ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOACÔNG NGHỆ THÔNG TIN

○♦♦♦ 🕮 ♦♦♦○

PROJECT 1 – SEARCH

PLAYING GAME WITH ADVERSARIAL SEARCH

Môn học: Cơ sở trí tuệ nhân tạo

GIÁO VIÊN HƯỚNG DẪN
TS. Nguyễn Hải Minh
TS. Nguyễn Ngọc Thảo
Trợ giảng. Nguyễn Thái Vũ

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOACÔNG NGHỆ THÔNG TIN

○♦♦♦ 🕮 ♦♦♦○

PROJECT 1 – SEARCH

PLAYING GAME WITH ADVERSARIAL SEARCH

Môn học: Cơ sở trí tuệ nhân tạo

SINH VIÊN THỰC HIỆN

20127662 - Nguyễn Đình Văn
20127166 – Nguyễn Huy Hoàn
20127061 – Lưu Minh Phát

MỤC LỤC

MỤC LỤC.....1
THÔNG TIN THÀNH VIÊN.....2
MỨC ĐỘ HOÀN THÀNH.....2
GIỚI THIỆU Ý TƯỞNG VÀHÀM TRONG ĐỒ ÁN.....3
MINH CHỨNG VÀ ĐỘ PHỨC TẠP.....5
TÀI LIỆU THAM KHẢO 8

MỤC LỤC HÌNH ẢNH

Hình 1: Board tic-tac-toe kích thước 3x3 5
Hình 2: Board tic-tac-toe kích thước 5x5 6
Hình 3: Board tic-tac-toe kích thước 7x7 7

THÔNG TIN THÀNH VIÊN

Mã số sinh viên	Họ và tên	Nhiệm vụ
20127662	Nguyễn Đình Văn	Design all size boards: 3x3, 5x5, 7x7 Design figures: ‘X’ & ‘O’ Design figures for selection
20127166	Nguyễn Huy Hoàn	Code minimax algorithm for tic-tac-toe AI Make video Check bug
20127061	Lưu Minh Phát	Make all detect, select(AI-Human, size, type), announce Code alpha-beta prunning algorithm for tic-tac-toe AI Report

MỨC ĐỘ HOÀN THÀNH

STT	Tên chức năng	Mức độ hoàn thành	Ghi chú
1	Size board 3x3, 5x5, 7x7	100 %	
2	Figures design	100 %	
3	Selects	100 %	
4	All detections	100 %	
5	Minimax	100 %	
6	Alpha-Beta prunning	100 %	

Tổng kết mức độ hoàn thành: 100%

Nhận xét:

- Thay đổi được các kích thước của board
- Thay đổi được loại adversarial search cho AI
- Thay đổi được thứ tự người chơi được đi trước
- Đánh dấu được chính xác figure ‘X’ hoặc ‘O’ tại vị trí mong muốn
- Các thuật toán tìm kiếm chạy bình thường, tốt, tuy khá chậm với board 7x7 cả minimax và alpha-beta prunning

GIỚI THIỆU Ý TƯỞNG VÀHÀM TRONG ĐỒ ÁN

- **Ngôn ngữ**: Python
- **Các thư viện sử dụng**: pygame, sys, random, math, time
- **Ý tưởng**: Cho AI chạy thử toàn bộ các bước tiếp theo sau đó tính toán chi phí lựa chọn ra chi phí tốt nhất thật sự để AI thực hiện, mỗi lần chạy thuật toán tìm kiếm cũng kiểm tra xem bước đi đó đã kết thúc chưa và phần thắng thuộc về ai. Công thức để tính được giá trị của từng bước em xin tham khảo nguồn [1]
  - Nếu là maximizing: nếu move dẫn đến chiến thắng thì trả về dương vô cực; nếu thua thì trừ vô cực
  - Nếu là minimizing: nếu move dẫn đến chiến thắng thì trả về trừ vô cực; thua thì dương vô cực
  - Trong alpha-beta prunning có kiểm tra giữa alpha và beta để giảm thiểu các bước
- **Các Hàm**:
  - *Về giao diện: kích thước khung trò chơi là 700x700, board được tạo bởi cái hình vuông tương ứng là 1 khung*
    - drawRectangles(win, size_board, mouse, width, xc, yc): để vẽ các hình vuông đại diện như 1 khung của 1 board (3x3: 9 recs; 5x5: 25 recs; 7x7: 49 recs)
    - (win, size_board, board, xc, yc, width): để vẽ 'X' và 'O'
    - drawUI(win, circle_position, circle_position_turn, circle_type_search, triumph): để thiết kế giao diện tương tác người chơi, cho phép chọn kích thước của board (3x3 / 5x5 / 7x7), hoặc chọn ai là người được đi đầu tiên, hoặc adversarial search mà AI sẽ sử dụng
    - squareDetection(mouse): hàm để kiểm tra vùng đặt của chuột để đặt figure
    - setting(mouse): để lấy và kiểm tra vị trí đặt chuột để điều chỉnh lựa chọn

