**TRƯỜNG ĐẠI HỌC SƯ PHẠM – ĐẠI HỌC ĐÀ NẴNG**

**KHOA TIN HỌC**

**-----o0o-----**



**BÁO CÁO**

**ĐỒ ÁN CHUYÊN NGÀNH**

***Đề tài:*** *Xây dựng game 2D - Brave trên nền tảng web sử dụng HTML, CSS và JavaScript*

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn** | Nguyễn Đình Lầu |
| **Sinh viên thực hiện** | Nguyễn Công Hoàng |
| **Lớp SH** | 19CNTTC |

**Đà Nẵng, 12/2022**

**NHẬN XÉT CỦA CÁN BỘ HƯỚNG DẪN** ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. .................................................................................................................................

*Đà Nẵng, ngày … tháng … năm …*

**Cán bộ hướng dẫn**

**NHẬN XÉT CỦA HỘI ĐỒNG PHẢN BIỆN** ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. ................................................................................................................................. .................................................................................................................................

*Đà Nẵng, ngày … tháng … năm …*

**Hội đồng phản biện**

**LỜI CAM ĐOAN**

Em tên là Nguyễn Công Hoàng, sinh viên lớp 19CNTTC. Em xin cam đoan đây là đề tài do em thực hiện. Các nội dung và kết quả nghiên cứu được trình bày trong Đồ án chuyên ngành này là trung thực và mọi tham khảo đều được trích dẫn, chỉ rõ nguồn tham khảo theo đúng quy định.

Sinh viên thực hiện

**Nguyễn Công Hoàng**

**LỜI CẢM ƠN**

Đầu tiên, em xin chân thành cảm ơn Quý Thầy/Cô trong khoa Tin học đã trang bị những kiến thức cho em trong suốt quá trình học tập tại Trường Đại học Sư phạm – Đại học Đà Nẵng vừa qua. Chính nhờ công lao giảng dạy, chỉ bảo tận tình của Quý Thầy/Cô mà em mới trang bị được kiến thức chuyên ngành công nghệ thông tin để có thể thực hiện tiếp chặng đường học tập, vận dụng và sáng tạo ra những sản phẩm hữu ích góp phần phục vụ các lĩnh vực khác nhau.

Với lòng kính trọng và biết ơn sâu sắc, em xin gửi lời cảm ơn đến giảng viên hướng dẫn, TS. Nguyễn Đình Lầu đã tận tình giúp đỡ em trong quá trình nghiên cứu, lập đề cương cũng như trong suốt quá trình thực hiện và hoàn thiện Đồ án chuyên ngành này.

Em cũng xin gửi lời cảm ơn chân thành tới các bạn ở Khoa Tin học, Trường Đại học Sư phạm đã tạo mọi điều kiện thuận lợi và giúp đỡ em trong thời gian học tập và nghiên cứu thực hiện Đồ án này.

Mặc dù em đã rất cố gắng và nổ lực để hoàn thành Đồ án chuyên ngành này nhưng trong phạm vi và khả năng cho phép, chắc chắn Đồ án sẽ không tránh khỏi những thiếu sót. Em rất mong nhận được sự thông cảm, góp ý và tận tình chỉ bảo của Quý Thầy/Cô để báo cáo được hoàn thiện nhất. Em xin chân thành cảm ơn!

Sinh viên thực hiện

**Nguyễn Công Hoàng**

**MỤC LỤC**

[**DANH MỤC HÌNH ẢNH**](#_Toc105863885) 9

[**LỜI NÓI ĐẦU** 10](#_Toc105863886)

[**A. Đề tài** 11](#_Toc105863887)

[I. Tên đề tài 11](#_Toc105863888)

[II. Phương pháp nghiên cứu 11](#_Toc105863889)

[III. Nhiệm vụ và kết quả đạt được 11](#_Toc105863890)

[IV. Bố cục đề tài 11](#_Toc105863891)

[**B. Tổng quan về HTML, CSS và JavaScript** 12](#_Toc105863892)

[I. HTML 12](#_Toc105863893)

[1. HTML là gì? 12](#_Toc105863894)

[2. CSS là gì? 12](#_Toc105863895)

[3. JavaScript là gì? 12](#_Toc105863896)

[**C. Quá trình thực hiện xây dựng game The Last Samurai** 13](#_Toc105863897)

