# AI Conceptual Gaming for Game Developers

***Antoin Ashraf Fathi, Baher Wael Youssef, Boula Awny Azmy, John Nabil Takla, Karim Mamdouh Fawzy, Wessam El-Behaidy****

Faculty of Computers & Artificial Intelligence, Helwan University
*Corresponding Email: w_behaidy@fci.helwan.edu.eg

**Abstract**

Our game consists of games which are encoded by humans and decoded by computers. In some of them the computers try to make the best possible outcome to prove that they are advanced, they won't settle only for less than the most efficient way to solve a problem in with which they pose a great challenge with human minds. In Pathfinding; We used A* algorithm to make a horse find the shortest path to the village, Maze; Using DFS and Backtracking we could find a way out of the maze, Tic Tac Toe; We used Minimax to make the AI play versus the player, 8 Puzzle Game; We solved using BFS, 8 Queens; Solved using branch & bound algorithm, Encrypting Text; We used Caesar Cipher cryptography algorithm to encrypt text, Combat AI; Using NavMesh in Unity engine to make AI find path to player then attacking him using Finite State Machines.

## 1. Introduction

Our game consists of games which are encoded by humans and decoded by computers. In some of them the computers try to make the best possible outcome to prove that they are advanced, they won't settle only for less than the most efficient way to solve a problem in with which they pose a great challenge with human minds.

We need to develop our own AI skills to make huge improvements within using any kind of game engines whether it be Unreal, GODOT or Unity through a) Pathfinding, b) applied in a real open world games like GTA, Watchdogs & Far Cry, c) encryption as banks get robbed and we want to increase the level of challenging as in 10 minutes the police will come so we must have a way to decrypt the text to be able to finish the text as fast as possible before the police arrives, d)If we want to represent the best possible human capable to finish the hardest intelligent task within only a few seconds faster than humans, learn from experience, has passion to work better than ever which is the gate of reinforcement learning which is applied in most of today's storyline games.

Our goal is implementing some 3D models from scratch and implementing AI Algorithms from scratch. The software used in implementation is Blender, Unity, Visual Studio, Mono Develop and Git VCS.

## 2. Methodology and Results snapshot

We developed a storyline to link all of the problems we want to solve in a fun and interesting way.

Our game's storyline consists of several consecutive scenes, as follows:

### 2.1 Forest maze (A* pathfinding) scene:

This scene uses a horse model to demonstrate the usage of A* pathfinding algorithm in guiding the horse through a maze (Fig. 1) shortest path.

A* algorithm calculates two important formulas:

- How much it takes from the beginning to reach that goal node
- Estimates how far current node is from the goal node using Euclidean distance as a heuristic function for estimation

It collects all surrounding nodes and selects the one with the least cost to the goal and iterates over them till it forms a complete path from starting node to goal node with least cost.

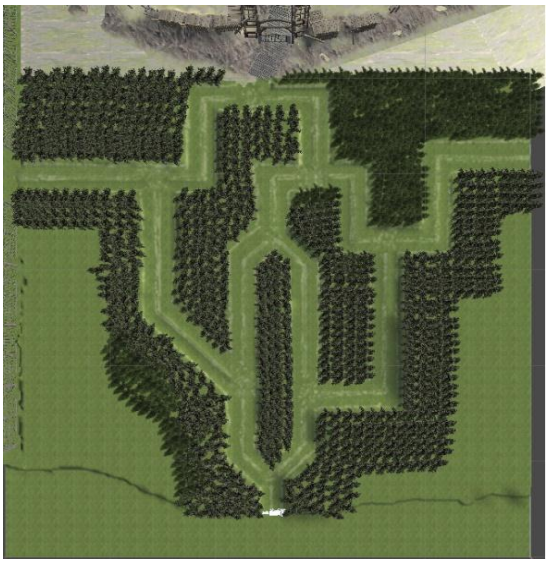

Fig. 1: Forest Maze

2.2 Eight Puzzle scene:

This scene uses a breadth first search to solve 8 puzzle game (Fig. 2).

