УДК 004.81

Разработка искусственного интеллекта в видеоиграх

Сергей Сергеевич Иванов[1], Елена Владиславовна Пономарева[2]

[1, 2] Уральский федеральный университет
имени первого Президента России Б. Н. Ельцина, Екатеринбург, Россия

[1] aeabr01@gmail.com

[2] ev.ponomareva@urfu.ru

Аннотация. Статья посвящена теме разработки искусственного интеллекта в видеоиграх. В статье рассматриваются основные технологии для создания ИИ в видеоиграх. Автор дает краткий обзор применения технологий на простом прототипе игры, а также сравнение этих методов.

Ключевые слова: искусственный интеллект, видеоигры, поведенческое дерево, конечный автомат, ИИ на основе полезности.

AI Development in Video Games

Sergey S. Ivanov[2], Elena V. Ponomareva[2]

[1, 2] Ural Federal University named after the First President of Russia B. N. Yeltsin, Ekaterinburg, Russia

[1] aeabr01@gmail.com

[2] ev.ponomareva@urfu.ru

Abstract. This paper is devoted to the development of artificial intelligence in video games. The article examines the main technologies for creating AI in video games. The author gives a brief overview of the technology application on a simple game prototype, as well as a comparison of these methods.

Keywords: artificial intelligence, video games, behavior tree, finite state machine, utility-based AI**.**

Introduction

The video game industry is one of the most perspective and popular areas in entertainment. From the very beginning of the video games history people have tried to create AI in games that could make the gameplay interesting and exciting. AI in video games does not have to be smart and outplay the players easily, it should seem smart and bring fun to the game. Video game developers have tried many techniques and algorithms for creating AI, but there is still no answer to what techniques are most suitable for different genres and whether it is possible to create a universal technique.

This paper examines different approaches for creating AI using the example of the turn-based strategy genre. This genre assumes that the AI will not have an advantage in reaction speed and accuracy, because the game is played by moves, so the AI must confront the player with tactics and calculating the player's actions. However, so far, the AI in games of this kind is significantly inferior to expert players, so to create difficulty situations in a game, it is needed to scale up enemy’s damage or something else. Two popular approaches will be tested and compared in context of simple turn-based game prototype.

Background

In the modern world, the video game industry is developing very rapidly. Some games can make more money than movies and other types of electronic entertainment. Along with the demand for games, the quality bar is also growing, because it is not enough for modern games to have a beautiful picture and a simple plot to please the player. Modern players are already difficult to surprise, so game developers focus on exciting and believable gameplay. Game AI plays a significant role in this.

Artificial intelligence in games serves various purposes, such as modeling plausible NPC behavior, bringing fun to the game, creating some difficulties for the player, randomly generating levels, and many others. The AI doesn't have to beat the player, but rather make the gameplay enjoyable and fun, because of that many

traditional AI design techniques may be useless in the context of video games because they focus on the result, such as winning, rather than the process.

Methods overview

The main algorithm for creating AI in old video games was finite state machine. A finite state machine is an abstract machine that can exist in one of several different and predefined states. A finite state machine can also define a set of conditions that determine when the state should change. The actual state determines how the state machine behaves. Finite state machines date back to the earliest days of computer game programming. For example, the ghosts in Pac Man are finite state machines. They can roam freely, chase the player, or evade the player. In each state they behave differently, and their transitions are determined by the player's actions. For example, if the player eats a power pill, the ghosts' state might change from chasing to evading. But the complex and interesting AI in modern games cannot be implemented using this algorithm. With a lot of parameters developers can simply get confused. It is very difficult to expand FTM.

Today, the behavior tree is the leader in the field of game AI. This method was used to design the behavior of NPCs in such popular games as Halo, Bioshock, and Spore. The behavior tree is a graph that shows all possible actions of the AI and how they can be reached. The behavior trees work like this: the first time they are evaluated (or they are reset) they start from the root (parent nodes act like selectors) and each child is evaluated from left to right. The child nodes are ordered based on their priority. If all of a child node's conditions are met, its behavior is started. When a node starts a behavior, that node is set to 'running', and it returns the behavior. The next time the tree is evaluated, it again checks the highest priority nodes, then, when it comes to a 'running' node, it knows to pick up where it left off. The node can have a sequence of actions and conditions before reaching an end state. If any condition fails, the traversal returns to the parent. The parent selector then moves on to the next priority child. However, this approach is difficult to expand like the FTM approach. Nevertheless, it is the most popular approach for the game AI.

