International Journal of Engineering and Information Systems (IJEAIS)
ISSN: 2643-640X
Vol. 3 Issue 11, November – 2019, Pages: 1-6

Designing and Implementation of Tic-Tac-Toe-4X4 based Artificial Intelligence using Python Programming

[1]**Abubakarsidiq Makame Rajab,** [2]**Nadhira Haji Hassan and** [3]**Abdul-rahimsidiq Makame Rajab**
[1]Department of Electronic Information Communication, Huazhong University of Science and Technology Hongshan District, Wuhan 430074, P.R. China
[2] Al-Farouk Private Institute, Zanzibar-Tanzania
[3] Institute of Finance Management (IFM), Dar-es-Salam-Tanzania

[1]Email: mrgovery@hotmail.com [2]Email: nadhirah_01@outlook.com [3]Email: abdul.mzale@yahho.com

Abstract *— In Tic-Tac-Toe of the classic game is built on computer-based flat form using python as a part of case study project using Artificial Intelligent techniques. The primary goal for this project is to create a computer artificial intelligent based on Tic-Tac-Toe 4x4 game that show two players on who will win and who will lose the game accordingly, using the standard Minimax algorithm, it was adopted and modified as a subset of rules from best gameplay practices: (1) attempt to win, (2) endeavor to keep a misfortune, (3) make a key move, and (4) make an irregular move. To make the game more fun and more winnable at easier difficulty levels, probabilities are introduced that the computer would find a valuable move and ignore it. In the end, the computer artificial intelligent uses a simple, lightweight decision tree to choose its next move, and the gameplay is fast, balanced, and enjoyable. furthermore, in the winning strategy in a chess game by means of symbolic model checking, and demonstrate the winning strategy in tic-tac-toe game through the symbolic model checking tool improved with the verification algorithm for winning strategy.*

Keywords: Tic -Tac –Toe, Design, Python**,** AI –Artificial Intelligent and algorithms.

I. INTRODUCTION

Tic-Tac-Toe is a famous basic amusement for two players everywhere throughout the world. Customarily one individual plays for "X" while another plays for "O". The diversion is played on the square leading body of **4x4 Tic-Tac-Toe** or significantly more size**.** The point of this amusement for player is to win by loading with "X" or "O" any line, section or corner to corner of the principle diversion board [1, 2]. The diversion sets players figuring since players ought to mirror a tad on methodology and foresee a few moves of the rival. So it instructs individuals to settle on choices rapidly. **Python is computer** programming for specialized figuring from the programming Works [3]. Utilized for wide assortment of logical and building estimations, the point of the report is to depict how Tic-Tac-Toe diversion reenactment was made with the assistance of **python**. The report portrays how program peruses the code and methodology for winning in the amusement. The objective of this undertaking is to give a Python program that plays a session of 4x4 Tic -Tac Toe.

The Tic-tac-toes amusement rationale does not make presumptions about the structure of the board. This enables us to effectively build Tic Toe diversions with various board arrangements. The board is spoken to as a two-dimensional cluster. A wide range of assessment capacities are given. Additionally included is a capacity for checking if the diversion has been won. We were actualizing two-dimensional exhibit Tic Tac 4x4 amusement. This ever-well known amusement can be played against two individuals and at last the diversion will end telling which individual is the victor and which individual has lost. The multi-dimensional cluster is utilized to store the X's and O's that the players enter [4]. The single dimensional cluster is utilized to store the circumstances that either X or O has won. The other thing that it stores number of feline's amusement that has been played, to do this, we began by characterizing both clusters. After this we started to request that the two players enter the qualities for their coveted move. Each time esteem is entered by a client the program test to ensure that spot hasn't been already picked just to ensure that no bamboozling goes on. No test for a 4 out of a line is performed inside the initial four moves since it isn't conceivable to win in the initial four moves regardless of how indiscreet an adversary might be. After the initial four moves another test is added to the system. This new test is to test if the qualities entered by the client are connected together, four out of a column, corner to corner, on a level plane, or vertically [5].

