PAPER • OPEN ACCESS

AI intelligent wayfinding based on Unreal Engine 4 static map

To cite this article: Yuhang Xie *et al* 2022 *J. Phys.: Conf. Ser.* **2253** 012016

AI intelligent wayfinding based on Unreal Engine 4 static map

Yuhang Xie[1], Yihong Chen[2], Ziyi Wang[2] and Yangjun Ou[1]

[1]School of Electronic Information Engineering, China West Normal University, Nanchong, China

[2]Internet of Things Perception and Big Data Analysis Key Laboratory of Nanchong, China West Normal University, Nanchong, China

E-mail: cyhswpi@126.com

Abstract. The development of games is closely related to the development of game engines.Unreal Engine is a mainstream game development engine.Unreal Engine has a powerful lighting rendering function and blueprint programming system.Therefore, using Unreal Engine for game development will better meet the requirements of modern game users for game quality and game content.In games, the AI path-finding system is often the core of a game.However, the path-finding component of the unreal engine is usually only applies to three-dimensional game and is used for intelligent path-finding in dynamic environment. Howerve, the 2D game usually finds the way in static environment.At present, there are a lareg number of path-finding algorithms and theoretical studies based on static environment, but there is no simulation experiment and detailed explanation of intelligent path-finding in static environment based on the Unreal Engine platform.This article firstly explains the principle of path-finding algorithms that are several commonly used in static environment.Then it compares the performance of different algorithms, and selects the optimal solution among several schemes by summarizing the experimental results.Then based on the optimal path-finding algorithm, using blueprint programming to implement AI path-finding in the Unreal engine.Finally, this article sort out and summarize the overall work process and indicate direction for future optimization.

1. Introduction

The development of artificial intelligence promotes to the development of video games.Video games have five advantages: real simulation, definition of boundary, non-destructive exploration, god standard and fun.With the continuous optimization of the high fidelity of the game engine,the game can provide highly simulated training environment[1].Therefore, the game can be used as the carrier of artificial intelligence algorithm testing, and the effect of artificial intelligence algorithm can also be verified in the game. Excellent artificial intelligence technology can offer more realistic and enjoyable gaming experience to players.AI intelligent path-finding is the core component of the game, and it is also the focus of research on game artificial intelligence[2].The nature of path-finding algorithm is path search that is also called path planning, that is, finding a feasible path between two points.In the game, NPC (Non-Player Character), that is, AI character, realizes movement in the game map according to the planned path based on path-finding algorithm[3].The game has high requirements for real-time responsiveness of AI characters. Solving the path search problem effcetively can increase the authenticity and interest of the game.

The path-finding planning problem is usually divided into local path-finding and global path-finding [4].Global path-finding is suitable for static grid maps and it can gennerate optimal path on static grid maps. The path-finding points will not change and the performance cost is low. Static grid maps are often used in the development of two-dimensional game maps.However, local path-finding is suitable for dynamic path-finding, and the path-finding points in dynamic path-finding are constantly changing. Therefore, local path-finding is usually suitable for three-dimensional game maps.However, in the Unreal Engine, the Navmesh path-finding component commonly used by developers is a kind of dynamic path-finding, and the path-finding component is not suitable for two-dimensional game maps.Due to the unreal engine's premier rendering capabilities, powerful particle system and physical collision system, games developed by unreal engine will have a sense of realism and playability far beyond the effect brought by other game engines. Therefore, this article aims to study how to realize AI intelligent path-finding in two-dimensional games through the unreal engine platform, which will make developers have deeper understanding of pathfinding algorithms, and provide game developers with the operation steps and experimental results to achieve this function.

2. Grid map

2.1. Definition of grid map

The grid is a spatial representation of a two-dimensional map, which converts the map into a data structure model that can be recognized by the computer. The grid processing can quickly divide the map into several square grids of the same length and width. The green indicates the starting point. The red indicates the target point, and the gray indicates the obstacle. In the computer, the map will be converted into a two-dimensional array, 0 and 1 respectively representing the walkable area and the obstacle area[5](Figure 1).

