```
def print_board(board):
    print("    0   1   2")
    print("  +---+---+---+")
    for i, row in enumerate(board):
        print(f"{i} | " + " | ".join(row) + " |")
        print("  +---+---+---+")
    print()


def check_winner(board):
    for i in range(3):
        if board[i][0] == board[i][1] == board[i][2] != ' ':
            return board[i][0]
        if board[0][i] == board[1][i] == board[2][i] != ' ':
            return board[0][i]
    if board[0][0] == board[1][1] == board[2][2] != ' ':
        return board[0][0]
    if board[0][2] == board[1][1] == board[2][0] != ' ':
        return board[0][2]
    if all(board[i][j] != ' ' for i in range(3) for j in range(3)):
        return 'Draw'
    return None


def minimax(board, depth, is_max, alpha, beta):
    winner = check_winner(board)
    if winner == 'X':
        return -10 + depth
    elif winner == 'O':
        return 10 - depth
    elif winner == 'Draw':
        return 0

    if is_max:
        max_eval = float('-inf')
        for i in range(3):
            for j in range(3):
                if board[i][j] == ' ':
                    board[i][j] = 'O'
                    eval = minimax(board, depth + 1, False, alpha, beta)
                    board[i][j] = ' '
                    max_eval = max(max_eval, eval)
                    alpha = max(alpha, eval)
                    if beta <= alpha:
```

```
                    break
        return max_eval
    else:
        min_eval = float('inf')
        for i in range(3):
            for j in range(3):
                if board[i][j] == ' ':
                    board[i][j] = 'X'
                    eval = minimax(board, depth + 1, True, alpha, beta)
                    board[i][j] = ' '
                    min_eval = min(min_eval, eval)
                    beta = min(beta, eval)
                    if beta <= alpha:
                        break
        return min_eval


def find_best_move(board):
    best_val = float('-inf')
    move = (-1, -1)
    for i in range(3):
        for j in range(3):
            if board[i][j] == ' ':
                board[i][j] = 'O'
                move_val = minimax(board, 0, False, float('-inf'), float('inf'))
                board[i][j] = ' '
                if move_val > best_val:
                    best_val = move_val
                    move = (i, j)
    return move


def play_game():
    board = [[' ' for _ in range(3)] for _ in range(3)]
    print("Tic-Tac-Toe: You are X, AI is O")
    print_board(board)

    while True:
        while True:
            try:
                row = int(input("Enter your row (0-2): "))
                col = int(input("Enter your column (0-2): "))
                if board[row][col] == ' ':
                    board[row][col] = 'X'
```

```
                break
            else:
                print("Cell already taken!")
        except:
            print("Invalid input, try again.")
    print_board(board)

    winner = check_winner(board)
    if winner:
        if winner == 'Draw':
            print("The game is a draw!")
        else:
            print(f"{winner} wins the game!")
        break

    print("AI is making a move...")
    row, col = find_best_move(board)
    board[row][col] = 'O'
    print_board(board)

    winner = check_winner(board)
    if winner:
        if winner == 'Draw':
            print("The game is a draw!")
        else:
            print(f"{winner} wins the game!")
        break


def main_menu():
    while True:
        print("\n===== TIC-TAC-TOE MENU =====")
        print("1. Play Game")
        print("2. Exit")
        choice = input("Enter your choice: ")

        if choice == '1':
            play_game()
        elif choice == '2':
            print("Exiting program successfully...")
            break
        else:
            print("Invalid choice! Try again.")
```

main_menu()