KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN

THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH

HỌC KỲ I, NĂM HỌC 2024 - 2025

PHÁT TRIỂN GAME XẾP GẠCH TRÊN WEB SỬ DỤNG MEDIAPIPE VỚI TƯƠNG TÁC ĐIỀU KHIỂN BẰNG CỬ CHỈ TAY

Giáo viên hướng dẫn:
TS. Nguyễn Bảo Ân

Sinh viên thực hiện:
Họ tên: Nguyễn Duy Tín
MSSV: 110122182
Lớp: DA22TTA

Trà Vinh, tháng 01 năm 2025

KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN

THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH

HỌC KỲ I, NĂM HỌC 2024 - 2025

PHÁT TRIỂN GAME XẾP GẠCH TRÊN WEB SỬ DỤNG MEDIAPIPE VỚI TƯƠNG TÁC ĐIỀU KHIỂN BẰNG CỬ CHỈ TAY

Giáo viên hướng dẫn:
TS. Nguyễn Bảo Ân

Sinh viên thực hiện:
Họ tên: Nguyễn Duy Tín
MSSV: 110122182
Lớp: DA22TTA

Trà Vinh, tháng 01 năm 2025

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 xin gửi lời cảm ơn chân thành đến Thầy Nguyễn Bảo Ân, người đã trực tiếp hướng dẫn, giúp đỡ và giải đáp thắc mắc trong suốt quá trình thực hiện đề tài. Sự tận tình Thầy đã giúp em hoàn thành đồ án này một cách tốt nhất.

Em cũng xin bày tỏ lòng biết ơn đến các Thầy, Cô trong Bộ môn Công nghệ thông tin, đã cung cấp những kiến thức nền tảng cần thiết trong suốt thời gian học tập và nghiên cứu.

Do lượng kiến thức và kinh nghiệm của em còn khiêm tốn nên vẫn có thể có những sai sót trong đồ án này. Kính mong quý Thầy/Cô thông cảm và em mong nhận được những nhận xét, đóng góp từ Thầy/Cô để em hoàn thiện hơn.

Em xin chân thành cảm ơn!

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

Sinh viên thực hiện

Nguyễn Duy Tín

MỤC LỤC

Trang

DANH MỤC HÌNH ẢNH – BẢNG BIỂU

DANH MỤC TỪ VIẾT TẮT

STT	Từ viết tắt	Từ gốc
1	API	Application Programming Interface
2	CSS	Cascading Style Sheets
3	CDN	Content Delivery Network
4	DOM	Document Object Model
5	HTML	Hyper Text Markup Language
6	JS	JavaScript
7	ML	Machine Learning
8	WASM	WebAssembly

TÓM TẮT ĐỒ ÁN CƠ SỞ NGÀNH

1. Vấn đề nghiên cứu

Đồ án này nghiên cứu các công nghệ về web như HTML, CSS, JS và công nghệ nhận diện cử chỉ tay bằng MediaPipe của JS. Đồ án hướng đến việc cải tiến game xếp gạch cổ điển điều khiển bằng phím, chuột bằng cách điều khiển bằng cử chỉ tay để cải thiện tương tác, nâng cao trải nghiệm người dùng.

2. Các hướng tiếp cận

Nghiên cứu lý thuyết về các công cụ và công nghệ sẽ sử dụng cho đồ án này như HTML, CSS, JS.

Áp dụng các lý thuyết đã nghiên cứu để cải tiến tựa game xếp gạch với logic game giữ nguyên, ứng dụng MediaPipe giúp nhận diện và theo dõi cử chỉ tay trong thời gian thực.

Phát triển giao diện game trên web: Xây dựng game bằng các công nghệ web như HTML, CSS, và JS.

Nghiên cứu dịch vụ cơ sở dữ liệu thời gian thực NoSQL (Not Only SQL) Firebase Realtime Database để lưu trữ thông tin người chơi và truy xuất trong thời gian thực.

Xử lý logic game: Tập trung phát triển thuật toán điều khiển và cơ chế chơi phù hợp với các cử chỉ như di chuyển gạch, xoay gạch và thả gạch xuống nhanh.

Tối ưu hóa trải nghiệm người dùng: Tích hợp âm thanh, hiệu ứng và giao diện bắt mắt để tăng tính hấp dẫn.

3. Cách giải quyết vấn đề

Nghiên cứu tài liệu và tham khảo qua các tựa game xếp gạch hiện có để áp dụng vào đồ án.

Xác định logic game như sau: Game sẽ có các logic như chạm trái, phải, xoay, hoàn thành hàng, game over,… Xác định các cử chỉ điều khiển: Giơ ngón cái/ngón út để di chuyển gach qua trái/phải (tùy bàn tay), xòe bàn tay để xoay, giơ ngón trỏ và ngón giữa để xuống nhanh.

Tìm hiểu về cách mà Firebase Realtime Database hoạt động cũng như cách tích hợp vào game.

Tìm hiểu xu hướng người dùng để thiết kế website với giao diện chơi game đẹp mắt, hài hòa, hiệu ứng sinh động, tích hợp âm thanh để cải thiện trải nghiệm người dùng.

Tiến hành xây dựng game với các logic và cử chỉ đã xác định bằng cách kết hợp các công nghệ đã nghiên cứu.

4. Kết quả đạt được

Game xếp gạch hoạt động trên web với các logic của game cổ điển, có thể điều khiển bằng cử chỉ tay trong thời gian thực. Thông tin người chơi được lưu trữ trên dịch vụ cơ sở dữ liệu Firebase Realtime Database.

Nhận diện chính xác cử chỉ tay, với độ trễ thấp và phản hồi nhanh.

Giao diện bắt mắt, hiệu ứng và âm thanh sống động thu hút người chơi.

MỞ ĐẦU

1. Lý do chọn đề tài

Trong thời đại công nghệ phát triển mạnh mẽ như hiện nay, cùng với đó các công nghệ như nhận diện cử chỉ tay, nhận diện khuôn mặt hay theo dõi chuyển động cơ thể đang dần được ứng dụng trong nhiều lĩnh vực từ y tế, giáo dục, dịch vụ đến giải trí và trò chơi. Các công nghệ đó mang lại lợi ích lớn trong việc tạo ra các giải pháp không chạm hay tương tác người - máy, nâng cao trải nghiệm người dùng và tính tiện lợi, nhanh chóng bằng cách sử dụng các chuyển động hay các đặc điểm của cơ thể để điều khiển thiết bị, nhận diện mặt người,... Trong bối cảnh đó, việc kết hợp công nghệ nhận diện cử chỉ tay với các ứng dụng giải trí, đặc biệt là trò chơi là xu hướng rất tiềm năng.

Game xếp gạch là một tựa game cổ điển được nhiều người yêu thích bởi tính giải trí, lối chơi đơn giản nhưng đầy tính tư duy. Tuy nhiên, các phiên bản hiện tại chủ yếu sử dụng bàn phím, chuột dẫn đến sự nhàm chán. Do đó em quyết định chọn đề tài "Phát triển game xếp gạch trên web sử dụng MediaPipe với tương tác điều khiển bằng cử chỉ tay" để tận dụng tiềm năng của công nghệ nhận diện cử chỉ, mang đến trải nghiệm chơi game mới mẻ. Không chỉ làm mới cách chơi của một tựa game quen thuộc, đề tài còn khẳng định tính ứng dụng thực tế của công nghệ nhận diện cử chỉ tay trong các sản phẩm giải trí, là tiền đề để góp phần thúc đẩy xu hướng tương tác người - máy trong tương lai.

2. Mục đích

Đổi mới cách chơi game xếp gạch cổ điển thông qua việc tích hợp công nghệ nhận diện cử chỉ tay, tạo nên một trải nghiệm điều khiển không cần bàn phím hay chuột.

Nghiên cứu và ứng dụng công nghệ MediaPipe, khai thác khả năng nhận diện cử chỉ thời gian thực để điều khiển trò chơi mượt mà trên nền tảng web.

Xây dựng một sản phẩm hoàn chỉnh, vừa đảm bảo tính sáng tạo vừa có tiềm năng mở rộng ứng dụng thực tế trong lĩnh vực giải trí với công nghệ tương tác.

3. Đối tượng nghiên cứu

HTML, CSS, JS: Cách tạo nên một trang web trò chơi.

Công nghệ MediaPipe: Tập trung nghiên cứu cách nhận diện cử chỉ tay và tích hợp vào ứng dụng web.

Dịch vụ cơ sở dữ liệu thời gian thực Firebase Realtime Database.

Game xếp gạch cổ điển: Các cơ chế chơi, luật chơi, và thuật toán liên quan đến việc di chuyển, xoay và thả gạch.

4. Phạm vi nghiên cứu

Về kỹ thuật: Tập trung phát triển và tích hợp MediaPipe vào game xếp gạch trên nền tảng web. Xây dựng giao diện và logic game phù hợp với tương tác cử chỉ.

Về tính năng: Cung cấp các tính năng cơ bản của game xếp gạch (di chuyển, xoay, thả gạch), đảm bảo trải nghiệm mượt mà với độ trễ thấp.

CHƯƠNG 1: TỔNG QUAN

Cùng với sự phát triển mạnh mẽ của công nghệ, các phương thức tương tác giữa con người và máy tính đã không ngừng được cải tiến. Từ các thiết bị truyền thống như bàn phím và chuột, công nghệ ngày nay đã chuyển sang các phương thức tương tác không chạm, trong đó có nhận diện cử chỉ, nhận diện khuôn mặt, và theo dõi chuyển động cơ thể. Những tiến bộ này không chỉ cải thiện tính tiện lợi mà còn tạo ra nhiều trải nghiệm mới mẻ và sáng tạo trong các lĩnh vực như y tế, giáo dục,… và đặc biệt là giải trí.

Trong lĩnh vực trò chơi, các tựa game kinh điển như xếp gạch đã gắn liền với nhiều thế hệ người chơi nhờ tính đơn giản, giải trí nhưng đầy thách thức. Tuy nhiên, các cách điều khiển quen thuộc như bàn phím, chuột dần trở nên nhàm chán, thiếu sự mới mẻ và không tận dụng hết khả năng của các công nghệ hiện đại. Điều này đặt ra yêu cầu về sự đổi mới cách chơi để nâng cao trải nghiệm người dùng.

Công nghệ MediaPipe, với khả năng nhận diện cử chỉ tay trong thời gian thực, cung cấp một giải pháp tiềm năng cho việc điều khiển trò chơi không cần thiết bị ngoại vi. Bằng cách sử dụng MediaPipe, có thể xây dựng một phiên bản game xếp gạch điều khiển bằng cử chỉ tay, người chơi chỉ cần sử dụng camera tích hợp để thực hiện các hành động như di chuyển, xoay hoặc thả gạch xuống nhanh mà không cần sử dụng đến các thiết bị ngoại vi rườm rà.

Các bước thực hiện đồ án

Bước 1: Nghiên cứu và thu thập thông tin

Tìm hiểu về trò chơi xếp gạch, bao gồm luật chơi, các cơ chế điều khiển, và cách hoạt động.

Nghiên cứu Firebase Realtime Database để lưu thông tin người chơi.

Nghiên cứu về công nghệ MediaPipe, tập trung vào khả năng nhận diện cử chỉ tay và tích hợp trên nền tảng web.

Tìm hiểu các công nghệ web cần thiết như HTML, CSS, JS để xây dựng giao diện và logic trò chơi.

