CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập – Tự do – Hạnh phúc

BÀI BÁO CÁO THỰC TẬP CƠ SỞ

Lập trình game Tetris – Xếp gạch

Giảng viên hướng dẫn : Phạm Thị Kim Ngoan

Họ và tên sinh viên : Võ Thành Luân

Lớp : 59CNTT-1

Mã số sinh viên : 59131333

Nha Trang, ngày 15 tháng 12 năm 2019.

NHẬN XÉT KẾT QUẢ THỰC TẬP

Sinh viên thực hiện: Võ Thành Luân – 59131333

Lớp: 59.CNTT-1

Khoa Công nghệ Thông tin, Trường Đại học Nha Trang.

Nội dung báo cáo:.....

.....

.....

.....

.....

.....

.....

.....

Đánh giá kết quả báo cáo: .....

.....

.....

.....

.....

.....

.....

Nha Trang, ngày ... Tháng 01 năm 2020

Giáo viên hướng dẫn

LỜI CẢM ƠN

Để hoàn thành bài báo cáo cho đề tài Thực tập Cơ sở, em xin gửi lời cảm ơn đến cô Phạm Thị Kim Ngoan, người đã tận ình hướng dẫn, giúp đỡ em trong quá trình học tập, tìm hiểu cũng như trong quá trình hoàn thành bài báo cáo lời cảm ơn sâu sắc nhất.

Trong quá trình làm bài báo cáo, khó tránh khỏi sai sót, rất mong các thầy bỏ qua. Đồng thời do kiến thức cũng như kinh nghiệm thực tiễn của bản thân còn hạn chế nên bài báo cáo này khó thể không tránh khỏi những thiếu sót, em rất mong nhận được ý kiến đóng góp của các thầy cô để em có thể học thêm được nhiều kinh nghiệm và sẽ hoàn thành tốt hơn bài báo cáo tốt nghiệp sắp tới.

Em xin chân thành cảm ơn!

MỤC LỤC

DANH MỤC CÁC SƠ ĐỒ VÀ HÌNH

CHƯƠNG 1. TỔNG QUAN

1.1. Đặt vấn đề

Lập trình Game đang là một nền công nghiệp phát triển mạnh trong thời đại hiện nay. Có thể thấy rõ, Game đang dần dần tăng tầm ảnh hưởng của mình trong các loại hình giải trí của con người trong thời đại mới. Về mặt tinh thần, Game là công cụ giải trí mới, mang lại niềm vui cho con người. Dưới góc độ kinh tế, nền công nghiệp Game là một ngành đem lại nhiều lợi nhuận và các ích lợi khác. Thứ nhất, nó tạo nên một nhu cầu mới cần nhiều lao động kỹ thuật cao, giải quyết nhiều công việc cho người lao động. Thứ hai, ngành này thúc đẩy nhiều ngành công nghiệp khác phát triển theo như viễn thông, thương mại điện tử, giáo dục, và một số ngành dịch vụ khác Thứ ba, ưu điểm của ngành này là một ngành công nghiệp không khói, nguyên liệu là tri thức, và đặc biệt đem lại lợi ích to lớn cho cộng đồng.

Theo thống kê của vài năm vừa qua, thị trường game trên toàn thế giới được ước tính lợi nhuận khoảng 75 tỉ dolar Mỹ mỗi năm. Đặc biệt, Việt Nam được xem là thị trường game lớn nhất của Đông Nam Á. Rất nhiều công ty game lớn ở Việt Nam như Vinagame – VNG, VTC Game, Soha Game,… có doanh thu nghìn tỷ đồng mỗi năm. Đồng thời cũng có rất nhiều cá nhân thành công trong lĩnh vực này như Nguyễn Hà Đông với Flappy Bird,…

Từ các ưu điểm trên em đã thực hiện đề tài: “Lập trình Game”. Khoá luận này cho em một hướng đi mới cho việc định hướng nghề nghiệp, cũng như phát triển thêm về kỹ năng lập trình, thiết kế hệ thống cũng như kỹ năng viết báo cáo.

1.2. Mục tiêu

Mục tiêu đầu tiên của đề tài Thực tập Cơ sở chính là để kiểm tra lại kỹ năng lập trình, khả năng áp dụng kiến thức đã học của bản thân vào thực tiễn, đồng thời cũng là cơ hội tốt để học hỏi, bổ sung, trau dồi thêm tri thức về chuyên môn. Đồng thời, đây là cơ hội rất tốt để có thêm kinh nghiệm về làm việc với dự án, là một trải nghiệm rất cần thiết để phục vụ cho công việc sau khi ra trường.

1.3. Đối tượng nghiên cứu

- Nghiên cứu ngôn ngữ lập trình Javascript.
- Nghiên cứu ngôn ngữ thiết kế HTML, CSS.
- Nghiên cứu kỹ thuật lập trình theo hướng đối tượng.
- Nghiên cứu phương pháp sử dụng phần mềm Visual Studio Code.
- Nghiên cứu trò chơi Tetris – Xếp gạch

1.4. Giới thiệu đề tài

Để thực hiện đề tài "Lập trình Game", em xin lấy game Tetris hay còn gọi là Xếp gạch – là một game rất quen thuộc với người Việt Nam vào những năm đầu của thế kỷ 21 làm nội dung của bài báo cáo.

1.4.1. Nguồn gốc

Tetris (tiếng Nga: Тетрис) là một trò chơi điện tử đầu tiên được thiết kế và phát triển bởi Alexey Pajitnov. Nó được tạo ra vào ngày 6 tháng 6 năm 1984,[1] trong lúc ông đang làm việc tại Trung tâm Tính toán Dorodnicyn của Viện hàn lâm Khoa học Liên Xô tại Moskva. Ông lấy tên của trò chơi từ tiền tố "tetra- của tiếng Hy Lạp, có nghĩa là "bốn" (mỗi bộ phần trong trò chơi, gọi là Tetromino, có bốn phần) và quần vợt (*tennis*), trò thể thao Pajitnov thích nhất.

Trò chơi (hay một biến thể của nó) được bán cho hầu hết mọi máy trò chơi điện tử và hệ điều hành máy tính, cũng như trong các máy tính đồ họa, điện thoại di động, máy nghe nhạc di động, PDA, và nhiều thứ khác. Tuy nhiều biến thể của *Tetris* được bán trên thị trường vào thập niên 1980, trò chơi cầm tay của Game Boy, ra mắt năm 1989 đã biến trò chơi thành một trong những trò chơi thịnh hành nhất. Số thứ 100 của tạp chí *Electronic Gaming Monthly* gọi Tetris là "Trò chơi Vĩ đại nhất trong Mọi thời đại". Nó đã bán được hơn 170 triệu phiên bản.

1.4.2. Lợi ích

Ba nhà khoa học: Erik Demaine, Susan Hohenberger và David Liben-Nowell tại Viện công nghệ nổi tiếng Massachusetts đã liên tục nghiên cứu để tìm ra sự bí ẩn phía sau của trò chơi này, và họ đã đưa ra kết luận với nhiều bất ngờ dành cho Tetris như sau:

"Tetris có nhiều điểm tương đồng với những câu hỏi toán học hóc búa nan giải nhất như kiểu Travelling Salesman Problem (NP-Hard problem). Nó như là việc tìm ra đường đi hiệu quả nhất cho một nhà buôn, khi ông ta phải đi qua rất nhiều địa điểm khác nhau".

Tôi được gì, nếu tôi chơi Tetris (Xếp gạch)?

Để ghi được điểm số trong Xếp gạch, bắt buộc bạn phải nhanh mắt và lẹ tay để sắp xếp các viên gạch lại với nhau thành những hàng kín. Càng lên cấp độ cao, tốc độ rơi của gạch càng tăng, do đó người chơi lâu trò Xếp gạch sẽ dần rèn luyện được cho bản thân khả năng tính toán và xử lý tình huống một cách thật nhạy bén để đạt được điểm số cao nhất trong thời gian nhanh nhất.

Với số lần chơi được lặp lại nhiều lần, Xếp gạch sẽ trở thành một "bộ môn" thật tuyệt vời để rèn luyện khả năng xử lý vấn đề của bộ não. Não liên tục hoạt động để giải quyết các bài toán hóc búa do Tetris đặt ra và luôn trong tư thế sẵn sàng trước các vấn đề người chơi chạm phải, đó sẽ tạo một tiền đề tốt để người chơi có thể nhanh chóng đưa ra lời lời đáp trước bất kỳ tình huống nào có thể xảy ra với bản thân ngay cả trong đời thực.

Học nhưng vẫn chơi – Chơi mà học lại tốt hơn! Bạn còn đợi gì mà không tham gia ngay thế giới của Tetris (Xếp gạch) ngay hôm nay?

1.4.3. Mô tả

a. Các loại khối hình

Trò chơi có bảy loại khối hình: I (thẳng đứng), J, L, O (vuông), S, T, Z. Ta thấy mỗi khối gạch được cấu tạo từ 4 hình vuông nhỏ xếp lại với nhau. Ta có thể coi các khối gạch đó như là những hình chữ nhật có kích thước khác nhau.

Các hình khác được tạo ra khi xoay các khối cơ bản này các góc tương ứng 90 độ, 180 độ, 270 độ.