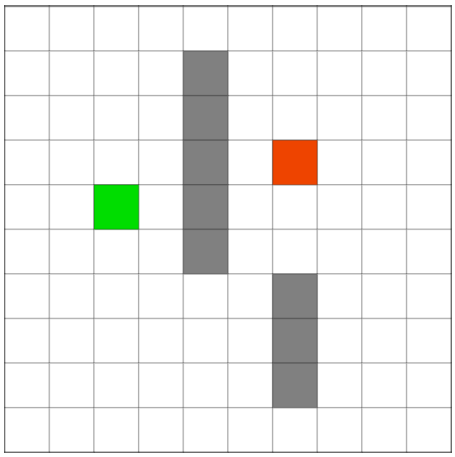

Figure 1. Figure 1.10*10 grid map.

2.2. Features of grid map

The advantage of a grid map is that it is simple to construct and easy to implement. By increasing or decreasing the number of grids or changing the size of the grids, the map can be modeled. If you want to add obstacles to the map, you only need to determine the range of the grid occupied by the obstacles, and change the attributes of the grid, that is, 0 and 1 in the two-dimensional array to complete the addition of obstacles.However, there are still some problems with grid maps. If the map is larger, more grids will be used, and the path-finding system will take up a lot of computational overhead, and the path-finding process will

not be smooth enough. However, since the map size and scale in two-dimensional games are usually fixed, developers only need to model according to the size and scale of the map, and select the appropriate path-finding algorithm for improvement, which can effectively alleviate the disadvantages of grid maps. Therefore, choosing a grid map for two-dimensional game development is still the most convenient and efficient method.

3. Basics of Pathfinding Algorithms

3.1. Common path-finding algorithms and their limitations

3.1.1. Breadth first search algorithm

Breadth-First-Search (BFS) is essentially a brute force search algorithm for blind search. This algorithm does not consider the location and direction of the end point, but traverses all nodes in the entire map to find the shortest path. Although breadth-first search can find the shortest path, it consumes a lot of running memory.

3.1.2. Dijkstra algorithm

The Dijkstra algorithm is essentially an improved algorithm based on the breadth first algorithm. By adding a weight to the edge between each vertex, it add the shortest edge currently to the shortest path plan and use the greedy algorithm to calculate and finally get the optimal path. It can avoid long routes and unwalkable areas, which is smarter than the breadth-first algorithm. However, the algorithm still has a tendency to search blindly, and the path-finding process still has no directionality.

3.2. A-star algorithm

Both of these two algorithms have a certain degree of blindness. Although they can find the shortest path, they will consume a lot of computation. A-star algorithm is a heuristic search algorithm[6]. Heuristic search is also called information search. Heuristic search guides the search by using the heuristic information based on the problem to reduce the scope of the search and reduce the complexity[7]. In the A-star algorithm, the heuristic information is represented by the evaluation function F(x):

$$F(x) = G(x) + H(x) \tag{1}$$

In the function, x represents the current evaluation point and G(x) represents the actual cost from the initial node to node x. H(x) is the heuristic function of the formula, and H(x) represents the estimated cost of optimal path from the current node x to the target node. H(x) must meet the following conditions to find the optimal solution:

$$H(x) \leq H^*(x) \tag{2}$$

H*(x) represents the cost of the optimal path from x to the target point, and H(x) is the lower bound of H*(x). When this condition is satisfied for any node, this path can be found as long as the optimal path exists. In grid maps, the heuristic function H(x) usually uses Manhattan distance to esimate distancen. The Manhattan distance formula is:

$$Dis(x, y) = \sum_{k=1}^{n} |x_k - y_k| \tag{3}$$

Manhattan distance calculates the distance between two vectors, and Manhattan distance only considers horizontal and vertical movement, and does not consider diagonal movement. The advantage of this method is that the computer processes integers faster than floating-point numbers.And when the unit distance is an integer, the diagonal distance is usually floating-point numbers. Using Manhattan distance can improve algorithm performance.The heuristic function