A. Main Objective

The aim of this project is to develop a Tic-Tac-Toe 4 game. The game is supposed to consist of two parts, one a single player game ‘X’ and the other a player game ‘O’ (both are the group members or anyone who is willing to play a game) playing against each other. The goal of the diversion is to put four ‘X’s’ or ‘O’s’ pieces in succession, either vertically, evenly, or corner to corner. On the off chance that this is accomplished, the diversion closes in a win, generally the amusement closes in a draw once no more moves are conceivable.

B. Motivation

a. To develop Tic-Tac-Toe 4X4 game for two players can come together and play for fun using computerized interfaced as well as based automatic as AI computerized game.
b. To develop a game that help a user to think fast when a problem occurs as well as to think and communicate.

C. Contributions

This is critical on the grounds that there are new calculations building up each day, along these lines the requirement for ideal and less intricate calculations has turned out to be essential. Our commitments to the Tic-Tac-Toe issue can be condensed as takes after; **(1)** we considering diverse ways to deal with the issue. **(2)** Building up an ideal calculation to take care of the issue. **(3)** Improving the effectiveness of the calculation while keeping up polynomial time multifaceted nature. This examination might be useful in enhancing the many-sided quality and giving ideal answers for comparable issues.

D. The scope of the study

There will be a straightforward square amusement board separated into 16 tiles or lattice spaces. At the point when the player taps on one of the lattice spaces, it will be relegated either a "X" or an "O". The diversion is over when one player claims 4 matrix spaces in succession or there are no moves left. The amusement will have a little measure of clean to make it finish. Toward the beginning of the diversion, the board won't be dynamic until the point that the primary player has picked whether they are to play "X" or "O". A board will demonstrate whose turn it is. At the point when the diversion is more than, a pennant will show the champ or declare a draw if nobody wins. A restart catch will be shown when the amusement is finished, restoring the diversion to the beginning state when clicked.

II. BACKGROUNG AND RELATED WORK

In this section the Tic-Tac-Toe game will be discussed in details. At the outset, the basic rules of the game are going to be covered. Then, there will be an evaluation on existing Tic-Tac-Toe games, which in turn will lead to dialogue about the current fashions of this game and the proposed model of this work [6]. Finally, this part is going to be concluded with a science lookup concentrated towards python program. The Roman Empire is known to have mounted the beginnings of the earliest recognized variant of tic-tac-toe. It originated round the first century BC (Crowley, 1993).

The Roman Empire is known to have established the beginnings of the earliest known variant of tic-tac-toe. It originated around the first century BC (Crowley, 1993). At that time, the sport was once known as Terni Lapilli. Instead of having any number of pieces, every player solely had three. The game was performed via moving them round to empty areas to keep playing. However, in accordance to Claudia Zaslavsky's book, the game Tic-Tac Toe is originating from ancient Egypt (Zaslavsky, 1982).An early version of Tic-tac-toe used to be played in the Roman Empire, around the fundamental century BC. It was once called Terni Lapilli and as adverse to having any range of portions 5. Every player just had three, hence they needed to move them around to discharge spaces to continue playing [1]. The amusement's framework markings have been discovered chalked all finished Rome. Be that as it may, as per Claudia Zaslavsky's book Tic-Tac Toe: And Other Three-In-A Row Games from Ancient Egypt to the Modern Computer Tic-Tac-Toe could begin back to antiquated Egypt. In 1864: The principal print reference to "nougats and crosses", the British name. In 1884: The primary print reference to a diversion called "tick-tack-toe" happened, however alluded to "a kids' amusement played on a slate, comprising in attempting with the eyes close to bring the pencil down on one of the quantities of a set, the number hit being scored" [2]. "Tic-tac-toe" may likewise get from "tick-tack", the name of an old variant of backgammon initially portrayed in 1558. The U.S. renaming of noughts and crosses as tic-tac-toe happened in the twentieth century.