Một chuỗi ngẫu nhiên của Tetriminos rơi xuống sân chơi (một trục đứng hình chữ nhật, được gọi là "tốt" hay "ma trận").

b. Cách chơi

Mục tiêu của trò chơi là di chuyển các khối gạch đang rơi từ từ xuống trong kích thước hình chữ nhật 20 hàng x 10 cột (trên màn hình). Chỗ nào có gạch rồi thì không di chuyển được tới vị trí đó. Người chơi xếp những khối hình sao cho khối hình lấp đầy 1 hàng ngang để ghi điểm và hàng ngang ấy sẽ biến mất.

Trò chơi gồm 2 phần:

- Phần bên trái là nơi để người xếp các khối gạch
- Phần bên phải là nơi để hiển thị các thông tin như điểm số, thời gian, tên,...

Theo thời gian chơi, một nhóm 4 khối sẽ rơi từ phía trên cùng của màn hình, người chơi có nhiệm vụ di chuyển các khối và xoay chúng cho đến khi chúng rơi xuống phía dưới cùng của màn hình, sau đó nhóm 4 khối tiếp theo sẽ rơi xuống.

Nếu để cho những khối hình cao quá màn hình, trò chơi sẽ kết thúc.

Phím tắt:

- Phím mũi tên lên / K : xoay khối.
- Phím mũi tên trái / A: di chuyển sang trái.
- Phím mũi tên phải / D: di chuyển sang phải.
- Phím mũi tên xuống / S : tăng tốc độ rơi.
- Phím Space / L : Khối gạch rơi xuống ngay lập tức.
- Phím Enter : Bắt đầu trò chơi.
- Phím P : Dừng / tiếp tục trò chơi.
- Phím E : Kết thúc trò chơi ngay lập tức

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. Môi trường cài đặt và ngôn ngữ sử dụng

Theo xu hướng tiện lợi của người dùng hiện nay, môi trường Web là môi trường có khả năng tiếp cận tới người dùng nhất. Chính vì thế, em chọn môi trường Web để cài đặt đề tài của mình.

Về mặt ngôn ngữ để cài đặt game theo hướng đối tượng – Javascript, đồng thời cũng sử dụng các ngôn ngữ thiết kế khác là HTML, CSS để thiết kế về mặt giao diện để thân thiện với người dùng.

** Tổng quát:*

- Môi trường: Web
- Ngôn ngữ lập trình: Javascript hướng đối tượng
- Ngôn ngữ thiết kế: HTML, CSS
- Các công cụ: Visual Studio Code (IDE), Word, Paint, Chrome,…

2.1.1. Ngôn ngữ Javascript

Là ngôn ngữ lập trình bậc cao (high-level) giống như: C/C++, Java, Python, Ruby,... Nó rất gần với ngôn ngữ tự nhiên của con người. Trong khi ngôn ngữ lập trình bậc thấp (low-level) như: Assembly... sẽ gần với máy tính hơn.

Là ngôn ngữ lập trình động (dynamic programming language): như Python, Ruby, Perl,... Chúng được tối ưu hoá nhằm nâng cao hiệu suất cho lập trình viên. Trong khi ngôn ngữ lập trình tĩnh (static programming language): như C/C++,... lại được tối ưu hoá để nâng cao hiệu suất cho phần cứng máy tính.

Là ngôn ngữ lập trình kịch bản (scripting language): nghĩa là không cần biên dịch (compile) hay liên kết (linked) giống như ngôn ngữ lập trình biên dịch (C/C++, Java,...) mà nó sẽ được dịch tại thời điểm chạy.

Là ngôn ngữ dựa trên đối tượng (object-based): tức nó gần giống như ngôn ngữ lập trình hướng đối tượng, ngoại trừ JavaScript không hỗ trợ tính kế thừa và đa hình.

Là ngôn ngữ dựa trên nguyên mẫu (prototype-based): là một kiểu của lập trình hướng đối tượng, trong đó các hành vi của đối tượng được sử dụng lại.

2.1.2. HTML và CSS

HTML (HyperText Markup Language) : là một ngôn ngữ đánh dấu được thiết kế ra để tạo nên các trang web, nghĩa là các mẩu thông tin được trình bày trên World Wide Web.

CSS (Cascading Style Sheets) : định nghĩa về cách hiển thị của một tài liệu HTML. CSS đặc biệt hữu ích trong việc thiết kế Web. Nó giúp cho người thiết kế dễ dàng áp đặt các phong cách đã được thiết kế lên bất kì page nào của website một cách nhanh chóng, đồng bộ.

2.1.3. Lập trình hướng đối tượng

Lập trình hướng đối tượng (tiếng Anh: Object-oriented programming, viết tắt: OOP) là một mẫu hình lập trình dựa trên khái niệm "công nghệ đối tượng", mà trong đó, đối tượng chứa đựng các dữ liệu, trên các trường, thường được gọi là các thuộc tính; và mã nguồn, được tổ chức thành các phương thức. Phương thức giúp cho đối tượng có thể truy xuất và hiệu chỉnh các trường dữ liệu của đối tượng khác, mà đối tượng hiện tại có tương tác (đối tượng được hỗ trợ các phương thức "this" hoặc "self"). Trong lập trình hướng đối tượng, chương trình máy tính được thiết kế bằng cách tách nó ra khỏi phạm vi các đối tượng tương tác với nhau. Ngôn ngữ lập trình hướng đối tượng khá đa dạng, phần lớn là các ngôn ngữ lập trình theo lớp, nghĩa là các đối tượng trong các ngôn ngữ này được xem như thực thể của một lớp, được dùng để định nghĩa một kiểu dữ liệu.

4 tính chất cơ bản của lập trình hướng đối tượng:

- Tính đóng gói (encapsulation) và che giấu thông tin (information hiding)
- Tính kế thừa (Inheritance)
- Tính đa hình (polymorphism)
- Tính trừu tượng (abstraction)

2.2. Phân tích – thiết kế

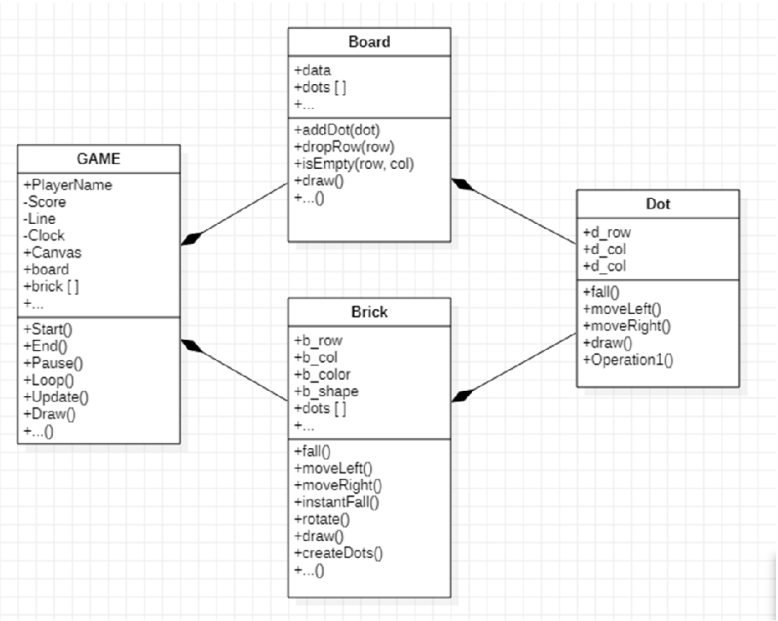
Game được phân tích, thiết kế theo hướng đối tượng

2.2.1. Mô hình chung

Trò chơi được cấu trúc từ 4 lớp chính:

- Game: là lớp chính của game, lưu trữ toàn bộ các dữ liệu về game như: tên người chơi, điểm số, thời gian chơi, các lớp khác như board, brick,… Là lớp xử lý tất cả các chức năng của game.
- Board: là lớp lưu trữ dữ liệu các ô vuông trên màn hình, lưu vị trí các ô gạch (Dot) theo mảng để kiểm tra ô đó rỗng hay không, trả về kết quả cho các hàm xử lý của các Lớp khác.
- Brick: là lớp khối gạch, lưu trữ các dữ liệu về khối gạch như: toạ độ, vị trí, màu, mảng các ô gạch(Dot), hình dạng khối gạch. Lớp có các hàm để di chuyển, thay đổi vị trí của khối gạch, là Lớp chính của game.
- Dot: là lớp cơ bản nhất của game, có nhiệm vụ lưu trữ vị trí ô gạch dưới dạng toạ độ hàng và cột. Lớp có các hàm xử lý cơ bản như di chuyển, thay đổi vị trí của ô gạch, kiểm tra các logic trong game, vẽ hình lên màn hình thông qua Canvas. Đây là lớp quan trọng nhất, là cơ sở cấu trúc cho các Lớp khác.