Starting from start state we put all the possible moves in the open list and as a child for the previous state then take every possible move to see if it's a possible move too and do the same as start state but check if possible move has repeated in the closed list if so don't put it in the open list till we reach the goal state then we backtrack the solution till we find the path to the goal.


Fig.2: Eight Puzzle

2.3 Maze Game (Depth First Search & Backtracking)

In this scene we generate and go through a maze using Depth first search (DFS) as follows:

- We bag up the initial node

- The algorithm looks around for other walkable nodes to decide arbitrarily to where it should step in
- It goes deep with the selected node until it maybe reaches the goal, or it gets stuck at the end of the road
- It comes back with a bag of all explored nodes and begin to bag the last one to explore and the process gets repeated from the second point

Backtracking:
As the word implies:
We take each step from the previous algorithm and go through them from bottom to top as to enhance on the results we obtained, as in Fig.3.

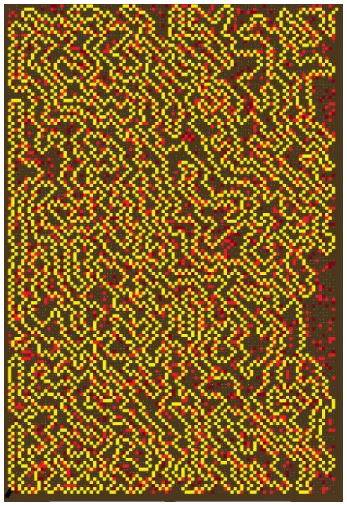

Fig. 3: Maze generated with DFS applied

2.4 Tic tac toe (Minimax algorithm)

- Minimax is a kind of backtracking algorithm that is used in decision making and game theory to find the optimal move for a player, assuming that your opponent also plays optimally, and it is widely used in two player turn-based games.
- In Minimax the two players are called maximizer and minimizer. The maximizer tries to get the highest score possible while the minimizer tries to do the opposite and get the lowest score possible.
- Every board state has a value associated with it. The values of the board are calculated by some heuristics which are unique for every type of game.

2.5 Eight Queens (Branch and Bound)

- As shown in Fig. 4, we place queens one by one in different columns, starting from the leftmost column.
- We check if queen can be placed in current row
- If the place is valid, we mark the column, row, and two diagonals as threatened
- If all queens are placed then a solution is found, we return true
- Else we proceed to the next queen/column (recursively)
- If no solution was found, we backtrack

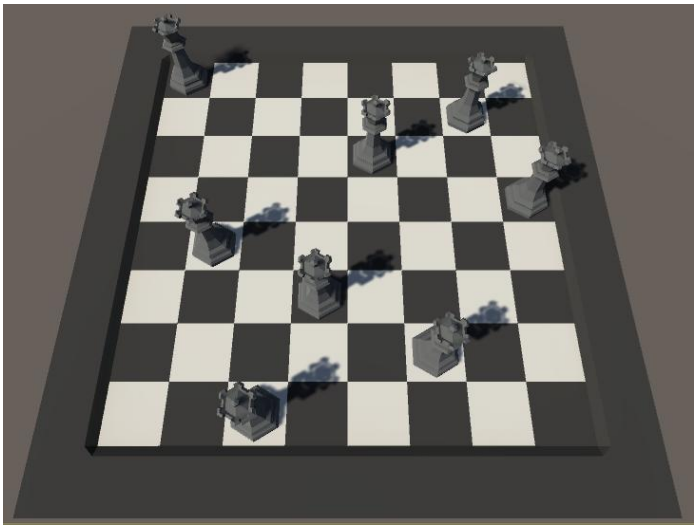

Fig. 4: 8 Queens Scene

## 2.6 Encrypted Sentence (Caesar Cipher)

The Caesar cipher is one of the known and simplest ciphers. It is a type of substitution cipher in which each character in a text is shifted a certain number of places down the alphabet. For example, with a shift "cipher" of 3 shown in Fig. 5, A would be replaced by D, B would become E, and so on.

