

# Strategy optimization for breakout games based on improved BFS algorithm and image recognition techniques

Zekai Li<sup>1</sup>, Chang Lu<sup>2</sup>, Andi Tong<sup>3,4</sup>

<sup>1</sup>Engineering, University of New South Wales, Sydney, 2032, Australia

<sup>2</sup>Brunel London school, north China University of Technology, Beijing, 100144, China

<sup>3</sup>College of Electronics and Information Engineering, Shenzhen University, Shenzhen, 518040, China

<sup>4</sup>2020282067@email.szu.edu.cn

**Abstract.** This paper develops a Dungeon Game and details the development process of the City Game. The Dungeon Game is an adventure-themed role-playing game in which the player takes on the role of a character he or she creates and engages in a variety of adventurous activities in a set fictional world. It also allows the use of various props to add interest and realism to the game. In the development of this game, the pathfinding algorithm is crucial to the character's movement and behavior, and the previous traditional pathfinding method can no longer meet the needs of the game, so we optimize the original algorithm, adding dynamic programming and backtracking method in the original algorithm to improve the original algorithm's pathfinding logic to make the character pathfinding in the game more reasonable, to improve the game's performance and user experience. performance and user experience. At the same time, this paper tries to incorporate an image recognition technique based on ANN neural network into the game by adding figures as images into the game, to enrich the content of the game and make it have more possibilities, which increases the playability for the players. In the path optimization test, although the response time is slightly inaccurate, the overall response time is gradually decreasing as the optimization algorithms improve. These algorithm improvements have a more significant increase in the efficiency of the game.

**Keywords:** Strategy Optimization, Breakout Games, Image Recognition.

## 1. Introduction

AI games are games that utilize artificial intelligence technology to simulate or enhance characters, environments, rules, or other elements of a game [1]. The history of AI games can be traced back to the birth of video games, such as the earliest oscilloscope tennis and tic-tac-toe. With the development of computing power and algorithms, AI games have gradually covered more genres and domains, such as shooting, strategy, adventure, and chess. And Pixel Breakout Game is one of the classics, which adopts pixel style and has a simple but impressive game type. Pixel breakout games have a certain demand and audience in both domestic and international markets, especially in Europe and the United States, where many players have a strong interest and affection for this retro and innovative style. Some excellent pixel breakout games can also get good sales and reviews, such as Dead Cells and Octagon Traveler. As

game engines and development tools continue to advance, the barriers to making pixel adventure games are lowering, and many indie developers and small studios can utilize the resources and platforms available to them to develop and publish their works. Some pixel adventure games have also been able to utilize advanced technology to enhance their graphics and gameplay experience, such as Terraria and Miniature Mage. Most of these games are combined with cross-board levelling gameplay, which has a strong retro flavor and makes them accessible to more players. At the same time, the horizontal breakout level is less difficult to get started, so players can easily adapt to the pixel breakout type of handheld game. In this kind of game, players need to control a character to move in the grid, avoid obstacles and enemies, and reach the target position. This kind of game usually has high difficulty and challenge and requires players to have good reaction ability and strategic thinking.

The BFS algorithm is a breadth-first search algorithm, which can be used to solve many problems in graph theory, such as hierarchical traversal, shortest paths, bipartite graph determination, and so on. In operating systems, the BFS algorithm can be used to implement a scheduler for desktop Linux; in data mining, the BFS algorithm can be used to implement a frequent pattern mining method based on subgraph matching. Since the dungeon is a 2D network as the background of the map, the algorithm can effectively realize the NPC pathfinding, and its efficiency is higher than other algorithms in this game.

And this initial pathfinding method has the following disadvantages:

1. The BFS algorithm visits all possible paths, even if some of them are obviously infeasible or inefficient. This causes the BFS algorithm to waste a lot of time and space resources, reducing pathfinding efficiency and performance.

2. The BFS algorithm is limited in its execution when processing images due to the irregularity in the number of nodes in the small part of the graph as well as the number of nodes in the layer [2].

3. The BFS algorithm can only find the shortest path, but it cannot consider the safety and advantages and disadvantages of the path, which is reflected in the game that some paths, although short, may encounter more obstacles and enemies, increasing the risk of failure.