Các lớp được thiết kế như sơ đồ Lớp ở hình 1 sau



Hình 2.1 Sơ đồ lớp

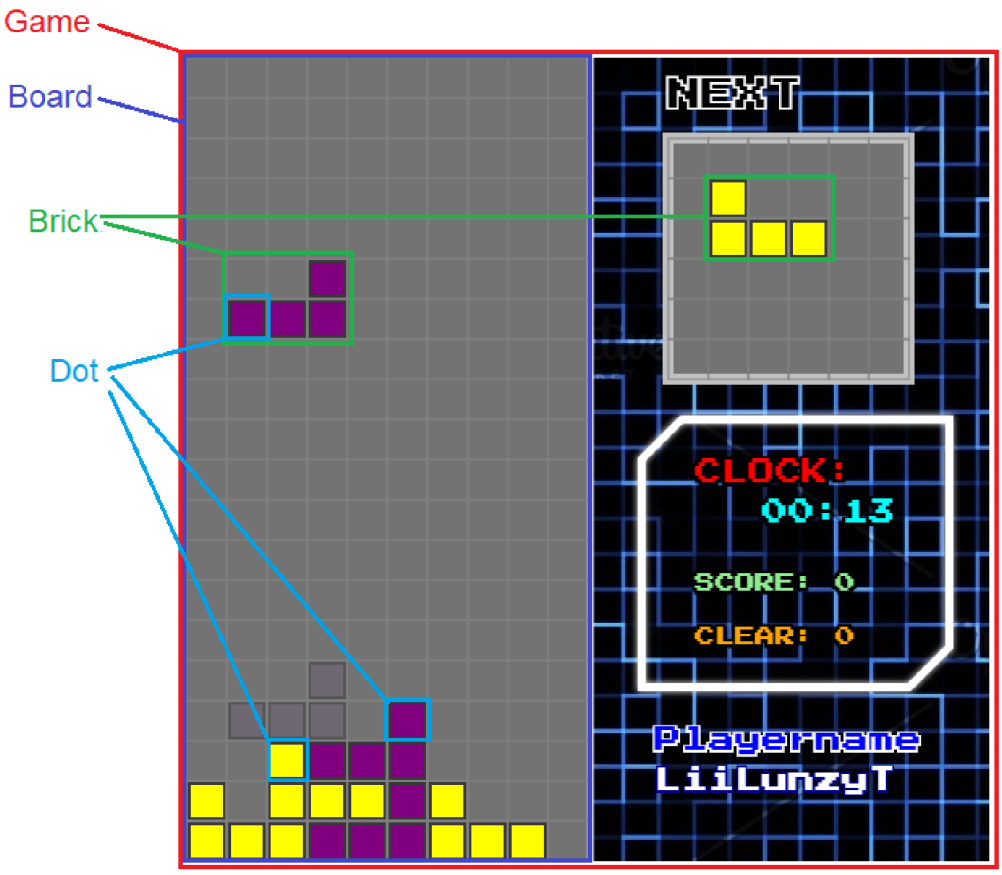
2.2.2. Giao diện

Về giao diện, được thể hiện bằng ngôn ngữ thiết kế như HTML, CSS. Về phần giao diện của Game, dùng element Canvas của HTML là chính.

Canvas là gì?

"Canvas là một phần tử của HTML5, cho phép thực hiện lập trình kết xuất đồ họa các đối tượng hai chiều trên trang web. Canvas chiếm một khu vực trong trang web với chiều rộng và chiều cao định trước. Sau đó sử dụng Javascript có thể truy cập vào khu vực này để vẽ thông qua một tập các hàm đồ họa tương tự như các API 2D khác."

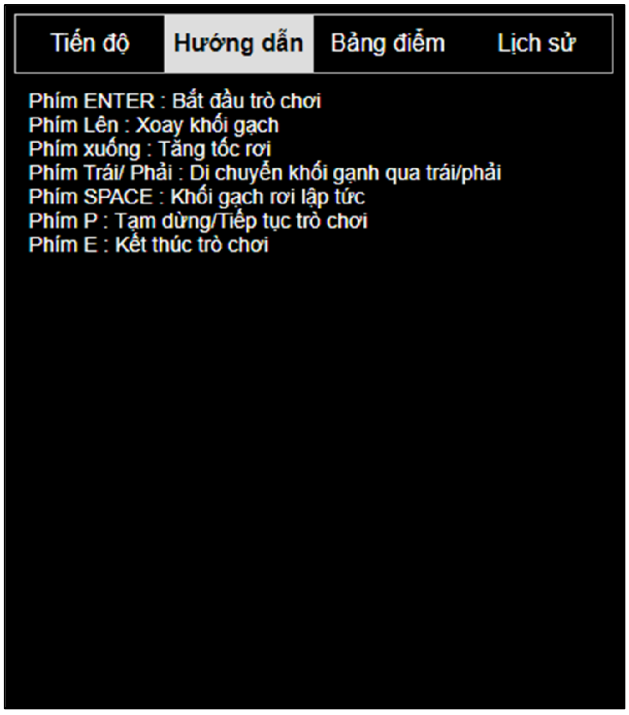
* Giao diện GAME



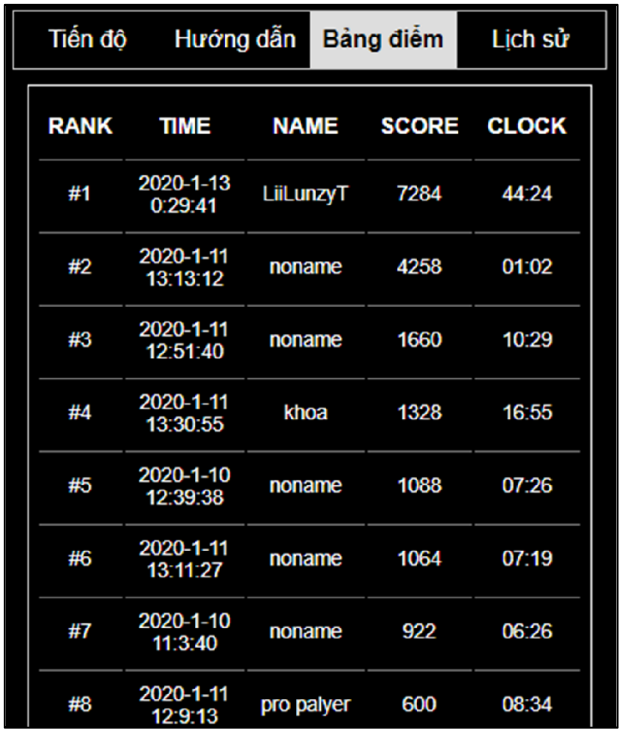
Hình 2.2 Giao diện của trò chơi

* Giao diện ngoài GAME (phụ):

