

**Yarmouk University**

**Faculty of Information Technology and Computer Sciences**

**Computer Science Department**

Tic Tac Toe advanced

**Project Team**

**Ahmad MousaKhaled Salem ( 2015901162 )**

**Hadeel Hassan Saleh Al-Naamneh( 2015801014 )**

**JumanIskandar Sari Haddad ( 2015901032 )**

**Supervisor: Dr. MalekBarhoush**

**2018-2019 First semester**

**Table of Contents**

[1.Abstract……………………………………………………………………………..3](#_Toc451937472)

[1.1.History………………………………………………………………………….3](#_Toc451937473)

[2.Introduction and overview………………………………………………………….7.](#_Toc451937473)

[2.1.Introduction:…………………………………………………………………….7](#_Toc451937474)

[2.2.Objective:……………………………………………………………………10](#_Toc451937475)

[3.Functional Requirements:………………………………………………………..11](#_Toc451937476)

[3.1.Requirement for Tic Tac Toe game…………………………………………11](#_Toc451937479)

[4.Equipments, HW and SW…………………………………………………………12.](#_Toc451937483)

[5.Use case Diagram………………………………………………………………….13.](#_Toc451937484)

[6.Use case Specification……………………………………………………………..15.](#_Toc451937485)

[7.Data flow digram………………………………………………………………….21..](#_Toc451937486)

[7.1.Data flow digram level 0………………………………………………………21.](#_Toc451937485)

[7.2.Data flow digram level 1………………………………………………………21.](#_Toc451937485)

[8.GeneralFramework………………………….…………………………………….22.](#_Toc451937485)

9.MinMax algorithm……………………………………………………………………………………………………………24

10.Psedou code……………………………………….................................................25

10.1.GameController………………………………………………………………...26

10.2. SmartAi………………………………………………………………………...27

11. Refferences……………………………………………………………………...30

# 1.Abstract

**TIC-TAC-TOE** is one of the most popular games in the world and it consists of two players ( X and O), it's based on winning one of two players or none of them, the player who succeeds in placing threeof their marks in horizontal, vertical , or diagonal row wins the game. Once they discover that the best play from both parties leads to a draw.

# 1.1.History:

The origins of the Tic-Tac-Toe game are back to ancient Egypt, where such game boards have been found on roofing tiles dating from around 1300 BCE. An early variation of tic-tac-toe was played in Roman empire, around the first century BC. Another closely related ancient game is Three men's morris which is played on a simple grid and requires three pieces in row to finish.

Thefirst print reference to a game called "noughts and crosses" occurred in 1884. Then the US renaming of noughts and crosses as a tic-tac-toe accurred in the 20th century.

In 1952, the game developed by British computer scientist Alexander S. Douglas for EOSAC , computer at the university of Cambridge, became one of the first known video games, the computer player could play perfect games of tic-tac-toe against human.

Then, it was discovered anew tic-tac-toe consist of 16 cells, This game takes place on a 4x4 board and involves two players, X and O. X starts and players take alternate turns. The object of the game is to place four of your pieces in a row, either vertically, horizontally, or diagonally. If this is achieved, the game ends in a win, otherwise the game ends in a draw once no more moves are possible.

Now adays the computer has been used to design tic-tac-toe game by using artificial intelligence, and in this project we are going to develop and modify this game and design it using four boards.

The first way to win in Tic Tac Toe in one-dimensional game is raw is containing of (o):

1- Items 1, 2, 3 and 4 contain the symbol (o).

|  |  |  |  |
| --- | --- | --- | --- |
| o | o | o | o |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

2- Items 5, 6, 7 and 8 contain the symbol (o).

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| o | o | o | o |
|  |  |  |  |
|  |  |  |  |

3- Items 9, 10, 11 and 12 contain the symbol (o).

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
| o | o | o | o |
|  |  |  |  |

4-items 13, 14, 15 and 16 contain the symbol (o).

