NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

Trà Vinh, ngày ..... tháng …… năm ……

Giáo viên hướng dẫn

(Ký tên và ghi rõ họ tên)

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

Trà Vinh, ngày ..... tháng …… năm ……

Thành viên hội đồng

(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Trước hết, em cũng xin gửi lời cảm ơn đến Thầy Nguyễn Bảo Ân đã hướng dẫn, giúp đỡ, góp ý và giải đáp những thắc mắc trong suốt quá trình thực hiện để em có thể hoàn thiện đề tài.

Em cũng xin chân thành cảm ơn các Thầy, Cô trong Bộ môn Công nghệ thông tin đã giảng dạy và trang bị cho em những kiến thức và kỹ năng cơ bản để có nền tảng thực hiện đề tài.

Cuối cùng, em xin gửi đến quý Thầy, Cô lời chúc sức khỏe, hạnh phúc và thành công trong sự nghiệp giảng dạy.

Em xin chân thành cảm ơn!

Trà Vinh, ngày .... tháng .... năm ....

Sinh viên thực hiện

Nguyễn Phước Hiệp

MỤC LỤC

DANH MỤC HÌNH ẢNH – BẢNG BIỂU

DANH MỤC TỪ VIẾT TẮT

STT	Từ viết tắt	Từ gốc
1	CSS	Cascading Style Sheets
2	HTML	Hyper Text Markup Language
3	BFS	Breadth First Search

TÓM TẮT ĐỒ ÁN CƠ SỞ NGÀNH

1. Vấn đề nghiên cứu

Phát triển game Lines trên nền tảng web bằng các công cụ như HTML, CSS, JavaScript và sử dụng cử chỉ tay để tương tác với trò chơi. Đồ án hướng đến việc cải tiến game Lines mà người chơi điều khiển bằng chuột theo một cách mới là điều khiển bằng cử chỉ tay để cải thiện tương tác, tạo trải nghiệm mới cho người chơi.

2. Các hướng tiếp cận

Đầu tiên, sử dụng HTML, CSS và JavaScript để xây dựng giao diện và logic của trò chơi.

Tiếp theo, tìm hiểu về thư viện MediaPipe Hands và thiết lập cử chỉ để thao tác với trò chơi. Sau đó, tập trung phát triển thuật toán điều khiển và cơ chế chơi bằng cử chỉ tay cho phù hợp như di chuyển bóng, chọn bóng.

Cuối cùng, tích hợp thêm một số hiệu ứng và cai thiện giao diện để tăng sự hấp dẫn của trò chơi.

3. Cách giải quyết vấn đề

Để xây dựng game Lines trên nền tảng web, đầu tiên phải xây dựng các chức năng cơ bản như tạo bảng, di chuyển bóng, ghi điểm khi bóng nổ và phát sinh bóng mới sau mỗi lần di chuyển không ghi điểm.

Đối với thuật toán di chuyển bóng, nghiên cứu tài liệu về thuật toán tìm đường đi BFS để thực hiện.

Tiếp đến, sử dụng thư viện MediaPipe để xử lý hình ảnh nhận được từ webcam, xây dựng các logic điều kiện cử chỉ của tay để tương tác với trò chơi như: chỉ giơ ngón trỏ để di chuyển, giơ đồng thời ngón trỏ và ngón cái để cố định vị trí chọn, giơ ngón cái và ngón trỏ co lại để chọn ngay vị trí đó.

4. Một số kết quả đạt được

Thiết kế thành công game Lines trên nền tảng web.

Ứng dụng thành công thư viện MediaPipe để nhận diện cử chỉ tay và tương tác với trò chơi.

Thiết kế thành công giao diện game dễ nhìn, trò chơi chạy ổn định.

MỞ ĐẦU

1. Lý do chọn đề tài

Trong thời đại công nghệ số phát triển nhanh chóng, các công nghệ tiên tiến như nhận diện cử chỉ tay, nhận diện khuôn mặt, và theo dõi chuyển động cơ thể đang ngày càng được ứng dụng rộng rãi trong nhiều lĩnh vực. Những công nghệ này không chỉ mang lại những giải pháp không chạm mà còn giúp tối ưu hóa trải nghiệm người dùng, tạo ra sự thuận tiện và tương tác mượt mà hơn giữa con người và máy móc. Việc áp dụng công nghệ nhận diện cử chỉ tay vào các trò chơi, đặc biệt là các trò chơi trên nền tảng web, đang trở thành một xu hướng hấp dẫn và đầy tiềm năng.

Game Lines là một trò chơi giải đố cổ điển, nổi bật với lối chơi đơn giản nhưng đầy thử thách. Truyền thống, game Lines chỉ có thể chơi được bằng điều khiển thông qua chuột, nhưng việc sử dụng các phương thức này dần tạo ra cảm giác nhàm chán và thiếu sự đổi mới. Vì vậy, phát triển game Lines trên nền tảng web với tính năng điều khiển bằng cử chỉ tay bằng cách sử dụng công nghệ MediaPipe để nhận diện cử chỉ trong thời gian thực. Việc này không chỉ làm mới cách chơi của trò chơi quen thuộc mà còn mang đến một trải nghiệm tương tác thú vị, độc đáo và đầy sáng tạo.

2. Mục đích

Đổi mới cách chơi game Lines cổ điển thông qua việc tích hợp công nghệ nhận diện cử chỉ tay, tạo nên một trải nghiệm điều khiển không dùng chuột.

Tận dụng công nghệ MediaPipe để phát triển khả năng nhận diện cử chỉ tay thời gian thực, giúp việc điều khiển trò chơi trên nền tảng web trở nên linh hoạt và hiệu quả.

Thiết kế một sản phẩm hoàn chỉnh, kết hợp sự sáng tạo với khả năng ứng dụng thực tế, mở ra tiềm năng lớn trong lĩnh vực giải trí và công nghệ tương tác.

3. Đối tượng nghiên cứu

HTML, CSS, JS: Cách tạo nên một trang web trò chơi và xử lí các sự kiện trò chơi.

Công nghệ MediaPipe: Tập trung tìm hiểu cách nhận diện cử chỉ tay và tích hợp vào trò chơi chạy trên web.

Game Lines: Các cơ chế chơi, luật chơi, và thuật toán liên quan đến việc tạo bóng ngẫu nhiên, di chuyển, và tính điểm.

4. Phạm vi nghiên cứu

Game Lines sẽ được xây dựng hoàn toàn trên nền tảng web, sử dụng các công nghệ cốt lõi như HTML, CSS và JavaScript để thiết kế giao diện người dùng và xử lý logic trò chơi.

Phát triển game Lines trên nền tảng web với việc tích hợp công nghệ MediaPipe, thiết kế giao diện và xây dựng logic trò chơi thích hợp với tương tác bằng cử chỉ tay.

CHƯƠNG 1: TỔNG QUAN

Với sự phát triển vượt bậc của công nghệ, các cách thức tương tác giữa con người và máy tính liên tục được cải tiến. Từ những thiết bị truyền thống như bàn phím và chuột, công nghệ hiện đại đã chuyển hướng sang các phương thức tương tác không chạm, bao gồm nhận diện cử chỉ, nhận diện khuôn mặt và theo dõi chuyển động cơ thể. Những bước tiến này không chỉ nâng cao sự tiện lợi mà còn mang đến nhiều trải nghiệm độc đáo và sáng tạo trong các lĩnh vực như y tế, giáo dục, và đặc biệt là giải trí.

Game Lines là một trò chơi giải trí cổ điển, đây là một trò chơi đơn giản nhưng cần sự tập trung cao nhờ vào lối chơi chiến thuật, yêu cầu người chơi phải tư duy và lên kế hoạch di chuyển hiệu quả. Trò chơi đã xuất hiện từ lâu và nhận được sự yêu thích của nhiều thế hệ. Tuy nhiên, với sự phát triển nhanh chóng của công nghệ, đặc biệt là trong lĩnh vực phát triển ứng dụng web và trí tuệ nhân tạo, việc đổi mới cách thức người chơi tương tác với game trở thành một hướng nghiên cứu đầy tiềm năng.

Đề tài này hướng đến việc phát triển phiên bản khác của game Lines trên nền tảng web, kết hợp với công nghệ MediaPipe để mang lại trải nghiệm mới mẻ cho người chơi. MediaPipe là một thư viện mã nguồn mở của Google, cung cấp các giải pháp AI như nhận diện bàn tay, khuôn mặt, cử chỉ và chuyển động cơ thể,... Việc tích hợp MediaPipe vào trò chơi không chỉ mang lại sự tương tác tự nhiên và linh hoạt mà còn mở ra cơ hội khám phá những ứng dụng của AI trong lĩnh vực giải trí, giáo dục và các lĩnh vực khác.

Trình tự thực hiện đồ án

Bước 1: Tìm hiểu game Lines và MediaPipe

Tìm hiểu về luật chơi, cơ chế trò chơi của game Lines.

Tìm hiểu về MediaPipe, nghiên cứu tập trung về MediaPipe Hands dùng để nhận diện cử chỉ tay.

Tìm hiểu về HTML, CSS, JavaScript, một số cách thức làm việc với sự kiện trong JavaScript.

Bước 2: Xây dựng trò chơi trên web

Thiết kế giao diện trò chơi

Sử dụng HTML, CSS để xây dựng giao diện nhập tên và giao diện chính của game.

Sử dụng JavaScript để xử lí các sự kiện để tương tác với người chơi, thiết kế các logic của game như tạo bóng và bóng dự đoán ngẫu nhiên, di chuyển bóng, cập nhật điểm, kết thúc trò chơi.

Sử dụng thuật toán tìm đường đi BFS để thực hiện việc di chuyển bóng giữa hai điểm bất kì.

Sử dụng Realtime Database của Firebase để minh họa một bảng xếp hạng.

Cài đặt và ứng dụng thư viện MediaPipe để xử lí cử chỉ tay, từ đó đặt ra điều kiện cho cử chỉ tay để người chơi tương tác với trò chơi.

Bước 3: Kiểm tra, tích hợp thêm các tính năng bổ sung

Kiểm thử độ chính xác của các tính năng được điều khiển bằng cử chỉ tay.