  - *Về thuật toán tìm kiếm*
    - minimax(maximizing, depth, alpha, beta)
    - alp_bet_prunning(maximizing, depth, alpha, beta)

  ➔ 2 thuật toán gần như giống nhau tuy nhiên alp_bet_prunning() lại có thêm vài dòng để kiểm tra giá trị alpha và beta để prune
    - evaluation(): để tính toán giá trị, chi phí của từng vị trí trong bảng. Công thức tính tham khảo từ nguồn [1] vì công thức này khá hợp lý và nó có thể dùng được, cách trình bày tham khảo từ nguồn [2]

      Ngắn gọn về công thức có thể như sau:

      Nếu là player 1 đặt ở giữa thì trả về chi phí cao

      eval += (1 / ($\sqrt{center - row} * 2 + \sqrt{center - col} * 2$)+1)

      Nếu row = col hoặc size-1-row = col hoặc size-1-col = row thì eval += 0.15

      Nếu player 2 đặt giữa thì trả về chi phí thấp

      eval -= (1 / ($\sqrt{center - row} * 2 + \sqrt{center - col} * 2$)+1)

      Nếu row = col hoặc size-1-row = col hoặc size-1-col = row thì eval -= 0.15

    ➔ Đi giữa càng có cơ hội thắng
    - victorycheck() : kiểm tra xem đã thắng hay chưa. Chia thành 4 loại tham khảo từ cả [1] và [2]
      - Theo chiều ngang '–' : đủ triumph figure liên tiếp nằm ngang