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| o | o | o | o |

The second way to win in Tic Tac Toe in one-dimensional game is column is containing of (o):

1- Items 1, 5, 9 and 13 contain the symbol (o).

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | o |
|  |  |  | O |
|  |  |  | O |
|  |  |  | O |

2- Items 2, 6, 10 and 14 contain the symbol (o).

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | o |  |
|  |  | o |  |
|  |  | o |  |
|  |  | o |  |

3- Items 3, 7, 11 and 15 contain the symbol (o).

|  |  |  |  |
| --- | --- | --- | --- |
|  | o |  |  |
|  | o |  |  |
|  | o |  |  |
|  | o |  |  |

4- Items 4, 8, 12 and 16 contain the symbol (o).

|  |  |  |  |
| --- | --- | --- | --- |
| o |  |  |  |
| o |  |  |  |
| o |  |  |  |
| o |  |  |  |

The Third way to win in Tic Tac Toe in one-dimensional game is diameter is containing of (o):

1- Items 1, 6, 8 and 16 contain the symbol (o).

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | o |
|  |  | o |  |
|  | o |  |  |
| o |  |  |  |

2- Items 4, 7, 10 and 13 contain the symbol (o).

|  |  |  |  |
| --- | --- | --- | --- |
| o |  |  |  |
|  | o |  |  |
|  |  | o |  |
|  |  |  | O |

# 2.Introduction and overview

## 2.1.Introduction:

In this project we are going to develop an advanced tic-tac-toe, it consists of two players one of them is the computer and the other is the human, with a four board and every board is in 4x4 cell.

it based on specific strategies has been designed using AI and Min-Max algorithm, and it's considered one of the hard games, because the computer uses AI pattern in playing to compete the human smartly.

Although this game is difficult, the elderly or young people can play it.

We designed this game so it will work on**PC**.

The main purpose of it is to help some people get rid of stress, increase the efficiency of the brain and develop learning skills.

1- Items 0, 21, 42 and 63 contain the symbol (o).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| o |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | o |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  | o |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | O |

6- Items 15, 26, 37 and 48 contain the symbol (o).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | O |
|  |  |  |  |  |  |  |  |  |  | o |  |  |  |  |  |
|  |  |  |  |  | o |  |  |  |  |  |  |  |  |  |  |
| o |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

# 2.2. Objective:

The game of Tic Tac Toe is a game of unconsciousness played by many people and they do not recognize them, where the human mystery of this game through interactive analysis, which is one of the main elements and necessary in the social life of all peoples in all parts of the world, It's an important game for all age groups including kids and adults because of itsbenefits**,** including**:**

* Develop a new style of thinking by motivating the mind to make smart and quick decisions to beat the computer.
* Reflect an important role in human life, where man is able to face problems and take a better way to solve them.
* Contribute to training the mind to think, and solve the most difficult mathematical issues.
* It is a exercise for the brain, as it is necessary to practice some brain exercises every day in different ways.

# 3.Functional Requirements



1.The system should allow the other player to select that plays the first or second .

2. The system must choose the right cell to prevent the other player from winning.

3.The system should choose the right corner to win.