The behavior tree is being replaced with the new approach called Utility AI. This approach assigns some score to the actions of AI and then chooses the action that has gained the biggest score. This method can help designing complex AI systems. This approach can be easily expanded; it is used in many popular games such as Killzone 2 and Apex. The advantages of this method are:

- design simplicity– first, the designer can easily explain to the programmer what needs to be done in a simple language, without any complex terms of states, decorators and sequences.
- scalability – unlike a finite state machine or a behavior tree, rules can be freely added on top of existing AI logic and no important connections or transitions will be broken.
- financial benefit – as a result of the above points, the game will be developed faster and with fewer errors, which will lead to an increase in profit.
- easy use – input parameters of the AI can be described using curves. The curves enable the Utility AI to make decisions across a large spectrum of inputs and give it a fuzzy-logic quality. In practice, the Utility AI can thus make rather good decisions even in scenarios that the AI programmer has not foreseen. The ease-of-use of the Utility AI is also extended into domains that are normally difficult to handle for Behavior Trees.

Experimental setup

To compare these two approaches, a simple prototype of a turn-based game was developed. The map of this game is presented in the form of a regular grid. A* algorithm was used for pathfinding. This is one of the most popular algorithms for finding the shortest path; it works quickly and can handle large maps. The example of game field is shown in Figure 1. The rules of the game prototype are simple:

- The blue cell is an AI, the goal is to win the black cell (the player).
- The game is turn-based, so the player and the AI can do 2 actions during each of their turns (attack and move).
- The player can move 4 squares, and the AI can only move 3 squares.

- The player has increased damage, but cannot use bonuses from the cells; this is necessary for the balance because the AI can win the player with good decisions and lose with bad.
- The green cells give the AI invulnerability to damage for 1 turn and disappear, then reappear elsewhere.
- The red cells give the AI double damage and just like green cells disappear and appear in another place.

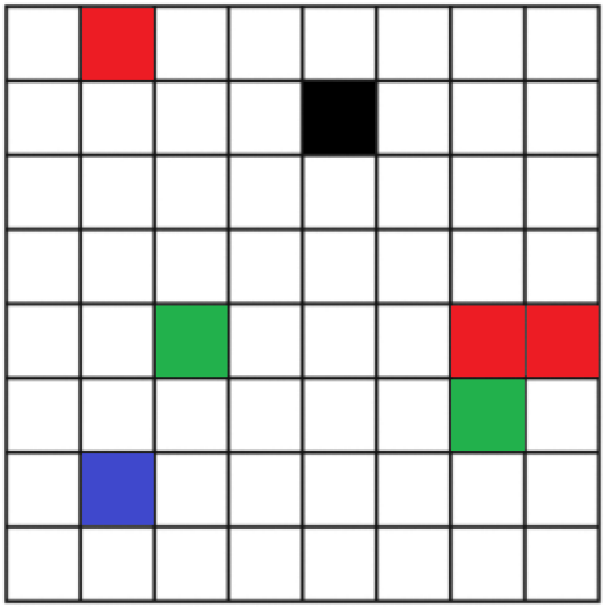

Figure 1: Game Prototype (Regular Grid).

First, let's try to describe the actions of the AI using the behavior tree. The AI must come to the green cell if the enemy is at an attacking distance and the green cell is nearby, the AI must come to the red cell if it is in range and it still has the strike action left. Priority is given to the red cells if the AI is ahead in health than the player and to the green cells in the opposite cases. The AI should give preference to the path where it can meet nearby (in the range of one step) green and red cells. The behavior tree is shown in Figure 2.

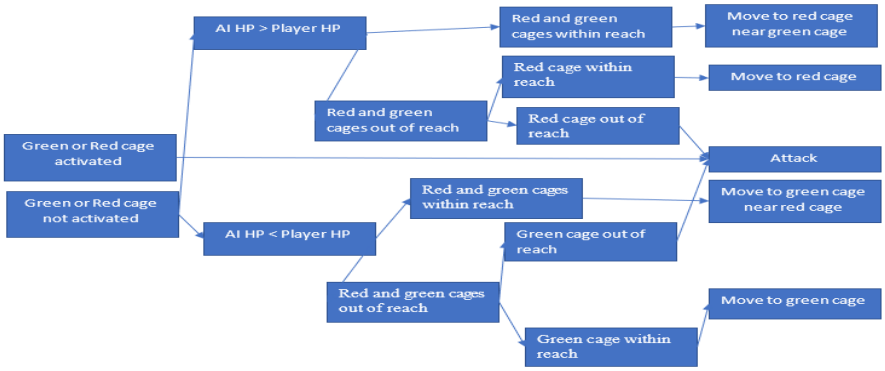

Figure 2: Behavior Tree for game prototype.