Khảo sát các game xếp gạch có sẵn để phân tích ưu, nhược điểm và định hướng cải tiến.

Bước 2: Phân tích yêu cầu và thiết kế hệ thống

Phân tích yêu cầu:

Xác định các cử chỉ tay cần nhận diện để thực hiện các hành động như: di chuyển, xoay, thả gạch xuống nhanh.

Xác định các tính năng chính của game, như bắt đầu, chơi, lưu điểm và kết thúc.

Thiết kế hệ thống kiến trúc tổng quan bao gồm:

Phần giao diện (Front-end): Xây dựng bố cục game, bảng điểm, và hiển thị gạch.

Phần xử lý cử chỉ: Tích hợp MediaPipe để xử lý dữ liệu từ camera.

Phần logic game: Điều khiển chuyển động của các khối gạch dựa trên cử chỉ.

Thiết kế giao diện người dùng trực quan, dễ sử dụng.

Bước 3: Xây dựng hệ thống

Phát triển giao diện web:

Sử dụng HTML và CSS để xây dựng bố cục game, bao gồm giao diện nhập tên người chơi và giao diện chơi game, đảm bảo tính thẩm mỹ và trực quan.

Sử dụng JS để tạo hiệu ứng động và lập trình các logic game.

Xây dựng logic game:

Lập trình các logic game như di chuyển, xoay khối gạch, thả gạch xuống nhanh. Xử lý các quy tắc chơi game như xóa hàng sau khi hoàn thành một hay nhiều hàng, cập nhật điểm số, điểm cao nhất, cập nhật tên người chơi lên bảng xếp hạng, lưu điểm và kết thúc trò chơi.

Tích hợp MediaPipe:

Cài đặt thư viện MediaPipe để cấu hình nhận diện cử chỉ tay thông qua camera trên thiết bị của người chơi.

Xử lí dữ liệu cử chỉ tay để chuyển đổi thành lệnh điều khiển (di chuyển, xoay, thả gạch xuống nhanh).

Tích hợp Firebase Realtime Database để lưu thông tin người chơi.

Bước 4: Kiểm thử và tối ưu hóa

Đảm bảo các cử chỉ tay được nhận diện chính xác và phản hồi đúng trong trò chơi.

Kiểm tra tính chính xác của các tính năng game (di chuyển, xoay, thả gạch xuống nhanh, xóa hàng, điểm số,…).

Bước 5: Hoàn thiện và tích hợp tích năng bổ sung

Cải thiện giao diện và trải nghiệm người dùng (thêm âm thanh, hiệu ứng, bảng xếp hạng).

Thiết kế hướng dẫn cho người chơi.

CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

2.1 HTML

2.1.1 Tổng quan

HTML (Hyper Text Markup Language) là một ngôn ngữ đánh dấu dùng để xây dựng các trang web trên World Wide Web. HTML thường được sử dụng cùng với các công nghệ khác như CSS để tạo kiểu và JS để thêm tính năng động cho trang web [1].

2.1.2 Nguyên lý hoạt động của HTML

Khi nhập tên miền vào trình duyệt, trình duyệt sử dụng hệ thống Domain Name System (DNS) để xác định địa chỉ IP của máy chủ web, sau đó kết nối và nhận dữ liệu dưới dạng tệp HTML. Trình duyệt phân tích các thẻ HTML trong tệp, chuyển đổi chúng thành nội dung hiển thị hoặc thông tin mà người và máy tính có thể hiểu [2].

2.1.3 Ứng dụng

HTML là ngôn ngữ quan trọng trong việc tổ chức bố cục và định hình các thành phần của trang web, đồng thời cho phép nhúng các tệp kỹ thuật số như nhạc, video và hình ảnh. Mặc dù có nhiều ngôn ngữ lập trình khác nhau, HTML vẫn giữ vai trò cốt lõi trong việc hiển thị nội dung trên mọi trang web, bất kể nền tảng hay ngôn ngữ lập trình được sử dụng [2].

2.2 CSS

2.2.1 Tổng quan

CSS (viết tắt của Cascading Style Sheets) là một ngôn ngữ được sử dụng để tìm và định dạng lại các phần tử được tạo ra bởi các ngôn ngữ đánh dấu HTML. CSS giúp thiết kế trang web trở nên trực quan, đẹp mắt và dễ tùy chỉnh hơn [3].

2.2.2 Chức năng chính

CSS được sử dụng để định dạng các phần tử HTML, bao gồm bố cục, màu sắc, phông chữ, kích thước, khoảng cách, hình nền, và nhiều thuộc tính khác.

Nó tách biệt nội dung (HTML) và hình thức (CSS), giúp quản lý trang web dễ dàng hơn.

2.2.3 Cấu trúc của CSS

CSS hoạt động thông qua các quy tắc (ruleset), bao gồm:

Selector (bộ chọn): Chỉ định phần tử HTML cần áp dụng kiểu.

Declaration (khai báo): Bao gồm thuộc tính và giá trị, được đặt trong cặp ngoặc {}.

2.2.4 Phân loại

CSS được chia thành 3 loại: Bảng kiểu trực tiếp (Inline Style Sheet), Bảng kiểu được nhúng vào tài liệu HTML (Internal Style Sheet), và Bảng kiểu bên ngoài (External Style Sheet). Những loại này có thể được sử dụng đồng thời, với thứ tự ưu tiên giảm dần như sau:

1. Bảng kiểu trực tiếp (*Inline Style Sheet*).
2. Bảng kiểu được nhúng vào tài liệu HTML (*Internal Style Sheet*).
3. Bảng kiểu bên ngoài (*External Style Sheet*).
4. Các thiết lập mặc định của trình duyệt (*Browser Default*).

Điều này có nghĩa là, nếu trong cùng một thẻ HTML có một thuộc tính được quy định bởi cả Inline Style Sheet và Internal Style Sheet, thì thuộc tính sẽ được áp dụng là thuộc tính trong Inline Style Sheet. Trong trường hợp một thẻ HTML được quy định nhiều thuộc tính bằng các loại CSS khác nhau, trình duyệt sẽ tổng hợp tất cả các định dạng để hiển thị. Nếu một thuộc tính không được định nghĩa trong bất kỳ loại CSS nào, trình duyệt sẽ áp dụng các thiết lập mặc định để định dạng thẻ đó [4].

2.3 JS

2.3.1 Tổng quan

JS (JavaScript) là một ngôn ngữ lập trình phổ biến trong phát triển website hiện nay. Được tích hợp và nhúng vào HTML, JS giúp các trang web trở nên sinh động và có tính tương tác. Ngôn ngữ này hoạt động như một phần của trang web, cho phép thực thi các đoạn mã phía máy khách (Client-side) và cả phía máy chủ (thông qua Node.js), từ đó tạo ra các trang web động và linh hoạt [5].

2.3.2 Các đặc điểm chính

JS cho phép các trang web thay đổi nội dung và hành vi mà không cần tải lại trang.

JS chạy được trên hầu hết các trình duyệt hiện đại như Chrome, Firefox, Safari, Edge.

JS phản hồi các hành động của người dùng như nhấp chuột, di chuyển chuột hoặc nhập liệu.

JS hỗ trợ xử lý các tác vụ không đồng bộ, chẳng hạn như tải dữ liệu từ máy chủ mà không làm gián đoạn giao diện người dùng.

Được nhúng trong HTML: JS có thể được nhúng trực tiếp trong tài liệu HTML thông qua thẻ <script>.

2.3.3 Ứng dụng

Tương tác người dùng: Tạo các hiệu ứng như menu thả xuống, slideshow, hoặc xác thực biểu mẫu.

Xử lý logic: Xây dựng các ứng dụng web phức tạp như trò chơi, ứng dụng quản lý công việc.

Kết nối máy chủ: Gửi và nhận dữ liệu từ máy chủ bằng Asynchronous JavaScript and XML (AJAX) hoặc các công cụ hiện đại hơn như fetch hoặc thư viện Axios.

Điều khiển DOM: Thay đổi nội dung, cấu trúc hoặc kiểu dáng của trang web mà không cần tải lại trang.

2.4 MediaPipe

2.4.1 Tổng quan

MediaPipe là một framework mã nguồn mở do Google phát triển, chuyên cung cấp các giải pháp ML cho xử lý video và hình ảnh thời gian thực trên nhiều nền tảng như Web, Android, iOS và các thiết bị nhúng. Đặc biệt trên JS, MediaPipe sử dụng WASM để chạy trực tiếp trong trình duyệt với hiệu suất cao [6].

MediaPipe cung cấp các giải pháp như:

Theo dõi bàn tay, khuôn mặt, cơ thể.

Phân đoạn ảnh nền/đối tượng.

Theo dõi vật thể 3D.

2.4.2 Lịch sử hình thành

Bối cảnh phát triển

Năm 2010, Google nhận thấy nhu cầu về các giải pháp Aritifical Intelligient (AI)/ML trong lĩnh vực xử lý video và hình ảnh ngày càng tăng cao, đặc biệt là các ứng dụng AR (Augmented Reality), VR (Virtual Reality) và nhận diện thời gian thực. Tuy nhiên, hầu hết các giải pháp vào thời điểm đó yêu cầu phần cứng mạnh và không dễ dàng triển khai trên các thiết bị đa nền tảng.

Sự ra đời của MediaPipe

Năm 2018 Google ra mắt MediaPipe như một công cụ nội bộ cho các nhà nghiên cứu tại hội nghị CVPR (Computer Vision and Pattern Recognition).

Năm 2019 MediaPipe được công khai mã nguồn trên GitHub, tập trung vào các giải pháp như Hands (nhận diện bàn tay) và Face Detection (nhận diện khuôn mặt).

Năm 2020 ra mắt các giải pháp mới như Holistic (nhận diện toàn bộ cơ thể) và Objectron (nhận diện vật thể 3D). Tích hợp tốt hơn với WASM để hỗ trợ JS và các ứng dụng trên Web.

Năm 2021 – 2022, MediaPipe được tối ưu hóa cho Web, cho phép chạy trực tiếp trong trình duyệt mà không cần backend xử lý. Cải tiến các giải pháp như Selfie Segmentation (phân đoạn ảnh nền) và Pose Tracking (theo dõi tư thế).

Hiện tại, MediaPipe trở thành framework tiên phong trong xử lý hình ảnh/video và ML thời gian thực trên đa nền tảng.

2.4.3 Đặc điểm của MediaPipe trên JS

Ưu điểm nổi bật

Đa nền tảng: Chạy trên trình duyệt mà không cần backend nhờ WASM.

Thời gian thực: Hiệu suất xử lý nhanh chóng, tối ưu cho các ứng dụng tương tác trực tiếp.

Hỗ trợ mã nguồn mở: Dễ dàng tích hợp và tùy chỉnh.

Các thành phần chính

WASM: Tăng tốc xử lý ML trên trình duyệt.

Camera API: Truy xuất dữ liệu video từ webcam.

DrawingUtils: Vẽ landmark (điểm đặc trưng) và kết nối trực tiếp trên canvas.

Các giải pháp

Hands: Nhận diện bàn tay và cử chỉ.

Face Mesh: Vẽ lưới 3D trên khuôn mặt.

Pose: Theo dõi tư thế cơ thể.

Holistic: Kết hợp nhận diện khuôn mặt, bàn tay và tư thế cơ thể.

Objectron: Theo dõi vật thể 3D.

Selfie Segmentation: Phân đoạn nền hoặc đối tượng trong ảnh.