4. The system should Marks the cell chosen by other player.

5. The system should use the Artificial intelligence techniques.

6. The computer should be one of the players.

7. The system should stop the game if it wins or the other player is win.

8. The system should check every winning path in every time the computer or the human is played.

9. The system should display the winning path in the end.

10. The system should give the final result to win or lose or tie the parties.

# 4.Equipments, HW and SW

**In this section we will describe the software and hardware tools which need for our system:**

1. Operating system: Windows XP / Windows 7 / Windows 8 / Windows 10.

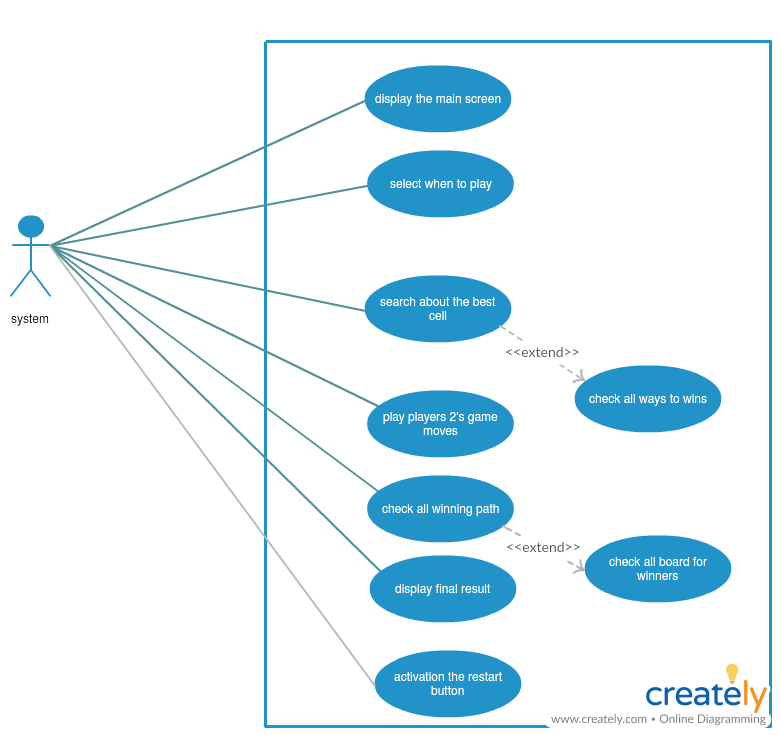
2. Microsoft Office Word 2010 to write documentation.

3. creately.com diagram software to draw UML diagrams.

4. Programming language: C#.

5. Design: Unity.

5.Use case diagram



**Figure 1: Use Case Diagram.**

**This diagram is called use case diagram, it shows all of the operations that the Tic Tac Toe game system can do, these operations are:**

* Search about the best cell: After a human chooses a single cell to play, the system searches for the best cell to play in order to reduce the chances of the other player's win, This process involves checking all the ways to win the other player
* Check winning path: Every time a computer or human plays, the system examines all the winning methods in order to provide the final result to win if any and The system examines all winning methods for all the boards in the game.
* Display the final result: After the system examines all the winning methods and finds truly one winning path, the system Declares win or loss other player, If all the cells are filled without a win, the system declares the two sides equal.
* Activate play again button: After the system shows the final result of the game, the system does the play button again, and after pressing this button all cells become effective for playing again.

6.Use case Specification

|  |  |
| --- | --- |
| Use Case ID: | Tic\_Tac\_toe -01 |
| Use Case Name: | Start game |
| Actors: | Human |
| Description: | Press on start game button to start playing. |
| Preconditions: | The game is on thedevice. |
| Post conditions: | All cells are effective to pick on. |
| Normal Flow: | 1. Open the game by clicking on the game icon.  2. Press on start game button. |
| Alternative Flows: | Reload the game then open it. |

**Table 1: Use Case for start game.**

|  |  |
| --- | --- |
| Use Case ID: | Tic\_Tac\_Toe-02 |
| Use Case Name: | Pick. |
| Actors: | Human, computer. |
| Description: | The computer or human select one cell in every time they play to pick on, then the cell is filled with the sympol x or o according to the player. |
| Preconditions: | 1. The cell is empty.  2. The player's role in playing. |
| Post conditions: | The cell is full with one sympol. |
| Normal Flow: | 1. The computer or human select one cell.  2. Pick on the cell.  3. The cell filled with sympol x or o. |
| Alternative Flows: | Select another cell |