To solve these problems to be able to obtain a more excellent game performance, and to save the game computing time and the efficiency of the game operation [3].

We have created a pixel puzzle game dungeon game, we have added two new features, dynamic programming, and backtracking, to the original algorithm to improve the efficiency and performance of pathfinding, so that the improvement can take into account the safety and advantages and disadvantages of paths, to make a better decision in the pixel puzzle game [4].

## **2. Game Design**

Game development is divided into three stages: framework architecture, game content implementation, game testing, and enhancing game integrity. Game content primarily includes objects, battles, and objectives. The close integration of these elements creates a diverse and engaging game with clear goals.

### *2.1. Game Framework*

The game framework consists of four modules: Dungeon Main, Entity, Goal, and Battle. Dungeon Main serves as the fundamental class for the game, encompassing game startup, saving, image importing, and in-game moments. Entity, Goal, and Battle are parent classes representing objects, objectives, and battles. Together, they form an integral part of Dungeon Main. Additionally, there are subclasses such as Moving Entity, battle ground, and Static Entity, which inherit from and override the methods of their parent classes, thereby enriching the game with more detailed content.

### *2.2. Game content*

The game's content is divided into three parts: objects, battles, and objectives.



From a technical standpoint, each element within the game is represented by a distinct class. These classes encapsulate parameters, methods, and unique functionalities relevant to each game element. Additionally, various images are associated with these classes through their specific parameters, allowing them to be displayed within the game. Objects constitute the vast majority within these three

categories, further classified as movable and immovable objects. Immobile objects are further divided into four subcategories: pickable, interactive, non-pickable, and non-interactive (Table 1).

**Table 1.** The designed game content.

Moving Entity	Figure	Introduction
Mercenary		In this game, the Mercenary is a unique movable object that employs the Strategy Pattern [5], resulting in two distinct movement strategies. Before being bribed, the Mercenary is in an antagonistic state toward the player. It will opt to find the shortest path to the player's location. The player has the option to spend a gold coin to bribe the Mercenary. After this, the Mercenary transitions to a friendly state and starts following the player's movements (tracing the player's path).
Player		the Player is the most distinctive entity. This object primarily moves through manual manipulation. The player can choose to use the W, A, S, and D keys on the keyboard to control the movement direction of the Player.
Spider		Spiders will randomly spawn within a 9x9 area around the player. Once spawned, they will choose to move clockwise around their spawn point. Any object other than a Boulder cannot obstruct the spider's movement. If a spider comes into contact with a Boulder, its circling direction will be altered.
Zombie		Zombies will spawn randomly around Zombie Toast (a spawn cage for zombies), and their movement direction is also randomized.
Static Entity	Figure	Introduction
Bomb		The bomb can be picked up by the player and placed anywhere. If placed close to an active Step Tile, they will destroy all nearby objects.
Key		Keys can only be picked up by the player, and there is an inherent function that matches keys with doors. In other words, a specific key can only open the corresponding door.
Gold		Gold can only be collected by the player. When the player's inventory contains gold and they are near a mercenary, they have the option to hire the mercenary.
Wood		Wood can only be collected by the player and can be used to craft bows or shields.
Arrow		Arrows can only be collected by the player and can be used to craft bows or shields.
Potion		Potions can only be collected by the player and can be used at any time to restore the player's health.
Sword		Swords can only be collected by the player, and once picked up, they will increase the player's attack power.

**Table 1.** (continued).

Step Tile		Step Tiles are activated when there is a stone placed on them.
Number One		Numbers ranging from 0 to 9 are randomly generated within the dungeon for image recognition purposes.

### 2.3. Route optimization

In the dungeon game, we utilize a 2D grid map, and as a result, we have chosen the Breadth-First Search (BFS) algorithm. For this type of map, the BFS algorithm proves to be highly effective [6]. Furthermore, we adapt the algorithm based on the game content to enhance accuracy and computational efficiency.

Improving algorithm efficiency through priority queues: Using the original BFS approach alone, employing brute force search without a clear goal could significantly increase the algorithm's execution time, leading to a decrease in player experience. By incorporating a priority queue, the algorithm prioritizes computing distances to grids closer to the player, thereby enhancing the efficiency of pathfinding to the target grid and subsequently the overall algorithm efficiency.

