system: create a script and name it Aspect, assign the script to all game preforms, define aspect enumeration in the Aspect class, define all game categories in the aspect enumeration, and finally define a common aspect enumeration variable.

Sense class for the perception system: Define the Aspect of the game object to be detected in the Sense class; define the perception detection rate and timer for periodically perceiving scene objects; define two virtual methods for initialization and sense perception update, and call the sense perception update method in Unity's Update method.

EnemyPerspective class of NPC's perception system: EnemyPerspective class inherits sense class Sense and overwrites the initialization virtual method and updating scene virtual method of Sense class to realize NPC's periodic perception of scene objects. Get all the original game objects of the game scene in the initialization method. In the update scene method, detect the game objects that the NPC needs to detect in real time and execute the corresponding code according to the detection result and the priority of the NPC state.

- When the player is identified in patrol state, the NPC will switch to attack state; if the player's weapon level is too high, the NPC will switch to escape state; when the weapon or medicine bottle is identified, the NPC will switch to pickup state;
- When weapons or medicine bottles are identified in the attack state, NPC will switch to the pickup state preferentially;
- When NPC is in pick state, does not change any state;
- Identify its own blood volume and switch to patrol state or get-away state according to its blood volume;
- In the get-away state, the recovery medicine bottle is identified, NPC will switch to the pickup state;
- In the escape state, the pickable weapon is identified, NPC will switch to the pickup state.

B. NPC finite state machine

This paper realizes the finite state machine of NPC through the "state pattern" in the design pattern.

First, abstract all states of NPC, define the basic state interface EnemyBaseState of NPC, and define the state behavior method Handle() and update method Update() in the interface. Define all specific state classes of NPC and implement the EnemyBaseState state interface.

Then add the finite state machine class Fsm script for the NPC object. Define the basic state variable state in Fsm.cs to represent the current state; call the state.Update() and state. Handle() statements in the Update() method to implement the invocation and update of the current state behavior method; create a patrol state class by default is assigned state, indicating that the initial state of the NPC after the game starts is the patrol state.

Finally, the input conditions of the state are obtained in the perception system. When the input conditions satisfy the state switching, a new specific state class object is created and assigned to the state variable state of the finite state machine, that is, the switching of the NPC state is realized by changing the state object.

Test by Instantiating a weapon near the NPC, and the NPC enters the pickup state after identifying the weapon from the patrol state. The test result is shown in Fig. 6.

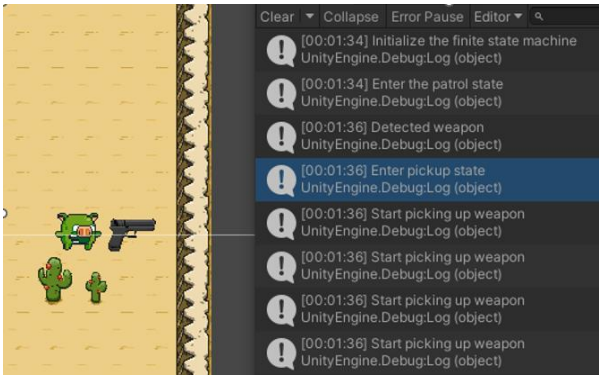

Figure 6. Test results of pickup state.

C. NPC blood volume fuzzy state machine

The idea of constructing a fuzzy state machine is similar to that of the NPC finite state machine. The difference is that the state variable defined by a fuzzy state machine is a set variable because a fuzzy state machine can have multiple states at the same time, while a finite state machine can only have one state at the same time.

Abstract all fuzzy states of NPC, define the NPC basic fuzzy state interface FusmBaseState, define the method Evaluate() to calculate whether the state activation factor reaches the activation level in the interface; define the update method Update() executed every frame; define three check methods to check the validity of the value of the activation factor, there are the CheckLowerBound (float lowerBound = 0.0f) method that checks the lowest boundary of the activation factor, the CheckUpperBound(float upperBound = 1.0f) method that checks the highest boundary of the activation factor, and the CheckBounds that checks the boundary of the activation factor (float lowerBound = 0.0f, float upperBound = 1.0f) method. Finally, define all the specific state classes of the NPC blood value fuzzy state machine and implement the FusmBaseState fuzzy state interface.