**Table 2: Use Case for Pick.**

|  |  |
| --- | --- |
| Use Case ID: | Tic\_Tac\_Toe-03 |
| Use Case Name: | Check winning path. |
| Actors: | System. |
| Description: | The system check all the winning path by check all the cell every time they are played,  And the system pick on the best cell in order to wins and the Tic Tac Toe game system give the final result, when it's found one winning path. |
| Preconditions: | At least one cell is full. |
| Post conditions: | The system select the best cell to play and give the final result, when it's found one winning path. |
| Normal Flow: | 1. The system check all winning path on the cells in every time the computer or human played.  2. The system select best cell to play, or display the final result. |
| Alternative Flows: | Restart the game. |

**Table 3: Use Case for chick winning path.**

|  |  |
| --- | --- |
| Use Case ID: | Tic\_Tac\_Toe-04 |
| Use Case Name: | Game over. |
| Actors: | System. |
| Description: | After check all winning path the system display the final result for win or loss or draw. |
| Preconditions: | At least five cell is full. |
| Post conditions: | The system display the final result. |
| Normal Flow: | 1. The system check all winning path.  2. The system display message containing final result. |
| Alternative Flows: | Activate play again button. |

**Table 4: Use Case for Game over.**

# 8.Data Flow Diagram

## 8.1.Data Flow Diagram Level 0

**pick**

Human /computer player

Tic-Tac-toe

Software

**display**

## 8.2.Data Flow Diagram Level 1

1.display board for the game

3.Pick the cell

Board

4.check all the winning path and display final result

2.check the cell is empty or full

# 8.General framework

**8.1.Unity**

**Use the Unity Editor to create 2D and 3D games, apps and experiences.**

Unity is, in short, a closed-source, cross-platform game development application. You create your game by manipulating objects in 3D and attaching various components to them. Even 2D games must be manipulated in 2D/3D. Scripts are written in C# (recommended), Boo or UnityScript (some peoples mistakenly call it JavaScript) and attached to 3D objects as components.

That said, once you’ve created a game with Unity, deployment is a cinch. With a couple of clicks, you can export your game to mobile, desktop and/or web (web currently requires the Unity player app to be installed). If you have the right license, you can even deploy to gaming consoles like Xbox, Playstation and Wii.

**A) THE GAME ENGINE FOR MOBILE**

Independent reports make clear that Unity is far and away the world’s favourite game engine for creating mobile games. Why is it so popular?

* One-click deployment to **Android**, **iOS**, **Windows Phone** and **Tizen**.
* Tons of optimizations thanks to features like occlusion culling, asset bundling and build size stripping.
* World class monetizations and retention services for mobile games.
* Dedicated, easy to use 3D and 2D tools and workflows.

**B) UNITY FOR DESKTOP**

*STRATEGY, PUZZLE, ACTION, SANDBOX AND MORE*

Thanks to Enlighten-powered Realtime Global Illumination and Unity’s physically-based shader, there’s never been a better time to make beautiful, immersive and entertaining desktop games with Unity.The Unity engine provides one-click deployment support across**PC, Mac** and **Linux** platforms.

**8.2.**Visual Studio #C

The Visual Studio integrated development environment is a creative launching pad that you can use to edit, debug, and build code, and then publish an app. An integrated development environment (IDE) is a feature-rich program that can be used for many aspects of software development. Over and above the standard editor and debugger that most IDEs provide, Visual Studio includes compilers, code completion tools, graphical designers, and many more features to ease the software development process.