As we can see, there are already quite a few nodes in the tree, although now it does not look very confusing, adding new types of cells or actions to the game will significantly increase the size of the tree, because behavior trees grow at a tremendous speed.

Attack	
Green or Red cage activated	+1000
Move to red cage	
Red cage out of reach	-160
AI hp > Player HP	+100
Green cage out of reach	+50
Move to green cage	
Green cage out of reach	-160
AI hp < Player HP	+100
Red cage out of reach	+50
Move to green cage near red cage	
Green and red cages are nearby	+200
AI hp < Player HP	+50
Move to red cage near green cage	
Green and red cages are nearby	+200
AI hp > Player HP	+50

Figure 3: Utility AI.

Now we will implement the case logic using Utility AI. The result is shown in Figure 3. It is a table with good extensibility, to add another action or cell, we will only need to add new entries to the table without deleting the old ones and not destroying the connections, unlike the approach with the behavior tree, where we will have to rebuild the tree almost entirely.

In the tests, both approaches showed good results and were able to beat the player, but despite the same result in the tests, designing an AI based on the Utility is much easier and more scalable.

Conclusion

The FTM method is well suited for implementing simple artificial intelligence with a small number of actions. To implement a more complex AI, it is better to use a behavior tree or Utility AI.

However, behavior tree is difficult to expand, so it is replaced by a new method of implementing AI in video games – Utility AI. The Utility AI will perform the best in complex behavior, group behavior, strategic planning and in expandable AI.

References

1. Xiao Cui , Hao Shi. Direction Oriented Pathfinding In Video Games. [Digital resource]. — URL: https://www.researchgate.net/publication/267405818_Direction_Oriented_Pathfinding_In_Video_Games (Reference date: 21.10.2021).

2. Antonio A. Sánchez-Ruiz, Ruiz Stephen, Lee-Urban Héctor, M Noz-Avila. Game AI for a Turn-based Strategy Game with Plan Adaptation and Ontology-based retrieval. [Digital resource]. — URL: https://www.researchgate.net/publication/228752082_Game_AI_for_a_Turn-based_Strategy_Game_with_Plan_Adaptation_and_Ontology-based_retrieval (Reference date: 21.10.2021).

3. Boming Xia, Xiaozhen Ye, Adnan O.M Abuassba. Recent Research on AI in Games. [Digital resource]. — URL: https://www.researchgate.net/publication/343244745_Recent_Research_on_AI_in_Games. (Reference date: 21.10.2021).

4. Marek Kopel, Tomasz Hajas. Implementing AI for Non-player Characters in 3D Video Games. [Digital resource]. — URL: https://www.researchgate.net/publication/323162058_Implementing_AI_for_Non-player_Characters_in_3D_Video_Games (Reference date: 21.10.2021).

5. Firas Safadi, Raphael Fonteneau, Damien Ernst. Artificial Intelligence in Video Games: Towards a Unified Framework. [Digital resource]. — URL:

https://www.researchgate.net/publication/273894693_Artificial_Intelligence_in_Video_Games_Towards_a_Unified_Framework (Reference date: 21.10.2021).

6. Ashwin Ram, Santiago Ontanon, Manish Mehta. Artificial Intelligence for Adaptive Computer Games. [Digital resource]. — URL: https://www.researchgate.net/publication/221439041_Artificial_Intelligence_for_Adaptive_Computer_Games (Reference date: 21.10.2021).

7. M Ranjitha, Kazaka Nathan, Lincy Joseph. Artificial Intelligence Algorithms and Techniques in the computation of Player-Adaptive Games. [Digital resource]. — URL: https://www.researchgate.net/publication/338353404_Artificial_Intelligence_Algorithms_and_Techniques_in_the_computation_of_Player-Adaptive_Games (Reference date: 21.10.2021).

8. Niels Justesen , Philip Bontrager , Julian Togelius , Sebastian Risi. Deep Learning for Video Game Playing. [Digital resource]. — URL: https://www.researchgate.net/publication/319327551_Deep_Learning_for_Video_Game_Playing (Reference date: 21.10.2021).

Информация об авторах

Иванов Сергей Сергеевич – студент студент кафедры Информационные технологии и автоматик Институт радиоэлектроники и информационных технологий Уральского федерального университета (Екатеринбург, Россия). E-mail: aeabr01@gmail.com.

Пономарева Елена Владиславовна - старший преподаватель кафедры иностранных языков и перевода Уральского гуманитарного института Уральского федерального университета (Екатеринбург, Россия). E-mail: ev.ponomareva@urfu.ru.