Ứng dụng thực tế

Trò chơi không chạm: Điều khiển game (xếp gạch, Flappy Bird) bằng cử chỉ tay.

Thực tế tăng cường (Augmented Reality): Thêm hiệu ứng khuôn mặt, tay trong ứng dụng AR.

Ứng dụng thể thao: Theo dõi tư thế và động tác trong yoga hoặc thể thao.

Thương mại điện tử: Thử đồ trang điểm hoặc kính trực tiếp qua camera.

Lợi ích của MediaPipe trong JS

Tiện lợi: Không yêu cầu backend mạnh.

Hiệu suất cao: Xử lý thời gian thực nhờ WASM.

Tích hợp dễ dàng: Làm việc tốt với các thư viện khác như Three.js hoặc TensorFlow.js.

2.4.4 Cách cài đặt và sử dụng

Để cài đặt và sử dụng MediaPipe trên JS, thực hiện các bước sau:

Cài đặt:

Sử dụng npm để cài đặt các package liên quan:

```
npm install @mediapipe/handpose
npm install @mediapipe/camera_utils
npm install @mediapipe/control_utils
npm install @mediapipe/drawing_utils
```

Hình 1: Hướng dẫn cài đặt MediaPipe qua npm

Hoặc sử dụng trực tiếp từ CDN qua <script>:

```
<script src="https://cdn.jsdelivr.net/npm/@mediapipe/hands"></script>
<script src="https://cdn.jsdelivr.net/npm/@mediapipe/camera_utils"></script>
<script src="https://cdn.jsdelivr.net/npm/@mediapipe/drawing_utils"></script>
```

Hình 2: Hướng dẫn cài đặt MediaPipe qua CDN

Gắn các <script> vào phần head của HTML.

Sử dụng:

Nhúng MediaPipe vào dự án, kết hợp với camera hoặc các nguồn video.

Sử dụng các công cụ như CameraUtils để lấy dữ liệu từ webcam và DrawingUtils để vẽ kết quả lên canvas.

Cấu hình module (ví dụ: HandPose, Pose, FaceMesh) và đăng ký callback để nhận kết quả phân tích.

Hiển thị kết quả bằng cách vẽ các điểm đặc trưng hoặc kết nối chúng.

Dùng một server cục bộ hoặc chạy file HTML trong trình duyệt để kiểm tra.

2.5 Firebase Realtime Database

Firebase Realtime Database là một cơ sở dữ liệu NoSQL trong nền tảng Firebase của Google, được thiết kế để lưu trữ và đồng bộ dữ liệu theo thời gian thực trên nhiều thiết bị. Nó lưu trữ dữ liệu dưới dạng cây JSON, cho phép truy cập và chỉnh sửa dễ dàng thông qua các nút (node). Khi dữ liệu được thay đổi, tất cả các thiết bị kết nối sẽ tự động nhận cập nhật ngay lập tức mà không cần gửi yêu cầu mới. Firebase Realtime Database cũng hỗ trợ hoạt động ngoại tuyến bằng cách lưu trữ tạm thời dữ liệu trên thiết bị và đồng bộ hóa khi có kết nối trở lại, phù hợp cho các ứng dụng như chat, bảng trạng thái thời gian thực, hoặc game đa người chơi [7].

CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU

3.1 Mô tả bài toán

Bài toán đặt ra là phát triển một trò chơi xếp gạch trên nền tảng web, nơi người chơi có thể điều khiển các khối gạch bằng cử chỉ tay thay vì bàn phím hay chuột. Trò chơi sẽ tích hợp công nghệ nhận diện cử chỉ tay sử dụng MediaPipe, giúp người chơi có thể thao tác một cách trực quan thông qua webcam. Bên cạnh đó, điểm số và tên người chơi sẽ được lưu trữ trong cơ sở dữ liệu Firebase Realtime Database, cho phép đồng bộ hóa và hiển thị bảng xếp hạng theo thời gian thực.

Người chơi sẽ sử dụng cử chỉ tay để điều khiển trò chơi. Người chơi có thể di chuyển khối gạch sang trái hoặc phải bằng cách giơ ngón cái hoặc ngón út tùy bàn tay. Cử chỉ xòe bàn tay sẽ được sử dụng để xoay khối gạch và cử chỉ giơ đồng thời ngón trỏ và ngón giữa để làm cho khối gạch rơi nhanh xuống dưới. Những hành động này cần được nhận diện chính xác để mang lại trải nghiệm mượt mà và tránh nhầm lẫn trong thao tác.

Giao diện trò chơi cần được thiết kế trực quan, bao gồm khu vực chơi, nơi các khối gạch rơi xuống và sắp xếp; một phần hiển thị điểm số hiện tại của người chơi và điểm cao nhất trong các lần chơi; bảng xếp hạng hiển thị tên cùng điểm số cao nhất. Điều này đảm bảo người chơi có thể dễ dàng theo dõi tiến trình của mình cũng như so sánh thành tích với những người khác.

Dữ liệu đầu vào:

Video từ webcam của người chơi.

Các cử chỉ tay được nhận diện bởi MediaPipe Hands (tọa độ bàn tay và các điểm landmark).

Tên và điểm số của người chơi.

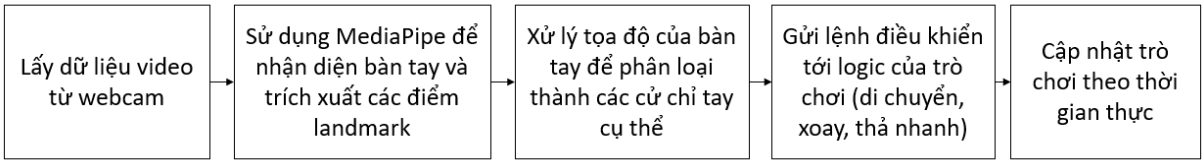
Dữ liệu đầu ra:

Các hành động trong trò chơi (di chuyển, xoay, thả nhanh).

Giao diện trò chơi hiển thị khối gạch đang rơi, điểm số và trạng thái trò chơi.

Dữ liệu từ Firebase được truy xuất và hiển thị trên giao diện dưới dạng danh sách, sắp xếp theo điểm số từ cao đến thấp, kèm theo tên người chơi.

Luồng xử lý:



Hình 3: Mô tả luồng xử lý

3.2 Các bước nghiên cứu đã tiến hành

3.2.1 Xây dựng bố cục

3.2.1.1 Lưới của trò chơi

Bản chất của trò chơi này là một lưới ma trận 2D gồm có 20 dòng và 10 cột. Mỗi ô trong lưới ma trận sẽ có 2 trạng thái:

7: nếu ô trống.

0 đến 6: tương ứng với id của gạch và màu sắc gạch.

(inde…	0	1	2	3	4	5	6	7	8	9
0	7	7	7	7	7	7	7	7	7	7
1	7	7	7	7	7	7	7	7	7	7
2	7	7	7	7	7	7	7	7	7	7
3	7	7	7	7	7	7	7	7	7	7
4	7	7	7	7	7	7	7	7	7	7
5	7	7	7	7	7	7	7	7	7	7
6	7	7	7	7	7	7	7	7	7	7
7	7	7	7	7	7	7	7	7	7	7
8	7	7	7	7	7	7	7	7	7	7
9	7	7	7	7	7	7	7	7	7	7
10	7	7	7	7	7	7	7	7	7	7
11	7	7	7	7	7	7	7	7	7	7
12	7	7	7	7	7	7	7	7	7	7
13	7	7	7	7	7	7	7	7	7	7
14	7	7	7	7	7	7	7	7	7	7
15	7	7	7	7	7	7	7	7	7	7
16	7	7	7	7	7	7	7	7	7	7
17	7	7	7	7	7	7	7	7	7	7
18	7	7	7	7	7	7	7	7	7	7
19	7	7	7	7	7	7	7	7	7	7

▶ Array(20)

Hình 4: Mô tả lưới ma trận khi chưa có gạch

Vai trò của lưới

Lưu trữ trạng thái: Mỗi ô trong lưới cho biết liệu vị trí đó đang trống hay đã bị chiếm bởi gạch.

Kiểm tra va chạm: Khi khối gạch di chuyển, trò chơi kiểm tra xem gạch có va chạm với:

Cạnh bảng (cạnh trái, phải, hoặc đáy).

Các ô đã bị chiếm bởi gạch khác.

Nếu một hàng trong lưới đầy đủ (tất cả ô đều bị chiếm), hàng đó sẽ bị xóa, và được thay thế bởi các số 7 (tương ứng với ô trống).

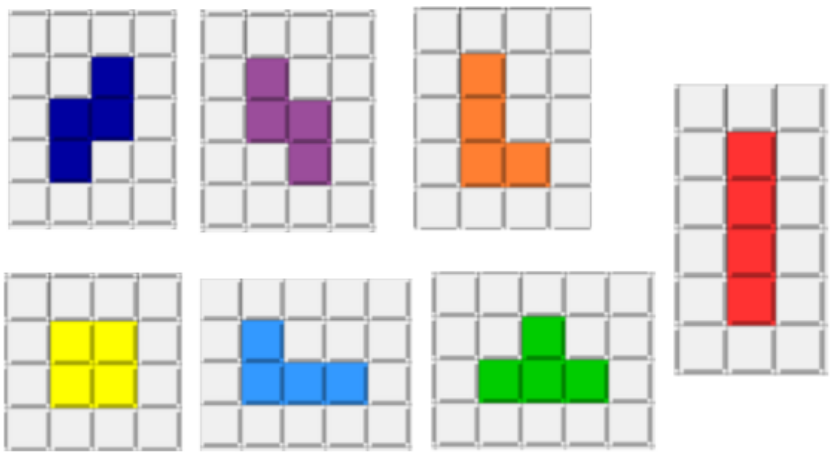
3.2.1.2 Các khối gạch

Mỗi khối gạch được định nghĩa bằng một mảng 2D nhỏ chứa giá trị 7 và 1:

1: Ô thuộc khối gạch.

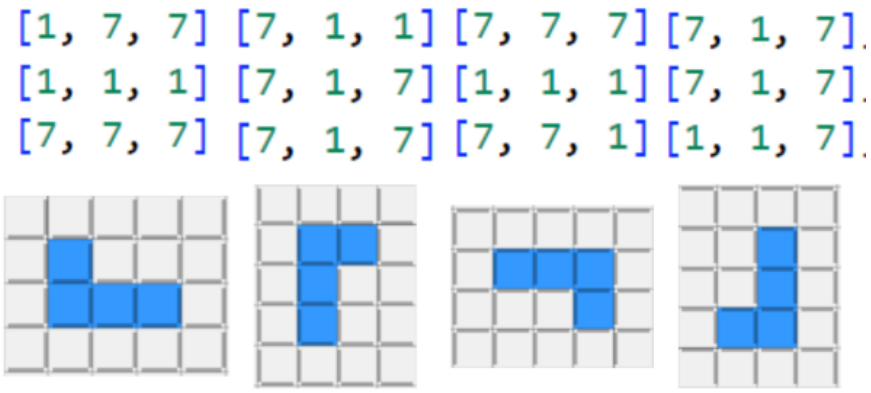
7: Ô trống.

Có 7 kiểu gạch khác nhau, mỗi kiểu gạch có một id và một mã màu tương ứng.



Hình 5: Mô tả 7 kiểu của khối gạch

Mỗi khối gạch sẽ có 4 hướng.