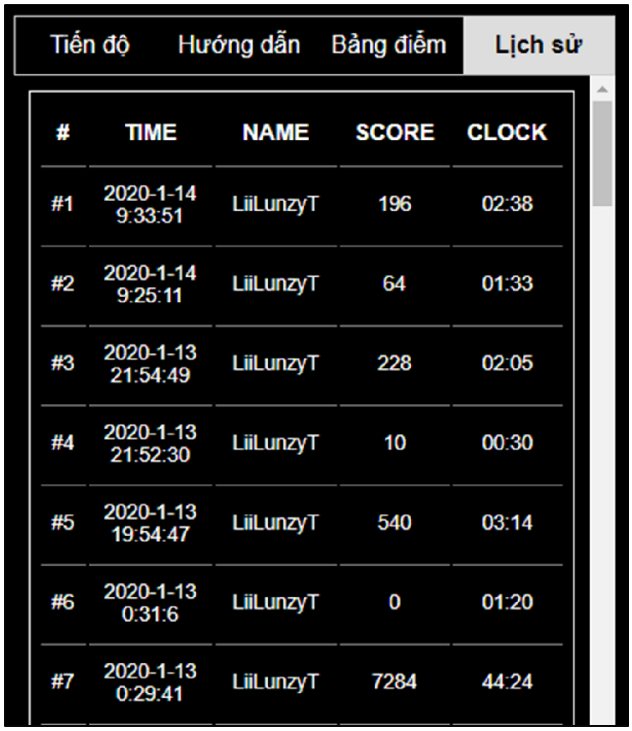
Hình 2.4 Bảng tiến độ



Hình 2.5 Bảng hướng dẫn



Hình 2.3 Bảng điểm cao



Hình 2.6 Bảng lịch sử

2.2.3. Thuật toán xử lý chính:

a. Thuộc tính chính của các đối tượng

```
class Game {
     // Hàm khởi tạo
     constructor() {
          this.cv = null; // phần tử Canvas
          this.ct = null; // phần tử Context
          this.font = null; // Font chính dùng trong game
          this.bg = null; // Hình nền

          this.isRunning = null; // flag: game hoạt động
          this.isPause = null; // flag: game tạm dừng
          this.drawInterval = null; // vòng lặp để vẽ
          this.fallInterval = null; // vòng lặp để khối gạch rơi
          this.clockInterval = null; // vòng lặp đếm thời gian chơi

          this.board = null; // khởi tạo board game của trò chơi
          this.brick = null; // khởi tạo khối gạch rơi
          this.#score = null; // điểm số
          this.#line = null; // số hàng đã phá được
          this.#clock = null; // thời gian chơi
          this.playerName = null; // tên của người chơi
          this.history = null; // lịch sử chơi
          this.highscore = null; // bảng điểm cao nhất
          this.init(); // gọi hàm init()
     }

     // Các hàm xử lý khác
     init(); // Hàm set các giá trị ban đầu cho các thuộc tính
     loadFont(); // Hàm tải font của game
     loadHistory(); // Hàm tải dữ liệu của game từ localStorage
     loadHistory2Table(); // Hàm truyền dữ liệu cho bảng html
     loadHighscore(); // Hàm tải dữ liệu của game từ localStorage
     loadHighscore2Table(); // Hàm truyền dữ liệu cho bảng html
     Start(); // Hàm bắt đầu
     Pause(); // Hàm tạm dừng
     unPause(); // Hàm tiếp tục
     End(); // Hàm dừng trò chơi
     clearBoard(); // Hàm làm sạch trên Canvas
     drawBoard(); // Hàm vẽ khung của trò chơi
     drawText(text, size, x, y, color, stroke, lwid, s_color);
          // Hàm vẽ chữ lên màn hình với tham số.
     Update(); // Hàm cập nhật điểm của trò chơi
     Draw(); // Hàm vẽ lên Canvas

      ...
}
```

```
class Board {
     // Hàm khởi tạo
     constructor(game) {
          this.game = game; // Tham số
          this.data = null; // mảng dữ liệu vị trí các ô gạch
          this.dots = null; // mảng các ô gạch (dot)
          this.init(); // gọi hàm init()
     }

     // Các hàm xử lý khác
     init(); // Hàm set các giá trị ban đầu cho các thuộc tính
     ...
     draw();
}
```

```
class Brick {
     constructor(game, row, col) {
          this.game = game; // Tham
          this.dots = null; // Mảng các thực thể Dot
          this.shadow = null; // bóng của khối gạch
          this.color = null; // màu của khối gạch
          this.s_color = null; // màu của bóng khối gạch
          this.isStop = null; // flag: khối gạch dừng rơi
          this.type = null; // dạng của khối gạch(IJLOZST)
          this.shape = null; // hình của khối gạch (4 hình mỗi dạng)
          this.dir = null; // hướng của khối gạch
          this.col = null; // toạ độ cột của khối gạch
          this.row = null; // toạ độ hàng của khối gạch
          this.haveFall = null; // số lần đã rớt
          this.init(); // gọi hàm init();
     }

     // Các hàm xử lý khác
     init(); // khởi tạo giá trị ban đầu của đối tượng
     ...
     draw(x_offset, y_offset); // vẽ đối tượng lên màn hình Canvas
}
```

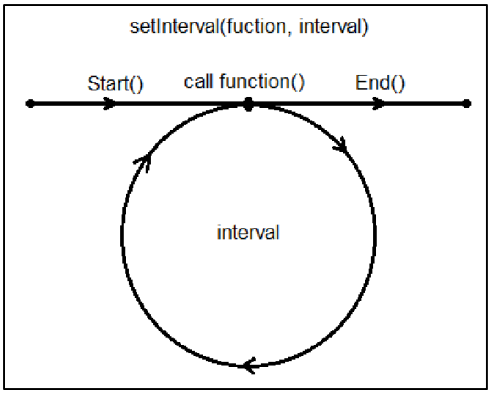
```
class Dot {
     constructor(game, row, col, color = 'red') {
          this.game = game; // truyền tham số cho đối tượng
          this.row = row; // truyền tham số cho đối tượng
          this.col = col; // truyền tham số cho đối tượng
          this.color = color; // truyền tham số cho đối tượng
     }

     // các hàm xử lý khác
     fall();
     moveLeff(), moveRight(), rotate(); // các hàm di chuyển, xoay
     ...
     draw(x_offset, y_offset); // vẽ đối tượng lên màn hình Canvas
}
```

b. Hàm loop chính của Game

Game hoạt động dựa theo 3 vòng lặp chính (loop), sử dụng bằng hàm setInterval của Javascript.

Hàm này được sử dụng để thiết lập độ trễ cho các hàm sẽ được thực hiện lặp lại như là hiệu ứng. Hàm setInterval() có cú pháp như sau:



Hình 2.7 Vòng lặp của trò chơi

```
setInterval ( function, interval );
```

Trong đó:

- function: Hàm được lặp lại theo thời gian.
- interval: thời gian trễ.

Khi trò chơi bắt đầu, sẽ gọi hàm Start()

```
Start() {
     if(!this.isRunning) {
          ...

          this.clockInterval = setInterval( () => {
               this.#clock += 1; // Thời gian trò chơi thêm 1s
          }, 1000);
          this.drawInterval = setInterval( () => {
               this.update(); // Gọi hàm cập nhật các thuộc tính
               this.draw(); // Gọi hàm vẽ các đối tượng
          }, 30);
          this.fallInterval = setInterval( () => {
               this.brick[0].fall(); // Khối gạch hiện tại rơi 1 ô
          }, 700);
     }
}
```

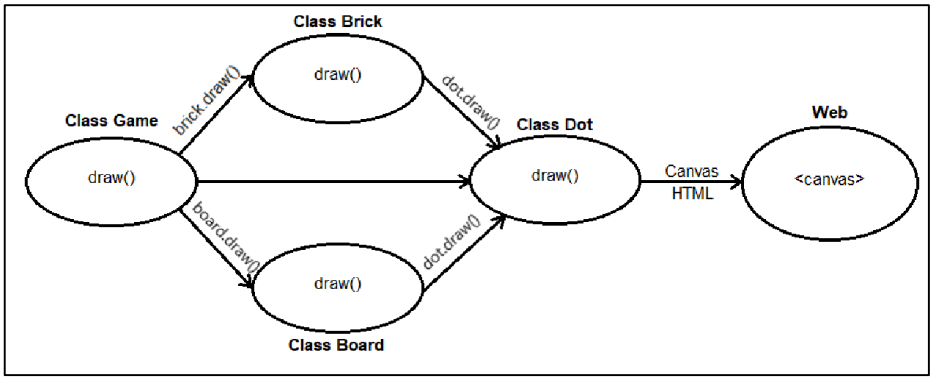
Khi trò chơi kết thúc, sẽ gọi hàm End()

```
End() {
     if( this.isRunning) {
          this.board = null;
          clearInterval(this.fallInterval);
          this.brick = null;
          clearInterval(this.drawInterval);
          this.isRunning = false;
          clearInterval(this.clockInterval);
          // Huỷ tất cả các hàm Loop

           ...
      }
```

c. Hàm vẽ lên màn hình

Để vẽ, thể hiện các đối tượng lên màn hình Game, em dùng Canvas để vẽ, các hàm vẽ được gọi theo mỗi vòng lặp được nêu ở trên.



Hình 2.8 Vẽ các đối tượng lên web

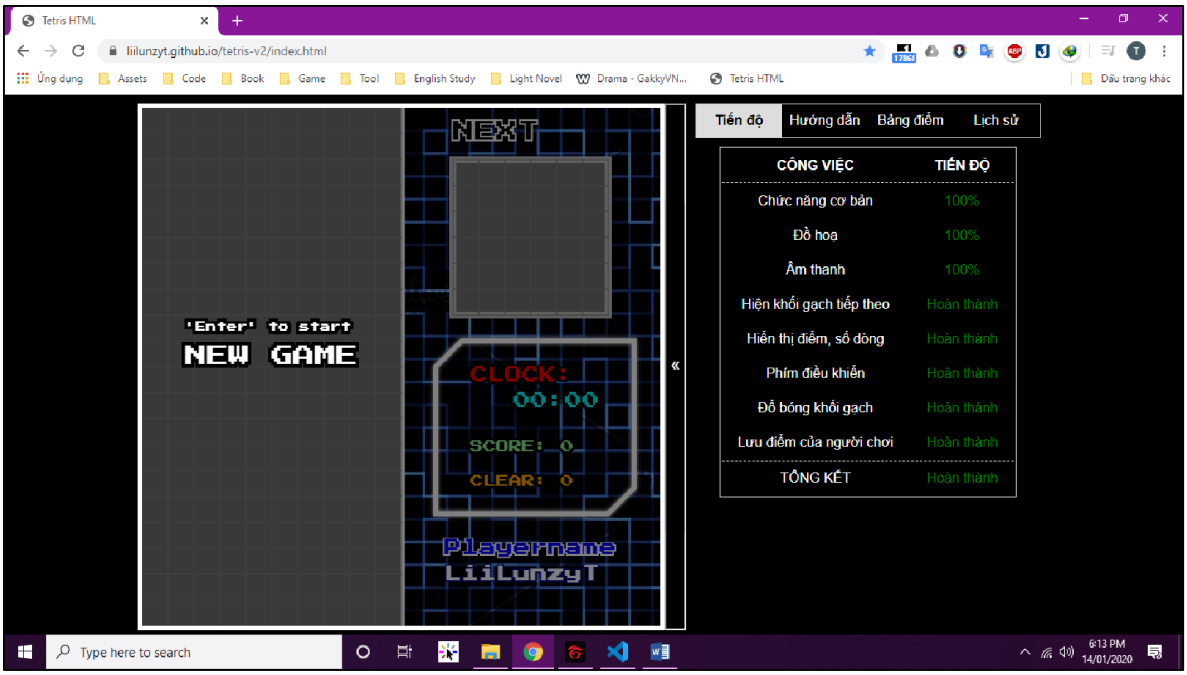
Thứ tự gọi hàm lần lượt: game.draw ➔ board.draw , brick.draw ➔ dot.draw

```
// Lớp GAME
draw() {
     this.drawBoard(); // gọi hàm vẽ khung của game.
     this.board.draw(); // gọi hàm draw() từ đối tượng Board.
     this.brick[0].draw(); // gọi hàm draw() từ đối tượng Brick.
     this.brick[1].draw(10,6); // gọi hàm draw() từ đối tượng Brick.
}
// LỚP BOARD
draw() {
     this.dots.forEach( (dot) => {
          dot.draw(); // Gọi hàm vẽ từ lớp Dot
     });
}
// LỚP BRICK
draw(x_offset = 0, y_offset = 0) {
     //draw brick
     this.dots.forEach( (dot) => {
          dot.draw(x_offset, y_offset); // Gọi hàm draw() từ lớp Dot
     });
}
// LỚP DOT
draw(x_offset = 0, y_offset = 0) {
     // x_offset, y_offset: độ lệch
     // Tính toán toạ độ của khối gạch
     let x = (this.col + x_offset) * SIZE;
     let y = (this.row + y_offset - 4) * SIZE;
     // Vẽ viền của khối gạch lên màn hình
     this.game.ct.strokeStyle = 'black';
     this.game.ct.strokeRect(x + 2, y + 2, SIZE - 4, SIZE - 4);
     //Vẽ khối gạch lên màn hình
     this.game.ct.fillStyle = this.color;
     this.game.ct.fillRect(x + 3, y + 3, SIZE - 6, SIZE - 6);
}
```

