```
import random


# Game Board Representation
board = [['-' for _ in range(3)] for _ in range(3)]


# Function to print the game board
def print_board():
  for row in board:
      print(' | '.join(row))
      print('---------')


# Function to check if the game is over
def game_over():
  # Check rows and columns
  for i in range(3):
      if board[i] == board[i] == board[i] != '-':
          return board[i]
      if board[i] == board[i] == board[i] != '-':
          return board[i]
  # Check diagonals
  if board == board == board != '-':
      return board
  if board == board == board != '-':
      return board
  # Check if the board is full
  if all(cell != '-' for row in board for cell in row):
      return 'Tie'
  return False
```

```
# Function to evaluate the board
def evaluate_board():
  result = game_over()
  if result == 'X':
     return 1
  elif result == 'O':
     return -1
  elif result == 'Tie':
     return 0
  else:
     return 0


# Minimax Algorithm with Alpha-Beta Pruning
def minimax(board, depth, is_maximizing, alpha, beta):
  result = game_over()
  if result:
     return evaluate_board()
  if is_maximizing:
     best_score = -float('inf')
     for i in range(3):
        for j in range(3):
           if board[i][j] == '-':
              board[i][j] = 'X'
              score = minimax(board, depth + 1, False, alpha, beta)
              board[i][j] = '-'
              best_score = max(score, best_score)
              alpha = max(alpha, best_score)
              if beta <= alpha:
                 break
     return best_score
  else:
     best_score = float('inf')
```

```
    for i in range(3):
        for j in range(3):
            if board[i][j] == '-':
                board[i][j] = 'O'
                score = minimax(board, depth + 1, True, alpha, beta)
                board[i][j] = '-'
                best_score = min(score, best_score)
                beta = min(beta, best_score)
                if beta <= alpha:
                    break
    return best_score


# Function to get the best move using Minimax
def get_best_move():
  best_score = -float('inf')
  best_move = (0, 0)
  for i in range(3):
      for j in range(3):
          if board[i][j] == '-':
              board[i][j] = 'X'
              score = minimax(board, 0, False, -float('inf'), float('inf'))
              board[i][j] = '-'
              if score > best_score:
                  best_score = score
                  best_move = (i, j)
  return best_move


# Main Game Loop
def play_game():
  while True:
      print_board()
```

```
    # User's turn
    move = input("Enter your move (row and column, e.g., 0 0): ")
    row, col = map(int, move.split())
    if board[row][col] != '-':
        print("Invalid move, try again.")
        continue
    board[row][col] = 'O'
    result = game_over()
    if result:
        print_board()
        if result == 'O':
            print("You win!")
        elif result == 'X':
            print("AI wins!")
        else:
            print("It's a tie!")
        break
    # AI's turn
    row, col = get_best_move()
    board[row][col] = 'X'
    result = game_over()
    if result:
        print_board()
        if result == 'O':
            print("You win!")
        elif result == 'X':
            print("AI wins!")
        else:
            print("It's a tie!")
        break


# Start the game
```

play_game()