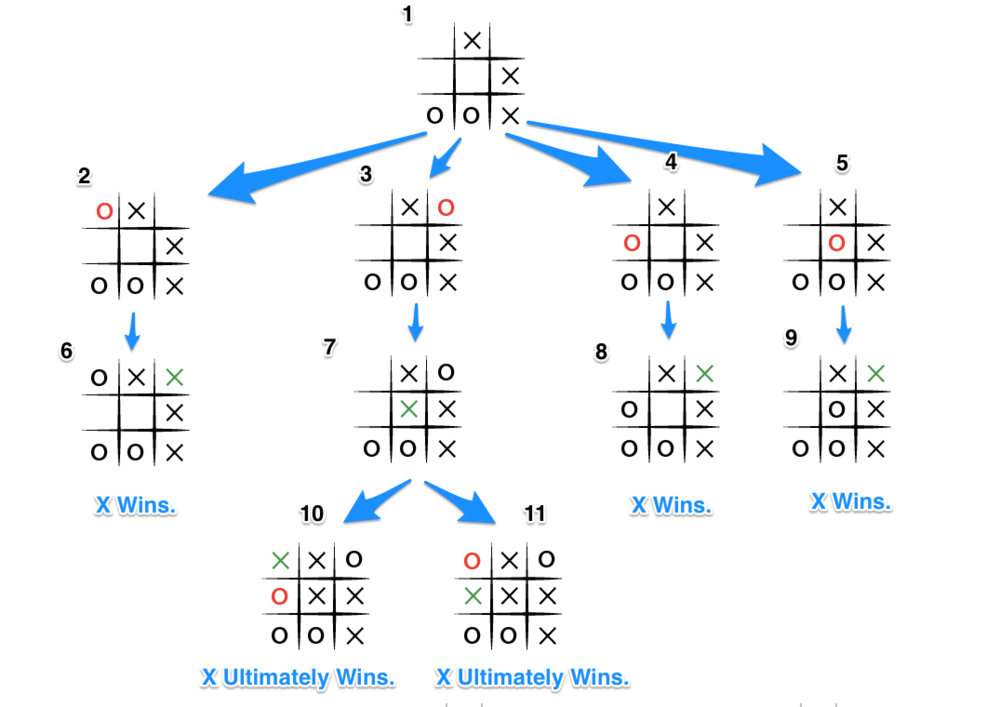
Visual Studio is available for Windows and Mac. Visual Studio for Mac has many of the same features as Visual Studio 2017, and is optimized for developing cross-platform and mobile apps. This article focuses on the Windows version of Visual Studio 2017.

There are three editions of Visual Studio 2017: Community, Professional, and Enterprise. See Compare Visual Studio IDEs to learn about which features are supported in each edition.

This overview article introduces you to the basic features of the IDE. We'll walk through some things you can do with Visual Studio, including creating a simple project, using IntelliSense as a coding aid, and debugging an app to see the value of a variable during the program's execution. We'll also take a tour of the various tool windows.

# 9.MinMax algorithm.

Minimax is a recursive algorithm which is used to choose an optimal move for a player assuming that the opponent is also playing optimally. As its name suggests, its goal is to minimize the maximum loss (minimize the worst case scenario).



# 10.Pseudocode

**10.1. GameController.**

1- Startgame()

{ if( player select play second)

The computer (o) is play first, and call AiTurn function

Else

The humanmark (x) is play first}

2- RestartClick()

{Hide the final result screen

Activate chooseplay screen

For(each all cell in all boards)

{ Dump the all cells}

Call StartGame function}

3- SelectPlayerSide(side)

{ for( each all cell in all boards)

{activate all cells

Re- color the cells white}

Call RestartGame function}

4- Awake()

{

humanmark = "x"

computer = "o"

Activate chooseplay screen

Hide the final result screen}

5- GetPlayerSide()

{

Return the player turn}

6- CheckWinningPath()

{

for( each all rows)

{ for ( each all boards)

{

if ( the values row are equal)

{ if( value == humanmark)

{

Call the finalresult function

Plus 1 in x score

Call AfterWinning function}

Else

If( value == computer)

{

Call the finalresult function

Plus 1 in o score

Call AfterWinning function}

}}}

}

For ( each all column)

{

For ( each all boards)

{

if ( the values column are equal)

{ if( value == humanmark)

{

Call the finalresult function

Plus 1 in x score

Call AfterWinning function}

Else

If( value == computer)

{

Call the finalresult function

Plus 1 in o score

Call AfterWinning function}

}}}

/\*

diagonal path 1

O \_ \_ \_

\_ O \_ \_

\_ \_ O \_

\_ \_ \_ O

\*/

For ( each all boards)