H(x) is applied to the Manhattan distance .And set the current node as A and the target node as B.Then the optimal path formula from the current node A to the target node B is:

$$H = |A_x - B_x| + |A_y - B_y| \tag{4}$$

At this time, the H value calculates the sum of the number of squares in the vertical and horizontal directions between the current node and the target node. Therefore, the path-finding process of the A-star algorithm means that the distance from the starting point to the target node is equal to the sum of the distance from the initial node to the current node and the distance from the current node to the end point[8]. The A-star algorithm will add two lists in the path-finding process, and one list is an open list, which is used to record all the units that can be selected. The other is the close list, which is used to record all the units that are no longer considered, to prevent the algorithm from repeatedly traversing the area that has been considered, and to reduce performance overhead. Taking the grid map in Figure 1 as an example, an optimal path is obtained by traversing the cost value of each square. As shown in Figure 2, the nodes passed by the yellow path constitute the optimal path, and the gray area is an obstacle area.

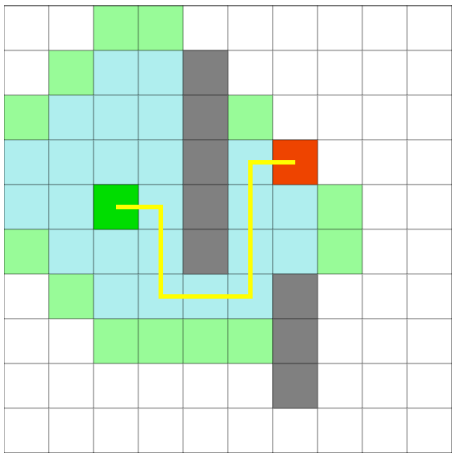

Figure 2. A-star algorithm realizes path-finding on grid map.

According to the principle of the algorithm, the pseudo code implemented by the A-star algorithm is as follows:

Algorithm 1 A-star Search

Input G, H, F
OpenList = [start point];
CloseList = [];
Start.G=0;
Start.F=Start.G+H(start,goal);

```
while Find the one with the smallest F value without being empty in OpenList do
    Get a node A in OpenList and remove it from OpenList
    if A is the target node then
        get and return the path
    for Traverses the child node B of each A do
        if B is not in the OpenList or in the CloseList then
            Find out the G value of B and calculate the F and H values of B
            Join the B node to the open list
        else if B is in the OpenList then
            if The F value of B is less than the F value of the OpenList then
                Update the F value of the OpenList
        else
            if The F value of B is less than the F value of the CloseList then
                Update the F value in the CloseList
                Remove the node from the CloseList and place it in the OpenList
        Node B is placed in the CloseList
        Sort by F value in OpenList
```

4. Realization of simulation experiment based on Unreal engine

4.1. Introduction to Unreal Engine

Unreal Engine 4 is a game engine that is developed based on C++. The engine has a powerful rendering function and blueprint programming system[9]. By using the real-time ray tracing function in UE4, the rendering time cost of the game environment is reduced[10], so using unreal engine4 to develop will make the game more realistic and playable[11]. Unreal Engine provides two different ways to develop games, and they are blueprint and C++. The two programming methods can be used interchangeably to develop projects. The blueprint system can create different executable commands in a modular form and compile them into virtual machine bytecode. Then use the mouse to arrange and link the modules to achieve programming[12]. With the improvement of modularity and visual programming, blueprint has also become the first development tool of the unreal engine[13].

