

Experiment - 3

Samkit Shah
2019130060
TE Comps
Batch - C

Aim:

To build a Tic-Tac-Toe using A* Algorithm.

Code:

```
game = {1: ' ', 2: ' ', 3: ' ',
        4: ' ', 5: ' ', 6: ' ',
        7: ' ', 8: ' ', 9: ' '}

def insert(letter, position):
    if is_free(position):
        game[position] = letter
        Display(game)
        if (is_draw()):
            print("Draw!")
            exit()
        if Win():
            if letter == comp:
                print("comp wins!")
                exit()
            else:
                print("Player wins!")
                exit()
        return
    else:
        print("Can't insert")
        position = int(input("enter position: "))
        insert(letter, position)
        return

def is_free(position):
    if game[position] == ' ':
        return True
    else:
        return False

def Display(game):
    print(game[1] + '|' + game[2] + '|' + game[3])
    print('-+-+-')
    print(game[4] + '|' + game[5] + '|' + game[6])
    print('-+-+-')
    print(game[7] + '|' + game[8] + '|' + game[9])
    print("\n")
```

```

def is_winner(mark):
    pos = False
    for j in range(1,8,3):
        check=True
        for i in range(j,j+2):
            if game[i]!=mark or game[i]!=game[i+1]:
                check=False
                break
        pos = pos|check
    for j in range(1,4):
        check=True
        for i in range(j,j+5,3):
            if game[i]!=mark or game[i]!=game[i+3]:
                check=False
                break
        pos = pos|check
    if(game[1] == mark and game[1]==game[5] and game[5]==game[9]):
        return True
    if game[3]==mark and game[3] == game[5] and game[5]==game[7]:
        return True
    return pos

def is_draw():
    for key in game.keys():
        if (game[key] == ' '):
            return False
    return True

def player_turn():
    position = int(input("Enter the position"))
    insert(player, position)
    return

def Win():
    return is_winner(comp)|is_winner(player)

def computer():
    Move = 0
    MaxScore = -1000
    for key in game.keys():
        if (game[key] == ' '):
            game[key] = comp
            score = find_score(game, 0, False)
            game[key] = ' '
            if (score > MaxScore):
                MaxScore = score
                Move = key

    insert(comp, Move)
    return

```

```

def find_score(game, h, is_bot):
    if (is_winner(comp)):
        return 1000
    elif (is_winner(player)):
        return -1000
    elif (is_draw()):
        return 0
    if (is_bot):
        bestScore = -1000
        for key in game.keys():
            if (game[key] == ' '):
                game[key] = comp
                score = find_score(game, h + 1, False)
                game[key] = ' '
                if (score > bestScore):
                    bestScore = score
        return bestScore

    else:
        bestScore = 1000
        for key in game.keys():
            if (game[key] == ' '):
                game[key] = player
                score = find_score(game, h + 1, True)
                game[key] = ' '
                if (score < bestScore):
                    bestScore = score
        return bestScore

def show_layout():
    print("This is the grid layout")
    print("1, 2, 3 ")
    print("4, 5, 6 ")
    print("7, 8, 9 ")
    print("\n")

if __name__=="__main__":
    Display(game)
    show_layout()
    print("Your turn")
    player = 'X'
    comp = 'O'
    while not Win():
        player_turn()
        computer()

```

Output:

```
PS C:\Users\Samkit\Desktop\College\sem5\AIML\Lab 3> python tictactoe.py
```

```
| |  
-+-+  
| |  
-+-+  
| |
```

This is the grid layout

1, 2, 3

4, 5, 6

7, 8, 9

Your turn

Enter the position5

```
| |  
-+-+  
|x|  
-+-+  
| |
```

.

O| |

```
-+-+  
|x|  
-+-+  
| |
```

Enter the position3

O| |x

```
-+-+  
|x|  
-+-+  
| |
```

O| |x

```
-+-+  
|x|  
-+-+  
O| |
```

Enter the position4

o| |x

-+-+-

x|x|

-+-+-

o| |

o| |x

-+-+-

x|x|o

-+-+-

o| |

Enter the position2

o|x|x

-+-+-

x|x|o

-+-+-

o| |

o|x|x

-+-+-

x|x|o

-+-+-

o|o|

Enter the position9

o|x|x

-+-+-

x|x|o

-+-+-

o|o|x

Draw!

Conclusion:

The goal of this project was to implement the tic-tac-toe game using the informed search technique. To determine the best location for the 'X' or 'O,' I first calculated the difference between the winning combinations of bot(O) and user(X) for each choice in that round, then calculated which choice would not lead to computer victory and which would lead to user victory using the maximum values obtained by comparing the number of moves required for victory for each choice and selecting the one with the fewest moves. If there are several equivalent movements, the piece is put at random.