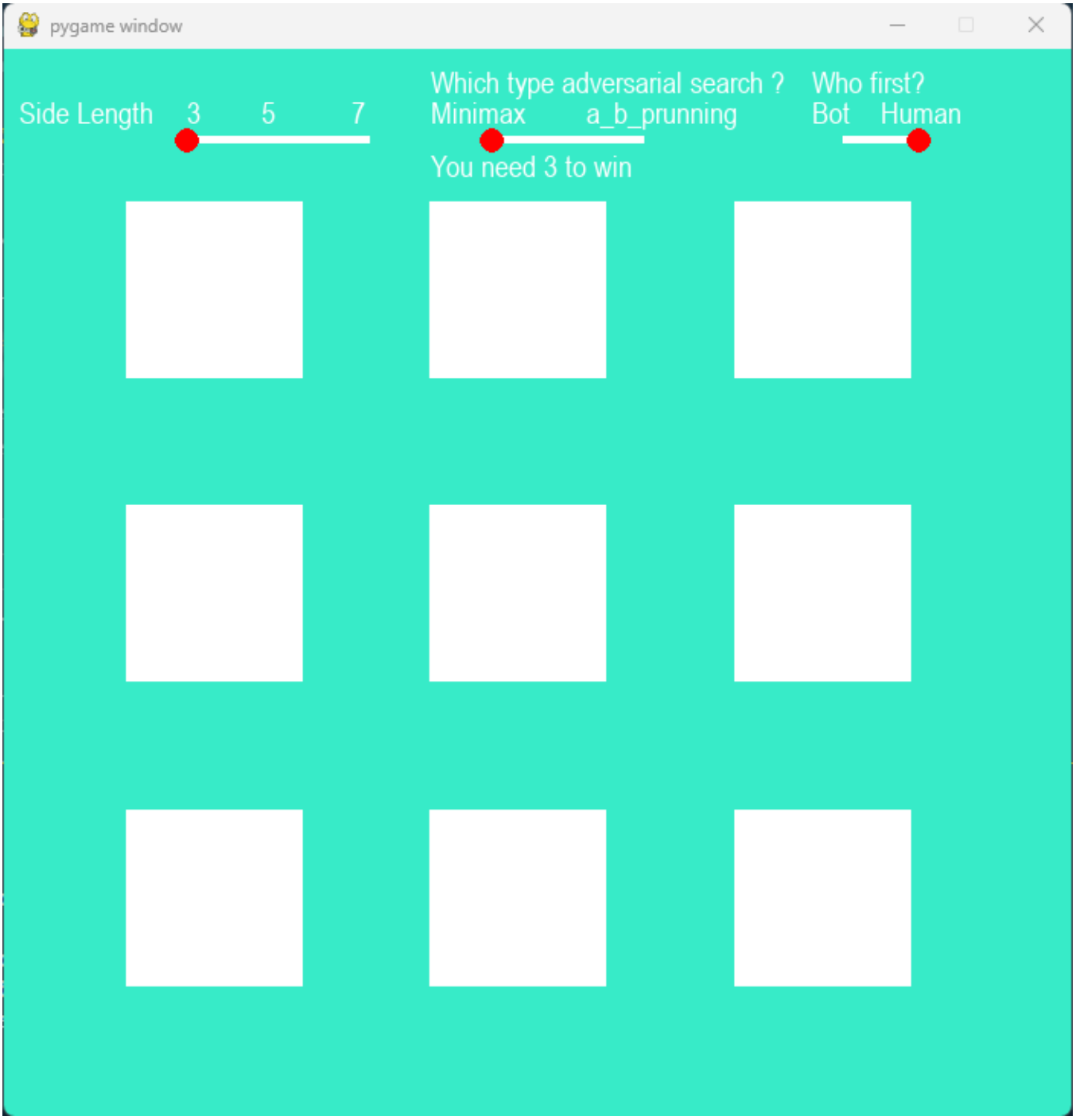
(horizontal)
    - Theo chiều dọc '|' : đủ triumph figure liên tiếp thẳng đứng (vertical)
    - Theo đường chéo chính '\' (chéo xuống từ trái sang phải) – đủ triumph figure liên tiếp theo đường chéo giảm (desc diagonal)
    - Theo đường chéo phụ '/' (chéo lên từ trái sang phải): đủ triumph figure liên tiếp theo đường chéo tăng (asc diagonal)
- doMove(move): để AI đi bước tiếp theo
- undoMove(move): nếu giá trị không hợp lệ hay không thích hợp nhất hay đi lại thì đây là tác dụng của hàm này
- possibleMoves(move): kiểm tra xem đã có figure ở bước đi này chưa
- presortingPossibleMoves(moveList): để sắp xếp lại cái bước đi có chi phí tốt nhất, có lợi nhất

- *Thông tin một số biến*
  - circle_position: vị trí của nút chỉnh size board
  - circle_pposition_turn: vị trí của nút chỉnh người đi đầu tiên (AI – Human)
  - circle_type_search: vị trí nút chỉnh loại adversarial search mà AI dùng để tìm kiếm
  - move, moveList: bước mà AI sẽ đi và danh sách các bước AI thử đi
  - width: là độ rộng của 1 khung trong board
  - Size_board: kích thước của bảng tic-tac-toe (3x3 / 5x5 / 7x7)
  - Triump: số lượng figures cần thiết liên tiếp để giành chiến thắng. 3x3 triumph = 3; 5x5 triumph = 4; 7x7 triumph = 5