Tic-Tac-Toe was additionally utilized by understudies to exhibit the computational energy of Tinker toy components. The Tinker toy PC, made out of (nearly) just Tinker toys, can play Tic-Tac-Toe flawlessly. It is right now in plain view at the Museums. Chess and Tic-Tac-Toe are one of the most famous games to which the moves are not left to chances, rather than pure mathematics and logical reasoning. In these games, a player wins by achieving a winning configuration first, like for instance: checkmate in chess, and 3-in-a-row in a basic Tic-Tac-Toe game in 3x3 boards[6].

In order to find a winning strategy, in theory all the paths could be explored. However, in practice this is not easy because the total number of strategies can be calculated a double exponential function of the size of the board. This is because each cells has 3 options; Marked by the first player, Marked by the second player, or Unmarked.

III. DESIGNING

A. Logical Designing

The program is simple to utilize and instinctive user interface, yet gives an abundance of highlights. There are a few levels of playing quality (the most noteworthy playing consummately), fix history, play against a human for either side, or self-play with selectable quality for each side. It additionally contains an arrangement of troublesome issue positions to engage the client intrigued by perplexes.

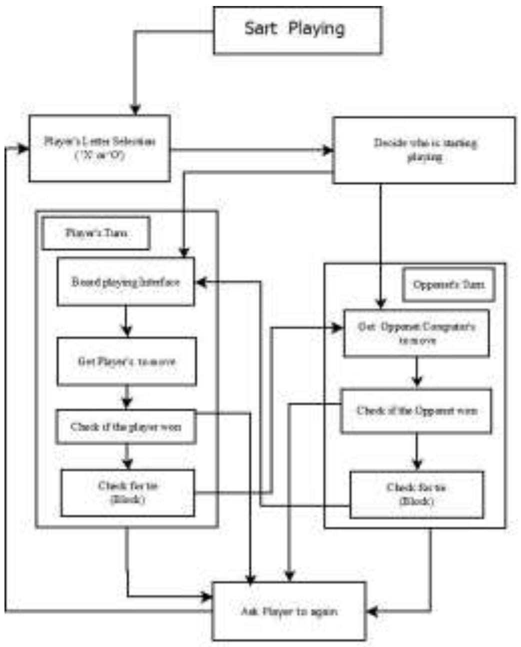


Figure 1: Flowchart for Tic-Tac-Toe

B. Board State Representation

Based on flowchart show what happens during the player's turn, and the ones on the right side show what happens during the computer's turn. After the player or computer makes a move, the program checks whether they won or caused a tie, and then the game switches turns. After the game is over, the program asks the player if they want to play again.

The Tic-Tac-Toe board is drawn as a pair of horizontal lines and a pair of vertical lines, with an *X*, *O*, or empty space in each of the nine spaces. In the program, the Tic-Tac-Toe board is represented as a list of strings like the ASCII art of Hangman. Each string represents one of the nine spaces on the board. The strings are either 'X' for the *X* player, 'O' for the *O* player, or a single space ' ' for a blank space.

C. Strategizing with the AI Game

The AI player utilizes a profundity constrained negamax calculation with alpha beta pruning. It's basic for the calculation to give a similar score to different youngster's hubs. On the off chance that there are various "best" move choices, the calculation will arbitrarily pick between the choices. This makes diversions somewhat more fascinating particularly when you have two AI players playing against each other [6].

The AI should have the capacity to take a gander at the board and choose which kinds of spaces it will proceed onward. To be clear, we will name three kinds of spaces on the Tic-Tac-Toe board: corners, sides, and the middle [7]. The AI's system for playing Tic-Tac-Toe will take after a straightforward calculation, a limited arrangement of guidelines to register an outcome. A solitary program can make utilization of a few unique calculations. A calculation can be spoken to with a flowchart. The Tic-Tac-Toe AI's calculation will process the best move to make [8].