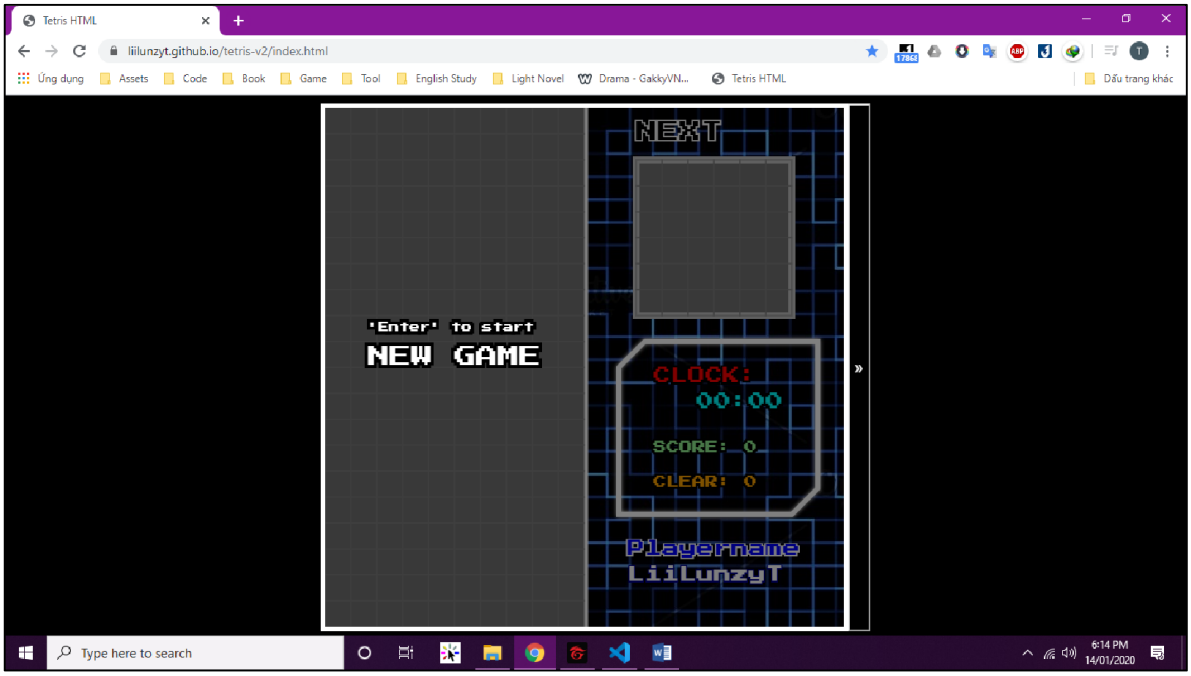
CHƯƠNG 3. KẾT QUẢ THỰC HIỆN ĐỀ TÀI

3.1. Giao diện

3.1.1. Màn hình bắt đầu game

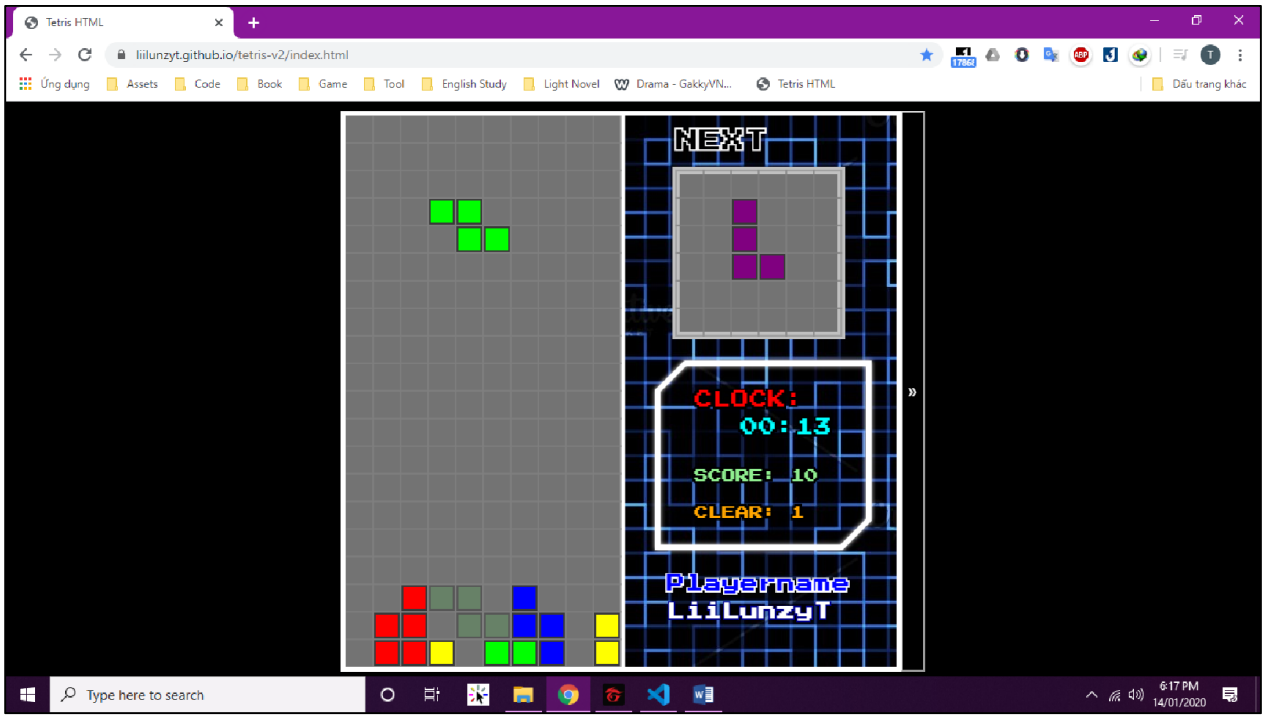


Hình 3.1 Toàn màn hình của Game



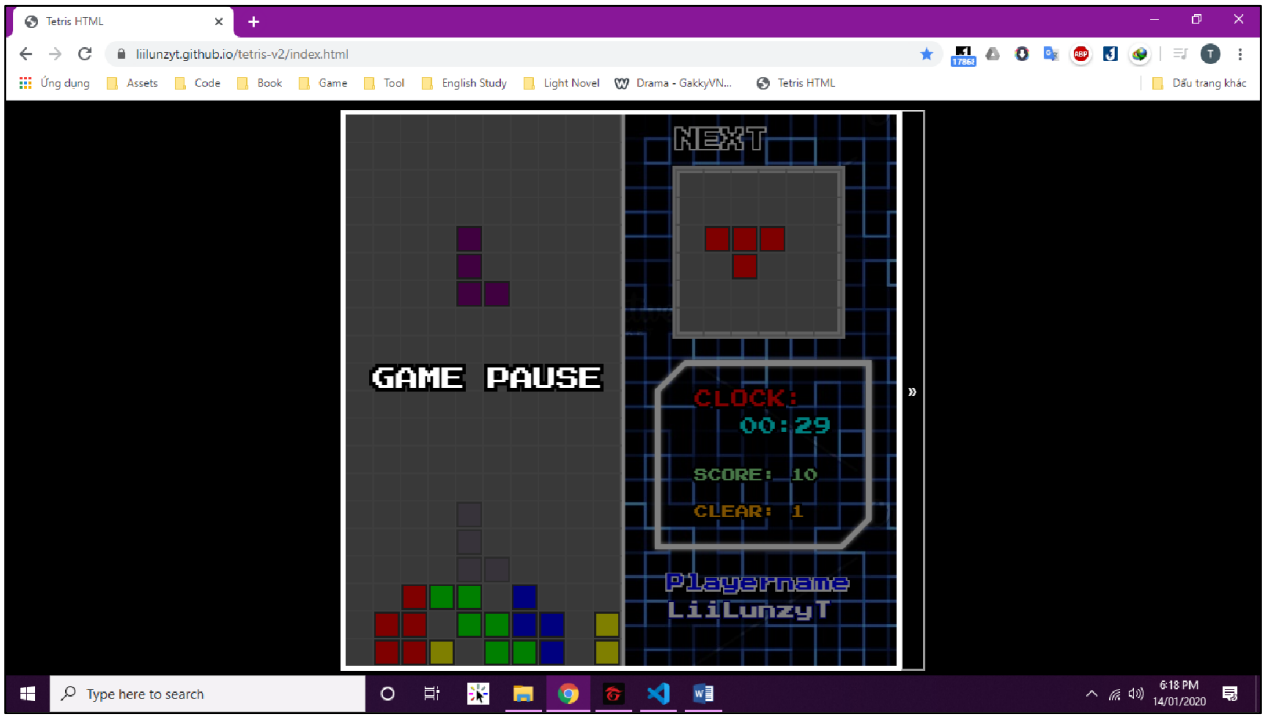
Hình 3.2 Toàn màn hình của Game sau khi rút gọn