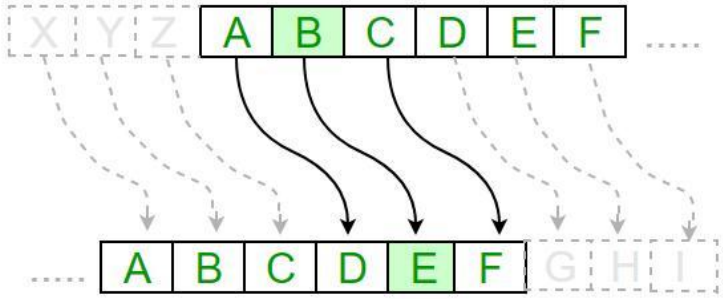


Fig. 5: Caesar Cipher with a shift "cipher" of 3


Fig 6: Combat Scene

2.7 Viking Village Combat Scene

- An Artificial agent is spawned every five seconds from a randomly chosen point of three predetermined spawn points (Fig. 6)
- The agent apply path finding techniques to find the shortest path to their goal (Player)
- Every character has a box collider (Fig. 7)
- An event "On Trigger Enter" is called when two colliders intersect
- We check if the enemy collider intersected with the collider of the player, and that the player is still alive
- We set the "Attack trigger" on the enemy (Fig. 8)
- The enemy attack animation is played (Fig. 9)
- Also play player hit animation by setting its trigger and decreasing its health