The exact opposite thing that the program does is telling which player has the most wins or if there is a draw. Two players, one allotted the image X (goes first) and one doled out the image O, alternate putting their imprints on a 4x4 board. The diversion either closes in a win, when one player gets three stamps in succession, or a draw, when the barricade is filled [9]. From a system stance, Tic-Tac-Toe is viewed as a "fathomed" diversion. You can give a PC an arrangement of guidelines that will dependably enable it to win or draw. Be that as it may, it's not extremely amusing to play against a great PC, so a vast piece of the test when making this amusement is making sense of how to make an AI that is shrewd yet blemished. Consequently the tenets of tic-tac toe diversion is sufficiently basic that we needn't bother with an intricate investigation of amusement designs in spite of being a straightforward amusement, the fundamental AI standards appeared here can be connected to more confounded recreations, for example, Checkers, Go and even chess. After the player or computer makes a move, the program checks if they won or caused a tie, and then the game switches turns. After the game is over, the program asks the player if they want to play again, see in the flow chart below;

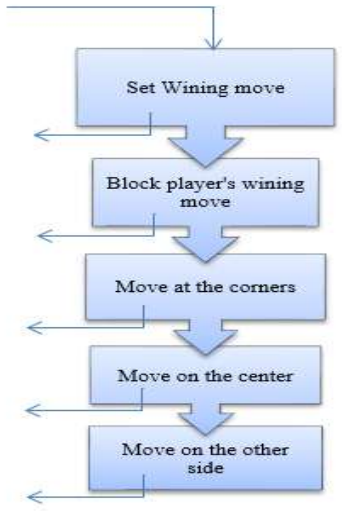


Figure 2: Board Flow chart Representation Syntax

D. Algorithms

This program utilizes the minimax calculation with discretionary alpha-beta pruning. The board is spoken to as a one-dimensional exhibit and multi-dimensional cluster. A wide range of assessment capacities are given. Likewise included is a capacity for checking if the game has been won.

The semantic algorithm is yet another approach towards the tic-tac-toe game. The semantic algorithm is in essence a learning algorithm, and it might be structured in the following way. It might have as initial information the ability to recognizing the 3 states of a game; lost won or a draw. The algorithm in this case would play the X, and it will play against another algorithm, i.e. the O. As soon as a game is finishes, the information if the game was won or lost is stored. Moreover, the moves are presented with the smaller letters "x" and "o" accordingly.

E. Win Detector

To recognize a success, this program basically takes a gander at each conceivable line, section and corner to corner to check whether any one player's pieces consume every one of the 4 spaces. This is accomplished utilizing a progression of conditionals. There are 76 distinct potential outcomes for a success (16 columns toward every path, 2 diagonals for every face toward every path (which makes 12 countenances), and after that 4 corner-to-corner diagonals), and this calculation checks everyone in grouping.

F. Static Evaluators

This program gives four diverse static evaluators. Everyone starts by taking a gander at each line, section, and corner to corner thus. On the off chance that the two players have pieces in that line, it is overlooked; if just a single player has pieces in it, the evaluator checks these n pieces. Toward the finish of this count, the evaluator has a rundown of what number of 1-in-a-lines, 2-in-a-columns, 3-in-a-lines and 4-in-a-lines every player has. Where the evaluators contrast is in what they do with this data, as depicted beneath.

IV IMPLEMENTATION

A. Minimax Algorithm

This is the algorithm that enables the player to look forward to future moves and pick the move that gives him the best alternatives for future moves. It has two fundamental capacities, the minmove and maxmove capacities, which are commonly recursive. Given a lot of potential moves, the player thinks about what the adversary's best move would be in every one of those potential states (utilizing the minmove capacity), and after that picks the move that gives his rival the most exceedingly awful choices.