3.1.2. Màn hình khi đang chơi



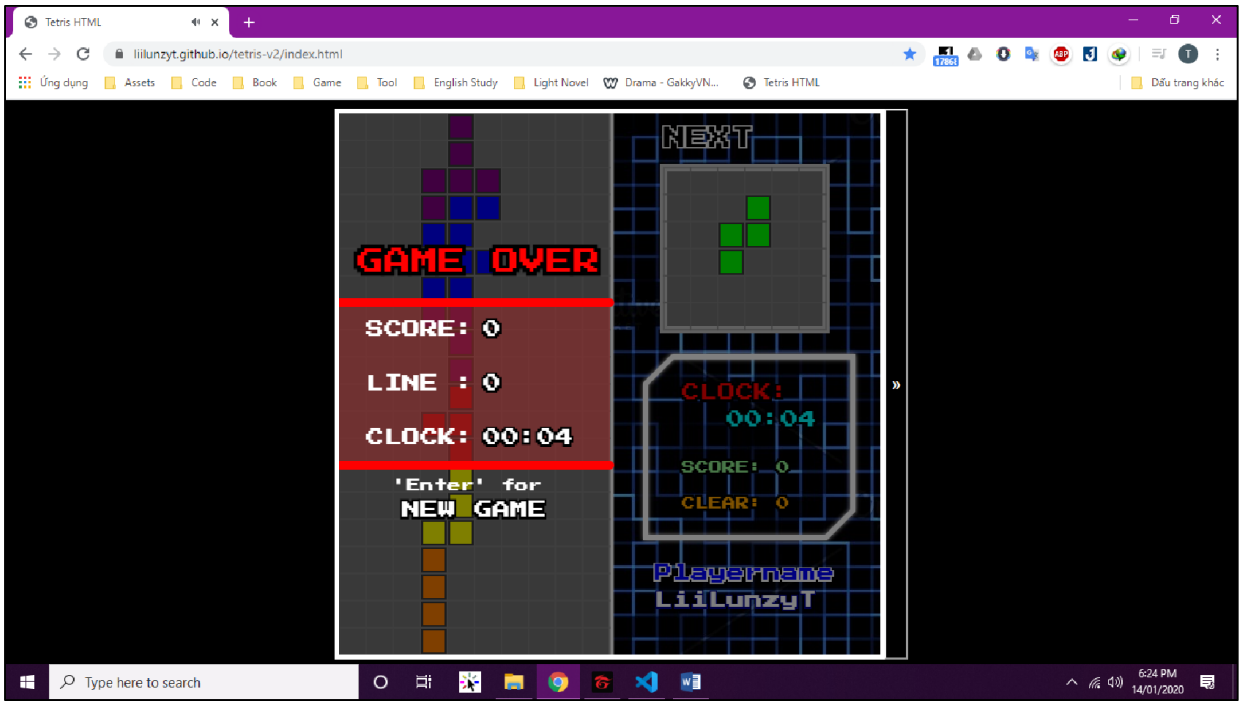
Hình 3.3 Màn hình Game khi đang chơi

3.1.3. Màn hình tạm dừng



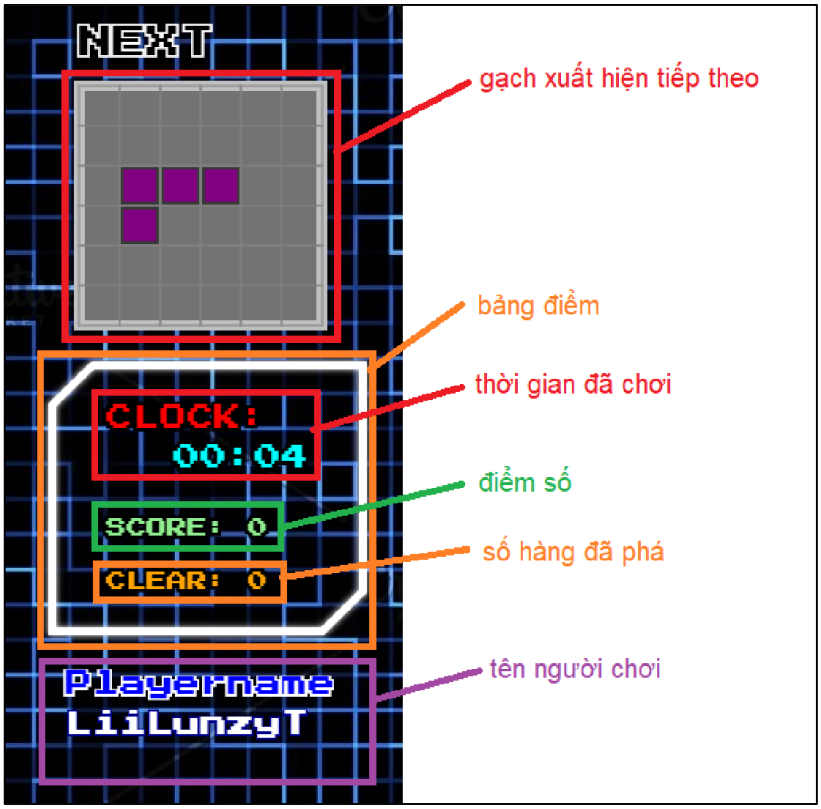
Hình 3.4 Màn hình Game khi đang tạm dừng

3.1.4. Màn hình kết thúc game



Hình 3.5 Màn hình kết thúc Game

3.1.5. Màn hình phụ



Hình 3.6 Chi tiết màn hình phụ

3.2. Các chức năng

3.2.1. Các chức năng trong Game

Các chức năng cơ bản trong Game:

- Di chuyển khối gạch bằng các phím **[🡐] [🡒]**
- Xoay khối gạch bằng phím [🡑]
- Khối gạch rơi ngay lập tức bằng phím [SPACE]
- Dừng / tiếp tục trò chơi bằng phím [P]
- Kết thúc nhanh trò chơi bằng phím [E]
- Có đồng hồ giúp người chơi xem đã chơi được bao lâu.
- Kẻ ô khung trò chơi để người chơi dễ nhìn.
- Cho người chơi biết được khối gạch tiếp theo.
- Đổ bóng khối gạch thuận tiện cho người chơi.
- Hoàn thành càng nhiều hàng (Combo) điểm cộng càng nhiều.
- Lưu lịch sử của người chơi (0 điểm thì không được lưu).

3.2.2. Các chức năng ngoài Game

Các chức năng phụ ngoài Game:

- Mặc định tên người chơi từ lần chơi gần nhất (Nếu là lần đầu, tên mặc định là “noname”).
- Bảng hướng dẫn giúp người chơi dễ dàng điều khiển.
- Bảng điểm cao, bảng lịch sử ghi lại lịch sử của người chơi, và xếp hạng điểm cao nhất của người chơi (xếp hạng tối đa 10 bản ghi).
- Bảng tiến độ thông báo quá trình cài đặt trò chơi (dành cho giảng viên).
- Nút [»] trên màn hình giúp mở rộng / thu gọn phần bên phải(phụ), tạo không gian thoải mái cho người chơi.