Hình 6: Mô tả chi tiết 1 khối gạch và 4 hướng của nó

3.2.2 Xây dựng logic game

3.2.2.1 Xây dựng lớp Board

Lớp Board được thiết kế để quản lý và vận hành các chức năng cốt lõi của trò chơi xếp gạch. Đây là một thành phần chính chịu trách nhiệm về trạng thái của bảng, các thao tác liên quan đến bảng và việc cập nhật giao diện trò chơi.

```
// Lớp Board
class Board {
  constructor(ctx) {
    this.ctx = ctx;
    this.grid = this.generateWhiteBoard();
    this.score = 0;
    this.gameOver = false;
    this.isPlaying = false;
  }
  // Reset lại bảng và điểm số khi trò chơi bắt đầu lại
  reset() {···
  }
  // Tạo bảng trắng (chưa có gạch)
  generateWhiteBoard() {···
  }
  // Vẽ một ô trên bảng
  drawCell(xAxis, yAxis, colorId) {···
  }
  // Vẽ lại toàn bộ bảng sau mỗi lần thay đổi
  drawBoard() {···
  }
  // Kiểm tra và xử lý khi hoàn thành một hàng
  handleCompleteRows() {···
  }
  // Hàm tạo hiệu ứng tan biến từ trái qua
  fadeOutRows(rows) {···
  }
  // Cập nhật điểm số
  handleScore(newScore) {···
  }
  // Xử lý khi trò chơi kết thúc
  handleGameOver() {···
  }
}
```

Hình 7: Mô tả lớp Board

Constructor:

Nhận vào một context để khởi tạo các giá trị mặc định khi tạo một bảng và gọi phương thức generateWhiteBoard() để tạo bảng trắng ban đầu.

ctx: Context vẽ trên canvas, cho phép vẽ các hình dạng và màu sắc trên bảng.

grid: Ma trận 2D biểu diễn trạng thái hiện tại của bảng (mỗi ô có một ID màu).

score: Điểm số hiện tại của trò chơi.

gameOver: Biến để xác định trạng thái kết thúc của trò chơi.

isPlaying: Biến để theo dõi xem trò chơi có đang chạy hay không.

Phương thức:

Vẽ bảng và ô

generateWhiteBoard():

```
generateWhiteBoard() {
  return Array.from({ length: ROWS }, () => Array(COLS).fill(WHITE_ID));
}
```

Hình 8: Phương thức generateWhiteBoard()

Tạo một bảng 2D với tất cả các ô được gán giá trị mặc định WHITE_ID.

drawCell():

```
// Vẽ một ô trên bảng
drawCell(xAxis, yAxis, colorId) {
  const color = COLOR_PALETTE[colorId] || COLOR_PALETTE[WHITE_ID];
  // Đổ nền màu cơ bản
  this.ctx.fillStyle = color;
  this.ctx.fillRect(···
  );
  // Đổ bóng ở dưới và bên phải để tạo chiều sâu
  this.ctx.fillStyle = 'rgba(0, 0, 0, 0.3)';
  this.ctx.fillRect(···
  );
  this.ctx.fillRect(···
  );
  // Vẽ đường viền xung quanh ô
  this.ctx.strokeStyle = 'rgba(0, 0, 0, 0.2)';
  this.ctx.strokeRect(···
  );
}
```

Hình 9: Phương thức drawCell()

Vẽ một ô cụ thể trên bảng với màu sắc từ COLOR_PALETTE với vị trí xAxis và yAxis được truyền vào. Tạo hiệu ứng đổ bóng ở dưới và bên phải để tạo cảm giác 3D, vẽ đường viền ô.

drawBoard():

```
drawBoard() {
  for (let row = 0; row < this.grid.length; row++) {
    for (let col = 0; col < this.grid[0].length; col++) {
      this.drawCell(col, row, this.grid[row][col]);
    }
  }
}
```

Hình 10: Phương thức drawBoard()

Lặp qua từng ô trong grid và vẽ toàn bộ bảng.

Xử lý hàng hoàn thành

handleCompleteRows():

```
handleCompleteRows() {
  const completedRows = []; // Danh sách các hàng hoàn thành
  for (let row = 0; row < this.grid.length; row++) {
    if (this.grid[row].every(cell => cell !== WHITE_ID)) {
      completedRows.push(row);
    }
  }
  if (completedRows.length > 0) {
    // Tạo hiệu ứng tan biến
    this.fadeOutRows(completedRows).then(() => {
      // Xóa các hàng đã hoàn thành sau hiệu ứng
      const newRows = Array.from({ length: completedRows.length }, () => Array(COLS).fill(WHITE_ID));
      const remainingRows = this.grid.filter((_, row) => !completedRows.includes(row));
      this.grid = [...newRows, ...remainingRows];
      if(completedRows.length ==1){
        this.handleScore(completedRows.length * 10); // Cập nhật điểm số
      }
      else if(completedRows.length == 2){
        this.handleScore(completedRows.length * 10 + 5);
        showBonus("+5 Bonus!");
      }
      else if(completedRows.length == 3){
        this.handleScore(completedRows.length * 10 + 7);
        showBonus("+7 Bonus!");
      }
      else{
        this.handleScore(completedRows.length * 10 + 10);
        showBonus("+10 Bonus!");
      }

      audioCplRows.play(); // Phát âm thanh hoàn thành hàng
      this.drawBoard(); // Vẽ lại bảng
    });
  }
}
```

Hình 11: Phương thức handleCompleteRows()

Kiểm tra và xử lý các hàng hoàn thành trong lưới trò chơi. Nó xác định các hàng không chứa ô trắng (WHITE_ID), sau đó kích hoạt hiệu ứng tan biến bằng fadeOutRows(). Sau hiệu ứng, các hàng hoàn thành được thay thế bằng các hàng mới toàn ô trắng, kết hợp với các hàng còn lại để cập nhật lưới. Phương thức cũng tăng điểm số dựa trên số lượng hàng xóa và thêm thưởng tùy số hàng hoàn thành, hiển thị thông báo thưởng với showBonus(), phát âm thanh bằng audioCplRows và vẽ lại bảng trò chơi để hiển thị lưới mới.

fadeOutRows():

```
fadeOutRows(rows) {
  return new Promise(resolve => {
    let col = 0; // Bắt đầu từ cột đầu tiên
    const fadeInterval = setInterval(() => {
      rows.forEach(row => {
        if (col < COLS) {
          this.grid[row][col] = WHITE_ID; // Làm "biến mất" từng ô từ trái qua
        }
      });
      this.drawBoard();

      col++;
      if (col === COLS) { // Kết thúc khi tất cả các cột đã tan biến
        clearInterval(fadeInterval);
        resolve();
      }
    }, 38); // Mỗi lần làm mờ 1 cột cách nhau 38ms
  });
}
```

Hình 12: Phương thức fadeOutRows()

Tạo hiệu ứng tan biến từng cột cho các hàng được chỉ định. Nó trả về một Promise, trong đó mỗi cột của các hàng được làm biến mất dần từ trái sang phải bằng cách đặt các ô trong cột đó về trạng thái trắng (WHITE_ID). Phương thức sử dụng setInterval để thực hiện hiệu ứng theo từng khoảng thời gian (38ms). Sau khi tất cả các cột đã được xử lý, setInterval được dừng lại bằng clearInterval, và Promise được giải quyết để báo hiệu hiệu ứng đã hoàn thành.

Điểm số

handleScore():

```
handleScore(newScore) {
  this.score += newScore;
  document.getElementById('score').innerHTML = this.score;
  if (this.score > curr_point_db) {
    fb.writeUserData(playerName, this.score);
    curr_point_db = this.score;
    fb.getUserData();
    document.getElementById("high-score").innerHTML = this.score;
  }
}
```

Hình 13: Phương thức handleScore()

Quản lý việc cập nhật điểm số và đồng bộ hóa dữ liệu người chơi. Đầu tiên, nó cộng newScore vào điểm hiện tại this.score và hiển thị điểm mới trên giao diện thông qua phần tử HTML có id là score. Sau đó, nó kiểm tra nếu điểm hiện tại lớn hơn điểm cao nhất đã lưu trong cơ sở dữ liệu curr_point_db. Nếu đúng, phương thức cập nhật điểm cao nhất trong cơ sở dữ liệu bằng fb.writeUserData(playerName, this.score) (fb là một namespace được tạo bằng cách import * as fb from "./connect_firebase.js";), đồng

thời gán curr_point_db bằng giá trị điểm hiện tại. Tiếp theo, phương thức gọi fb.getUserData() để sắp xếp và hiển thị lên bảng xếp hạng. Cuối cùng, nó hiển thị điểm cao mới trên giao diện qua phần tử HTML có id là high-score.

Kết thúc trò chơi

handleGameOver():

```
handleGameOver() {
  this.gameOver = true;
  this.isPlaying = false;
  showGameOverMessage();
  audioGameOver.play();
}
```

Hình 14: Phương thức handleGameOver()

Xử lý khi trò chơi kết thúc bằng cách đặt trạng thái gameOver thành true và dừng trò chơi (isPlaying = false). Nó hiển thị thông báo kết thúc bằng cách gọi hàm showGameOverMessage() và phát âm thanh game over qua audioGameOver.

Reset():

```
reset() {
  this.score = 0;
  document.getElementById('score').innerText = 0; // Reset điểm số về 0
  this.grid = this.generateWhiteBoard();
  this.gameOver = false;
  this.drawBoard();
}
```

Hình 15: Phương thức reset()

Được sử dụng để khởi tạo lại trạng thái ban đầu của trò chơi. Nó đặt lại điểm số về 0, cập nhật hiển thị điểm số trên giao diện, tạo một bảng trò chơi mới toàn ô trắng bằng cách gọi generateWhiteBoard(), thiết lập trạng thái trò chơi không kết thúc (gameOver = false), và vẽ lại bảng trò chơi trên giao diện.

3.2.2.2 Xây dựng lớp Brick

Lớp Brick được thiết kế để đại diện cho từng khối gạch trong trò chơi xếp gạch. Đây là thành phần quản lý cách mà khối gạch xuất hiện, di chuyển, xoay và tương tác với bảng (Board).

```
// Lớp điều khiển các khối gạch
class Brick {
  constructor(id) {
    this.id = id;
    this.layout = BRICK_LAYOUT[id];
    this.activeIndex = 0;//Hướng
    this.colPos = 3;
    this.rowPos = -2;
  }
  // Vẽ khối gạch lên bảng
  draw() {···
  }
  // Xóa khối gạch khi nó di chuyển
  clear() {···
  }
  // Di chuyển khối gạch sang trái
  moveLeft() {···
  }
  // Di chuyển khối gạch sang phải
  moveRight() {···
  }
  // Di chuyển khối gạch xuống dưới
  moveDown() {···
  }
  // Xoay khối gạch
  rotate() {···
  }
  // Kiểm tra va chạm của khối gạch
  checkCollision(nextRow, nextCol, nextLayout) {···
  }
  // Xử lý khi khối gạch đáp xuống
  handleLanded() {···
  }
}
```

Hình 16: Mô tả lớp Brick

Constructor:

Nhận vào một id để khởi tạo các giá trị mặc định khi tạo khối brick.

id: Đại diện cho loại khối gạch (mỗi loại có màu sắc và hình dạng riêng, được xác định trong BRICK_LAYOUT).

layout: Mảng 2D chứa thông tin bố cục của khối gạch (các ô nào được lấp đầy).

activeIndex: Chỉ số để xác định hướng hiện tại của khối gạch (xoay 0°, 90°, 180°, 270°).

colPos và rowPos: Vị trí hiện tại của khối gạch trên bảng, được xác định bởi cột và hàng (ở đây chọn vị trí xuất hiện là cột thứ 3 và hàng là -2 tức là bên ngoài bảng).