Cải thiện lại giao diện, thêm âm thanh và bảng xếp hạng.

Bước 4: Báo cáo kết quả thực hiện

Hoàn chỉnh quyển báo cáo, slide thuyết trình, poster và chuẩn bị sẵn sàng để báo cáo.

CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

2.1 Giới thiệu các công nghệ dùng để thiết kế trò chơi.

2.1.1 Giới thiệu về HTML

HTML là một ngôn ngữ đánh dấu siêu văn bản, thường được sử dụng để tạo và cấu trúc các phần trong trang web và ứng dụng. HTML tổ chức nội dung thành các phần tử như tiêu đề, đoạn văn, hình ảnh, liên kết hoặc các thành phần khác để cho trình duyệt có thể hiển thị rõ ràng.

Cấu trúc cơ bản của HTML gồm các thẻ (tags) và thuộc tính (attributes). Các thẻ (tags) giúp trình duyệt có thể hiểu và hiển thị các phần tử ra trang web như thế nào. Các thuộc tính (attributes) dùng để định kiểu cho thẻ (tags) hiển thị ra trang web, nó có thể được viết trực tiếp trong thẻ hoặc viết ở một file riêng được liên kết đến trang web.

HTML không phải là một ngôn ngữ lập trình mà chỉ là một ngôn ngữ đánh dấu siêu văn bản. Do đó, HTML chỉ có thể hiển thị nội dung đã được cố định trong mã, nội dung bên trong trang HTML sẽ không thay đổi cho dù có thực hiện hành động nhấp chuột, nhập liệu hay các hành động khác.

Trong game Lines, HTML dùng để thiết kế các thành phần giao diện chính cho trò chơi như bố cục, bảng chơi, điểm số và các nút cần thiết.

2.1.2 Giới thiệu về CSS

CSS là ngôn ngữ dùng để định kiểu và trình bày nội dung trên trang web, giúp làm cho trang web trở nên sinh động, hấp dẫn thông qua việc điều chỉnh cách các phần tử HTML được hiển thị. Nó cho phép dùng các thuộc tính để tùy chỉnh các phần tử HTML như màu sắc, phông chữ, cỡ chữ, hiệu ứng động, kích thước, vị trí và các thuộc tính khác nhằm tạo ra giao diện đẹp mắt và dễ sử dụng.

CSS hoạt động dựa trên việc chọn phần tử HTML dựa trên các thuộc tính như lớp, ID và áp dụng các kiểu định dạng cho chúng. Ngoài ra, CSS hỗ trợ phân tách giữa nội dung và giao diện nên việc thay đổi và tối ưu hóa thiết kế trở nên dễ dàng hơn.

Ngoài ra, CSS còn có ưu điểm là phân tách giữa các thẻ HTML và các thuộc tính CSS, điều đó giúp cho việc thay đổi và tối ưu hóa trở nên dễ dàng. Đồng thời giúp tối ưu hóa mã nguồn, giảm thiểu sự trùng lặp trong việc định dạng.

Trong game Lines, CSS được sử dụng để định dạng và tạo kiểu cho các thành phần giao diện được xây dựng bằng HTML. CSS giúp trò chơi trở nên trực quan, hấp dẫn và dễ sử dụng hơn.

2.1.3 Giới thiệu về JavaScript

JavaScript là ngôn ngữ lập trình phổ biến và mạnh mẽ, được sử dụng rộng rãi trong phát triển website hiện nay. Nó được tích hợp trực tiếp vào mã HTML và giúp làm cho các trang web trở nên sống động và tương tác hơn. JavaScript đóng vai trò quan trọng trong việc tạo ra các trang web động, thực thi mã lệnh ngay trên trình duyệt của người dùng (client-side) và có thể sử dụng cả phía máy chủ với nền tảng như Node.js.

Nhờ vào JavaScript, website không chỉ đơn thuần là một tài liệu tĩnh mà còn có thể phản hồi nhanh chóng với người dùng thông qua các chức năng tương tác được thiết lập. Các trang web có thể trở nên sống động hơn nhờ vào việc xử lý sự kiện, giúp trang web phản hồi ngay lập tức khi người dùng thực hiện các hành động như nhấp chuột, di chuột, nhập liệu.

JavaScript còn có thể thay đổi nội dung động và giao tiếp với máy chủ mà không cần phải tải tại trang, điều này giúp tạo ra một trải nghiệm mượt mà cho người dùng.

JavaScript còn hỗ trợ rất nhiều thư viện và framework giúp phát triển ứng dụng web một cách nhanh chóng và hiệu quả. JavaScript không chỉ cải thiện khả năng tương tác trên web mà còn mang lại tính linh hoạt, hiệu quả trong phát triển và khả năng mở rộng ứng dụng, giúp tiết kiệm rất nhiều thời gian.

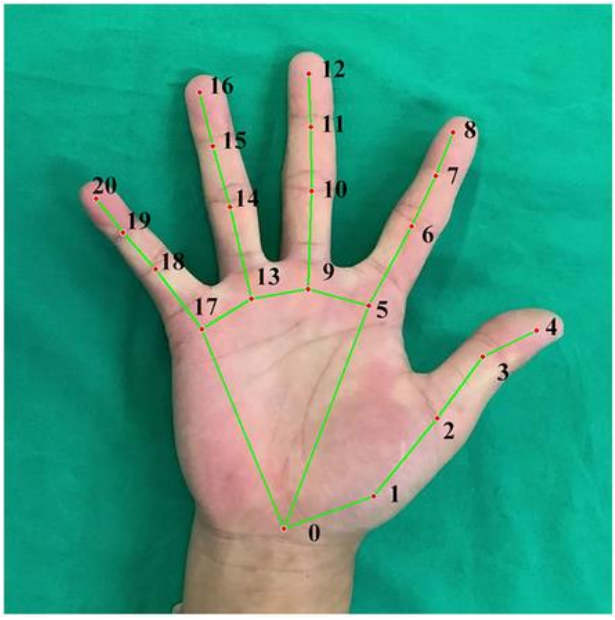
Trong game Lines, sử dụng JavaScript để làm việc với các sự kiện để tương tác với người chơi, thiết kế các logic của game như tạo bảng, tạo bóng, di chuyển bóng, ghi điểm và ứng dụng thư viện MediaPipe.

2.2 Thư viện MediaPipe

2.2.1 Giới thiệu về thư viện MediaPipe

Mediapipe là một thư viện mã nguồn mở của Google phát triển, được thiết kế để xử lý các tác vụ như nhận diện khuôn mặt, theo dõi bàn tay, phân tích dáng điệu, và nhiều ứng dụng khác trực tiếp trên trình duyệt web.

Với sự hỗ trợ của JavaScript, MediaPipe có thể được tích hợp trực tiếp vào các ứng dụng web, cho phép xử lý dữ liệu thời gian thực ngay trên trình duyệt mà không cần cài đặt thêm phần mềm. Điều này mở ra tiềm năng lớn cho việc xây dựng các ứng dụng tương tác, sử dụng các công nghệ AI một cách linh hoạt và dễ dàng.



Hình 1: Mô tả các điểm landmark của MediaPipe Hand
(nguồn Internet)

Thư viện MediaPipe sử dụng các landmarks dựa trên mô hình xương bàn tay để tạo thành một mô hình đơn giản, giúp xác định vị trí và chuyển động của bàn tay trong không gian. Các điểm được đánh số từ 0 đến 20 đại diện cho các khớp của bàn tay.

Ngoài ra, các đường nối giữa các điểm đặc trưng tạo thành một mô hình xương bàn tay trực quan, giúp công cụ này dễ dàng nhận diện cử chỉ tay và phân tích chuyển động. Đây là nền tảng cho nhiều ứng dụng hiện đại như điều khiển bằng cử chỉ, thực tế tăng cường (AR), thực tế ảo (VR), và các giao diện tương tác người-

máy. Khả năng chính xác cao và hiệu quả của MediaPipe Hands đã làm cho nó trở thành một công cụ phổ biến trong việc phát triển các ứng dụng thông minh liên quan đến nhận diện và xử lý hình ảnh bàn tay.

Trong game Lines, thư viện MediaPipe được sử dụng để có thể giúp người chơi tương tác với trò chơi bằng cử chỉ tay theo thời gian thực.

2.2.2 Cách cài đặt thư viện MediaPipe

Có hai cách phổ biến để cài đặt thư viện MediaPipe.

Cách đầu tiên là sử dụng "npm" nếu đang làm việc với dự án sử dụng Node.js. Chẳng hạn như để cài đặt module Hands thì chỉ cần gõ lệnh "npm install @mediapipe/hands" trong terminal.

Cách thứ hai có phần đơn giản hơn, chỉ cần nhúng trực tiếp đoạn mã "<script src="https://cdn.jsdelivr.net/npm/@mediapipe/hands"></script>" vào tệp HTML thì đã có thể sử dụng được module Hands của MediaPipe.

Có thể cài đặt thêm các module khác của MediaPipe tùy theo nhu cầu và mục đích sử dụng.

2.2.3 Cách sử dụng thư viện MediaPipe

Để sử dụng module Hands của thư viện MediaPipe, đầu tiên cần khởi tạo một đối tượng hands được hàm locateFile trả về các tệp cần thiết và các mô hình dư liệu liên quan.

```
const hands = new Hands({
  locateFile: (file) =>
    `https://cdn.jsdelivr.net/npm/@mediapipe/hands/${file}`,
});

hands.setOptions({
  maxNumHands: 1,
  modelComplexity: 1,
  minDetectionConfidence: 0.7,
  minTrackingConfidence: 0.5,
});
```

Hình 2: Mô tả cách sử dụng MediaPipe Hand

Tiếp đến sử dụng phương thức setOption(), phương thức này cho phép tùy chỉnh cấu hình hoạt động của mô hình Hands.

maxNumHands là số lượng bàn tay tối đa mà mô hình có thể phát hiện trong một khung hình.

modelComplexity là một tham số dùng để điều chỉnh độ phức tạp của mô hình. Giá trị của tham số này có được tính từ 0 trở lên, tương ứng với giá trị được thiết lập thì mức độ chính xác và khả năng sử dụng tài nguyên sẽ tăng dần.

minDetectionConfidence là một tham số dùng để xác định giá trị tối thiểu để mô hình xác nhận rằng nó đã phát hiện được bàn tay. Giá trị của tham số này dao động từ 0.0 đến 1.0, thông thường tham số này sẽ có giá trị mặc định là 0.5, nếu dữ liệu được lấy từ webcam có kết quả dự đoán thấp hơn giá trị của tham số này thì mô hình sẽ không nhận diện bàn tay.

minTrackingConfidence là một tham số để thể hiện ngưỡng tin cậy tối thiểu để theo dõi bàn tay, nếu ngưỡng tin cậy thấp hơn giá trị này thì cần phát hiện lại bàn tay. Giá trị của tham số này dao động từ 0.0 đến 1.0, thông thường tham số này sẽ có giá trị mặc định là 0.5.