In addition, the algorithm employs a bidirectional search strategy, further enhancing its efficiency. It divides the search into two parts and combines them to achieve optimization. Specifically, two simultaneous BFS searches are conducted: one from the player towards the starting point, and the other from the starting point towards the player. When the same point (midpoint) appears in both queues, the algorithm returns the path, thus achieving efficiency gains through this approach.

### 2.4. Number Recognition

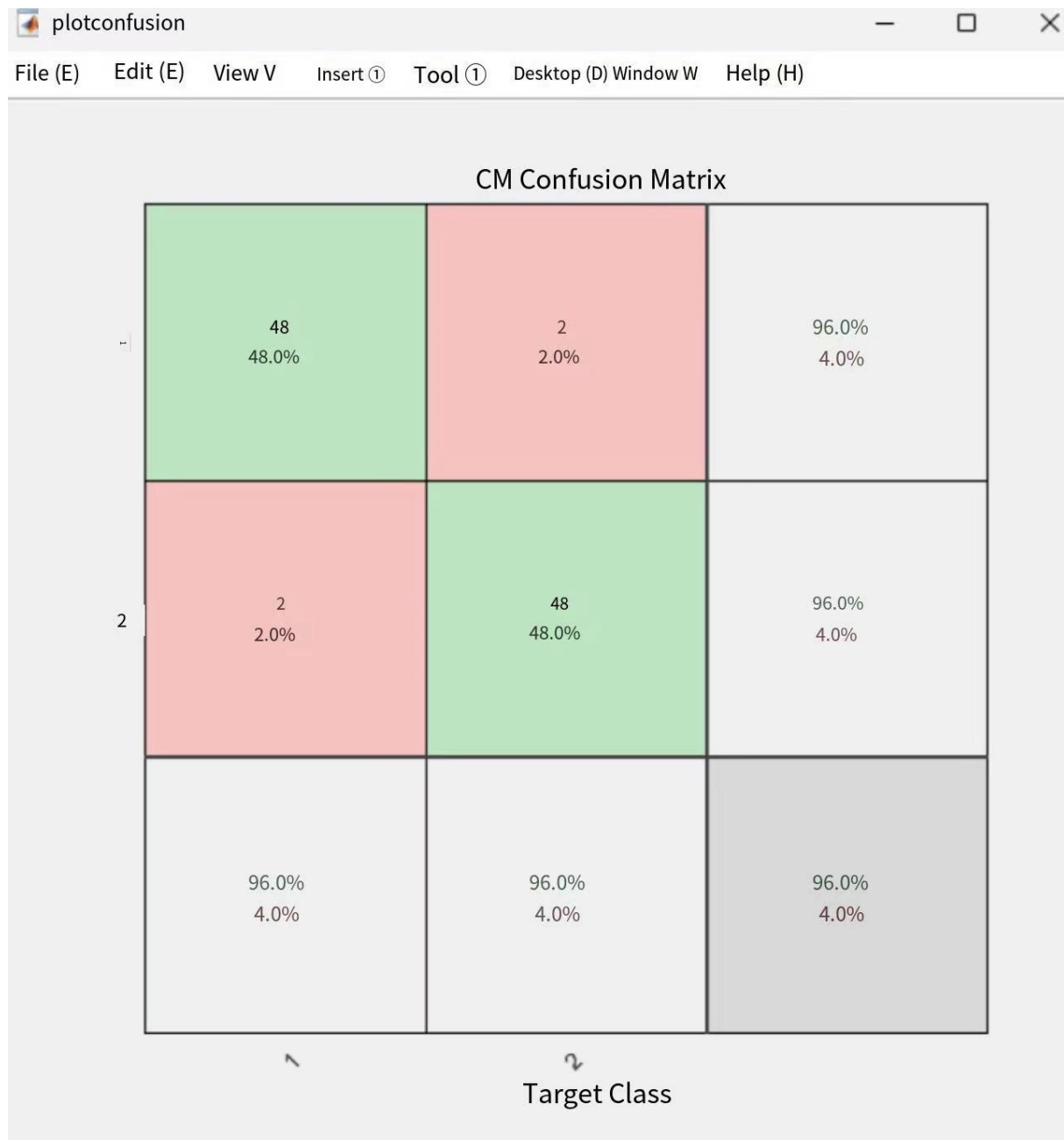
To enhance the gameplay and fun factor of the game, as well as increase the sense of accomplishment upon completion, the authors incorporated an image recognition component into the game [7]. By touching the numbers present on the game map, successful recognition triggers changes in game mechanics and difficulty levels, including boosting monster health and attack power, thus elevating the game's playability.