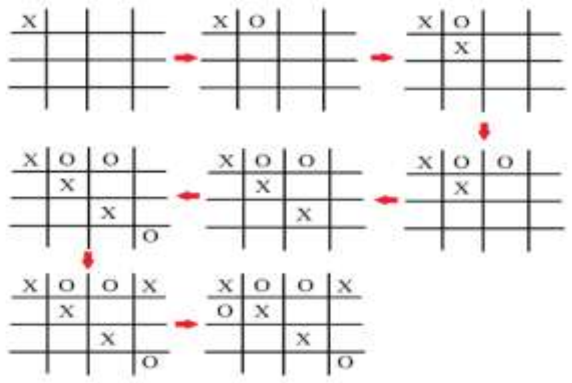


Figure 3: Minimax Algorithm

Additionally, the minmove capacity considers all the potential moves from a given state, thinks about what the player's best move would be from those (utilizing maxmove), and returns the estimation of his most exceedingly terrible choice [10]. This common recursion proceeds until as far as possible is come to, so, all in all the capacity just assesses all the potential states (utilizing one of the static evaluators).

B. Alpha-Beta Pruning Algorithm

This calculation is essentially a fancier form of minimax. It calls minmove and maxmove precisely as portrayed above, yet in addition goes around two factors, alpha and beta. Alpha, at first set to - 650001 (one not exactly the estimation of the most noticeably terrible conceivable move, one where the rival wins), monitors the best move the rival will give us a chance to get up until now. In the event that the minmove capacity experiences a more terrible move than alpha, it can stop immediately, since it realizes that the player can improve. Correspondingly, beta, at first set to 600001 (one more than the estimation of a success for the player), monitors the most exceedingly awful move the rival can make the player take. On the off chance that the maxmove capacity experiences something better, it can stop immediately, since it realizes the adversary will never give it a chance to take that move. Together, these factors enable the calculation to look at numerous less hubs, in this manner accelerating execution significantly [11].

C. Symmetries of the 4 ×4 Game

One innocent way to deal with this issue is to decide each conceivable condition of the board and afterward structure a reviewed poset with the one of a kind maximal passage comparing to an unfilled board and after that as we go down we take a gander at all potential approaches to legitimately embed one number until either there is a success or the game outcomes in a tie[12]. Given such a poset we could then effectively decide the victor of the game by working from the base to the top. Anyway an estimation for the quantity of sheets at profundity k is;

$$\binom{16}{k}\binom{16}{\frac{k}{2}}\binom{16}{\frac{k}{2}}k!$$

That is, pick k positions out of 16 conceivable positions, at that point pick which chances are to be played, which levels are to be played and put them on the board in all conceivable ways. Summing up finished all conceivable k at that point gives us 2.7 × 1015 sheets. For examination the 3 × 3 form of Numerical Tic-Tac-Toe has 9.3 × 106 and traditional Tic-Tac-Toe has 6, 046. (This check gives every single conceivable board, yet a portion of these sheets would not happen in gameplay as they contain inside them at least two disjoint winning lines which would show play has officially ceased.) Even with progresses in registering force and memory stockpiling this is as yet restrictive to approach an investigation of the 4 × 4 board. We will utilize a few systems to decrease the span of this issue to a point where the calculation can be done productively. A standout amongst the most vital instruments that we have is to utilize the symmetry of the board. That is a bijection from the board to itself which jelly lines. To be more exact when we have the accompanying board;

Then the lines are: {1,2,3,4}, {5,6,7,8}, {9, 10,11,12}, {13,14,15,16}, {1,5, 9,13}, {2,6, 10,14}, {3,7,11,15}, {4, 8,12, 16}, {1, 6,11, 16},

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

The bijections of the board consist of compositions of the following maps: rotations, reflections, and the two maps shown below*;*