Phương thức

Vẽ và xóa khối gạch

draw():

```
draw() {
  for (let row = 0; row < this.layout[this.activeIndex].length; row++) {
    for (let col = 0; col < this.layout[this.activeIndex][0].length; col++) {
      if (this.layout[this.activeIndex][row][col] !== WHITE_ID) {
        board.drawCell(col + this.colPos, row + this.rowPos, this.id);
      }
    }
  }
}
```

Hình 17: Phương thức draw()

Vẽ hình dạng hiện tại của một khối trên bảng trò chơi. Nó lặp qua từng hàng và cột của ma trận bố cục khối (this.layout[this.activeIndex]) và xác định các ô không phải màu trắng (WHITE_ID). Với mỗi ô hợp lệ, nó tính toán vị trí chính xác trên bảng dựa vào this.colPos và this.rowPos, sau đó gọi phương thức board.drawCell() để vẽ ô tại vị trí đó với ID của khối (this.id).

clear():

```
clear() {
  for (let row = 0; row < this.layout[this.activeIndex].length; row++) {
    for (let col = 0; col < this.layout[this.activeIndex][0].length; col++) {
      if (this.layout[this.activeIndex][row][col] !== WHITE_ID) {
        board.drawCell(col + this.colPos, row + this.rowPos, WHITE_ID);
      }
    }
  }
}
```

Hình 18: Phương thức clear()

Xóa khối hiện tại khỏi bảng trò chơi bằng cách lặp qua ma trận bố cục khối (this.layout[this.activeIndex]). Với mỗi ô không phải màu trắng (WHITE_ID), nó tính toán vị trí trên bảng dựa vào this.colPos và this.rowPos, sau đó gọi phương thức board.drawCell() để vẽ lại ô đó với màu trắng (WHITE_ID), trả bảng về trạng thái không chứa khối ở vị trí này.

Di chuyển khối gạch

moveLeft() và moveRight():

```
moveLeft() {
  if (!this.checkCollision(this.rowPos, this.colPos - 1, this.layout[this.activeIndex])) {
    this.clear();
    this.colPos--;
    this.draw();
  }
}
moveRight() {
  if (!this.checkCollision(this.rowPos, this.colPos + 1, this.layout[this.activeIndex])) {
    this.clear();
    this.colPos++;
    this.draw();
  }
}
```

Hình 19: Phương thức moveLeft() và moveRight()

Di chuyển khối gạch sang trái hoặc phải. Trước khi di chuyển, chúng kiểm tra va chạm bằng checkCollision() ở vị trí dự kiến. Nếu không có va chạm, phương thức sẽ xóa khối khỏi vị trí hiện tại bằng clear(), cập nhật cột vị trí (colPos) theo hướng tương ứng, và vẽ lại khối ở vị trí mới bằng draw().

moveDown():

```
moveDown() {
  if (!this.checkCollision(this.rowPos + 1, this.colPos, this.layout[this.activeIndex])) {
    this.clear();
    this.rowPos++;
    this.draw();
    return;
  }
  this.handleLanded(); // Xử lý khi khối gạch đáp xuống
  generateNewBrick();
}
```

Hình 20: Phương thức moveDown()

Di chuyển khối gạch xuống một hàng nếu không có va chạm tại vị trí dự kiến. Nếu không có va chạm, nó xóa khối khỏi vị trí hiện tại bằng clear(), tăng giá trị hàng (rowPos), và vẽ lại khối bằng draw(). Nếu xảy ra va chạm, phương thức gọi phương thức handleLanded() để xử lý khi khối đáp xuống và tạo khối gạch mới bằng phương thức generateNewBrick().

Xoay khối gạch

rotate():

```
rotate() {
  if (!this.checkCollision(this.rowPos, this.colPos, this.layout[(this.activeIndex + 1) % 4])) {
    this.clear();
    this.activeIndex = (this.activeIndex + 1) % 4;
    this.draw();
  }
}
```

Hình 21: Phương thức rotate()

Xoay khối gạch theo chiều kim đồng hồ nếu không có va chạm với các ô khác sau khi xoay. Nó kiểm tra va chạm với hình dạng kế tiếp của khối (bằng cách sử dụng chỉ số xoay activeIndex + 1 modulo 4 để đảm bảo quay vòng qua các trạng thái hình dạng của khối). Nếu không có va chạm, phương thức sẽ xóa khối khỏi vị trí hiện tại, cập nhật chỉ số hình dạng (activeIndex), và vẽ lại khối ở vị trí mới.

Kiểm tra va chạm

checkCollision():

```
checkCollision(nextRow, nextCol, nextLayout) {
  for (let row = 0; row < nextLayout.length; row++) {
    for (let col = 0; col < nextLayout[row].length; col++) {
      if (nextLayout[row][col] !== WHITE_ID) { // Chỉ kiểm tra các ô thực sự
        if (
          col + nextCol < 0 || // Vượt biên trái
          col + nextCol >= COLS || // Vượt biên phải
          row + nextRow >= ROWS || // Vượt biên dưới
          (row + nextRow >= 0 && board.grid[row + nextRow][col + nextCol] !== WHITE_ID) // Đụng gạch khác
        ) {
          return true; // Có va chạm
        }
      }
    }
  }
  return false; // Không có va chạm
}
```

Hình 22: Phương thức checkCollision()

Kiểm tra xem khối gạch có va chạm với biên hoặc các ô đã tồn tại trên bảng hay không bằng cách duyệt qua từng ô trong bố cục mới (nextLayout). Nó bỏ qua các ô trống (giá trị WHITE_ID) và kiểm tra các điều kiện: ô nằm ngoài biên trái, phải, hoặc dưới của bảng, hoặc trùng với một ô đã có gạch trong board.grid. Nếu phát hiện bất kỳ va chạm nào, phương thức trả về true, ngược lại trả về false.

Xử lý khi khối gạch đáp xuống

handleLanded():

```
handleLanded() {
  if (this.rowPos <= 0) {
    board.handleGameOver();
    return;
  }
  for (let row = 0; row < this.layout[this.activeIndex].length; row++) {
    for (let col = 0; col < this.layout[this.activeIndex][0].length; col++) {
      if (this.layout[this.activeIndex][row][col] !== WHITE_ID) {
        board.grid[row + this.rowPos][col + this.colPos] = this.id;
      }
    }
  }
  board.handleCompleteRows();
  board.drawBoard();
}
```

Hình 23: Phương thức handleLanded()

Xử lý khi khối gạch đã đáp xuống vị trí cuối cùng. Nếu khối gạch nằm ở hàng trên cùng (rowPos <= 0), trò chơi kết thúc bằng cách gọi handleGameOver(). Ngược lại, nó gắn các ô của khối gạch hiện tại vào bảng (board.grid) dựa trên vị trí hiện tại (rowPos, colPos). Sau đó, hàm kiểm tra và xử lý các hàng đầy bằng board.handleCompleteRows() rồi vẽ lại bảng bằng board.drawBoard().

3.2.2.3 Xây dựng các hàm cần thiết

```
// Hàm tạo và hiển thị khối gạch mới
function generateNewBrick() {···
}
// Hàm hiển thị khối gạch tiếp theo
function drawNextBrick() {···
}
function showBonus(bonusText) {···
}
// Hàm lấy tên người chơi
export function getName(){···
}
//Hàm hiển thị thông báo Game Over
function showGameOverMessage() {···
}
```

Hình 24: Mô tả các hàm có trong game

Hàm generateNewBrick():

```
function generateNewBrick() {
  brick = nextBrick || new Brick(Math.floor(Math.random() * BRICK_LAYOUT.length));
  nextBrick = new Brick(Math.floor(Math.random() * BRICK_LAYOUT.length));
  drawNextBrick();
}
```

Hình 25: Hàm generateNewBrick()

Chức năng: Tạo và cập nhật khối gạch hiện tại (brick) và khối gạch kế tiếp (nextBrick).

Cách hoạt động: Đầu tiên, hàm kiểm tra nếu biến nextBrick đã tồn tại, viên gạch hiện tại (brick) sẽ được gán bằng giá trị của nextBrick; nếu không, brick sẽ được khởi tạo mới bằng cách tạo một đối tượng Brick với kiểu ngẫu nhiên. Kiểu ngẫu nhiên này được xác định bởi biểu thức Math.floor(Math.random() * BRICK_LAYOUT.length), trong đó:

Math.random() tạo một số ngẫu nhiên trong khoảng từ 0 (bao gồm) đến 1 (không bao gồm).

Nhân số này với BRICK_LAYOUT.length giúp mở rộng phạm vi thành từ 0 đến độ dài của mảng BRICK_LAYOUT (không bao gồm giá trị lớn nhất).

Math.floor() làm tròn xuống số này để đảm bảo nó là một số nguyên hợp lệ, dùng làm chỉ số cho kiểu viên gạch trong mảng BRICK_LAYOUT.

Sau khi xác định viên gạch hiện tại, viên gạch tiếp theo (nextBrick) cũng được khởi tạo với cách tương tự để chuẩn bị cho lượt tiếp theo. Cuối cùng, hàm gọi drawNextBrick() để hiển thị viên gạch tiếp theo lên giao diện.

Hàm drawNextBrick():

```
function drawNextBrick() {
  nextCtx.clearRect(0, 0, nextBrickCanvas.width, nextBrickCanvas.height);
  for (let row = 0; row < nextBrick.layout[0].length; row++) {
    for (let col = 0; col < nextBrick.layout[0][0].length; col++) {
      if (nextBrick.layout[0][row][col] !== WHITE_ID) {
        nextCtx.fillStyle = COLOR_PALETTE[nextBrick.id];
        nextCtx.fillRect(
          col * (BLOCK_SIZE / 2),
          row * (BLOCK_SIZE / 2),
          BLOCK_SIZE / 2,
          BLOCK_SIZE / 2
        );
        nextCtx.strokeRect(
          col * (BLOCK_SIZE / 2),
          row * (BLOCK_SIZE / 2),
          BLOCK_SIZE / 2,
          BLOCK_SIZE / 2
        );
      }
    }
  }
}
```

Hình 26: Hàm drawNextBrick()

Chức năng: Hiển thị khối gạch kế tiếp trong vùng canvas phụ.

Cách hoạt động: Xóa sạch canvas hiện tại của khối gạch kế tiếp. Vẽ từng ô của khối gạch kế tiếp với màu sắc tương ứng từ COLOR_PALETTE và các kích thước bằng phân nửa kích thức của gạch trên lưới.

Hàm showBonus():

```
function showBonus(bonusText) {
  const bonusMessage = document.getElementById("bonus-message");
  // Hiển thị điểm bonus
  bonusMessage.textContent = bonusText;
  bonusMessage.style.opacity = "1";
  bonusMessage.style.bottom = "45%";
  // Ẩn sau 1 giây
  setTimeout(() => {
    bonusMessage.style.opacity = "0";
    bonusMessage.style.bottom = "-50px";
  }, 1000);
}
```

Hình 27: Hàm showBonus()

Chức năng: Hiển thị thông báo điểm thưởng khi hoàn thành nhiều hàng.

Cách hoạt động: Cập nhật nội dung (bonusText) và vị trí của thông báo thưởng. Sau 1 giây, ẩn thông báo.