Add Fusm script to NPC object to represent fuzzy state machine, define variable FsmEnemy fsmEnemy in the script to represent NPC object; define public List<FusmBaseState> states to represent all state sets of fuzzy state machine; define public List<FusmBaseState> activated to represent the set of all active states in fuzzy state machine. Traverse the activated collection in the Update() method of the fuzzy state machine to get each activated state activatedState, and call the activatedState.Update() method to call the state behavior method of each activated state.

Test by attacking NPC to reduce its HP: NPC is completely healthy when the game is start, and keep attacking NPC. When the HP of NPC is 0.75, it is both healthy and injured, but it belongs to the healthy state, as shown in Fig. 7.

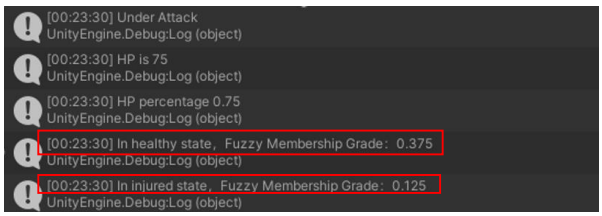

Figure 7. NPC belongs to the healthy state.

D. ANN game scene

In the get-away or escape state, NPC will stop when it encounters an obstacle without training by the neural network and genetic algorithm, as shown in Fig. 8.

Figure 8. NPC is stopping when it encounters obstacles in the get-away state without training by the neural network and genetic algorithm.

This paper mainly uses the artificial neural network and genetic algorithm to obtain the best escape route in the get-away or escape state.

Define the ANN.cs script to represent the artificial neural network of NPC. In this script, define and initialize the basic structure of the artificial neural network, including the input layer, hidden layer, output layer, weights, and bias terms. Except for the input layer and output layer, the rest components are initialized with random values; then define the method (float, float) RunNetwork(float a, float b, float c) of running the artificial neural network, and provide 3 parameters representing the distance between the NPC and the obstacles in 3 directions, and obtain the output layer data by running the neural network, that is, the x-coordinate and y-coordinate values of the NPC's next movement direction.

Add the artificial neural network controller script ANNController.cs for the NPC. In this script, define the NPC artificial neural network object ANN network; define 3 float variables aSensor, bSensor, cSensor to represent 3 sensors of the NPC, and combine the ray detection to obtain the distance between the NPC and the obstacles in the current direction and the left and right directions as the input value of the artificial neural network; call the method network.RunNetwork(aSensor, bSensor, cSensor) to obtain the x-coordinate and y-coordinate of the next movement direction; define the Escape method to move to the next position according to the obtained x-coordinate and y-coordinate.

The operation of a random artificial neural network of an NPC cannot obtain the best results, so it is necessary to combine the genetic algorithm. Define the genetic manager script GeneticManager.cs, in which a large number of ANN game objects are defined as the NPC population (each NPC individual has a neural network, but the ultimate goal of the program is to obtain the optimal solution artificial neural network, not NPC individuals. So initializing the population only needs to initialize a large number of ANN objects). Train the random artificial neural network of the initial population individuals, and define the fitness function according to the total distance recorded during the operation of the artificial neural network and the distance between the NPC and the portal. Obtain the fitness of the artificial neural network. The greater the fitness, the better the performance

of the neural network. Sort the fitness of all neural networks of the first generation, select the best performing artificial neural network for cross-breeding, and introduce genetic mutations to obtain the second-generation population. And so on until an artificial neural network that can correctly calculate the escape route is obtained.

NPC trained with neural networks and genetic algorithms will bypass obstacles to reach the portal. As shown in Fig. 9, when in the escape state, the NPC will go through an arc line to bypass the obstacle and reach the portal to complete the escape.

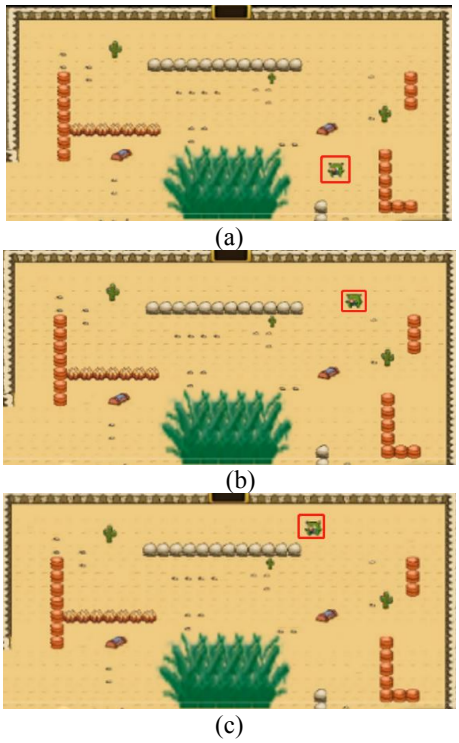

Figure 9. NPC (a)starts to escape, (b) bypasses the obstacle, (c) moves to the portal after bypassing the obstacle.