Cuối cùng, thông qua sự kiện onResults() thì kết quả sẽ chứa thông tin về các đối tượng được phát hiện. Từ đó cho phép triển khai các chức năng như hiển thị kết quả trên giao diện người dùng hoặc thực hiện các thao tác phức tạp hơn.

2.3 Thuật toán tìm đường đi BFS.

2.3.1 Định nghĩa về thuật toán tìm đường đi BFS.

Thuật toán BFS là một phương pháp tìm kiếm và duyệt qua các đỉnh trong cấu trúc dữ liệu đồ thị hoặc cây theo chiều rộng.

BFS thường được sử dụng để giải quyết các bài toán tìm đường đi ngắn nhất trong đồ thị không trọng số, kiểm tra tính liên thông của đồ thị, hoặc phân tích cấu trúc mạng. Điểm đặc biệt của BFS là nó duyệt các đỉnh theo thứ tự khoảng cách từ đỉnh bắt đầu, nghĩa là tất cả các đỉnh ở cùng một mức độ (distance level) sẽ được duyệt trước khi chuyển sang các mức độ tiếp theo. Thuật toán này rất hữu ích trong nhiều lĩnh vực như trí tuệ nhân tạo, xử lý đồ thị, phân tích dữ liệu mạng, và giải

quyết các bài toán thực tế như định tuyến đường đi, kiểm tra các kết nối mạng, hoặc tìm kiếm trong trò chơi.

2.3.2 Cách hoạt động của BFS.

Thuật toán BFS hoạt động bằng cách duyệt qua các đỉnh trong đồ thị hoặc cây theo chiều rộng, bắt đầu từ một đỉnh nguồn. Thuật toán sử dụng một hàng đợi để quản lý các đỉnh sẽ được duyệt tiếp theo. Đầu tiên, đỉnh nguồn được thêm vào hàng đợi và đánh dấu là đã thăm. Sau đó, thuật toán lần lượt lấy từng đỉnh từ hàng đợi, duyệt qua các đỉnh kề chưa được thăm, đánh dấu chúng và đưa vào hàng đợi. Quá trình này tiếp tục cho đến khi hàng đợi rỗng, đảm bảo rằng các đỉnh được duyệt theo thứ tự từ gần đến xa so với đỉnh nguồn, giúp BFS trở thành một phương pháp hiệu quả để tìm kiếm và khám phá đồ thị hoặc cây.

Trong game Lines, thuật toán BFS là một thuật toán quan trọng dùng để tìm đường đi giữa hai ô trong bảng, từ đó có thể thực hiện hành động di chuyển bóng của trò chơi.

2.4 Realtime Database của Firebase

2.4.1 Giới thiệu về Firebase

Firebase là một nền tảng phát triển ứng dụng được cung cấp bởi Google, giúp và mở rộng ứng dụng trên các nền tảng như Android, iOS và web. Firebase cung cấp một loạt các công cụ và dịch vụ để hỗ trợ toàn bộ quy trình phát triển ứng dụng.

Firebase nổi bật nhờ khả năng tích hợp sâu với các dịch vụ của Google và hỗ trợ xử lý thời gian thực. Điều này giúp các ứng dụng có thể đồng bộ dữ liệu nhanh chóng giữa nhiều người dùng.

2.4.2 Giới thiệu về Realtime Database của Firebase

Realtime Database của Firebase là một cơ sở dữ liệu NoSQL trên nền tảng đám mây, được thiết kế để lưu trữ và đồng bộ hóa dữ liệu giữa các ứng dụng trong thời gian thực. Một trong những điểm mạnh nổi bật của Realtime Database là khả năng đồng bộ hóa dữ liệu tức thì giữa các thiết bị và làm việc offline; dữ liệu sẽ tự động được đồng bộ khi kết nối mạng được khôi phục.

Realtime Database phù hợp với các ứng dụng yêu cầu xử lý thời gian thực nhanh chóng. Đây là giải pháp lý tưởng cho các ứng dụng nhỏ và vừa, cho phép tập trung vào trải nghiệm người dùng mà không cần phải lo lắng về quản lý máy chủ backend.

Trong việc phát triển game Lines, Realtime Database được sử dụng để lưu danh sách các người chơi và hiển thị ra một mục bảng xếp hạng. Điều này làm tăng tính thực tế của trò chơi và tính cạnh tranh cho các người chơi.

CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU

3.1 Mô tả và phân tích bài toán

3.1.1 Mô tả bài toán

Game Lines là một trò chơi đơn giản trên bàn chơi dạng lưới, nơi người chơi cần di chuyển các quả bóng sao cho chúng tạo thành các hàng ngang, cột dọc hoặc đường chéo liên tiếp với số lượng từ đủ 5 quả bóng trở lên để ghi điểm.

Bàn chơi được thiết kế dạng lưới 9x9 ô vuông, mỗi ô chỉ có thể chứa một quả bóng hoặc để trống. Người chơi chọn một quả bóng và di chuyển nó đến một ô trống theo đường đi hợp lệ, đảm bảo không có vật cản. Sau mỗi lượt di chuyển, 3 quả bóng mới xuất hiện ngẫu nhiên trên lưới nếu người chơi không ghi được điểm.

Khi người chơi tạo được một hàng bóng cùng loại từ đủ 5 quả trở lên, các quả bóng trong hàng đó sẽ biến mất và điểm số được cập nhật. Trò chơi kết thúc khi không còn ô trống nào trên bàn chơi.

3.1.2 Phân tích bài toán

Để hiện thực bài toán này ta cần tổ chức một ma trận tương ứng với bảng chơi, mỗi phần tử ma trận sẽ là một số biểu thị cho màu của quả bóng. Bảng chơi ban đầu sẽ có 6 quả bóng được đặt ngẫu nhiên và 3 bóng dự đoán.

Các quả bóng sẽ xuất hiện ngẫu nhiên trên bảng chơi, nhiệm vụ của người chơi là di chuyển bóng để ghi điểm. Người chơi phải chọn một quả bóng và di chuyển quả bóng đó đến ô trống mới, sử dụng thuật toán tìm đường đi BFS để thực hiện việc này. Nếu không có đường đi hợp lệ, hủy trạng thái đang chọn của quả bóng đó.

Sau mỗi bước đi, cần thực hiện thao tác kiểm tra các quả bóng thẳng hàng bằng cách quét toàn bộ ma trận. Nếu có một hàng nào đó có từ đủ 5 quả bóng cùng loại trở lên, các quả bóng đó sẽ được xóa và điểm của người chơi sẽ được cập nhật, điểm của người chơi sẽ được cộng theo mức bình thường nếu hàng các quả bóng cùng loại là 5 quả, từ 6 quả trở lên thì sẽ có thêm điểm bonus cho người chơi. Nếu không có hàng nào được tìm thấy, trò chơi sẽ đặt 3 quả bóng mới vào 3 vị trí đã dự đoán trước đó và tạo thêm 3 bóng dự đoán mới.

Sau mỗi lần người chơi di chuyển bóng, quét lại toàn bộ ma trận, nếu số lượng bóng lắp đầy bảng chơi, người chơi sẽ bị xử thua và điểm mà người chơi đạt được sẽ được hiển thị ra màn hình.

Phân tích các thuật toán cần thiết:

Tạo bảng ban đầu.

Tạo bóng và bóng dự đoán mới.

Kiểm tra đường đi hợp lệ.

Di chuyển bóng giữa hai vị trí bất kì trên bảng chơi.

Kiểm tra hàng liên tiếp.

Ghi điểm.

Điều kiện kết thúc trò chơi.

Luồng xử lý của trò chơi:

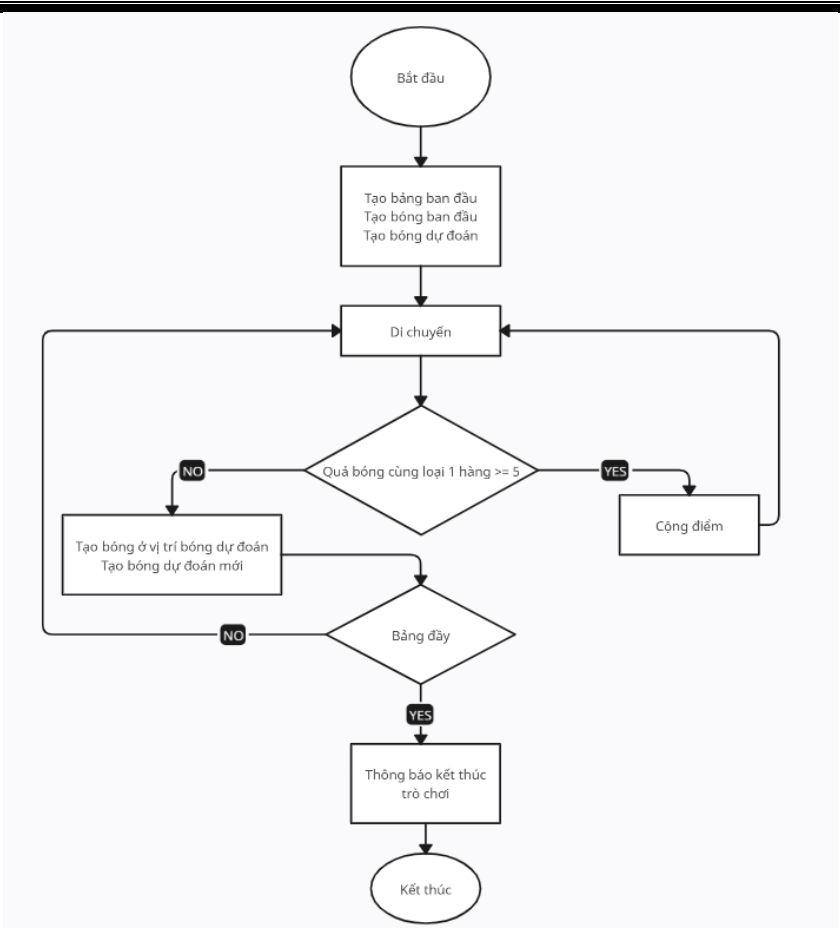
Đầu tiên, trò chơi được khởi tạo và hình ảnh sẽ được thu thập từ webcam của người chơi. Sau đó, sử dụng MediaPipe Hands để nhận diện và theo dõi bàn tay của người chơi.