3.3. Mã nguồn

Link sản phẩm: https://liilunzyt.github.io/tetris-v2/index.html

Link source-code: https://github.com/LiiLunzyT/liilunzyt.github.io/tree/master/tetris-v2

Cấu trúc thư mục gồm:

- File index.html: trang web.
- Thư mục css: chứa các file css và các file font.
- Thư mục js: chứa các file
- Thư mục img: chứa các file hình ảnh trong game.
- Thư mục sound: chứa các file âm thanh trong game.

index.html	css	js	img	sound
	style.css pressstart2p.woff2	const.js function.js brickType.js dot.js board.js game.js	bg-1.jpg	clear.wav down.mp3 fall.wav gameover.wav move.wav rotate.wav

3.3.1. game.js

Là file javascript cài đặt lớp Game (~400 line), xử lý tất các hàm, các sự kiện chính trong game, bao gồm:

a. Bắt đầu

```
Start() {
     this.isRunning = true;
     console.log('GAME START');
     this.board = new Board(this);
     this.brick = [new Brick(this), new Brick(this)];
     this.#score = 0;
     this.#line = 0;
     this.#clock = 0;
     this.loadHistory();
     this.loadHighscore();

     // If gameover sound is playing, stop it
     sound[4].pause();
     sound[4].currentTime = 0;
          // setInterval đã nêu ở trên
     ...
}
```

b. Kết thúc

```
End() {
     this.isRunning = false;
     // if the game is pausing, unpause it before
     if( this.isPause ) {
          this.unPause();
          this.draw();
     }
     // play sound
     sound[4].play();
     // clear atrribute đã nói ở trên
     ...
     // draw Scoreboard
     this.ct.restore();
     this.ct.globalAlpha = 0.5;
     this.ct.fillStyle = 'black';
     this.ct.fillRect(0, 0, 600, 600);
     this.ct.fillStyle = 'brown';
     this.ct.fillRect(0, 210, 300, 180);
     this.ct.globalAlpha = 1.0;
     this.ct.beginPath();
     this.ct.moveTo(0,210);
     this.ct.lineTo(300,210);
     this.ct.moveTo(0, 390);
     this.ct.lineTo(300,390);
     this.ct.lineWidth = 10;
     this.ct.strokeStyle = 'red';
     this.ct.stroke();
     this.drawText('GAME OVER', 30, 20, 180, 'red', true, 8, 'black');
     // Draw Score
     this.ct.restore();
     this.drawText('SCORE: ', 20, 30, 250, 'white');
     this.drawText(this.#score, 20, 160, 250, 'white', true, 5);
     this.drawText('LINE : ', 20, 30, 310, 'white');
     this.drawText(this.#line, 20, 160, 310, 'white', true, 5);
     this.drawText('CLOCK: ', 20, 30, 370, 'white');
     this.drawText(toHHMMSS(this.#clock), 20, 160, 370, 'white', true,
5);
     this.drawText("'Enter' for", 15, 60, 420, 'white',);
     this.drawText('NEW GAME', 20, 70, 450, 'white', true, 5);
     // get Day Time
     if(this.#score != 0) {
          let today = new Date();
          let date = today.getFullYear() + '-' + (today.getMonth()+1) + '-' +
today.getDate();
          let time = today.getHours() + ":" + today.getMinutes() + ":" +
today.getSeconds();
          let dateTime = date+'\n'+time;
          // Save Score
          let data = {
               "time"  : dateTime,
               "name" : this.playerName,
               "score" : this.#score,
               "clock" : toHHMMSS(this.#clock)};
          this.history.unshift(data);
          setLocalStorage('highscore', this.history);
     }
}
```

c. Phím điều khiển

```
readKeyPress() {
     document.addEventListener('keydown', (event) => {
          if( !this.isRunning ) { // while game isn't running
               switch(event.code) {
                    case K_ENTER: this.Start(); break;
               }
          } else {
               switch(event.code) { // while game is running
                    case K_UP: case K_K:
                          this.brick[0].rotate(); break;
                    case K_DOWN: case K_S:
                          this.brick[0].fall(); break;
                    case K_LEFT: case K_A:
                          this.brick[0].moveLeft(); break;
                    case K_RIGHT: case K_D:
                          this.brick[0].moveRight(); break;
                    case K_SPACE: case K_L:
                          this.brick[0].instantFall();
                          sound[5].play(); break;
                    case K_P: this.Pause(); break;
                    case K_E: this.End(); break;
               }
          }
     });
}
```

d. Lấy dữ liệu từ LocalStorage

```
loadHistory() {
     if (checkLocalStorage('highscore') ) {
          this.history = loadLocalStorage('highscore');
          this.loadHistory2Table();
     }
}

function setLocalStorage(key, data) {
    localStorage.setItem(key, JSON.stringify(data));
}

function loadLocalStorage(key) {
    return JSON.parse(localStorage.getItem(key));
}

function checkLocalStorage(key) {
    return localStorage.getItem(key) != null &&
localStorage.getItem(key).length != 0;
}
```

3.3.2. const.js

Là file javascript dùng để lưu trữ các hằng, các biến được đặt giá trị mặc định đầu game

a. Bảng phím điều khiển

```
const K_UP = 'ArrowUp';
const K_LEFT = 'ArrowLeft';
const K_DOWN = 'ArrowDown';
const K_RIGHT = 'ArrowRight';
const K_SPACE = 'Space';
const K_ENTER = 'Enter';
const K_SHIFTLEFT = 'ShiftLeft';
const K_P = 'KeyP';
const K_E = 'KeyE';
const K_W = 'KeyW';
const K_A = 'KeyA';
const K_S = 'KeyS';
const K_D = 'KeyD';
const K_L = 'KeyL';
const K_K = 'KeyK';
```

b. Bảng màu

```
const B_COLOR = [
     'rgba(255, 0  , 0  , 1.0)',      // 0 - RED
     'rgba(0  , 255, 0  , 1.0)',      // 1 - GREEN
     'rgba(0  , 0  , 255, 1.0)',      // 2 - BLUE
     'rgba(128, 0  , 128, 1.0)',      // 3 - PURPLE
     'rgba(255, 255, 0  , 1.0)',      // 4 - YELLOW
     'rgba(255, 127, 0  , 1.0)',      // 5 - ORANGE
     'rgba(128, 76 , 51 , 1.0)',      // 6 - BROWN
];
```

c. Bảng âm thanh

```
var sound = [
     new Audio('sound/fall.wav'),
     new Audio('sound/clear.wav'),
     new Audio('sound/rotate.wav'),
     new Audio('sound/move.wav'),
     new Audio('sound/gameover.wav'),
     new Audio('sound/down.mp3')
];

sound.forEach( (s) => {
     s.volume = 0.1;
});
```

3.3.3. BrickType.js

Là file javascript lưu các hình dạng của các khối gạch

Vd: Mẫu của khối T

```
const T = [
          [
               [_,X,_],
               [X,X,X],
               [_,_,_]
          ],
          [
               [_,X,_],
               [_,X,X],
               [_,X,_]
          ],
          [
               [_,_,_],
               [X,X,X],
               [_,X,_]
          ],
          [
               [_,X,_],
               [X,X,_],
               [_,X,_]
          ]
];
```

3.3.4. dot.js

Là file javascript cài đặt lớp Dot, chứa các hàm xử lý toạ độ, vẽ ô gạch lên Web.

```
canMoveLeft() {
     return this.game.board.isEmpty(this.row, this.col - 1) && this.col >
0;
}

moveLeft() {
     this.col--;
}

canMoveRight() {
     return this.game.board.isEmpty(this.row, this.col + 1) && this.col <
9;
}

moveRight() {
     this.col++;
}

isHitBottom() {
     return this.row == 23;
}
```

```
isBelowEmpty() {
     return this.game.board.isEmpty(this.row + 1, this.col);
}

isCanFall() {
     let isCan = true;
     if( this.isHitBottom() ) isCan = false;
     else if( !this.isBelowEmpty() ) isCan = false;

     return isCan;
}

fall() {
     this.row++;
}
```

3.3.5. board.js

Là file javascript cài đặt lớp Board, có các hàm xử lý, kiểm tra các hàng gạch, xoá hàng đã đầy gạch,…

a. Kiểm tra bảng gạch

```
checkBoard() {
     this.clearBoard();
     let count = 0;
     let rowFull = [];
     for(let row = 4; row < ROWS; row++) {
          if(this.checkFullRow(row)) {
               rowFull.push(row);
               this.fullRowChangeColor(row);
               count++;
          }
     }
          if(count) {
          this.dropData(rowFull);
          setTimeout( () => {
               this.dropDots(rowFull);
          }, 10 * 1000 / 60);
     }
     return count;
}

checkFullRow(row) {
          let isFull = true;
          for(let col = 0; col < COLS; col++) {
               if(this.data[row][col] == _) isFull = false;
          }

          return isFull;
     }

isEmpty(row, col) {
     return !this.data[row][col];
}
```

b. Xoá các hàng gạch đã đầy

```
dropData(rowFull) {
     rowFull.forEach( (row) => {
          this.data.splice(row, 1);
          this.data.unshift([_,_,_,_,_,_,_,_,_,_]);
     });
}

dropDots(rowFull) {
     rowFull.forEach( (row) => {
          this.dots = this.dots.filter( (dot) => {
               return dot.row != row;
          });

          this.dots.forEach( (dot) => {
               if(dot.row < row) dot.row++;
          });
     });
}
```

c. Chuyển màu các hàng gạch đã đầy

```
fullRowChangeColor(row) {
     let i = 0;
     this.dots.forEach( (dot) => {
          if( dot.row == row ) {
               setTimeout( () => {
                    let r = Math.floor(Math.random() * 7);
                    dot.color = C_COLOR[r];
               }, i * 1000 / 60);
               i += 1;
          }
     });
}
```

d. Thêm ô gạch vào bảng

```
addBrick(dots) {
     dots.forEach( (dot) => {
          this.dots.push(dot);
          this.data[dot.row][dot.col] = X;
     });
}
```

e. Vẽ các ô gạch

```
draw() {
     this.dots.forEach( (dot) => {
          dot.draw();
     });
}
```