{

if ( the values diagonal path 1 are equal)

{ if( value == humanmark)

{

Call the finalresult function

Plus 1 in x score

Call AfterWinning function}

Else

If( value == computer)

{

Call the finalresult function

Plus 1 in o score

Call AfterWinning function}

}}

/\*diagonal path 2

\_ \_ \_ O

\_ \_ O \_

\_ O \_ \_

O \_ \_ \_

\*/

For ( each all boards)

{

if ( the values diagonal path 2 are equal)

{ if( value == humanmark)

{

Call the finalresult function

Plus 1 in x score

Call AfterWinning function}

Else

If( value == computer)

{

Call the finalresult function

Plus 1 in o score

Call AfterWinning function}

}}

if ( the values diagonal path 1 mix boards are equal)

{ if( value == humanmark)

{

Call the finalresult function

Plus 1 in x score

Call AfterWinning function}

Else

If( value == computer)

{

Call the finalresult function

Plus 1 in o score

Call AfterWinning function}

}}

}

if ( the values diagonal path 1 mix boards are equal)

{ if( value == humanmark)

{

Call the finalresult function

Plus 1 in x score

Call AfterWinning function}

Else

If( value == computer)

{

Call the finalresult function

Plus 1 in o score

Call AfterWinning function}

}}

}

if (Not found empty cell in all boards)

{

The final result is draw

Plus 1 in draw score

}

Else

Call changeside function

}

6- ChangeSide()

{

If(turn X)

{be turn O}

Else{

Be turn X}

If(turn O)

{

Call function LastMoveX

Call function Aiturn

}

}

7-AiTurn()

{

If ( check () == 0)

{

Call MinMax for all board }

If (there is no empty cell in board 0)

move0.score = large negative value}

If (there is no empty cell in board 1)

move1.score = large negative value }

If (there is no empty cell in board 2)

move2.score = large negative value }

If (there is no empty cell in board 3)

move3.score = large negative value }

if (last move human(X) in board 0 )

{

If ( there is no empty cell in board 0)

{

Max value = there is result score minimax on board 1

Mx= there is result index minimax on board 1

For ( each all boards )

{ if(there is result minimax on board large of Max value and not board 0){

Max value = there is best result score minimax on board

Mx= there is best result index minimax on board

}

}

move computer in cell index mx

Unactive cell index mx}

Else If ( the best cell is 0 if empty)

{ move computer in cell is 0 in board 0

Unactive cell 0

}else If ( the best cell is 12 if empty)

{ move computer in cell is 12 in board 0

Unactive cell 12

}

Else{

move computer in cell result score minimax on board 0

Unactive cell result index minimax on board 0

}

}else

if (last move human(X) in board 1 )

{

If ( there is no empty cell in board 1)

{

Max value = there is result score minimax on board 0

Mx= there is result index minimax on board 0

For ( each all boards )

{ if(there is result minimax on board large of Max value and not board 1){

Max value = there is best result score minimax on board

Mx= there is best result index minimax on board

}

}

move computer in cell index mx

Unactive cell index mx

}

Else If ( the best cell is 21 if empty)

{ move computer in cell is 21 in board 1

Unactive cell 21

}else If ( the best cell is 25 if empty)

{ move computer in cell is 25 in board 1

Unactive cell 25

}

Else{

move computer in cell result score minimax on board 1

Unactive cell result index minimax on board 1

}

else

if (last move human(X) in board 2 )

{

If ( there is no empty cell in board 2)

{

Max value = there is result score minimax on board 0

Mx= there is result index minimax on board 0

For ( each all boards )

{ if(there is result minimax on board large of Max value and not board 2){

Max value = there is best result score minimax on board

Mx= there is best result index minimax on board

}

} move computer in cell index mx

Unactive cell index mx

}

Else If ( the best cell is 38 if empty)

{ move computer in cell is 38 in board 2

Unactive cell 38

}else If ( the best cell is 42 if empty)