Tiếp theo, các tọa độ của các điểm trên bàn tay được xử lý nhằm xác định cử chỉ tay mà người chơi thực hiện, dựa trên cử chỉ tay được nhận diện, hệ thống sẽ áp dụng logic của trò chơi để thực hiện các hành động tương ứng để tương tác với trò chơi.

Cuối cùng, trạng thái của trò chơi được cập nhật liên tục theo thời gian thực để phản hồi lại các thao tác của người chơi.

Hình 3: Mô tả luồng xử lý

Dựa vào các thuật toán đã phân tích, có thể biểu diễn thành một lưu đồ thuật toán của trò chơi.



Hình 4: Lưu đồ thuật toán của trò chơi

Lưu đồ thuật toán mô tả tổng quan về cách trò chơi hoạt động. Từ lưu đồ này, có thể dễ dàng nhận biết các thành phần tương tác với nhau. Việc này không chỉ giúp tiết kiệm thời gian mà còn mang lại tính nhất quán trong quá trình triển khai.

3.2 Các bước nghiên cứu đã tiến hành

3.2.1 Tạo bảng và các mảng cơ bản ban đầu

Để có thể xây dựng game Lines trên nền tảng web thì việc đầu tiên ở trang HTML là xây dựng các thẻ bố cục của game. Cần khởi tạo một thẻ dùng để chứa ma trận 9x9 và được đặt ID để có thể làm việc với JavaScript.

Ở file JavaScript, khởi tạo một mảng hai chiều arr có giá trị của từng phần tử bằng 0, chức năng của mảng hai chiều này dùng để vẽ ra ma trận 9x9 để hiển thị ở trang HTML.

Tiếp đến, tạo một mảng một chiều ColorBall mà mỗi phần tử của mảng là một đường dẫn đến file hình ảnh dùng để hiển thị cho loại bóng đó. Đồng thời, khởi

tạo thêm một mảng một chiều khác đặt tên là ValueColorBall được đánh số từ 1 đến n quả bóng. Có nghĩa là ứng với mỗi chỉ số mảng của từng phần tử trong ColorBall thì sẽ có một giá trị tương ứng cho từng loại bóng, các giá trị này dùng để tạo ra các logic game.

3.2.2 Hiển thị ma trận ra ngoài trang HTML

```
function CreateBroad() {
  const matrixContainer = document.getElementById("Board");

  for (let i = 0; i < arr.length; i++) {
    for (let j = 0; j < arr[i].length; j++) {
      const cell = document.createElement("div");
      cell.classList.add("cell");
      matrixContainer.appendChild(cell);
    }
  }
}
```

Hình 5: Mô tả hàm CreateBroad()

Hàm CreateBroad được thiết kế để tạo giao diện của một bảng (Board) trên trang web bằng cách sử dụng DOM manipulation. Đầu tiên, hàm lấy một phần tử HTML có ID là Board thông qua phương thức document.getElementById. Phần tử này được sử dụng làm vùng chứa để thêm các ô (cell) của bảng.

Tiếp theo, hàm lặp qua mảng arr (là một ma trận 9x9 đã được định nghĩa trước đó). Trong mỗi lần lặp qua hai vòng lặp lồng nhau, nó tạo một phần tử mới cho từng ô trong ma trận. Mỗi ô được gán lớp CSS là cell thông qua phương thức classList.add. Sau đó, các phần tử này được thêm vào vùng chứa bảng bằng phương thức appendChild.

3.2.3 Dự đoán bóng và đặt bóng vào ô được dự đoán

Để có thể thực hiện được chức năng dự đoán bóng, đầu tiên cần phải khởi tạo thêm một mảng một chiều arr_pre dùng để chứa chỉ số dòng, cột và màu sắc của bóng được dự đoán tại vị trí đó.

```
let arr_pre = [];
function SetBall_pre(index1, index2, rdc_index) {
  if (arr[index1][index2] === 0) {
    const cells = document.getElementsByClassName("cell");
    const index = index1 * arr.length + index2;
    const circle = document.createElement("div");
    circle.classList.add("circle-pre");

    circle.style.backgroundImage = `url(${ColorBall[rdc_index]})`;

    cells[index].appendChild(circle);
    arr[index1][index2] = -1;
  }
}
```

Hình 6: Mô tả hàm SetBall_pre()

Hàm SetBall_pre được thiết kế để hiển thị một quả bóng tạm thời trong bảng ma trận. Khi được gọi, hàm kiểm tra trạng thái của một ô tại vị trí được xác định bởi hai chỉ số index1 và index2. Nếu ô này đang trống (tức là giá trị trong ma trận arr tại vị trí đó bằng 0), hàm sẽ tiếp tục xử lý. Đầu tiên, nó xác định vị trí của ô tương ứng trong giao diện HTML thông qua danh sách các phần tử có lớp cell. Sau đó, chỉ số của ô trong danh sách được tính toán dựa trên công thức: index = index1 * arr.length + index2.

Lớp circle-pre được định dạng kích thước và vị trí cụ thể mục đích làm cho bóng dự đoán luôn nằm ở giữa ô và có kích thước luôn nhỏ hơn bóng chính.

Tiếp theo, hàm tạo một phần tử mới để đại diện cho quả bóng, gán cho nó lớp circle-pre và thiết lập hình nền bằng hình ảnh từ mảng ColorBall, chọn theo chỉ số rdc_index. Phần tử quả bóng này được thêm vào bên trong ô tương ứng trong giao diện bằng phương thức appendChild.

Cuối cùng, trạng thái của ô trong ma trận arr được cập nhật thành -1 để đánh dấu rằng ô này đang được sử dụng tạm thời (tránh việc dự đoán 2 quả bóng trong cùng một ô). Hàm này thường được sử dụng để dự đoán vị trí tiếp theo của các quả bóng trong trò chơi, giúp người chơi biết trước được những việc cần chuẩn bị tiếp theo.

Sau khi dự đoán vị trí của các quả bóng, khi người chơi di chuyển mà không ghi được điểm thì các quả bóng dự đoán sẽ được thay đổi thành các quả bóng thật, xuất hiện tại vị trí được dự đoán với loại bóng không thay đổi. Khi đó, hàm SetBall sẽ thực hiện việc đặt bóng vào ô dự đoán.

```
function SetBall() {
  for (let i = 0; i < arr_pre.length; i += 3) {
    let index1 = arr_pre[i];
    let index2 = arr_pre[i + 1];
    let rdc_index = arr_pre[i + 2];

    if (arr[index1][index2] !== -1 && arr[index1][index2] !== 0) {
      removeBalls_pre_notvalue(index1, index2);
    } else {
      removeBalls_pre(index1, index2);
      const cells = document.getElementsByClassName("cell");
      const index = index1 * arr[0].length + index2;

      // Thêm bóng chính thức
      const circle = document.createElement("div");
      circle.classList.add("circle");
      circle.style.backgroundImage = `url(${ColorBall[rdc_index]})`;
      cells[index].appendChild(circle);

      // Cập nhật mảng chính thức
      arr[index1][index2] = ValueColorBall[rdc_index];
    }
  }
  arr_pre = [];
}
```

Hình 7: Mô tả hàm SetBall()

Hàm SetBall được thiết kế để chính thức đặt các quả bóng vào bảng ma trận dựa trên dữ liệu tạm thời lưu trong mảng arr_pre. Mỗi nhóm ba phần tử liên tiếp trong arr_pre lần lượt đại diện cho chỉ số hàng (index1), chỉ số cột (index2), và chỉ số màu sắc (rdc_index).

Hàm thực hiện lặp qua mảng arr_pre với bước nhảy là 3. Trong mỗi lần lặp, nó kiểm tra trạng thái của ô tại vị trí (index1, index2) trong ma trận arr. Nếu ô này đã chứa một giá trị khác -1 hoặc 0 (tức là vừa có bóng di chuyển vào), hàm sẽ gọi phương thức removeBalls_pre_notvalue để xử lý việc loại bỏ bóng dự đoán tại ô đó.

Ngược lại, nếu ô đang trống hoặc chỉ chứa bóng tạm thời, hàm sẽ gọi phương thức removeBalls_pre để xóa nội dung tạm thời trong ô.

Tập hợp các phần tử HTML có lớp cell được lấy thông qua document.getElementsByClassName('cell'). Chỉ số của ô trong danh sách được tính toán bằng công thức index = index1 * arr[0].length + index2, đảm bảo rằng chỉ số đúng tương ứng với vị trí trong ma trận.

Sau khi đảm bảo ô đã được làm trống, hàm tạo một phần tử mới để đại diện cho quả bóng và gán cho nó lớp CSS circle. Tiếp theo, hàm sẽ thiết lập hình

nền của quả bóng bằng hình ảnh từ mảng ColorBall, dựa trên giá trị rdc_index. Phần tử quả bóng này sau đó được thêm vào giao diện thông qua phương thức appendChild của ô tương ứng trong danh sách các phần tử HTML có lớp cell. Đồng thời, giá trị của ô trong ma trận arr được cập nhật thành giá trị từ mảng ValueColorBall tại vị trí rdc_index, biểu thị màu sắc hoặc trạng thái của quả bóng.

Sau khi hoàn thành việc đặt tất cả các quả bóng, hàm sẽ xóa toàn bộ dữ liệu trong mảng arr_pre để chuẩn bị cho lần dự đoán tiếp theo.