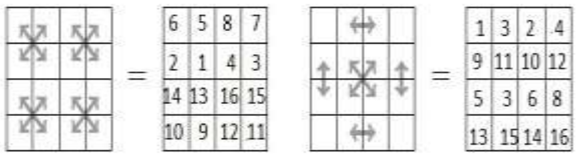
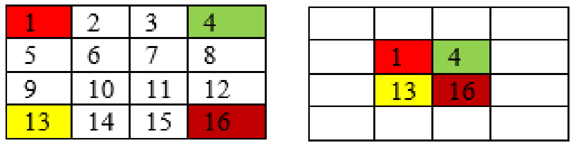


Figure 4 Bijection Board Composition

For the two maps given above we will consider the one on the left the "cross evenness" and the one Child the privilege the "X-balance" When these are consolidated we end up with 32 diverse bijections that save the lines of the 4 × 4 board. Proof: A straightforward check will confirm that every single one of those maps will save lines, so it gets the job done to demonstrate that in the event that we have saved lines that we should be a synthesis of these maps. Next we note that each guide we have illustrated is reversible (i.e., for revolution we switch the heading and for different maps we just apply the guide a subsequent time). Subsequently to demonstrate that we have all conceivable bijections by joining these maps, it does the trick to indicate how we can apply these maps to return to the beginning board.

So assume we have a board that has safeguarded lines. At that point by applying revolutions we can put the 1 in the upper left corner. Note that 1, 4, 13 and 16 will dependably need to frame the sides of a square thus if necessary we can apply reflection to put 4 in the upper right corner. Reflection will accomplish this since 1 and 4 can't be on inverse corners, since that would drive 2 and 3 to be one of the four focus squares. The middle squares are associated with three slines; however 2 and 3 are just engaged with two, which means they can never be focus tiles. Along these lines we are in one of the accompanying two circumstances:-



Similarly, *6*, *7*, *10*, and *11* will shape another square and their situating must concur with the above sheets in protecting corner to corner lines. So in the principal case we presently have the accompanying four potential outcomes:-

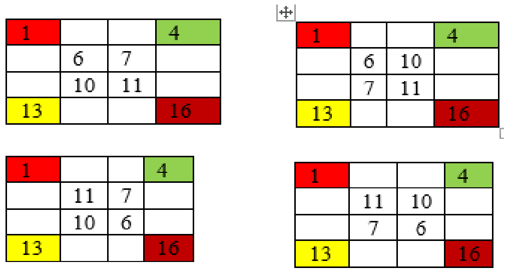


Figure 5 Demo composition board State Scenario

Each staying unfilled square is contained in two lines and we can figure out what esteem (if conceivable) must go in the square by taking the convergence of the two lines. The principal probability diminishes to the character, the fourth gives the X evenness, and the other two are inconceivable.

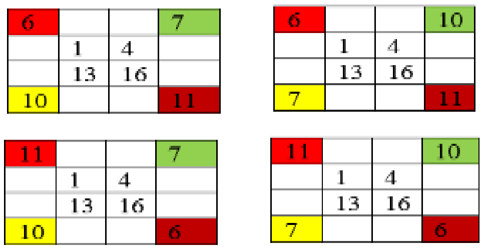


Figure 6 Demo composition board State Scenario

Once more, the procedure as before we can fill in any residual squares by taking a gander at the crossing point lines. The primary plausibility decreases to cross-balance, the fourth probability is the aftereffect of structure of X evenness and cross-balance maps, and the other two are incomprehensible. Hence utilizing just our given maps we have represented each legitimate bijection of the board.

There is one other common contender for evenness including control of the numbers themselves instead of the area of every passage. The evenness was utilized by Murkowski [4] in the 3 × 3 form and was found because of the reality:-

$$a + b + c = 15 \leftrightarrow (10 - a) + (10 - b) + (10 - c) = 15.$$

Simple algebra takes the original sum $n(n^2 + 1)/2$ and finds that in the general case; subtracting each filled entry from n^2 +1 presents a possible symmetry. For the 4×4 version, each filled cell q would then be replaced by $17 - q$.