3.3.6. brick.js

Là file cài đặt lớp Brick, gồm có các hàm xử lý, di chuyển khối gạch đang rơi

a. Rơi

```
isCanFall() {
     let isCan = true;
     this.dots.forEach( (dot) => {
          if( !dot.isCanFall() ) isCan = false;
     });
     return isCan;
}


fall() {
     if ( this.isCanFall() ) {
          this.dots.forEach( (dot) => {
               dot.fall();
          });
          this.row++;
          this.haveMove++;
          this.updateShadow();
          return true;
     }
     else {
          if(this.haveMove == 0) return this.game.End();
          sound[0].play();

          this.game.board.addBrick(this.dots);
          this.game.brick.shift();
          this.game.brick.push(new Brick(this.game));
          return false;
     }
}
```

b. Di chuyển trái phải

```
canMoveLeft() {
     let isCan = true;
     this.dots.forEach( (dot) => {
          if( !dot.canMoveLeft() ) isCan = false;
     });
     return isCan;
}
moveLeft() {
     if( this.canMoveLeft() && this.canMove()) {
          sound[2].play();
          this.dots.forEach( (dot) => {
               dot.moveLeft();
          });
          this.col--;
          this.haveMove++;
          this.updateShadow();
     }
}
// Bên phải tương tự
```

c. Tạo ô gạch

```
createDots() {
     for(let row = 0; row < this.shape.length; row++) {
          for(let col = 0; col < this.shape[0].length; col++) {
               if( this.shape[row][col] == X) {
                    let dot = new Dot(this.game, row + this.row, col +
this.col, this.color     );
                    this.dots.push(dot);
               }
          }
     }
}

createShape() {
     let r = Math.floor(Math.random() * 7);
     this.dir = Math.floor(Math.random() * 4);

     let listType = [ T , Z , S , L , J , I , O];
     this.type = listType[r];
     this.shape = this.type[this.dir];
     this.color = B_COLOR[r];
     this.s_color = S_COLOR[r];
}
```

d. Tạo bóng gạch

```
updateShadow() {
     this.shadow = [];
     let y = 0;

     for(y = this.dots[3].row + 1; y <= 23; y++) {
          let check = true;
          this.dots.forEach( (dot) => {
               if( !this.game.board.isEmpty(y - this.dots[3].row +
dot.row, dot.col) ) check  = false;
          });
          if( !check ) break;
     }

     y--;
     this.dots.forEach( (dot) => {
          this.shadow.push([y - this.dots[3].row + dot.row, dot.col]);
     });
}
```

e. Xoay

```
canRotate() {
     let newDir = this.dir + 1;
     if(newDir == 4) newDir = 0;

     let newShape = this.type[newDir];
     let newDots = [];
     for (let row = 0; row < newShape.length; row++) {
          for (let col = 0; col < newShape[0].length; col++) {
               if(newShape[row][col] == X) {
                    let newDot = new Dot(this.game, this.row + row,
this.col + col, this.color);
                    newDots.push(newDot);
               }
          }
     }

     let isCan = true;
     newDots.forEach( (dot) => {
          if(!this.game.board.isEmpty(dot.row, dot.col)) isCan = false;
          if(dot.col < 0 || dot.col > 9) isCan = false;;
     });
     return isCan;
}

rotate() {
     if(this.canRotate() && this.canMove()) {
          sound[3].play();
          this.dots = [];
          this.dir++;
          this.haveMove++;
          if(this.dir == 4) this.dir = 0;
          this.shape = this.type[this.dir];
          this.createDots();
          this.updateShadow();
     }
}
```

f. Rơi ngay lập tức

```
instantFall() {
     if(this.fall() ) {
          return setTimeout( () => {
               this.instantFall();
               this.isFastMove = true;
          }, 1000 / 100);
     } else {
          this.isFastMove = false;
     }
}
```

3.3.7. index.html

```
<!DOCTYPE html>
<html>
<head>
<title>Tetris HTML</title>
<link rel="stylesheet" type="text/css" href="css/style.css">

</head>
<body>
<canvas id="game-canvas"></canvas>
<button id="btn-hide" value="hide" onclick="hideGuide()"
onfocus="this.blur();">&raquo;</button>
<div id="game-info">
     <div class="tab">
          <button class="tablinks active" onclick="openTab(event, 'game-
progress')">Tiến độ</button>
          <button class="tablinks" onclick="openTab(event, 'game-
guide')">Hướng dẫn</button>
          <button class="tablinks" onclick="openTab(event, 'game-
highscore')">Bảng điểm</button>
          <button class="tablinks" onclick="openTab(event, 'game-
history')">Lịch sử</button>
     </div>
     <div id="game-progress" class="tabcontent">
          <table>
          ...
          </table>
     </div>
     <div id="game-guide" class="tabcontent">
     ...
     </div>
     <div id="game-highscore" class="tabcontent score-table">
     ...
     </div>
     <div id="game-history" class="tabcontent score-table">
          <table>
          ...
          </table>
          <button id="erase-history">Xoá Lịch sử</button>
     </div>
</div>
</body>

<script type="text/javascript" src="js/const.js"></script>
<script type="text/javascript" src="js/function.js"></script>
<script type="text/javascript" src="js/brickType.js"></script>
<script type="text/javascript" src="js/dot.js"></script>
<script type="text/javascript" src="js/board.js"></script>
<script type="text/javascript" src="js/brick.js"></script>
<script type="text/javascript" src="js/game.js"></script>

</html>
```

3.3.8. style.css

```
* {
     margin: 0;
     padding: 0;
     background-color: black;
     color: white;
     box-sizing: border-box;

     font-family: Arial, Helvetica, sans-serif;
}

body {
     position: relative;
     padding-top: 10px;
     display: flex;
     flex-direction: row;
     justify-content: center;
     overflow: hidden;
}

#game-info {
     width: 400px;
     height: 100%;
     margin: 0 10px;
     margin-left: -500px;
     transition: 0.5s margin;
}

#btn-hide {
     padding: 0 5px;
     font-size: 20px;
     color: white;
     background-color: black;
     outline: none;
     margin-left: -20px;
     transition: 0.5s margin;
}

#btn-hide:hover {
     margin-left: 0;
}

#game-progress {
     width: 100%;
     height: 30%;
     text-align: center;
     display: block;
}

#game-guide {
     padding: 10px;
     font-size: 14px;
}
#game-highscore {
     height: 520px;
     width: 100%;
}
```

CHƯƠNG 4. KẾT LUẬN

- Kết quả đạt được:
  - ✓ Hoàn thành đầy đủ các mục tiêu đề ra.
- Ưu điểm:
  - ✓ Hoàn thành đúng thời hạn, tiến độ được giao.
  - ✓ Đã cố gắng bám sát được các tiêu chí đề ra.
  - ✓ Phát triển trên môi trường web nên rất thân thiện với người chơi
  - ✓ Các bạn học rất thích thú với game này (thành công).
- Nhược điểm:
  - ✓ Hạn chế về mặt mỹ thuật nên giao diện không được đẹp lắm.
  - ✓ Chức năng lưu điểm chỉ ở mức độ cục bộ, do không có thời gian để tìm hiểu và phát triển phía Server
  - ✓ Vẫn tồn tại một vài bug nhỏ nhưng chưa biết nguyên do dẫn đến chưa thể xử lý (bug không ảnh hưởng lớn đến trò chơi).
- Thu hoạch:
  - ✓ Trau dồi được nhiều kinh nghiệm, kỹ năng trong quá trình hoàn thành đề tài.
  - ✓ Hiểu rõ hơn về ngôn ngữ Javascript (trước đây chỉ nắm được ở mức cơ bản).
  - ✓ Đề tài này có thể được xem như một dự án trong CV.
  - ✓ Kỹ năng viết báo cáo được rèn luyện, hỗ trợ cho bài luận tốt nghiệp sắp tới.
- Hướng phát triển trong tương lai:
  - ✓ Mở rộng, thêm chức năng cho Game.
  - ✓ Phát triển về phía Server.
  - ✓ Quảng bá Game, tăng lượng người chơi.
  - ✓ Thử nghiệm cài đặt trên các môi trường khác (App, Mobile, Console,…).
  - ✓ Tiếp tục nghiên cứu, phát triển các dự án khác với quy mô lớn hơn.

TÀI LIỆU THAM KHẢO

[1] Tài liệu về game Tetris: https://en.wikipedia.org/wiki/Tetris

[2] Bài viết về game Tetris:

https://vietgiaitri.com/tetris-bai-toan-chua-co-ket-qua-20091019i12093/?

[3] Tổng quan về Javascript:

https://completejavascript.com/gioi-thieu-tong-quan-ve-ngon-ngu-lap-trinh-javascript

[4] HTML và CSS cơ bản:

https://viblo.asia/p/tim-hieu-ve-html-va-css-co-ban-phan-1-7ymwGXV0R4p1

[5] Lập trình hướng đối tượng:

https://en.wikipedia.org/wiki/Object-oriented_programming

[6] Phương hướng cài đặt game:

https://www.youtube.com/watch?v=BNwNdo6y9dw&t=3678s