3.2.4 Tạo bóng ngẫu nhiên

```
function CreateBall() {
  let ballCount = 3;

  const ball_check = 81 - isBoardFull();
  if (ball_check < 3) {
    ballCount = ball_check;
    if (ball_check === 1) {
      endgame++;
    }
  }

  let createdBalls = 0;

  while (createdBalls < ballCount) {
    const xball = Math.floor(Math.random() * 9);
    const yball = Math.floor(Math.random() * 9);
    const rdc_index = Math.floor(Math.random() * ColorBall.length);

    if (isBoardFull() == 81 || endgame === 2) {
      document.getElementById("gameOverModal").style.display = "flex";
      return [];
    } else {
      if (arr[xball][yball] === 0) {
        arr_pre.push(xball, yball, rdc_index);
        SetBall_pre(xball, yball, rdc_index);
        createdBalls++;
      }
    }
  }
}
```

Hình 8: Mô tả hàm CreateBall()

Hàm bắt đầu bằng cách đặt số lượng bóng cần tạo vào biến ballCount với giá trị là 3 và khởi tạo biến createdBalls để đếm số bóng đã tạo. Sau đó, nó sử dụng vòng lặp while để tiếp tục tạo bóng cho đến khi đạt đủ số lượng yêu cầu.

Trong mỗi lần lặp, hàm tạo ra các tọa độ ngẫu nhiên xball và yball trong phạm vi từ 0 đến 8 (tương ứng với các chỉ số hàng và cột của ma trận 9x9). Ngoài ra, chỉ số ngẫu nhiên rdc_index cũng được tạo ra để chọn màu sắc cho bóng từ mảng ColorBall.

Trước khi đặt bóng, hàm kiểm tra xem bảng ma trận đã đầy hay chưa bằng cách gọi hàm isBoardFull(). Nếu bảng đầy, một thông báo kết thúc trò chơi được hiển thị và hàm trả về một mảng rỗng, ngừng mọi xử lý.

Nếu bảng chưa đầy, hàm tiếp tục kiểm tra trạng thái của ô tại tọa độ (xball, yball) trong ma trận arr. Nếu ô này trống (giá trị là 0), nó sẽ thêm tọa độ và chỉ số màu sắc vào mảng dự đoán arr_pre. Sau đó, hàm gọi SetBall_pre để hiển thị bóng tạm thời tại vị trí này trên giao diện. Biến createdBalls được tăng lên mỗi khi một bóng mới được tạo thành công.

Hàm này kết hợp việc tạo dữ liệu (tọa độ và màu sắc của bóng) với việc cập nhật giao diện một cách đồng bộ, đảm bảo rằng người dùng có thể nhìn thấy các bóng xuất hiện trên bảng khi chúng được tạo. Kết hợp với hàm SetBall_pre, CreateBall đóng vai trò chính trong việc khởi tạo các phần tử trò chơi và cập nhật trạng thái ban đầu của bảng.

3.2.5 Di chuyển bóng

Sau khi hoàn thành việc tạo bảng và hiển thị bóng thành công, việc cần suy nghĩ đến tiếp theo là làm thế nào để có thể di chuyển bóng đến một vị trí trống bất kì trên bảng trò chơi.

```
let selectedBall = null;
let selectedX = -1;
let selectedY = -1;
```

Hình 9: Mô tả khởi tạo biến SelectedBall

Đầu tiên, cần tiến hành việc chọn quả bóng nào để di chuyển. Khởi tạo một biến SelectedBall được gán giá trị là null và hai biến khác dùng để lưu vị trí hàng và cột của quả bóng được chọn, khi có quả bóng bất kì nào đó được chọn thì biến Selected sẽ được gán giá trị là vị trí ô đang được người chơi chọn bóng.

Tiếp đến, sau khi người chơi chọn một quả bóng, cần phải xác định xem đó là lần chọn bóng đầu tiên hay là người chơi đổi quả bóng khác để di chuyển. Lúc đó cần phải thiết kế một hàm để phân tích hai trường hợp trên.

```
function selectOrMoveBall(index1, index2) {
  const cells = document.getElementsByClassName("cell");
  if (selectedBall === null && arr[index1][index2] !== 0) {
    selectedBall = cells[index1 * arr.length + index2].querySelector(".circle");
    selectedX = index1;
    selectedY = index2;
    selectedBall.classList.add("selected");
  } else if (
      (selectedBall !== null && arr[index1][index2] === 0) ||
      (selectedBall !== null && arr[index1][index2] === -1)
  ) {
    if (hasPath(selectedX, selectedY, index1, index2)) {
      moveBall(index1, index2);
    } else {
      selectedBall.classList.remove("selected");
      selectedBall = null;
      selectedX = -1;
      selectedY = -1;
    }
  } else if (
    selectedBall !== null && arr[index1][index2] !== 0 && arr[index1][index2] !== -1
  ) {
    selectedBall.classList.remove("selected");
    selectedBall = cells[index1 * arr.length + index2].querySelector(".circle");
    selectedX = index1;
    selectedY = index2;
    selectedBall.classList.add("selected");
  }
}
```

Hình 10: Mô tả hàm SelectOrMoveBall()

Hàm SelectOrMoveBall được sử dụng để phân tích hai trường hợp kể trên. Ban đầu hàm sẽ kiểm tra nếu chưa có quả bóng nào được chọn trước và ô người chơi đang chọn là một ô chứa bóng thì vị trí quả bóng của ô đang được chọn sẽ được gán cho biến SelectedBall và gán lại tọa độ của ô đang được chọn đó.

Tiếp đến, nếu người chơi nhấn vào một ô bất kì đang trống thì hàm sẽ tiếp tục kiểm tra xem có một đường đi nào đó từ điểm đang chọn đến điểm mà người chơi muốn di chuyển đến hay không.

Nếu có tồn tại một đường đi nào đó, hàm sẽ thực hiện việc di chuyển bóng bằng phương thức MoveBall.

Trong trường hợp người chơi đã chọn bóng trước đó và muốn thực hiện hành động đổi quả bóng khác để di chuyển, lúc này hàm sẽ xóa trạng thái và vị trí của quả bóng trước đó và thay bằng quả bóng mới vừa được chọn. Hàm sẽ gán lại tọa độ của quả bóng đó để chắc chắn rằng các logic xử lí tiếp theo không có sai sót.

Việc di chuyển quả bóng từ điểm này đến điểm khác và phải xét đường đi vừa tìm được là ngắn nhất thì có thể ứng dụng bằng thuật toán BFS (tìm kiếm theo chiều rộng.

```
const directions = [
  [-1, 0], // Lên
  [1, 0], // Xuống
  [0, -1], // Trái
  [0, 1], // Phải
];
```

Hình 11: Mô tả ma trận hướng di chuyển

Việc đầu tiên là khởi tạo hướng di chuyển để kiểm tra đường đi. Ma trận directions được khởi tạo ban đầu mà một ma trận 4x2 gồm 4 hướng lên, xuống, trái và phải, mỗi dòng của ma trận sẽ chứa một hướng di chuyển.

```
const queue = [[startX, startY]];
const visited = Array.from({ length: arr.length }, () =>
  Array(arr[0].length).fill(false)
);
visited[startX][startY] = true;
const path = Array.from({ length: arr.length }, () =>
  Array(arr[0].length).fill(null)
);
```

Hình 12: Mô tả khởi tạo các cấu trúc dữ liệu

Tiếp đến là khởi tạo các cấu trúc dữ liệu. Một hàng đợi sẽ được khởi tạo và giá trị đầu tiên sẽ được gán bằng tọa độ của ô bắt đầu. Sau đó, khởi tạo tiếp một ma trận 9x9 (bằng với số dòng và số cột của bảng) Visited có giá trị là false cho mỗi phần tử, mục đích là đánh dấu các vị trí đã được duyệt để tránh vòng lặp vô hạn. Ngay sau đó, tại vị trí của ô bắt đầu thì Visited tại ô đó sẽ được gán lại thành true.

Tiếp tục khởi tạo một ma trận Path có kích thước bằng với ma trận Visited và cũng khởi tạo cho tất cả các phần tử trong ma trận Path có giá trị là null.

Khởi tạo xong các cấu trúc dữ liệu trên, tiến hành sử dụng BFS để tìm đường đi.

```
function hasPath(startX, startY, targetX, targetY) {
  if (arr[targetX][targetY] !== 0 && arr[targetX][targetY] === -1) {
  }

  const directions = [
    [-1, 0], // Lên
    [1, 0], // Xuống
    [0, -1], // Trái
    [0, 1], // Phải
  ];

  const queue = [[startX, startY]];
  const visited = Array.from({ length: arr.length }, () =>
    Array(arr[0].length).fill(false)
  );
  visited[startX][startY] = true;
  const path = Array.from({ length: arr.length }, () =>
    Array(arr[0].length).fill(null)
  );

  while (queue.length > 0) {
    const [currentX, currentY] = queue.shift();

    if (currentX === targetX && currentY === targetY) {
      const steps = [];
      let x = targetX;
      let y = targetY;
      while (x !== startX || y !== startY) {
        steps.push([x, y]);
        [x, y] = path[x][y];
      }
      steps.reverse();
      return steps;
    }

    for (let i = 0; i < directions.length; i++) {
      const [dx, dy] = directions[i];
      const newX = currentX + dx;
      const newY = currentY + dy;

      if (isValid(newX, newY) && !visited[newX][newY]) {
        queue.push([newX, newY]);
        visited[newX][newY] = true;
        path[newX][newY] = [currentX, currentY];
      }
    }
  }
  return null;
}
```

Hình 13: Mô tả hàm hasPath() theo thuật toán tìm đường đi BFS

Nếu hàng đợi không rỗng, thì tọa độ của ô tại vị trí đầu tiên sẽ được lấy ra khỏi hàng đợi. Tiếp đến kiểm tra xem ô vừa lấy ra có phải là ô đích đến hay không, nếu không phải thì sẽ tiếp tục xây dựng đường đi đến ô đích.

Để xây dựng đường đi, một danh sách steps được tạo ra dùng để lưu các tọa độ từ ô đích quay ngược về ô bắt đầu dựa trên mảng path. Sau khi danh sách hoàn thành, nó được đảo ngược để đưa ra đường đi đúng thứ tự từ bắt đầu đến đích, sau đó được trả về.