Hàm showGameOverMessage():

```
function showGameOverMessage() {
  const gameOverOverlay = document.getElementById('game-over');
  gameOverOverlay.style.display = 'flex'; // Hiển thị phần tử thông báo
  audioBg.pause();
  audioBg.currentTime = 0;
  // Xử lý sự kiện khi nhấn nút OK
  document.getElementById('game-over-btn').onclick = () => {
    gameOverOverlay.style.display = 'none'; // Ẩn thông báo
    board.reset(); // Reset trò chơi
    document.getElementById('play').innerText = 'Play';
  };
}
```

Hình 28: Hàm showGameOverMessage()

Chức năng: Hiển thị thông báo khi trò chơi kết thúc.

Cách hoạt động: Nó thay đổi thuộc tính style.display của phần tử HTML có ID game-over thành "flex" để hiển thị thông báo, đồng thời tạm dừng và đặt lại nhạc nền (audioBg). Khi người dùng nhấn nút "OK" (phần tử có ID game-over-btn), hàm ẩn thông báo, đặt lại trò chơi bằng board.reset(), và thay đổi nút "Play" trên giao diện về trạng thái ban đầu.

3.2.3 Xây dựng cơ sở dữ liệu Firebase Realtime Database

Import một số hàm cần thiết

```
import { initializeApp } from "https://www.gstatic.com/firebasejs/11.1.0/firebase-app.js";
import { getAnalytics } from "https://www.gstatic.com/firebasejs/11.1.0/firebase-analytics.js";
import {getDatabase,set,get,ref,onValue,child,
} from "https://www.gstatic.com/firebasejs/11.1.0/firebase-database.js";
import * as scrt from "./script.js";
```

Hình 29: Import một số hàm cần thiết

Khởi tạo Firebase: Hàm initializeApp dùng để khởi tạo ứng dụng Firebase với các cấu hình được cung cấp.

Sử dụng Firebase Analytics: Hàm getAnalytics cho phép sử dụng dịch vụ Analytics của Firebase để theo dõi và phân tích hoạt động của ứng dụng.

Sử dụng Firebase Realtime Database:

getDatabase: Lấy một tham chiếu đến cơ sở dữ liệu thời gian thực của Firebase.

set, get, ref, onValue, child: Các hàm này hỗ trợ việc đọc, ghi, và theo dõi dữ liệu trong cơ sở dữ liệu thời gian thực.

import * as scrt from "./script.js": Nhập một module JS và tải tất cả các phần tử được xuất từ file script.js dưới namespace scrt để sử dụng trong các hàm tiếp theo.

Khởi tạo ứng dụng Firebase:

```
const firebaseConfig = {
  apiKey: "AIzaSyD8XgkCQVCztRKGCQ0Cmr0mErwftfeYTG4",
  authDomain: "dbleaderboard-3a36a.firebaseapp.com",
  databaseURL: "https://dbleaderboard-3a36a-default-rtdb.firebaseio.com",
  projectId: "dbleaderboard-3a36a",
  storageBucket: "dbleaderboard-3a36a.firebasestorage.app",
  messagingSenderId: "807096461470",
  appId: "1:807096461470:web:bed78545c894b1ddc2338f",
  measurementId: "G-0V82C37234"
};
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
```

Hình 30: Khởi tạo ứng dụng Firebase

firebaseConfig: Đây là một đối tượng chứa thông tin cấu hình cần thiết để kết nối ứng dụng với dự án Firebase trên đám mây, được cung cấp khi khởi tạo cơ sở dữ liệu.

Dùng hàm initializeApp để khởi tạo ứng dụng Firebase với thông tin cấu hình từ firebaseConfig. Biến app là một đối tượng đại diện cho ứng dụng Firebase đã được khởi tạo.

Hàm getAnalytics lấy một thể hiện của dịch vụ Google Analytics, liên kết với ứng dụng Firebase vừa khởi tạo. Biến analytics có thể được dùng để theo dõi và phân tích dữ liệu hoạt động trong ứng dụng (như sự kiện người dùng, thông tin phiên,…).

Hàm writeUserData():

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

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

Hình 31: Hàm writeUserData()

Sử dụng để lưu thông tin người chơi vào Firebase Realtime Database. Nó nhận hai tham số, name (tên người chơi) và point (điểm số), sau đó kết nối đến cơ sở dữ liệu bằng getDatabase() và tạo một tham chiếu đến nhánh leaderboard theo tên người chơi ("leaderboard/" + name). Dữ liệu được ghi vào cơ sở dữ liệu thông qua hàm set, trong đó lưu trữ một đối tượng gồm tên (name) và điểm số (score: point).

Hàm getUserData():

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

  onValue(postsRef, (snapshot) => {
    const data = snapshot.val();
    let leaderboard = Object.keys(data).map((key) => {
      return { name: key, ...data[key] };
    });
    leaderboard.sort((a, b) => b.score - a.score);
    WriteScore("leaderboard", leaderboard);
  });
}
```

Hình 32: Hàm getUserData()

Kết nối đến cơ sở dữ liệu Firebase qua getDatabase(), tạo tham chiếu đến nhánh leaderboard qua ref(db, "leaderboard"). Sau đó, hàm sử dụng onValue để lắng nghe thay đổi tại nhánh này. Khi có thay đổi, callback được gọi với snapshot, chứa dữ liệu bảng xếp hạng. Dữ liệu này ban đầu là một đối tượng, được chuyển thành một mảng các đối tượng có cấu trúc { name: key, score: data[key].score }, sau đó sắp xếp mảng theo điểm

số giảm dần. Cuối cùng, mảng đã sắp xếp được truyền vào hàm WriteScore() để hiển thị lại xếp hạng.

Hàm WriteScore():

```
export function WriteScore(tagName, player) {
  document.getElementById(tagName).innerHTML = "";
  let scoreText = "";
  for (let i = 0; i < 10; i++) {
    scoreText +=
      "Hạng " + (i + 1) + ": " + player[i].name + "-" + player[i].score + "<br>";
  }
  for (let i = 0; i < player.length; i++) {
    if(player[i].name === scrt.getName()){
      scoreText += ("Bạn: Hạng " + (i+1))
    }
  }
  document.getElementById(tagName).innerHTML = scoreText;
}
```

Hình 33: Hàm WriteScore()

Sử dụng để hiển thị bảng xếp hạng trên trang web. Đầu tiên, nó xóa nội dung hiện tại của phần tử HTML có id là tagName. Sau đó, hàm xây dựng một chuỗi scoreText chứa thông tin về 10 người chơi đứng đầu, bao gồm tên và điểm số của họ, và thêm vào nội dung HTML. Tiếp theo, nó kiểm tra nếu tên người chơi hiện tại (lấy từ hàm scrt.getName()) có trong danh sách bảng xếp hạng, thì thêm dòng "Bạn: Hạng X" vào bảng xếp hạng, với X là vị trí của người chơi đó. Cuối cùng, hàm cập nhật lại phần tử HTML với bảng xếp hạng mới.

Hàm getUserScoreOrDefault():

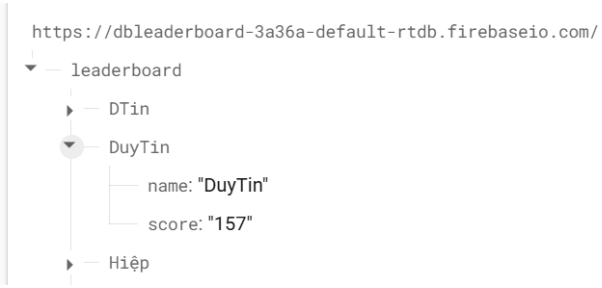
```
export async function getUserScoreOrDefault(name) {
  const db = getDatabase();
  const userRef = ref(db);
  const snapshot = await get(child(userRef, `leaderboard/${name}`));
  if (snapshot.exists()) {
    const userData = snapshot.val();
    return userData.score; // Người chơi đã tồn tại, trả về điểm
  } else {
    return 0; // Người chơi không tồn tại, trả về 0
  }
}
```

Hình 34: Hàm getUserScoreOrDefault()

Là một hàm bất đồng bộ dùng để lấy điểm số của người chơi từ cơ sở dữ liệu theo tên. Đầu tiên, hàm kết nối tới cơ sở dữ liệu bằng getDatabase() và tạo tham chiếu tới toàn bộ cơ sở dữ liệu thông qua ref(db). Sau đó, nó sử dụng get() kết hợp với child() để truy cập đến đường dẫn cụ thể leaderboard/{name}, nơi lưu thông tin của người chơi.

Tiếp theo, hàm kiểm tra xem dữ liệu tại đường dẫn đó có tồn tại hay không bằng snapshot.exists(). Nếu tồn tại, nó trích xuất và trả về giá trị điểm số của người chơi có tên tương ứng bằng snapshot.val().score. Nếu dữ liệu không tồn tại, tức là người chơi chưa được ghi nhận trong bảng xếp hạng, hàm trả về giá trị mặc định là 0.

Mô tả cấu trúc cơ sở dữ liệu của người chơi lưu trên Firebase Realtime Database:



Hình 35: Mô tả cấu trúc cơ sở dữ liệu lưu trên Firebase Realtime Database

3.2.4 Lập trình điều khiển trò chơi bằng cử chỉ tay

Khởi tạo các phần tử và biến

```
const videoElement = document.querySelector('.input_video');
const canvasElement = document.querySelector('.output_canvas');
const canvasCtx = canvasElement.getContext('2d');
const message2Hand = document.getElementById('message-2hand');

let finger_id = [4, 8, 12, 16, 20];
let previousMoveTime = 0; // Thời gian thực thi trước đó
```

Hình 36: Khởi tạo các phần tử và biến

Chọn các phần tử DOM cần thiết (video, canvas, message-2hand).

finger_id: Mảng chứa các chỉ số của các ngón tay trong MediaPipe.

previousMoveTime: Lưu lại thời gian lần cuối thực thi cử chỉ để giới hạn tốc độ thực thi.

Hàm nhận diện cử chỉ detect_gesture():

```
function detect_gesture(fingers) {
  // So sánh trực tiếp từng phần tử
  if (fingers[0] === 1 && fingers[1] === 1 && fingers[2] === 1 && fingers[3] === 1 && fingers[4] === 1) {
    return "ROTATE";
  } else if (fingers[0] === 0 && fingers[1] === 1 && fingers[2] === 1 && fingers[3] === 0 && fingers[4] === 0) {
    return "DOWN";
  } else if (fingers[0] === 1 && fingers[1] === 0 && fingers[2] === 0 && fingers[3] === 0 && fingers[4] === 0) {
    return "LEFT";
  } else if (fingers[0] === 0 && fingers[1] === 0 && fingers[2] === 0 && fingers[3] === 0 && fingers[4] === 1) {
    return "RIGHT";
  }
  return null;  // Trả về null nếu không khớp
}
```

Hình 37: Hàm detect_gesture()

Nhận đầu vào là mảng fingers đại diện trạng thái của 5 ngón tay (1: giơ lên, 0: cụp xuống).

So sánh fingers với các mẫu xác định:

[1, 1, 1, 1, 1]: Tất cả ngón tay giơ lên → Lệnh "ROTATE".

[0, 1, 1, 0, 0]: Ngón trỏ và ngón giữa giơ lên → Lệnh "DOWN".

[1, 0, 0, 0, 0]: Chỉ ngón cái/ngón út giơ lên(bàn phải/trái)→ Lệnh "LEFT".

[0, 0, 0, 0, 1]: Chỉ ngón cái/ngón út giơ lên(bàn tay trái/phải) → Lệnh "RIGHT".

Trả về null nếu không có cử chỉ khớp.

Cấu hình MediaPipe Hands

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

hands.setOptions({
  maxNumHands: 2,
  modelComplexity: 1,
  minDetectionConfidence: 0.5,
  minTrackingConfidence: 0.5
});
```

Hình 38: Cấu hình MediaPipe

Tạo đối tượng Hands từ MediaPipe, cấu hình để tải tệp từ CDN.