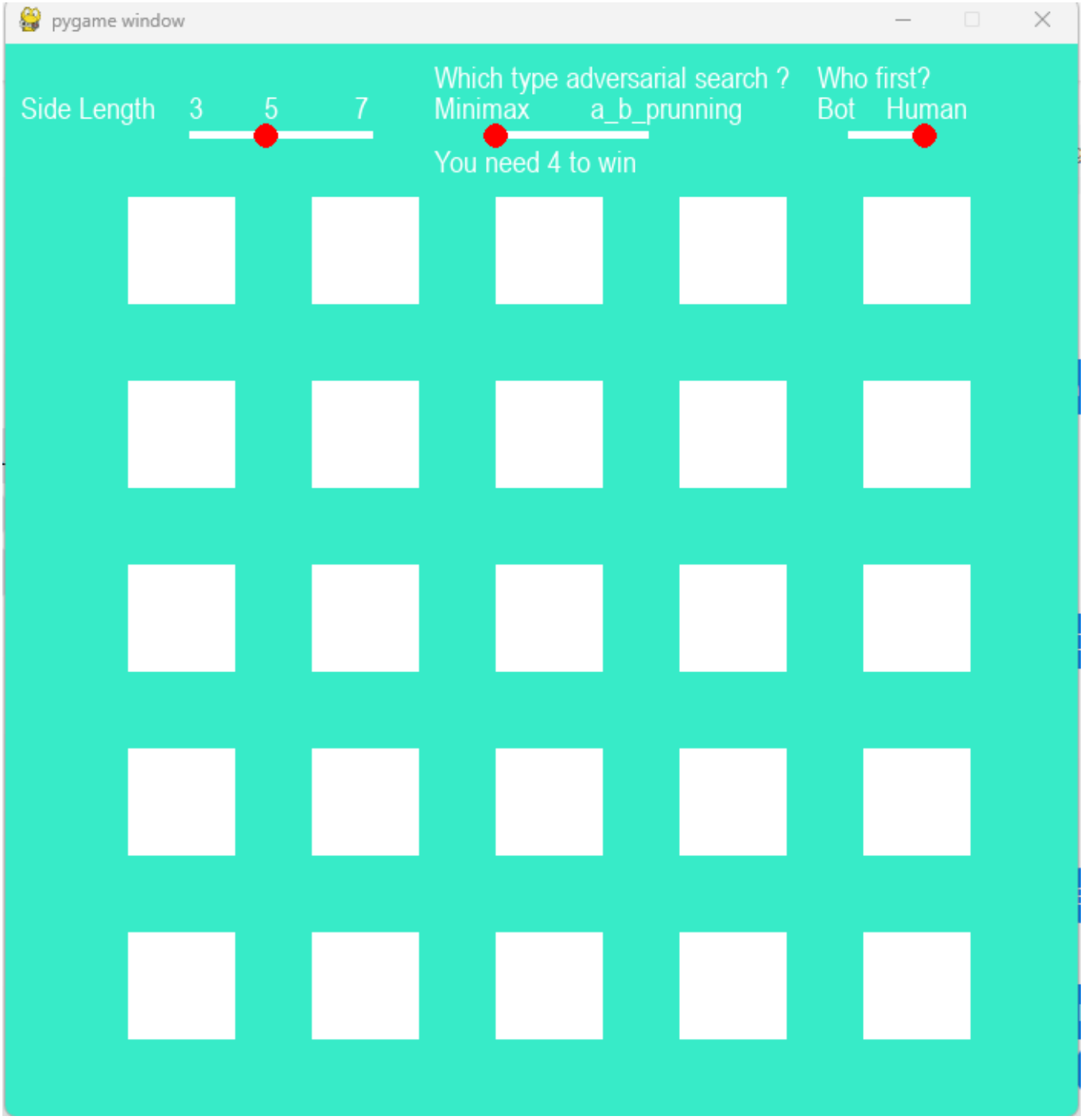
  ➔ triumph = size_board / 2 + 1.77 [1]