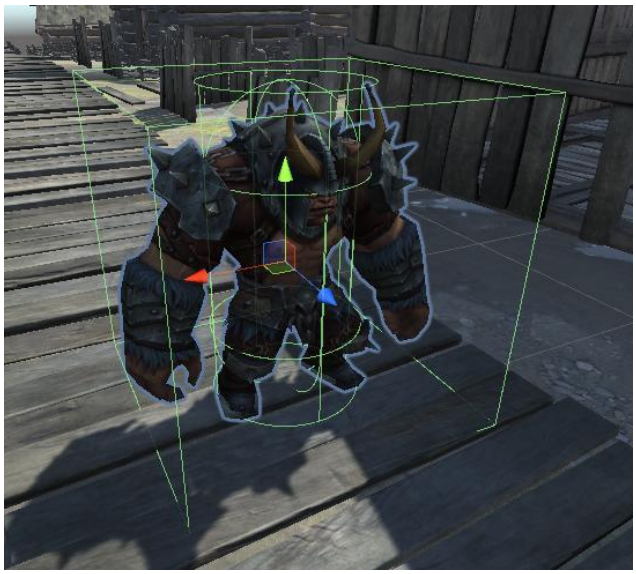

Fig. 7: Character colliders

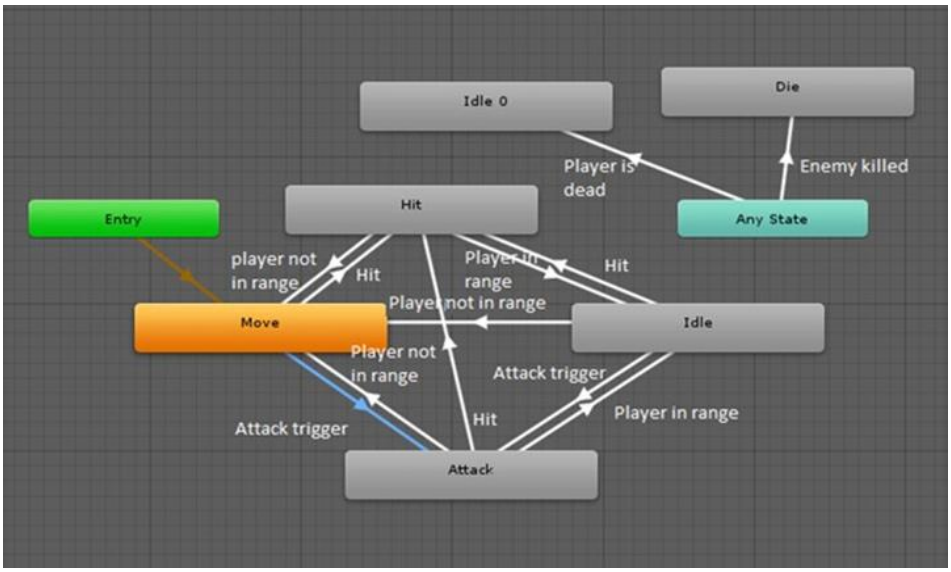


Fig. 8: Enemy finite state machine

- "Time Between Attacks" to set the frequency of enemy attacking player
- "Delta Time" to get the time taken to complete the last frame
- We add "Delta Time" to "counter"
- Check if the "counter" reached "Time between attacks"
- When and enemy is dead we sink its body into the ground
- UI score and player health are updated accordingly

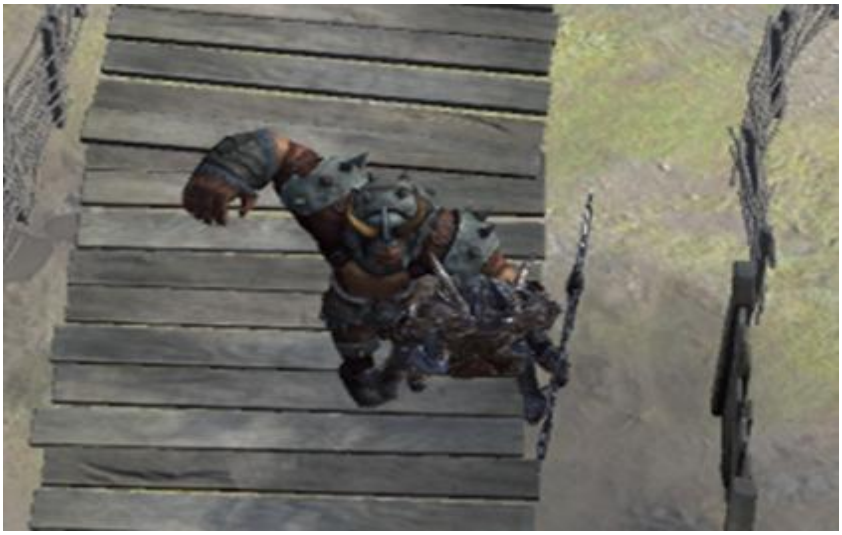

Fig. 9: Enemy attacking player

The full game map is shown in Fig. 10.

Fig. 10: Full game map

## 3. Conclusion

We succeeded to develop our own AI skills through game engines. We developed a storyline to link all of the problems we want to solve in a fun and interesting way. We implemented 3D models from scratch as the horse, enemy player, and the forest maze. In addition, we implemented AI Algorithms from scratch like DFS and backtracking, Tic-tac-toe, eight queens, and Caesar Cipher.

## References

[1] Ian Millington, "Artificial Intelligence for Games," Elsevier, 2006.

[2] AI Role Playing Game Development, http://lbms03.cityu.edu.hk/studproj/ee/2007eecyk362.pdf

[3] A*-based Pathfinding in Modern Computer Game, http://paper.ijcsns.org/07_book/201101/20110119.pdf

[4] Simulating Car Racing Game by Applying Pathfinding Algorithms, http://www.ijmlc.org/papers/82-A1090.pdf

[5] Pathfinding in Strategy Games and Maze Solving Using A* Search Algorithm, https://file.scirp.org/pdf/JCC_2016090815023801.pdf