Cài đặt tùy chọn:

maxNumHands: Nhận diện tối đa 2 bàn tay.

modelComplexity: Độ phức tạp của mô hình (1 là cao nhất).

minDetectionConfidence: Ngưỡng tối thiểu để phát hiện bàn tay (0.5).

minTrackingConfidence: Ngưỡng tối thiểu để theo dõi bàn tay (0.5).

Kích hoạt camera

```
const camera = new Camera(videoElement, {
  onFrame: async () => {
    await hands.send({ image: videoElement });
  },
  width: 1280,
  height: 720
});

camera.start();
```

Hình 39: Khởi tạo và kích hoạt camera

Khởi tạo và bắt đầu một đối tượng Camera để sử dụng video từ camera của người dùng. Mỗi khi có một khung hình mới từ video (onFrame), hình ảnh đó sẽ được gửi đến mô-đun nhận diện bàn tay (hands.send). Video được phát ở độ phân giải 1280x720 pixel, và camera sẽ bắt đầu quay khi gọi camera.start().

Vẽ các điểm landmark lên canvas

```
for (const landmarks of results.multiHandLandmarks) {
  drawLandmarks(canvasCtx, landmarks, { color: '#FF0000', lineWidth: 2 });
}
```

Hình 40: Vẽ các điểm landmark lên canvas

Vẽ bàn tay lên canvas bằng cách sử dụng dữ liệu từ results.multiHandLandmarks. Với mỗi tập hợp tọa độ bàn tay (landmarks), hàm drawLandmarks() vẽ các điểm đặc trưng này bằng màu đỏ với độ dày đường viền là 2, tạo hình ảnh trực quan về bàn tay trên canvas.

Xử lý kết quả nhận diện

```
hands.onResults((results) => {

  const currentTime = performance.now(); // Thời gian hiện tại

  canvasElement.width = videoElement.videoWidth;
  canvasElement.height = videoElement.videoHeight;

  canvasCtx.save();
  canvasCtx.clearRect(0, 0, canvasElement.width, canvasElement.height);
  canvasCtx.translate(canvasElement.width, 0); // Dịch chuyển toàn bộ canvas sang phải
  canvasCtx.scale(-1, 1); // Lật khung hình theo trục dọc
  canvasCtx.drawImage(results.image, 0, 0, canvasElement.width, canvasElement.height);
```

Hình 41: Xử lí kết quả nhận diện

Phần tử currentTime để cập nhật thời gian hiện tại, sử dụng để tính toán thời gian thực thi lệnh trước đó.

Cập nhật kích thước canvas (width và height) theo video.

Xóa nội dung cũ trên canvas và lật hình ảnh từ camera để tương ứng góc nhìn người dùng, sau đó vẽ lại canvas vơi hình ảnh đã được lật.

Nhận diện trạng thái các ngón tay

```
if (results.multiHandLandmarks) {
  results.multiHandLandmarks.forEach((landmarks, index) => {
    const handedness = results.multiHandedness[index].label; // 'Left' hoặc 'Right'
    if (results.multiHandLandmarks.length >= 2 && MPipe.style.display == 'block') {
      message2Hand.style.display = 'block';
      return;
    } else {
      let fingers = [0, 0, 0, 0, 0];
      if (handedness === 'Left') { // Do lật khung hình nên Left sẽ là bàn tay phải và ngược lại
          // // Xử lí sự kiện cho ngón cái - khối gạch qua trái
        if (landmarks[finger_id[0]].x > (landmarks[finger_id[0] - 2].x + 0.015))
          fingers[0] = 1;
        // Xử lí cho ngón út - khối gạch qua phải
        if (landmarks[finger_id[4]].y < landmarks[finger_id[4] - 2].y)
          fingers[4] = 1;
      }
      else{ // Bàn tay trái
          // Xử lí sự kiện cho ngón cái - khối gạch qua phải
        if (landmarks[finger_id[0]].x < (landmarks[finger_id[0] - 2].x - 0.015))
          fingers[4] = 1;
        // Xử lí cho ngón út - khối gạch qua trái
        if (landmarks[finger_id[4]].y < landmarks[finger_id[4] - 2].y)
          fingers[0] = 1;
      }
      // Xử lí cho các ngón còn lại
      for(let i = 1; i < 4; i++){
        if(landmarks[finger_id[i]].y < landmarks[finger_id[i] - 2].y)
          fingers[i]=1;
      }
```

Hình 42: Xử lí nhận diện trạng thái các ngón tay

Kiểm tra số lượng bàn tay

MediaPipe có thể phát hiện nhiều bàn tay. Nếu phát hiện nhiều hơn 1 bàn tay, hiển thị thông báo yêu cầu người chơi chỉ sử dụng một bàn tay và dừng xử lý.

Xác định trạng thái ngón tay

Mảng fingers được khởi tạo với 5 phần tử, mỗi phần tử biểu thị trạng thái của một ngón tay:

1 là giơ lên.

0 là cụp xuống.

Phân biệt bàn tay trái và phải

Sử dụng thông tin handedness từ MediaPipe để xác định bàn tay:

Nếu handedness là Left, nghĩa là bàn tay phải trong thực tế (do hình ảnh đã lật trên canvas).

Ngược lại, nếu handedness là Right, đó là bàn tay trái.

Xử lý trạng thái ngón cái và ngón út

Với bàn tay phải (label = "Left"):

Nếu tọa độ x (landmarks = 4) của ngón cái lớn hơn tọa độ x (landmarks = 2), ngón cái được đánh dấu hoạt động (di chuyển khối gạch sang trái).

Nếu tọa độ y (landmarks = 20) của ngón út nhỏ hơn tọa độ y (landmarks = 18), ngón út hoạt động (di chuyển khối gạch sang phải).

Với bàn tay trái (label = "Right"):

Kiểm tra ngược lại ngón cái/út để di chuyển khối gạch sang trái/phải.

Xử lý trạng thái các ngón còn lại

Với cả hai tay, nếu tọa độ y (landmarks = 12, 14, 16) của ngón nhỏ hơn tọa độ y (landmarks = 10, 12, 14), ngón đó được đánh dấu hoạt động.

Thực thi lệnh trò chơi

```
const command = detect_gesture(fingers);
if (!board.gameOver && board.isPlaying && !isPaused && (currentTime - previousMoveTime > 300)) {
  switch (command) {
    case 'LEFT':
      brick.moveLeft();
      break;
    case 'RIGHT':
      brick.moveRight();
      break;
    case 'DOWN':
      brick.moveDown();
      break;
    case 'ROTATE':
      brick.rotate();
      break;
  }
  previousMoveTime = currentTime;
}
```

Hình 43: Thực thi lệnh trò chơi dựa trên cử chỉ

Tạo mới một phần tử command và gán giá trị bằng hàm detect_gesture (truyền vào hàm mảng fingers sau khi duyệt qua các điều kiện trạng thái của ngón tay) để nhận về các giá trị LEFT, RIGHT, DOWN, ROTATE tương ứng với di chuyển và xoay khối gạch. Nếu lệnh từ command trả về thuộc 1 trong các lệnh LEFT, RIGHT, DOWN, ROTATE thì gọi hàm tương ứng.

Kiểm tra trạng thái trò chơi (gameOver, isPlaying, isPaused) và thời gian thực thi lệnh trước đó để tránh lệnh được thực hiện liên tục nếu người chơi không cụp tay kịp.

Đặt previousMoveTime lại bằng currentTime để tiếp tục vòng lặp mới.

CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

Sau quá trình thực hiện đồ án "Phát triển game xếp gạch trên web sử dụng MediaPipe với tương tác điều khiển bằng cử chỉ tay", các kết quả chính đạt được bao gồm:

Hiệu năng của hệ thống

Độ chính xác tương đối cao: Việc nhận diện cử chỉ tay đạt độ chính xác tương đối cao trong điều kiện ánh sáng đủ và người chơi ở đủ gần, các thao tác điều khiển game sẽ diễn ra mượt mà.

Hệ thống chạy ổn định trên trình duyệt web mà không yêu cầu phần cứng cao cấp hoặc cài đặt phần mềm bổ sung.

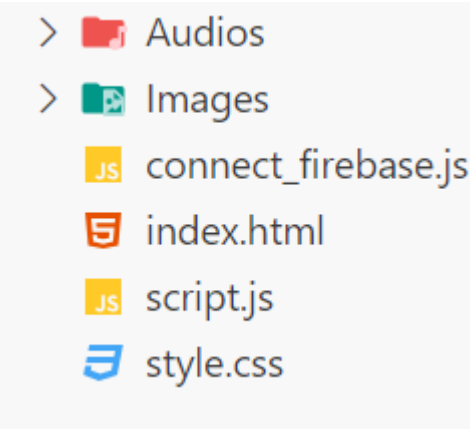
Trải nghiệm người dùng

Người chơi có thể sử dụng các cử chỉ tay đơn giản để điều khiển các khối gạch, tạo cảm giác mới mẻ và hứng thú so với cách chơi truyền thống bằng bàn phím.

Game có giao diện trực quan, dễ sử dụng, giúp người chơi nhanh chóng làm quen với cách điều khiển.

Chạy tốt trên các trình duyệt phổ biến như Chrome, Firefox, và Edge, đồng thời tương thích với các thiết bị như laptop, PC có webcam.

Tổ chức thư mục:



Hình 44: Tổ chức thư mục

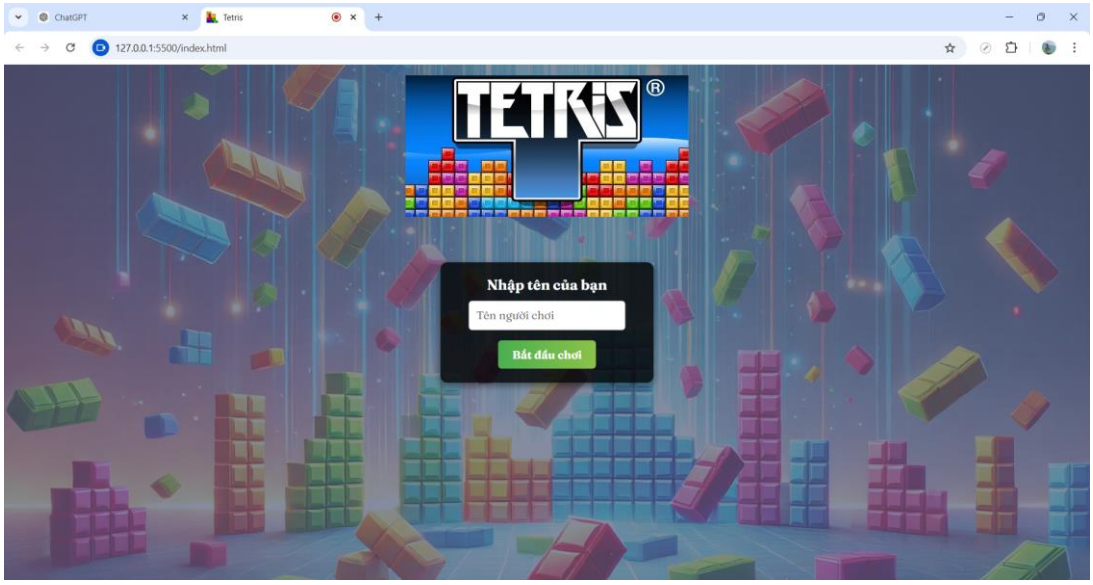
Thư mục Audios chứa âm thanh của trò chơi.