V. CONCLUSION

This paper realizes seven states of the NPC through the finite state machine. On the basis of the finite state machine, the fuzzy logic is used to construct a fuzzy state machine for the NPC. Finally, in the get-away or escape state, combining artificial neural network and genetic algorithm to achieve the best escape route.

Although the above functions have been implemented, there are still many aspects that can be optimized and expanded. For example, the NPC personality system can be introduced. NPCs with different personalities will have different states under the same conditions. This expansion method can greatly increase the NPC's diversity and authenticity; you can also add auditory and even smell perception systems to NPCs, so even if the player is not within the sight of the NPC, as long as the distance is close enough, the NPC can detect the player through hearing and smell, and so on.

In the current information age, the research of artificial intelligence technology in games is a very important direction. Since the relevant research materials on the application of artificial intelligence in game development are still relatively lacking, the artificial intelligence proposed in this paper has certain reference value for future research on game artificial intelligence and future intelligent behavior research.

ACKNOWLEDGMENT

The work described in this paper was supported by National Natural Science Foundation of China (No.62172452), Science and Technology Projects in Guangzhou (No.202103010004), Guangzhou Basic and Applied Basic Research Foundation (No. 202102021240), Research platforms and research projects of ordinary universities in Guangdong Province (No. 2021KQNCX061), Natural Science Foundation of Guangdong Province (No.2018A0303130169).

REFERENCES

[1] I. Millington, and J. Funge, Artificial intelligence for games, CRC Press, 2018.

[2] G. Skinner, and T. Walmsley, “Artificial intelligence and deep learning in video games a brief review,” IEEE 4th International Conference on Computer and Communication Systems (ICCCS), 2019, pp. 404-408.

[3] I. Zarembo, “Analysis of artificial intelligence applications for automated testing of video games,” Eevironment Technologies Resourses Proceedings of the International Scientific and Practical Conference, 2019, pp. 170-174.

[4] A. Solihin, E. W. Hidayat, and A. P. Aldya, “Application of the finite state machine algorithm on 2D platformer rabbit games vs zombies,” Jurnal Online Informatika, 2019, vol. 4(1), pp. 33-38.

[5] M. Pirovano, “The use of fuzzy logic for artificial intelligence in games,” University of Milan, 2012.

[6] M. Rozikin, R. Dijaya, C. Taurusta, “Education game indonesian old museum explorer using fuzzy state machine,” Journal of Physics: Conference Series, 2021, vol. 1764(1), pp. 012059.

[7] Y. K. Hu, X. Fan, L. T. Yu, and Z. X. Luo, “Graph based neural network regression strategy for facial image super-resolution,” Journal of Software, 2018, vol.29(4), pp.914-925.

[8] Z. H. Lv, J. Y. Wang, and X. N. Luo, “Editorial: Neural computing in next-generation virtual reality technology,” Neural Computing & Applications, 2018, pp. 1-4.

[9] S. B. Maind, and P. Wankar, “Research paper on basic of artificial neural network,” International Journal on Recent and Innovation Trends in Computing and Communication, 2014, vol. 2(1), pp. 96-100.

[10] S. C. Wang, “Artificial neural network,” Interdisciplinary computing in java programming Springer, Boston, MA, 2003, pp.81-100.

[11] A. D. Dongare, R. R. Kharde, and A. D. Kachare, “Introduction to artificial neural network,” International Journal of Engineering and Innovative Technology (IJEIT), 2021, vol. 2(1), pp. 189-194.

[12] M. J. Kim, and C. W. Ahn, “Hybrid fighting game AI using a genetic algorithm and Monte Carlo tree search,” Proceedings of the genetic and evolutionary computation conference companion, 2018, pp. 129-130.

[13] B. Q. Zhao, H. H. Li , R. M. Wang , and X. N. Luo, “Automatic generation of informative video thumbnail,” 2020 8th International Conference on Digital Home (ICDH), 2020.