In this study, the Artificial Neural Network (ANN) algorithm was employed to achieve number recognition within the game [8]. ANN was chosen due to its self-learning capabilities. During training, diverse image patterns are fed into the artificial neural network, which gradually learns to recognize such images through its self-learning functionality. Moreover, this algorithm offers high efficiency, leveraging computer processing power for analysis. MATLAB, a scientific computing language, was used for programming in this project. The dataset comprises 1000 training examples of hand-written digits in a 784x1000 format. After thorough testing and adjustments, the selected parameters include 10 training epochs, a learning rate of 0.14,  $Bp = 1$ , and  $cf = 1$ . The activation function utilized at this point is sigmoid. When  $cf = 2$ , the learning rate shifts to 0.014 to ensure sufficient convergence during training, with the activation function being switched to softmax.

To achieve the highest possible recognition accuracy, a comparison of different parameters was conducted for the three-layer neural network. The aim was to select the configuration with the highest accuracy. The three-layer neural network comprises an input layer with 784 nodes, a hidden layer with 100 nodes, and an output layer with two nodes.

In this context, for the three-layer network with  $bp = 1$  and  $cf = 1$ , the recognized digits are 0, 1, 3, 5, and 7. Among the results, the leftmost number in the third row (96%) represents the network's successful recognition rate for 500 sets of images containing the digits 0, 1, 3, 5, and 7, known as sensitivity. Additionally, there are 500 sets of 784x1000 hand-written images containing the digits 2, 4, 6, 8, and 9. The network's recognition rate for this set is 96%, termed specificity (Figure 1). The gray-shaded data on the right depict the model's recognition success rate for 100 sets of 784x1000 hand-written digits. Different parameters have a significant impact on the experimental results [9]. Therefore, a three-layer neural network with 96% recognition accuracy will be selected as the final version for

integration into the game. This level of accuracy ensures successful recognition in most scenarios, and any failure to recognize would only result in the inability to alter game difficulty. In subsequent maps, at least two numbers will be placed to ensure successful digit recognition in every game.



**Figure 1.** At this point it is a three-layer network, bp=1, cf=1, and the recognized number is 01357.

### 3. Gaming Tests

To ensure the accuracy and integrity of the game, we conducted black-box testing [9]. Initially, we subjected the refined algorithm to testing by examining the routes of mercenaries, spiders, and zombies, assessing the correctness of game content. We chose a map with a side length of 20 units, randomly generated mercenaries, and ran the game automatically. We compared the resulting game path with the expected path to determine the accuracy of pathfinding.

This process was aimed at validating the correctness of our improved algorithm, particularly in scenarios involving mercenaries, spiders, and zombies. By conducting these tests and verifying the alignment of actual game results with the expected outcomes, we could ascertain the accuracy and reliability of the pathfinding mechanism (Table 2).

**Table 2.** Game Path Optimization Test.

Components	Original BFS	BFS with priority queues	BFS with priority queues and bidirectional search
Length	40	40	40
Time1	0.85s	0.67s	0.46s
Time2	0.89s	0.72s	0.44s
Time3	1.08s	0.60s	0.54s
Time4	0.93s	0.82s	0.49s
Average Time	0.94s	0.70	0.48

The Figure 2 and table 3 clearly illustrates that with the integration of the improved algorithm, the average time consistently decreases as the optimization algorithm is applied. This substantial enhancement in pathfinding efficiency ensures that players can enjoy a significantly improved gaming experience, even within larger game maps.