4.2. Construction of grid map environment

Firstly a grid map is created in the unreal engine. A new AI character is created through blueprint, and it will be placed anywhere on the grid map.where the AI is located is the starting point of the path finding. A target point is setted, and the Arrow component in Unreal Engine is attached to the target point and it will be placed anywhere on the grid map without overlapping the starting point. The get actor location in blueprint is used to obtain the AI starting point and target point coordinates respectively . Then three boolean variables are setted in blueprint, and they are block, issinged and iscolored. Isblock is used to recognize obstacles in the map. Issinged is used to store data detected by grid, and iscolored is used to detect and calculate the optimal path. Then click on a single grid in the map and use isblock to determine each

grid. When there is an obstacle in the grid, it will be marked in red. Finally, the angle of view of the camera component in the engine will be adjusted to perform simulation rendering at an appropriate angle. The effect of consruction is shown in Figure 3.

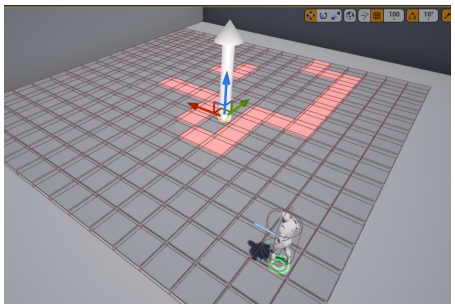

Figure 3. Map building effect in Unreal Engine.

4.3. Detection function of Manhattan method

In the grid map, the Manhattan method can effectively save the computer performance overhead and improve the running performance of the game. The Manhattan method needs to detect the neighboring nodes of the current node in four directions, that is ,up, down, left, and right. Also in the grid map built by Unreal Engine, using the Manhattan method needs to detect the front, rear, left, and right nodes of the AI character's location. Firstly, the current node position of the AI character is gotton. Then a cube collision box is attached to the AI character, and it will be rotated 45°. The size of the cube collision box will be adjusted, so that the boundary of the cube detection box is between the top, bottom, left, and right squares. Then using blueprint to program, and loop and disperse the results detected by each node (Figure 4).

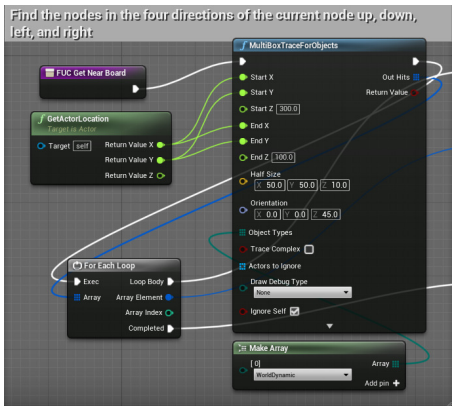

Figure 4. Blueprint code for cube collision detection.

4.4. Using blueprint to write A-star algorithm

The principle and structure of the algorithm are learned through the pseudo-code of A-star, and using blueprint to program the algorithm. Firstly, the SET module and the Target module are created. And then, the G value and H value will be calculated.Finally,the F value will be calculated based on the G value(Figure 5).The following is the blueprint code of the A-star algorithm:

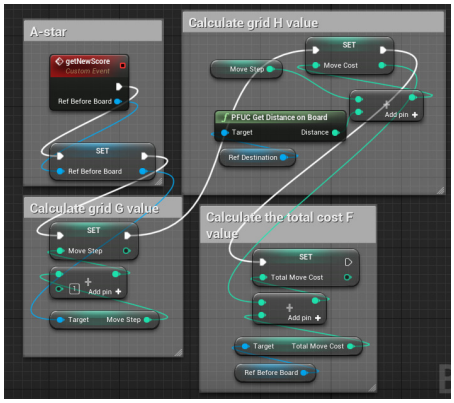

Figure 5. Blueprint code of the A-star algorithm.

4.5. Experimental test and result analysis

Run the above code and click compile to compile. If the blueprint is wrong, there will be a prompt that it can not run. One method is to use debug to detect the process problem of blueprint module. If the operation is successful, the system will traverse the F value of the node, and sort and filter the nodes. The green path is the path found by the algorithm. The result of the operation is shown in Figure 6. Through the analysis of the experimental results in Figure 6, A-star path-finding can accurately plan the path from start point to end point, and can avoid obstacle grids. The change of the start point and the end point does not affect the accuracy of the algorithm. However, there is still room for optimization in this experiment, and the Manhattan method has limitations in its direction. This limitation will cause the motion state of the AI character to be unrealistic. Part of the path can reach a shorter path through the diagonal distance, so this experiment can only achieve a relatively optimal path.