Đối với mỗi ô hiện tại, thuật toán xem xét tất cả các ô liền kề theo bốn hướng (lên, xuống, trái, phải) bằng cách sử dụng một danh sách directions.

Với mỗi hướng, tọa độ của ô mới (newX, newY) được tính toán bằng cách cộng thêm tọa độ của phần tử trong danh sách directions. Nếu ô mới hợp lệ (nằm trong bảng, chưa được thăm và có thể đi qua), nó được thêm vào hàng đợi để xử lý trong các bước tiếp theo. Đồng thời, trạng thái đã duyệt của ô này được cập nhật lại thành true.

```
function isValid(x, y) {
  let t = x >= 0 && y >= 0 && x < arr.length && y < arr[0].length;
  return (t && arr[x][y] === 0) || (t && arr[x][y] === -1);
}
```

Hình 14: Mô tả hàm isValid() kiểm tra tính hợp lệ

Path[newX][newY] = [currentX, currentY] dùng để lưu lại vị trí hiện tại làm điểm xuất phát của ô (newX, newY). Điều này giúp xây dựng đường đi ngược từ đích về nguồn nếu tìm thấy đích.

Vòng lặp sẽ liên tục chạy cho đến khi tìm được đường đi giữa hai ô. Nếu hàng đợi trở nên trống trước khi tìm thấy đích, điều này có nghĩa là không có đường đi từ ô bắt đầu đến ô đích. Hàm sẽ trả về null.

Nếu hàm hasPath trả về một đường đi thì hàm MoveBall sẽ tiếp tục thực hiện việc di chuyển bóng. Một vòng lặp for sẽ được lặp qua từng phần tử trong danh sách path và khởi tạo thêm hằng số dùng để gán tọa độ được duyệt từ path, đây là tọa độ tiếp theo mà quả bóng sẽ di chuyển tới.

```
async function moveBall(index1, index2) {
  const cells = document.getElementsByClassName("cell");
  const path = hasPath(selectedX, selectedY, index1, index2);

  if (!path) {
    return;
  }

  if (selectedBall) {
    selectedBall.classList.remove("selected");
  }

  const ballToMove =
    cells[selectedX * arr.length + selectedY].querySelector(".circle");

  for (let step of path) {
    const [nextX, nextY] = step;
    const nextCell = cells[nextX * arr.length + nextY];

    nextCell.appendChild(ballToMove);

    arr[nextX][nextY] = arr[selectedX][selectedY];
    arr[selectedX][selectedY] = 0;

    selectedX = nextX;
    selectedY = nextY;

    await new Promise((resolve) => setTimeout(resolve, 30));
  }
```

Hình 15: Mô tả hàm Moveball()

Sau khi di chuyển đến ô bóng mới hàm sẽ cập nhật lại giá trị tại vị trí ô đến sao cho giá trị tại ô đến sẽ là giá trị của ô bắt đầu và giá trị tại ô bắt đầu sẽ được gán lại thành 0.

Sau mỗi lần thực hiện hành động di chuyển bóng, nếu không có hàng, cột hay đường chéo nào từ đủ 5 quả bóng cùng màu trở lên thì hàm sẽ tạo ra thêm 3 quả bóng tại 3 ô dự đoán và dự đoán tiếp 3 ô mới cho người chơi. Nếu có, thực hiện việc cộng điểm cho người chơi và không sinh bóng mới.

3.2.6 Kiểm tra sự kiện ghi điểm

Sau mỗi lần người chơi di chuyển một quả bóng từ điểm này đến điểm khác, cần kiểm tra xem sau mỗi lần di chuyển đó người chơi có tạo thành một hàng dọc, ngang, chéo các quả bóng cùng màu có số lượng từ đủ 5 trở lên hay không. Nếu có, sẽ xóa các quả bóng đó ra khỏi bảng và thực hiện cộng điểm cho người chơi.

```
function checkForFiveBalls(x, y) {
  const directions = [
    [0, 1], // Ngang
    [1, 0], // Dọc
    [1, 1], // Chéo chính
    [1, -1], // Chéo phụ
  ];

  for (let i = 0; i < directions.length; i++) {
    const [dx, dy] = directions[i];
    const positions = checkDirection(x, y, dx, dy);

    if (positions) {
      return positions;
    }
  }
  return null;
}
```

Hình 16: Mô tả hàm checkForFiveBalls()

Đầu tiên, tạo một hàm checkForFiveBalls có nhiệm vụ kiểm tra một phần tử cụ thể trong ma trận theo bốn hướng: ngang, dọc và hai đường chéo. Nó sử dụng thông tin về hướng di chuyển để xác định xem từ vị trí ban đầu có hình thành được nhóm các quả bóng liên tiếp có cùng loại hay không.

Nếu quả bóng trong quá trình duyệt có đặc điểm giống với quả bóng ban đầu thì vị trí của nó sẽ được lưu lại, và số lượng phần tử liên tiếp được tăng thêm. Quá trình này tiếp tục cho đến khi gặp phải phần tử không phù hợp hoặc khi vượt ra khỏi biên của ma trận khởi tạo bảng chơi. Nếu số lượng quả bóng cùng loại liên tiếp đạt tối thiểu là năm, hàm trả về danh sách các vị trí của chúng. Ngược lại, nó trả về giá trị rỗng.

```
function checkAllForFiveBalls() {
  for (let i = 0; i < arr.length; i++) {
    for (let j = 0; j < arr[i].length; j++) {
      if (arr[i][j] !== 0) {
        const positions = checkForFiveBalls(i, j);
        if (positions) {
          removeBalls(positions);
          if (positions.length > 5) {
            score = score + positions.length * 10;
            score += positions.length * 2;
          } else {
            score = score + positions.length * 10;
          }
          document.getElementById("score").textContent = score;
          lib.get_Point_ByID(useridgame).then((point) => {
            curr_point_db = point;
          });

          if (curr_point_db < score) {
            lib.writeUserData(useridgame, GetName(), GetPoint());
            lib.getUserData();
          }
          lib.getUserData();

          return true;
        }
      }
    }
  }
  return false;
}
```

Hình 17: Mô tả hàm checkAllForFiveBalls()

Tiếp đến, khi nhóm các quả bóng liên tiếp được phát hiện, chương trình chuyển sang bước xử lý chúng. Đầu tiên, các phần tử trong nhóm này sẽ được loại bỏ khỏi ma trận, và điểm số của người chơi sẽ được cập nhật.

Số điểm được tính dựa trên số lượng quả bóng trong nhóm, với công thức khác nhau cho các nhóm có kích thước vượt quá năm phần tử, nhằm thưởng thêm điểm. Điểm số sau khi tính toán được hiển thị ngay trên giao diện, giúp người chơi theo dõi kết quả của mình.

Để thực hiện việc lưu điểm số, sử dụng Realtime Database của Firebase để thực hiện. Đầu tiên, tạo một file JavaScript và khởi tạo các hàm được Firebase yêu cầu trong quá trình tạo Realtime Database trên trang web của Firebase.

```
export function writeUserData(userID, name, point) {
  const db = getDatabase();
  const reference = ref(db, "users/" + userID);

  set(reference, {
    Name: name,
    Point: point,
  });
}
```

Hình 18: Mô tả hàm writeUserData()

Tiếp đến, tạo một hàm writeUserData() và truyền các tham số cần thiết như ID, name và point để ghi vào Realtime Database.

```
export function getUserData() {
  const db = getDatabase();
  const postsRef = ref(db, "users");

  onValue(postsRef, (snapshot) => {
    const data = snapshot.val();
    let usersArray = Object.keys(data).map((key) => {
      return { id: key, ...data[key] };
    });

    usersArray.sort((a, b) => b.Point - a.Point);
    WriteScore("highscore", usersArray);
  });
}
```

Hình 19: Mô tả hàm getUserData()

Hàm getUserData dùng để đọc tất cả dữ liệu từ Realtime Database, sau đó sắp xếp lại và ghi ra một thẻ HTML có id là highscore.

Mỗi lần người chơi ghi điểm, điểm số sẽ luôn được ghi mới vào Realtime Database và bảng xếp hạng sẽ được cập nhật lại.

3.2.7 Kiểm tra thua

```
function isBoardFull() {
  let countball = 0;
  for (let i = 0; i < arr.length; i++) {
    for (let j = 0; j < arr[i].length; j++) {
      if (arr[i][j] != 0 && arr[i][j] != -1) {
        countball++;
      }
    }
  }
  return countball;
}
```

Hình 20: Mô tả hàm isBoardFull()

Khi người chơi chơi đến một thời điểm mà bóng mới không còn vị trí để có thể tạo ra trên bảng, có nghĩa lúc này người chơi đã thua. Sự kiện này được kiểm tra liên tục sau khi người chơi thực hiện hành động di chuyển bóng.

Lúc này, hàm sẽ duyệt qua từng phần tử của ma trận chứa bảng, nếu các giá trị tại mỗi ô của bảng khác 0 và khác -1 (giá trị của bóng dự đoán) thì thông báo thua của trò chơi sẽ được hiện ra và điểm mà người chơi đạt được cũng sẽ được hiển thị ra ngoài.

3.3 Tích hợp cử chỉ tay vào game Lines.

3.3.1 Khởi tạo một camera và hiển thị ra màn hình.

Đầu tiên, thiết kế một vùng bằng thẻ <canvas> dùng để hiển thị hình ảnh từ webcam ra ngoài trang HTML. Quá trình bắt đầu bằng việc yêu cầu quyền truy cập vào camera của người dùng thông qua trình duyệt. Khi quyền truy cập được cấp, mã tạo một phần tử video và liên kết nó với luồng dữ liệu video từ camera. Phần tử video sau đó được kích hoạt và phát nội dung từ camera, cho phép hiển thị hình ảnh trực tiếp.

```
const video = document.createElement("video");

const stream = await navigator.mediaDevices.getUserMedia({
  video: { width: 640, height: 480 },
});

video.srcObject = stream;
```

Hình 21: Mô tả khởi tạo một camera

Khi video được thiết lập, mã chuyển sang sử dụng một canvas HTML để xử lý và hiển thị nội dung. Nhưng lúc này hình ảnh hiển thị ra ngoài trang HTML lại bị ngược so với thực tế, cần phải lật ngược lại hình ảnh hiển thị để đảm bảo rằng hướng di chuyển bàn tay sẽ không bị ngược theo hình ảnh hiển thị ngoài trang HTML. Điều này đặc biệt hữu ích trong các ứng dụng yêu cầu tương tác trực quan, giúp người chơi cảm thấy trực quan và quen thuộc hơn khi sử dụng.