**Figure 2.** Game Content Showcase.

**Table 3.** The assessment result (Survey of 10 players, 0-10 points).

Audience	Overall rating	display	responsiveness	Game content
1	8	9	10	10
2	9	9	9	10
3	8	10	9	7
4	7	8	9	9
5	8	9	8	8
6	7	10	9	8
7	6	7	9	9
8	9	8	8	7
9	9	8	10	8
10	7	8	8	8
Average	7.8	8.6	8.9	8.4

#### 4. Conclusion

We designed a dungeon game as the base platform for our research. This dungeon game is a third-person adventure game, and the idea of the game is that the player directs the character in the game to win the game by moving, picking up items, synthesizing items, avoiding or killing monsters, and finally achieving the goal in the game. In the game, we added two new features, dynamic programming, and backtracking, to the original path algorithm to improve the efficiency and performance of pathfinding, which can take into account the safety and advantages and disadvantages of paths, to make better decisions in the pixel adventure game. After the optimization and improvement of the game, the gameplay was enhanced. After the improvement, we tested the game and we were able to find that after the optimization, the evaluation of the game shows that the players are more satisfied with the response speed of the game, and the overall playability of the game is higher and richer. In the path optimization test, although the response time is slightly inaccurate, the overall response time with the improvement of the optimization algorithm is gradually reduced, which can be seen that these algorithmic improvements on the efficiency of the game have a more significant increase. Most players in the questionnaire were satisfied with the response time and game content.

The advantage of this game is that after optimization, the interaction between NPCs and players is more powerful than in similar games, and the difficulty of the game can be adjusted through the combination of the game and image recognition, which can enhance the richness of the game. The shortcoming of this game is that the game is based on 2D pixel graphics, so its image quality and expression are weak, and it is difficult to attract players in terms of image and graphics. The future research direction for this game is to try to improve the pathfinding algorithm for NPCs and also to try to recognize the player's path through image recognition technology and change the difficulty of the game through the path, which can be more attractive to the players.

#### Authors Contribution

All the authors contributed equally and their names were listed in alphabetical order.

#### References

- [1] V. K. Akram, M. Asci and O. Dagdeviren, "Design and Analysis of a Breadth First Search based Connectivity Robustness Estimation Approach in Wireless Sensor Networks," 2018 6th *Inter. Conf. Control. Engin. Inform. Tech.*, Turkey, 2018, pp. 1-6.
- [2] H. Wen and W. Zhang, "Improving Parallelism of Breadth First Search (BFS) Algorithm for Accelerated Performance on GPUs," 2019 *IEEE High-Perfor. Extre. Comput. Conf.*, 2019, pp. 1-7.
- [3] N. J. A. Jesuthas and S. Somaskandan, "Path-finding and Planning in a 3D Environment An Analysis Using Bidirectional Versions of Dijkstra's, Weighted A\*, and Greedy Best First Search Algorithms," 2022 *2nd Asian Conf. Innov. Tech.*, 2022, pp. 1-8.
- [4] K. Ibrahim, "Optimizing Breadth-First Search at Scale Using Hardware-Accelerated Space Consistency," 2019 *IEEE High-Perfor. Extre. Comput. Conf.*, 2019, pp. 23-33.
- [5] D. Plankalkül and K. Zuse, Internet Archive, 1972, pp. 96-105.
- [6] S. D. H. Permana and K. B. Y. Bintoro and B. Arifitama and A. Syahputra, "Comparative Analysis of Pathfinding Algorithms A \*, Dijkstra, and BFS on Maze Runner Game", *Inter. J. Of Infor. Sys. Tech.* **1. 2**, 2018, pp.1-8.
- [7] Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research S Agatonovic-Kustrin, *Rose. Beresford J. pharm. Biomed. Anal.* **22 (5)**, 2000, pp. 717-727.
- [8] Massimo Buscema- A Brief Introduction to Artificial Neural Networks Artificial Adaptive Systems in Medicine, *New Theor. Model. New Appl.*, 2009, pp. 5.
- [9] C. P. Schultz, R. D. Bryant, "Game Test: All in one", 2016, *Inter. J. Of Infor. Sys. Tech* pp. 1-26.