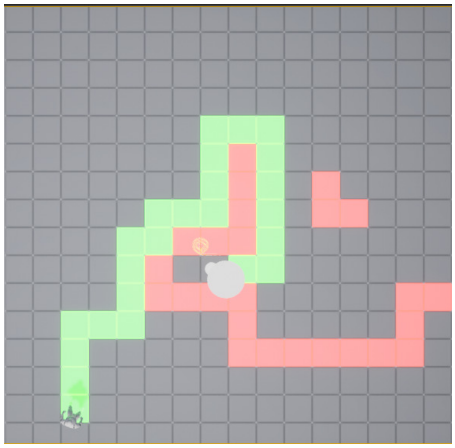

Figure 6. Display of running results in Unreal Engine.

5. Summary and outlook

A-star pathfinding is an efficient path-finding algorithm. A reasonable use of the A-star algorithm in a two-dimensional game can achieve a good path-finding effect. This article firstly elaborates on the static grid map, and the experimental simulation and simulation are carried out in the unreal engine through the research on the principle of the A-star algorithm.And the

experimental results of the operation are obtained. This simulation experiment can be used in a number of two-dimensional games, such as RPG, real-time strategy games, role-playing games. It can also be used in urban road planning and navigation systems. In short, through continuous research and exploration of path-finding algorithms, more optimization and applications of path-finding algorithms will surely be discovered.

References

[1] Ezzeddine A , Soto B . Connecting teams in modular construction projects using game engine technology[J]. Automation in Construction, 2021, 132:103887.
[2] Huang H . Intelligent Pathfinding Algorithm in Web Games[M]. 2020.
[3] Zeng Y , Xu K , Qin L , et al. A Semi-Markov Decision Model with Inverse Reinforcement Learning for Recognizing the Destination of a Maneuvering Agent in Real Time Strategy Games[J]. IEEE Access, 2020, PP(99):1-1.
[4] Zhang C , Zhang K , Zhang Y Q , et al. An Algorithm Applying to NPC Path Planning of Web Games[J]. Applied Mechanics & Materials, 2014, 556-562:3420-3423.
[5] Yang B , Ding Z , Yuan L , et al. A Novel Urban Emergency Path Planning Method Based on Vector Grid Map[J]. IEEE Access, 2020, PP(99):1-1.
[6] Sedighi S , Nguyen D V , Kuhnert K D . Guided Hybrid A-star Path Planning Algorithm for Valet Parking Applications[C]// 2019 5th International Conference on Control, Automation and Robotics (ICCAR). IEEE, 2019.
[7] Jie H , Wang G W , Yu X . A pathfinding algorithm in real-time strategy game based on Unity3D[C]// International Conference on Audio. IEEE, 2012.
[8] Cui X , Shi H . A*-based Pathfinding in Modern Computer Games[J]. International Journal of Computer ence & Network Security, 2011, 11(1):p.125-130.
[9] M Müller, Casser V , Lahoud J , et al. Sim4CV: A Photo-Realistic Simulator for Computer Vision Applications[J]. International Journal of Computer Vision, 2017(8).
[10] Martinez-Gonzalez P , Oprea S , Garcia-Garcia A , et al. UnrealROX: an extremely photorealistic virtual reality environment for robotics simulations and synthetic data.
[11] Qiu W , Yuille A . UnrealCV: Connecting Computer Vision to Unreal Engine[M]. Springer, Cham, 2016.
[12] Boyd R . Implementing Reinforcement Learning in Unreal Engine 4 with Blueprint. 2017.
[13] Ec A , Lz B . Exploring alternatives with Unreal Engine's Blueprints Visual Scripting System[J]. Entertainment Computing, 2020, 36.