Thư mục Images chứa ảnh có trong trò chơi.

Các file index.html, script.js và style.css chứa mã nguồn để tạo nên trò chơi, bao gồm giao diện, logic game và điều khiển bằng cử chỉ tay. File connect_firebase.js chịu trác nhiệm kết nối và thao tác với cơ sở dữ liệu thời gian thực (đọc, ghi dữ liệu).

Các giao diện và chức năng chính

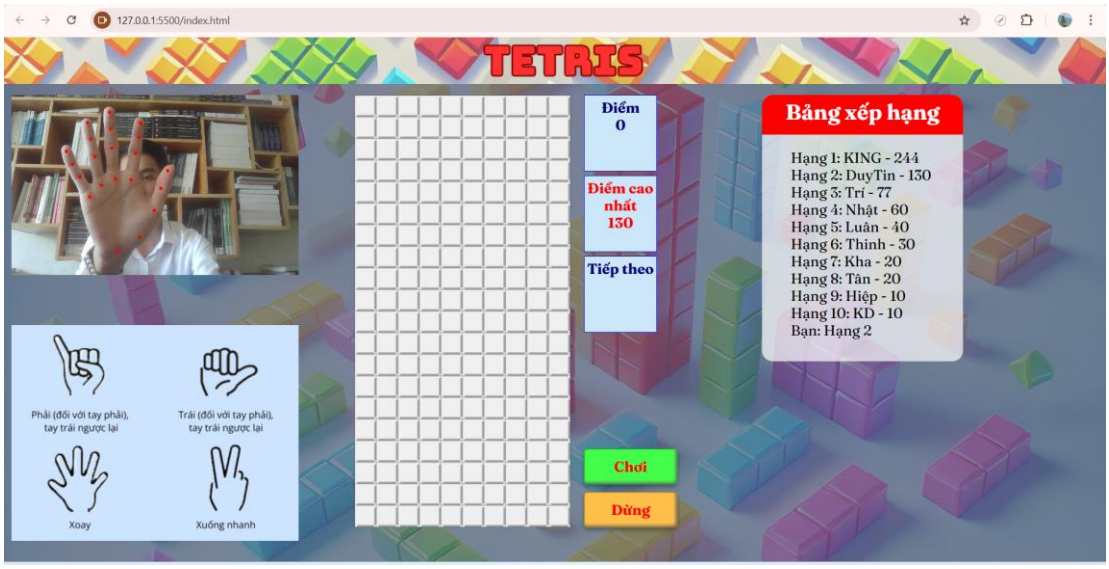
Giao diện đăng nhập:



Hình 45: Giao diện đăng nhập

Người chơi sẽ nhập tên đăng nhập để truy cập vào trò chơi. Khi người chơi nhấn nút "Bắt đầu chơi", hệ thống sẽ kiểm tra nếu tên đăng nhập rỗng thì sẽ hiện hộp thoại thông báo.

Màn hình chính:



Hình 46: Giao diện chơi game

Hiển thị giao diện chơi game bao gồm: Khung camera (phía trên góc trái), hướng dẫn chơi, giao diện trò chơi và bảng xếp hạng.

Chế độ chơi xếp gạch:

Người chơi nhấn nút "Chơi" để bắt đầu trò chơi. Khi nút "Chơi" được nhấn, gạch sẽ xuất hiện trên lưới và gạch tiếp theo sẽ xuất hiện ở ô "Tiếp theo". Cùng lúc đó, nhạc nền sẽ được bật.

Người chơi sẽ phải dùng cử chỉ tay để điều khiển các khối gạch: giơ ngón cái/ngón út của bàn tay trái để điều khiển gạch qua phải/trái, giơ ngón cái/ngón út của bàn tay phải để điều khiển gạch qua trái/phải, giơ ngón trỏ và ngón giữa để di chuyển gạch xuống nhanh và xòe bàn tay để xoay gạch. Nếu phát hiện nhiều hơn 2 bàn tay, một thông báo "Chỉ sử dụng 1 bàn tay" sẽ xuất hiện tại khung camera và mọi hành động điều khiển sẽ tạm dừng cho đến khi hệ thống chỉ phát hiện một bàn tay.

Khi muốn dừng trò chơi, người chơi nhấm nút "Dừng", trò chơi và nhạc nền sẽ dừng. Để tiếp tục, người chơi nhấn vào nút "Tiếp tục", trò chơi và nhạc nền sẽ phát tiếp tục. Ngoài ra, nếu người chơi muốn làm mới lại trò chơi thì nhấn vào nút "Chơi lại", toàn bộ gạch trên bảng sẽ bị xóa và điểm hiện tại sẽ bằng 0.

Nếu người chơi hoàn thành số hàng bất kì, hàng hoàn thành sẽ có hiệu ứng tan biến từ trái qua và âm thanh xóa hàng sẽ phát. Cách tính điểm như sau: Nếu người chơi hoàn thành 1 hàng, người chơi được cộng 10 điểm vào điểm hiện tại; nếu hoàn thành 2 hàng, người chơi sẽ được cộng 25 điểm, bao gồm điểm hoàn thành hàng là 20 điểm và điểm thưởng là 5 điểm, khi đó sẽ xuất hiện thông báo điểm thưởng trong giao diện chơi; tương tự với số hàng hoàn thành là 3, người chơi sẽ được cộng 37 điểm (trong đó có 7 điểm thưởng); số hàng hoàn thành là 4, người chơi sẽ được cộng 50 điểm (trong đó có 10 điểm thưởng). Do số hàng hoàn thành 1 lần xếp gạch tối đa là 4 (số ô gạch có chiều dài lớn nhất của một khối gạch) nên điểm thưởng tối đa có thể được cộng là 10 điểm. Khi người chơi có điểm hiện tại lớn hơn điểm cao nhất, điểm cao nhất sẽ được cập nhật bằng điểm hiện tại ngay lập tức.

Đối với bảng xếp hạng, tên người chơi sẽ được cập nhật lên bảng xếp hạng khi đăng nhập vào trò chơi với vị trí phụ thuộc vào điểm cao nhất đã chơi trước đó hoặc điểm bằng 0 nếu lần đầu chơi. Trong quá trình chơi, nếu người chơi đạt được điểm số

cao hơn so với bất kì người chơi nào có trên bảng xếp hạng, người chơi sẽ được thăng hạng.

Trò chơi sẽ kết thúc khi người chơi để gạch chạm dòng trên cùng của lưới. Khi đó, một hộp thoại thông báo "Game over" sẽ xuất hiện cùng với âm thanh. Khi người chơi nhấn vào nút "OK", toàn bộ gạch trên bảng sẽ xóa, điểm hiện tại sẽ được cập nhật về 0, người chơi nhấn vào nút chơi để bắt đầu lại từ đầu.

Những điểm nổi bật

Hệ thống tận dụng khả năng của MediaPipe để nhận diện và theo dõi cử chỉ tay trong thời gian thực cùng với đó là lưu trữ dữ liệu thời gian thực, đảm bảo tính ổn định.

Kết hợp HTML, CSS, và JS để phát triển game, mang lại sự thuận tiện và khả năng mở rộng trong tương lai.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Đồ án đã đạt được những kết quả sau:

Xây dựng thành công game xếp gạch trên nền tảng web với khả năng điều khiển bằng cử chỉ tay thay thế cho phím bấm truyền thống, thông tin người chơi được lưu trên Firebase Realtime Database.

Tích hợp công nghệ MediaPipe để nhận diện cử chỉ tay trong thời gian thực, đảm bảo tính chính xác và hiệu quả.

Những đóng góp mới của đồ án:

Cung cấp một giải pháp điều khiển game mới, không cần thiết bị ngoại vi như chuột hay bàn phím.

Ứng dụng MediaPipe hiệu quả trong việc xử lý và theo dõi cử chỉ tay trực tiếp trên trình duyệt web.

Hướng phát triển

Tăng tính khả dụng:

Tối ưu hóa để trang web hỗ trợ reponsive, chạy tốt trên nhiều thiết bị khác nhau như smartphone, máy tính bảng.

Chuyển từ chạy local sang triển khai trực tuyến để người dùng dễ dàng truy cập qua internet.

Cải thiện trải nghiệm người dùng:

Bổ sung các chế độ chơi khác nhau, tạo sự đa dạng và hấp dẫn.

Nâng cấp giao diện trực quan và thân thiện hơn.

Nâng cao hiệu năng:

Cải thiện khả năng nhận diện cử chỉ trong môi trường ánh sáng yếu hoặc khi người chơi ở khoảng cách xa webcam.

Tích hợp thêm các thuật toán tối ưu để tăng tốc độ xử lý cử chỉ.

Hỗ trợ đa người chơi trực tuyến.

DANH MỤC TÀI LIỆU THAM KHẢO

[1] Wikipedia, "HTML," 15 11 2024. [Online]. Available: https://vi.wikipedia.org/wiki/HTML. [Accessed 12 20 2024].

[2] V. IDC, "HTML là gì? Nguyên lý hoạt động của HTML," 10 09 2024. [Online]. Available: https://viettelidc.com.vn/tin-tuc/html-la-gi-nguyen-ly-hoat-dong-cua-html. [Accessed 20 12 2024].

[3] TopDev, "CSS là gì?," [Online]. Available: https://topdev.vn/blog/css-la-gi/. [Accessed 20 12 2024].

[4] Đ. P. Miền v. P. T. T. Mai, Thiết kế và Lập trình Web, Bộ môn Công nghệ Thông tin, Khoa Kỹ thuật và Công nghệ, Trường Đại học Trà Vinh, 2014.

[5] C. L. V. Tiến, "JavaScript là gì? Kiến thức chi tiết về JavaScript cơ bản," 08 01 2024. [Online]. Available: https://vietnix.vn/javascript-la-gi/?gad_source=1&gclid=Cj0KCQiA4fi7BhC5ARIsAEV1YibnmBE7tAsZF_QvEEnx5LexIIDcPyQ4kIiS2q0_SLwVLgT1yFrWwGwaArAlEALw_wcB. [Accessed 20 12 2024].

[6] Q. Trần, "Mediapipe - Live ML solutions và ứng dụng vẽ bằng hands gestures," 12 10 2021. [Online]. Available: https://viblo.asia/p/mediapipe-live-ml-solutions-va-ung-dung-ve-bang-hands-gestures-gAm5ymOV5db. [Accessed 22 12 2024].

[7] C. L. V. Tiến, "Firebase là gì? Các chức năng cơ bản cần nên biết của Firebase," 24 04 2024. [Online]. Available: https://vietnix.vn/firebase-la-gi/?gad_source=1&gclid=CjwKCAiAg8S7BhATEiwAO2-R6l27HlM3yuQdIvi-3UIOfGlO1nMopRSsVNACfSOUA5nlNVy0Bb1k2xoCG58QAvD_BwE. [Accessed 22 12 2024].