V. RESULTS

Player X can win or power a draw from any of these beginning imprints; be that as it may, playing the corner gives the rival the littlest selection of squares which must be played to abstain from losing. The second player, whom we might assign "O", must react to X's opening imprint so as to keep away from the constrained win. Player O should dependably react to a corner opening with an inside stamp, and to a middle opening with a corner check. An edge opening must be addressed either with a middle check, a corner stamp by the X, or an edge check inverse the X.

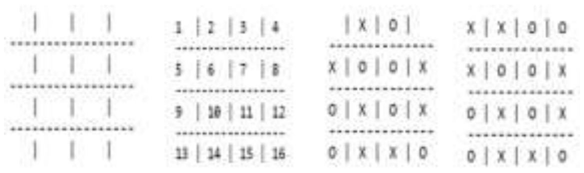


Figure 7: Tic-Tac Toe based on manual programmed demo

Some other reactions will enable X to compel the win. Once the opening is finished, O's undertaking is to take after the above rundown of needs so as to constrain the draw, or else to pick up a win if X makes a frail play. To ensure a tie in case you're O however, in the event that X doesn't play focus (playing a corner is the best opening move), take focus, and afterward a side center.

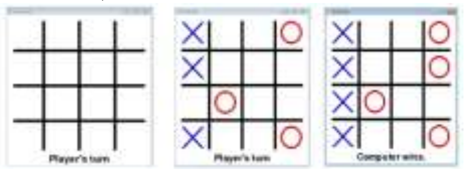


Figure 8: Tic-Tac Toe based on AI demo

This will prevent any forks from happening. On the off chance that you play a corner, an immaculate X player has officially played the corner inverse his first and continues to play a third corner, halting your 3-in-a-rowand making his own fork. In the event that X plays focus opening move, simply keep your eyes open and he won't have the capacity to fork you. In the event that you are X, play a corner first. In the event that O takes focus (best move for him), take the corner inverse you're unique, and continue as point by point above. On the off chance that O plays a corner or side-center to begin with, you are ensured to win. On the off chance that corner, essentially take any of the other 3 corners, and after that the last. You've cleft him. In the event that he plays a side-center, take the main corner that his blocking won't make 2 out of a line. He'll square, yet the best of the other two, you'll see which one, and you'll fork him. The main path for X not to win is for O to play center and after that a side-center.

VI. REFERENCE

[1] D. B. Klitsner and B. P. Clemens, "Electronic tic-tac-toe game having three function control," ed: Google Patents, 2003.

[2] P. Turner, "Combination tic-tac-toe and question and answer game," ed: Google Patents, 1987.

[3] E. Kaplan, "Interactive tic-tac-toe slot machine," ed: Google Patents, 1999.

[4] S. C. Raphael, A. S. Raphael, and R. R. King, "Three-dimensional tic-tac-toe game," ed: Google Patents, 1995.

[5] P. R. Anderson, J. D. Flint, J. J. Giobbi, S. P. Joshi, and E. A. Frohm, "Gaming machine with pattern-driven bonus array," ed: Google Patents, 2005.

[6] T. G. Daly, "Game and gaming machine having tic-tac-toe type feature," ed: Google Patents, 2016.

[7] J. Blake and J. Goodman, "Computer-based learning: games as an instructional strategy," *ABNF JOURNAL*, vol. 10, pp. 43-45, 1999.

[8] C. Schlieder, P. Kiefer, and S. Matyas, "Geogames: Designing location-based games from classic board games," *IEEE Intelligent Systems*, vol. 21, pp. 40-46, 2006.

[9] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, pp. 450-465, 2018.

[10] A. Saffidine, H. Finnsson, and M. Buro, "Alpha-beta pruning for games with simultaneous moves," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[11] I. Roizen and J. Pearl, "A minimax algorithm better than alpha-beta? Yes and no," *Artificial Intelligence*, vol. 21, pp. 199-220, 1983.

[12] J. B. Pollack, "Binomial and multinomial-based slot machine," ed: Google Patents, 2012.