[I. Sơ lược về sản phẩm 13](#_Toc105863898)

[II. Các đối tượng trong game 13](#_Toc105863899)

[III. Thực hiện xây dựng game 14](#_Toc105863900)

[1. Thiết lập giao diện game menu 14](#_Toc105863901)

[1.1. Lên ý tưởng giao diện 14](#_Toc105863902)

[1.2. Cài đặt giao diện 14](#_Toc105863903)

[2. Xây dựng cơ chế game 19](#_Toc105863904)

[2.1. Lên ý tưởng 19](#_Toc105863905)

[2.2. Thực hiện xây dựng project 19](#_Toc105863906)

[2.2.1. Cài đặt project: 19](#_Toc105863907)

[2.2.2. Khởi tạo các nhân vật 20](#_Toc105863908)

[2.2.3. Di chuyển nhân vật bằng Event Listeners (keys) 25](#_Toc105863909)

[2.2.4. Cơ chế tấn công (gây sát thương) 27](#_Toc105863910)

[2.2.5. Thanh HP (healthbar) và giao diện đồ họa của game 31](#_Toc105863911)

[2.2.6. Bộ đếm ngược thời gian và sự kiện để trò chơi kết thúc 32](#_Toc105863912)

[2.3. Cài đặt các Sprite và animation cho game 35](#_Toc105863913)

[2.3.1. Background Sprite 35](#_Toc105863914)

[2.3.2. Sprite trạng thái idle (đứng yên tại chỗ) 37](#_Toc105863915)

[2.3.3. Sprite trạng thái run (chạy) 38](#_Toc105863916)

[2.3.3. Sprite trạng thái jump (nhảy) 39](#_Toc105863917)

[2.3.4. Sprite trạng thái attack (tấn công) 41](#_Toc105863918)

[2.3.4. Sprite trạng thái receive hit (bị sát thương) 42](#_Toc105863919)

[2.3.4. Sprite trạng thái death (chết) 42](#_Toc105863920)

[3. Push game lên live server 43](#_Toc105863921)

[**D. Demo sản phẩm** 43](#_Toc105863922)

[**TÀI LIỆU THAM KHẢO** 44](#_Toc105863923)

# **DANH MỤC HÌNH ẢNH**

[Hình 1. Hai nhân vật trong game. 13](#_Toc105865285)

[Hình 2. Giao diện Main menu 16](#_Toc105865286)

[Hình 3. Giao diện chọn màn chơi 17](#_Toc105865287)

[Hình 4. Apply thuộc tính hover cho giao diện chọn màn chơi 17](#_Toc105865288)

[Hình 5. Giao diện hướng dẫn chơi 18](#_Toc105865289)

[Hình 6. gameon.html và gameon.js 19](#_Toc105865290)

[Hình 7. Định dạng canvas 20](#_Toc105865291)

[Hình 8. Hai nhân vật trong game vừa khởi tạo 22](#_Toc105865292)

[Hình 9. Hai nhân vật khi có velocity 23](#_Toc105865293)

[Hình 10. Hai nhân vật khi có đủ velocity và gravity 25](#_Toc105865294)

[Hình 11. Attack box của 2 nhân vật 28](#_Toc105865295)

[Hình 12. Hoàn thiện attack box của 2 nhân vật 31](#_Toc105865296)

[Hình 13. Lên ý tưởng cho thanh healthbar 32](#_Toc105865297)

[Hình 14. Thanh healthbar 32](#_Toc105865298)

[Hình 15. Trường hợp hòa 34](#_Toc105865299)

[Hình 16. Trường hợp player 2 thắng 34](#_Toc105865300)

[Hình 17. Trường hợp player 1 thắng 35](#_Toc105865301)

[Hình 18. Trường hợp player 2 hết sạch HP và player 1 thắng 35](#_Toc105865302)

[Hình 19. Sắp xếp folder project 35](#_Toc105865303)

[Hình 20. Apply background thành công 36](#_Toc105865304)

[Hình 21. Sprite idle của nhân vật kenji 37](#_Toc105865305)

[Hình 22. Sprite idle của nhân vật samuraiMack 37](#_Toc105865306)

[Hình 23. Sprite run 38](#_Toc105865307)

[Hình 24. Sprite jump 39](#_Toc105865308)

[Hình 25. Sprite attack 41](#_Toc105865309)

# **LỜI NÓI ĐẦU**

Trong thời đại ngày nay, ngành công nghiệp game trên thế giới nói chung và ở Việt Nam nói riêng đang ngày càng phát triển một cách mạnh mẽ. Số lượng trò chơi được xuất bản ngày càng tăng và mức doanh thu cực khủng đã đóng góp một phần không nhỏ đến sự tăng trưởng kinh tế toàn cầu.