MINH CHỨNG, ĐỘ PHỨC TẠP VÀ LINK VIDEO DEMO

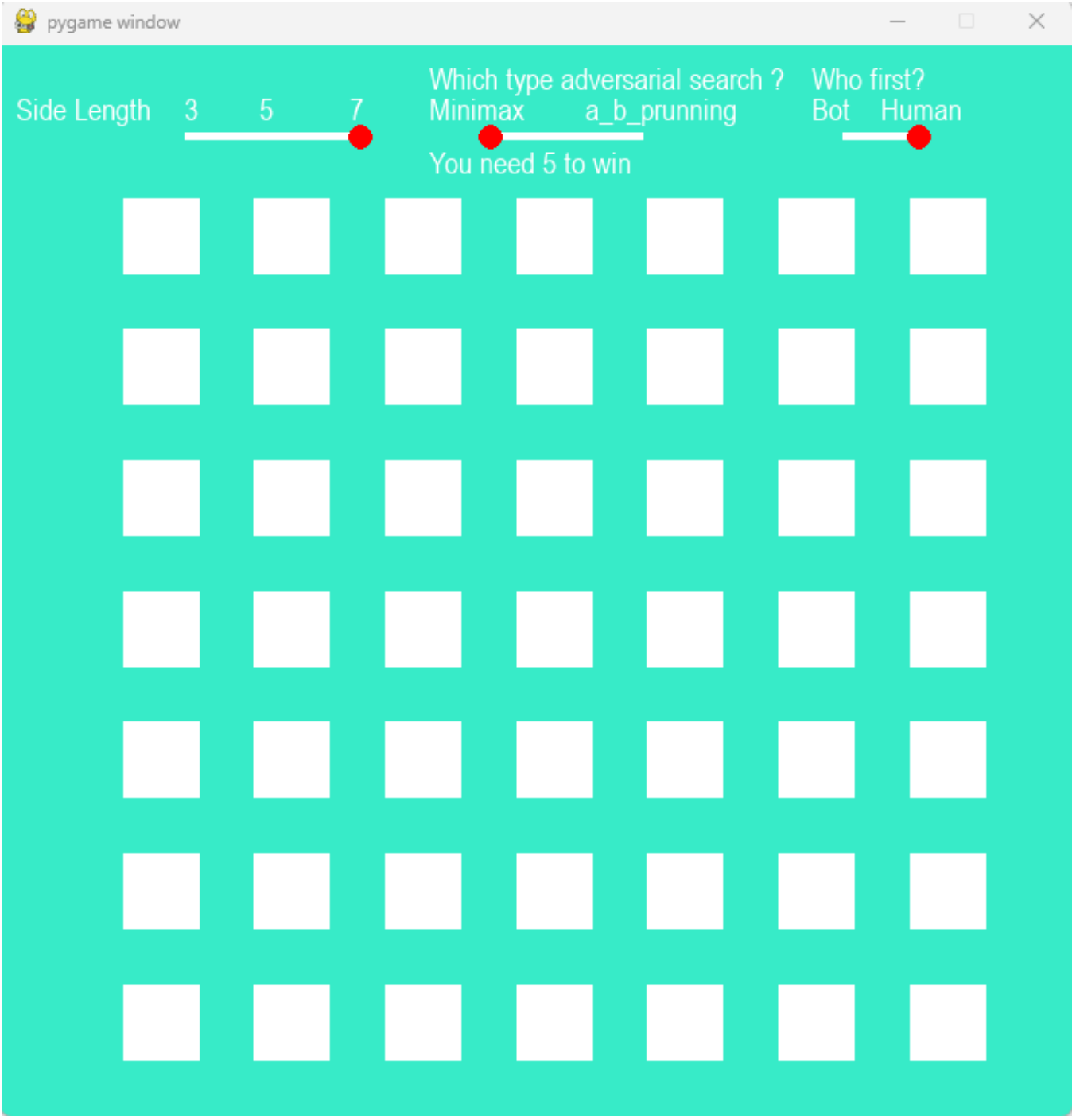
Thiết kế tham khảo từ nguồn [1] và [2]



Hình 1: Board tic-tac-toe kích thước 3x3



Hình 2: Board tic-tac-toe kích thước 5x5



Hình 3: Board tic-tac-toe kích thước 7x7

Time Complexity: $O(b^m)$
Space Complexity: O(b m)
Trong đó:

b: hệ số phân nhánh của cây (length of moveList)
m: là độ sâu tối đa của cây (depth)

Linh video demo: https://drive.google.com/drive/folders/1uMWANbOWw6VbudOhCNRIV-xl_RVBiU-C?usp=sharing

TÀI LIỆU THAM KHẢO

[1] Anthuril12, "https://github.com/," 31 10 2021. [Online]. Available: https://github.com/Anthuril12/Tic-Tac-Toe-Bot?fbclid=IwAR0inOxOn2gn7f_EOC4a8pRxot0vvIYUvM9FJeLSqAUfrq8lclJbiG_odHI.

[2] "https://www.youtube.com," 21 1 2022. [Online]. Available: https://www.youtube.com/watch?v=Bk9hlNZc6sE&t=468s.