{ move computer in cell is 42 in board 2

Unactive cell 42

}

Else{

move computer in cell result score minimax on board 2

Unactive cell result index minimax on board 2

}

else

if (last move human(X) in board 3 )

{

If ( there is no empty cell in board 3)

{

Max value = there is result score minimax on board 0

Mx= there is result index minimax on board 0

For ( each all boards )

{ if(there is result minimax on board large of Max value and not board 3){

Max value = there is best result score minimax on board

Mx= there is best result index minimax on board

}

}

move computer in cell index mx

Unactive cell index mx

}

Else If ( the best cell is 51 if empty)

{ move computer in cell is 51 in board 3

Unactive cell 51

}else If ( the best cell is 63 if empty)

{ move computer in cell is 63 in board 3

Unactive cell 63

}

Else{

move computer in cell result score minimax on board 3

Unactive cell result index minimax on board 3

}

Call CheckWinningPath function

8- FinalResult( string value)

{

If ( value = draw )

{

Display final result screen for draw

Else

Display final result screen for x or o

}

}

9- AfterWinning( three values )

{

Re- active the win cells green

Re- color the win cells green

}

10- Check\_befor\_lastOneMove\_happened\_winningPath ()

{

For ( each all rows)

{

For ( each all boards )

{

If ( three values in row ==" o" )

{

Index cell Remaining = " o"}}

For ( each all columns)

{

For ( each all boards )

{

If ( three values in column ==" o" )

{

Index cell Remaining = " o"}}

For ( each all boards )

{

If ( three values in diagonal ==" o" )

{

Index cell Remaining = " o"}

If ( three values in diagonal mix board==" o" )

{

Index cell Remaining = " o"}

For ( each all rows)

{

For ( each all boards )

{

If ( three values in row ==" X" )

{

Index cell Remaining = " o"}}

For ( each all columns)

{

For ( each all boards )

{

If ( three values in column ==" X" )

{

Index cell Remaining = " o"}}

For ( each all boards )

{

If ( three values in diagonal ==" X" )

{

Index cell Remaining = " o"}

If ( three values in diagonal mix board==" X" )

{

Index cell Remaining = " o"}

**10.2. SmartAi**

1- Move minmax( 7 values)

{

If ( human is win)

{

Score = -10

}

Else

If( computer is win)

{ score = 10}

Else

If (no cell is empty and no winner)

{

Score =0

}

Plus 1 on the depth

If ( depth == 10 )

{

If ( the player is a computer)

{

Score = alpha

}

Else

{

Score = beta

}

For ( each all free cells)

{

Put playermark in the empty cell

If ( turn == computer )

{

Score=Call minimax function.

If(Score large alpha )

{

Store new result score and index best move

}

}

Else

{

Score=Call minimax function.

If(Score small beta )

{

Store new result score and index best move

}

}

2- ListEmptyCells(button list ,lo ,hi){

Check if the empty cell and add list

}

3- CheckWin4( array of text, player)

{

Search about all winning path on row or column or diagonal}

}

**11.References**

1. [www.creately.com](http://www.creately.com)
2. <https://www.quora.com/What-is-unity-game-engine>
3. <https://docs.microsoft.com/en-us/visualstudio/ide/visual-studio-ide?view=vs-2017>
4. [https://en.wikipedia.org/wiki/Tic-tac-toe](https://en.wikipedia.org/wiki/Tic-tac-toe?fbclid=IwAR2T_S9FOg39oRBPM7s5pQGoRlmz1ShwaU7ZDz6Q89TqG4R0kME2AnuR7KI)
5. [https://www.almrsal.com/post/575030](https://www.almrsal.com/post/575030?fbclid=IwAR2pxHv8iTAySymIu5zXNyrQiprnTBO5kdEMSnX4nkmyRa0oIAZucRmD5Uo)
6. <https://towardsdatascience.com/tic-tac-toe-creating-unbeatable-ai-with-minimax-algorithm-8af9e52c1e7d>