Để phân tích hình ảnh video, cần sử dụng thư viện MediaPipe dụng nhằm nhận diện bàn tay dựa trên các kỹ thuật học máy. Thư viện được tải từ một nguồn trực tuyến, đảm bảo rằng các tệp cần thiết luôn được lấy từ phiên bản mới nhất.

Tạo thêm một con trỏ ảo để có thể dựa vào con trỏ ảo đó mà người chơi có thể tương tác với trò chơi.

```
function setupCustomCursor() {
  const cursor = document.createElement("div");
  cursor.id = "customCursor";
  cursor.style.position = "absolute";
  cursor.style.width = "20px";
  cursor.style.height = "20px";
  cursor.style.backgroundImage = 'url("../another/picture/handup.png")';
  cursor.style.backgroundSize = "contain";
  cursor.style.backgroundRepeat = "no-repeat";
  cursor.style.backgroundPosition = "center";
  cursor.style.pointerEvents = "none";
  document.body.appendChild(cursor);
}
```

Hình 22: Mô tả hàm setupCustomCursor()

Tọa độ x, y của con trỏ ban đầu được xác định dựa trên tọa độ của ngón tay trỏ và được tùy chỉnh sao cho tương ứng với kích thước của màn hình.

```
const x = (1 - indexFingerTip.x) * window.innerWidth;
const y = indexFingerTip.y * window.innerHeight;
```

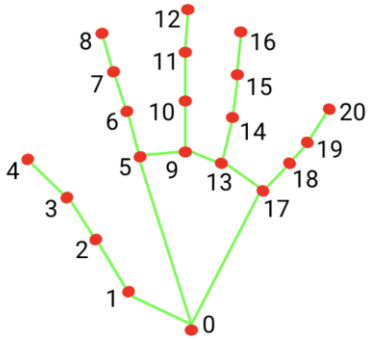
Hình 23: Mô tả tọa độ của ngón tay trỏ

Khi thiết lập xong những yêu cầu cần thiết để có thể ứng dụng thư viện MediaPipe thì có thể tiến hành xử lí nhận diện cử chỉ tay.

3.3.2 Các thao tác điều khiển một con trỏ ảo theo cử chỉ tay.

Thư viện MediaPipe sẽ được ứng dụng để xử lí dữ liệu bàn tay dựa trên landmarks để thực hiện lần lượt các thao tác: di chuyển và chọn.

Chức năng di chuyển: Sau khi nhận được hình ảnh từ webcam, lần lượt xét xem cử chỉ mà người chơi hiện đang làm là sẽ thực hiện hành động gì. Điều kiện để di chuyển con trỏ ảo là chỉ có ngón trỏ đang giơ thẳng và các ngón còn lại phải khép lại.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Hình 24: Mô tả các điểm landmark của từng khớp tay
(Nguồn Internet)

Để xét được như vậy, lần lượt xét tọa độ trục y của các ngón trỏ, giữa, nhẫn và út. Điều kiện xét là ngón trỏ đang mở là nếu tọa độ y của landmarks[8] (tức là indexFingerTip) phải nhỏ hơn tọa độ y của landmarks[6] (tức là indexFingerPip) và các ngón giữa, nhẫn, út phải đang co lại có nghĩa là tọa độ y của landmarks[12], [16], [20] (tức là middleFingerTip, ringFingerTip và pinkyFingerTip) phải lớn hơn tọa độ y của landmarks[10], [14], [18] (tức là middleFingerPip, ringFingerPip và pinkyFingerPip).

```
if (handedness === "Left") {
  if (
    indexFingerTip.y < indexFingerPip.y &&
    middleFingerTip.y > middleFingerPip.y &&
    ringFingerTip.y > ringFingerPip.y &&
    pinkyFingerTip.y > pinkyFingerPip.y &&
    thumbTip.x < thumbIp.x
  ) {
    cursorX = x + canvas.offsetLeft;
    cursorY = y + canvas.offsetTop;
    moveCursor(cursorX, cursorY);
  }
```

Hình 25: Mô tả điều kiện di chuyển chuột theo bàn tay phải

Ngón cái là một ngón tay mà điều kiện xét của nó phải dựa vào trục x chứ không giống như trục y của các ngón còn lại.

Để xét được ngón cái có đang khép hay không, nếu tọa độ x của landmarks[4] (tức là thumbTip) phải lớn hơn tọa độ x của landmarks[3] (tức là thumbIp). Nhưng vì camera đã bị lật ngược lại nên điều kiện cũng sẽ thay đổi thành thumbTip.x < thumbIp.x và dẫn đến handedness == Left.

```
function moveCursor(x, y) {
  const cursor = document.getElementById("customCursor");
  cursor.style.left = `${x - 10}px`;
  cursor.style.top = `${y}px`;
}
```

Hình 26: Mô tả hàm moveCursor()

Hàm moveCursor() sẽ dựa vào tọa độ x, y mà con trỏ ảo đang ở và di chuyển con trỏ ảo theo vị trí tương ứng với vị trí ngón tay hiện trên màn hình.

Việc đặt thêm điều kiện cho ngón cái là để cho trường hợp phát sinh. Trường hợp phát sinh ở đây là nếu người chơi bắt đầu thay đổi cử chỉ từ di chuyển sang chọn bóng thì vị trí mà người chơi chọn có thể bị dịch chuyển sau khi thay đổi cử chỉ. Vì vậy khi người chơi muốn thực hiện việc chọn một quả bóng nào đó thì có thể tiến hành giơ ngón cái ra ngoài để con trỏ chuột có thể đứng yên ngay tại vị trí mà người chơi muốn chọn.

Chức năng chọn: So với chức năng di chuyển thì chức năng chọn có phần dễ dàng hơn. Khi người chơi xác định được ô chứa quả bóng cần chọn, người chơi chỉ cần thực hiện thao tác co ngón trỏ lại là có thể chọn được quả bóng họ muốn. Khi di chuyển con trỏ ảo đến vị trí mới, người chơi chỉ cần lặp lại thao tác co ngón trỏ lại thì đã có thể thực hiện thành công việc di chuyển bóng từ ô này đến ô khác trong bảng bằng cử chỉ tay.

```
if (indexFingerTip.y > indexFingerPip.y && thumbTip.x > thumbIp.x) {
  const cell = getCellFromPosition(cursorX, cursorY);
  if (cell && canClick) {
    canClick = false;
    cell.click();

    setTimeout(() => {
      canClick = true;
    }, 1000);
  }
}
```

Hình 27: Mô tả sự kiện chọn bóng

Tạo thêm một khoảng thời gian 1 giây để giản cách hành động chọn sau mỗi lần chọn, tránh tạo sự khó chịu cho người chơi.

Điều kiện để thực hiện hành động chọn là tọa độ x của landmarks[4] (tức là thumbTip) phải lớn hơn tọa độ x của landmarks[3] (tức là thumbIp), đồng thời tọa độ y của landmarks[8] (tức là indexFingerTip) phải lớn hơn tọa độ y của landmarks[6] (tức là indexFingerPip).

```
function getCellFromPosition(x, y) {
  const cells = document.querySelectorAll(".cell");
  for (let i = 0; i < cells.length; i++) {
    const cell = cells[i];
    const rect = cell.getBoundingClientRect();

    if (
      x >= rect.left &&
      x <= rect.right &&
      y >= rect.top &&
      y <= rect.bottom
    ) {
      return cell;
    }
  }
  return null;
}
```

Hình 28: Mô tả hàm getCellFromPosition()

Hàm getCellFromPosition() được sử dụng để tính toán vị trí ô được chọn, hàm sẽ nhận hai tọa độ x, y là vị trí con trỏ ảo hiện tại và dựa vào đó tính toán xem ô nào đang được chọn và trả về vị trí ô ngay vị trí con trỏ ảo. Nếu không có ô nào ngay vị trí con trỏ ảo thì hàm sẽ trả về null.

Trong một số trường hợp, người chơi không đơn thuần chỉ sử dụng bàn tay phải để chơi trò chơi mà lại đổi sang bàn tay trái để chơi. Lúc này sẽ phát sinh thêm một trường hợp cần phải giải quyết.

Khi webcam đọc được thông tin hình ảnh nhận được từ người chơi, trong kết quả trả về từ MediaPipe sẽ có thêm một danh sách multiHandedness. Danh sách này dùng để chứa các thông tin về từng bàn tay được phát hiện từ webcam của người chơi.

```
if (results.multiHandLandmarks && results.multiHandLandmarks.length > 0) {
  const handedness = results.multiHandedness[0].label;
```

Hình 29: Mô tả điều kiện xác định bàn tay phải hoặc trái

Vì trò chơi chỉ cho phép di chuyển theo một bàn tay nhất định nên trong danh sách multiHandness, bàn tay đầu tiên được phát hiện chính là phần tử đầu tiên của danh sách multiHandness (tức là multiHandness[0]) sẽ là phần tử được xét xem người chơi đang sử dụng bàn tay nào để điều khiển trò chơi. MediaPipe sử dụng mô hình học máy để xác định dựa trên những đặc điểm hình dáng và vị trí của ngón tay để xác định xem bàn tay được nhận diện đó là bàn tay trái hay phải. Sau đó chỉ cần lấy giá trị của phần tử đầu tiên trong danh sách multiHandness thì nó sẽ trả về tương ứng là Left với bàn tay trái và Right với bàn tay phải.

```
} else if (handedness === "Right") {
  if (
    indexFingerTip.y < indexFingerPip.y &&
    middleFingerTip.y > middleFingerPip.y &&
    ringFingerTip.y > ringFingerPip.y &&
    pinkyFingerTip.y > pinkyFingerPip.y &&
    thumbTip.x > thumbIp.x
  ) {
    cursorX = x + canvas.offsetLeft;
    cursorY = y + canvas.offsetTop;
    moveCursor(cursorX, cursorY);
  }
```

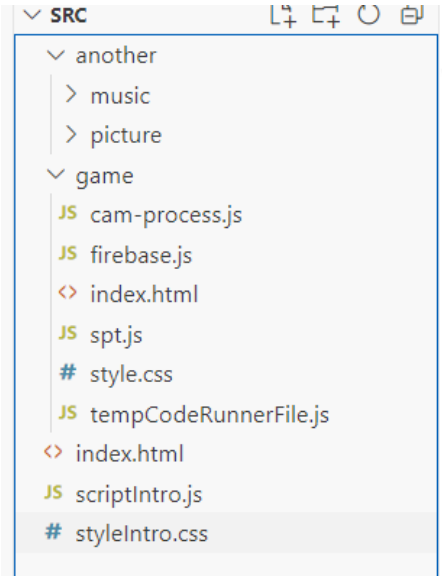
Hình 30: Mô tả điều kiện di chuyển chuột theo bàn tay trái

Đối với bàn tay trái, chỉ cần thay đổi điều kiện của ngón cái ở hai phương thức di chuyển và chọn bóng sao cho ngược lại so với điều kiện của ngón cái ở bàn tay phải là đã có thể chơi được trò chơi với một trong hai bàn tay bất kì.

CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

Sau khi thực hiện các bước nghiên cứu, thiết kế thì cơ bản đã thiết kế thành công game Lines có thể sử dụng cử chỉ tay để tương tác với trò chơi.

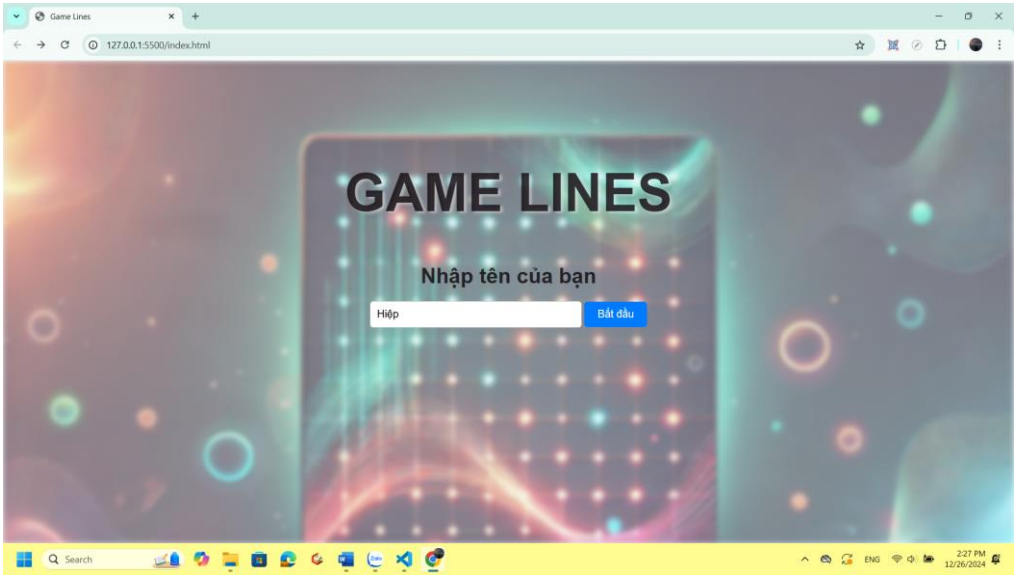
Cách tổ chức cây thư mục:



Hình 31: Mô tả tổ chức cây thư mục

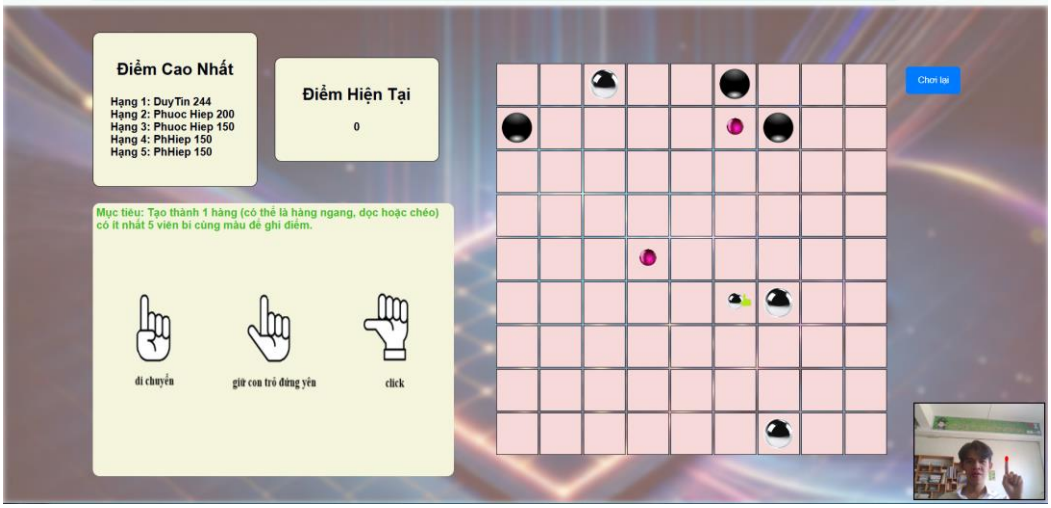
Trong đó, thư mục game là nơi chứa các tệp chuyên về game Lines, tệp spt.js chứa các logic của game, tệp cam-process.js chứa các logic về MediaPipe.

Giao diện lúc vừa truy cập vào trang web trò chơi, yêu cầu người chơi nhập trên rồi mới bắt đầu trò chơi.



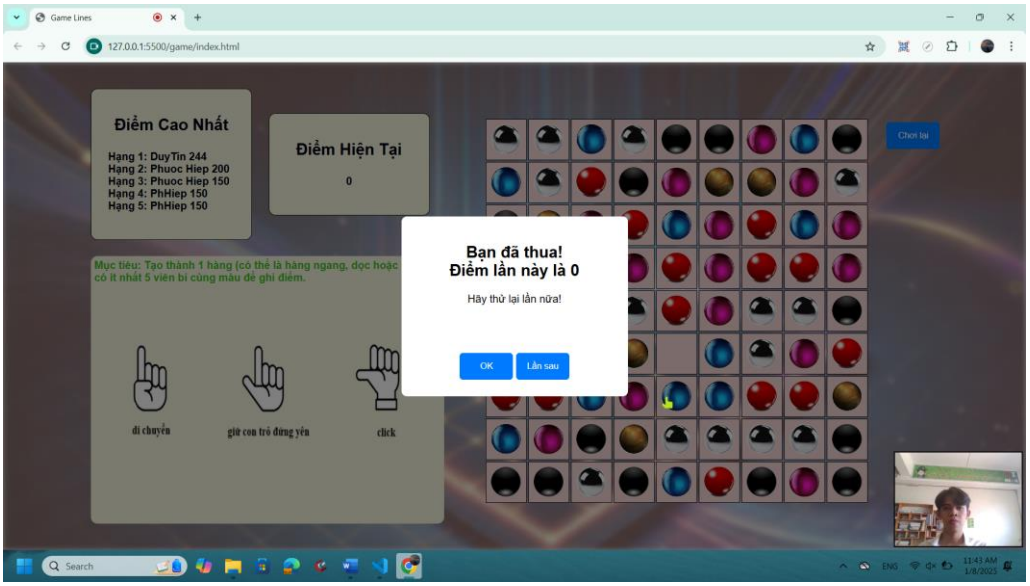
Hình 32: Giao diện nhập tên

Giao diện tổng quan của trò chơi.



Hình 33: Giao diện tổng quan của trò chơi

Giao diện của người chơi sau khi để bóng đầy trò chơi.



Hình 34: Giao diện sau khi thua

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Kết quả đạt được của đồ án:

Phát triển thành công game Lines trên nền tảng web có thể tương tác bằng cử chỉ tay thay cho tương tác bằng chuột truyền thống.

Ứng dụng MediaPipe để xử lý hình ảnh từ webcam và từ đó đặt điều kiện cử chỉ tay để tương tác với trò chơi.

Biết xử lí một số sự kiện của JavaScript và xây dựng thành công một số thuật toán của trò chơi.

Biết sử dụng dịch vụ cơ sở dữ liệu Firebase Realtime Database của Google.

Những đóng góp mới của đồ án:

Không cần sử dụng thiết bị chuột truyền thống nhưng vẫn có thể tương tác được với trò chơi.

Hướng phát triển:

Cải thiện và nâng cấp giao diện cũng như một số hiệu ứng mới trong trò chơi.

Khả năng hỗ trợ đa nền tảng để có thể chơi trên các thiết bị di động.

Phát triển thêm chế độ chơi mới.

DANH MỤC TÀI LIỆU THAM KHẢO

[1] Đoàn Phước Miền, Phạm Thị Trúc Mai (2014), Thiết kế và Lập trình Web, Bộ môn Công nghệ Thông tin, Khoa Kỹ thuật và Công nghệ, Trường Đại học Trà Vinh.

[2] Trầm Hoàng Nam (2013), Lý thuyết đồ thị, Bộ môn Công nghệ Thông tin, Khoa Kỹ thuật và Công nghệ, Trường Đại học Trà Vinh.

[3] M. Haverbeke, Eloquent JavaScript: A Modern Introduction to Programming, 4th edition. No Starch Press, 2024.

[4] Quang Trần, "MediaPipe: Live ML Solutions và ứng dụng vẽ bằng Hands Gestures," 2021. [Trực tuyến]. Địa chỉ: https://viblo.asia/p/mediapipe-live-ml-solutions-va-ung-dung-ve-bang-hands-gestures-gAm5ymOV5db. [Truy cập 23/12/2024].

[5] Trần Gia Nhuận, "Tìm hiểu về Firebase Realtime Database," 2018. [Trực tuyến]. Địa chỉ: https://viblo.asia/p/tim-hieu-ve-firebase-realtime-database-Az45bxzVZxY. [Truy cập 25/12/2024].