Bên cạnh đó, ngành công nghiệp game cũng có những mặt tiêu cực. Nếu chúng ta biết cân bằng và kiểm soát chúng thì game thực sự trở thành một món ăn tinh thần, một thú vui và đồng thời là một công cụ để giải trí tuyệt vời sau những khoảng thời gian học tập và làm việc căng thẳng. Game cũng có khả năng kết nối người chơi một cách kỳ diệu, kích thích não bộ con người trở nên nhạy bén và thông minh hơn.

Các dòng game dễ chơi, dễ tiếp cận đang là xu hướng hiện tại của cả thế giới, điển hình là các dòng game mobile khi smartphone là một công cụ không thể thiếu đối với mỗi người. Nhưng không chỉ riêng game mobile, game web cũng là một sự lựa chọn không tồi khi nó dễ dàng đáp ứng hầu như tất cả các nhu cầu mà người chơi mong muốn, không cần cấu hình máy quá mạnh, chỉ cần kết nối mạng internet.

Chính vì những lẽ đó, mà em đã bắt tay vào thực hiện đề tài xây dựng game 2D - **Brave** trên nền tảng web với mong muốn nó sẽ trở thành một trong những trò chơi mang tính giải trí nhẹ cho mọi người. Được xây dựng bằng ngôn ngữ HTML, CSS và JavaScript thân thiện, đơn giản, cho nên game sẽ chỉ đáp ứng được những nhu cầu tối thiểu về gameplay, hình ảnh và âm thanh. Nhưng em vẫn hy vọng rằng đây là một sản phẩm tốt để mang đến cho mọi người.

Do kiến thức thực tế và hiểu biết còn hạn chế nên trong quá trình hoàn thành báo cáo không tránh khỏi sai sót, em rất mong sự góp ý đóng góp của các thầy cô để trong tương lai các bài tập của em được tốt hơn.

Em xin chân thành cảm ơn!

# **A. Đề tài**

## I. Tên đề tài

Xây dựng game đối kháng Brave trên nền tảng web sử dụng HTML, CSS và JavaScript.

## II. Phương pháp nghiên cứu

* Về mặt lý thuyết: Nghiên cứu các tài liệu liên quan đến các nội dung nghiên cứu, tài liệu về các ngôn ngôn ngữ lập trình, hiểu được cơ chế của game.
* Về mặt lập trình: Nghiên cứu, xây dựng bằng ngôn ngữ HTML, CSS, JavaScript.
* Về mặt hoạt động: Xây dựng game Brave sẽ là game trực tuyến, người chơi cần phải có internet để truy cập vào domain web game.

## III. Nhiệm vụ và kết quả đạt được

* Nghiên cứu giao diện của game.
* Tìm hiểu cơ chế khi thiết kế game.
* Tìm hiểu các công cụ để xây dựng ứng dụng game.

## IV. Bố cục đề tài

Nội dung gồm 4 phần:

* Phần A: Mở đầu (giới thiệu tên đề tài và nêu bố cục đề tài)
* Phần B: Tìm hiểu về HTML, CSS và JavaScript
* Phần C: Quá trình thực hiện xây dựng game Brave
* Phần D: Demo sản phẩm

# **B. Tổng quan về HTML, CSS và JavaScript**

## I. HTML

## 1. HTML là gì?

HTML (viết tắt của từ HyperText Markup Language, hay là "Ngôn ngữ Đánh dấu Siêu văn bản") là một ngôn ngữ đánh dấu được thiết kế ra để tạo nên các trang web trên World Wide Web.

HTML có thể được trợ giúp bởi các công nghệ như CSS và các ngôn ngữ kịch bản giống như JavaScript.

HTML không phải là ngôn ngữ lập trình, đồng nghĩa với việc nó không thể tạo ra các chức năng “động” được. Nó chỉ giống như Microsoft Word, dùng để bố cục và định dạng trang web.

Khi làm việc với HTML, chúng ta sẽ sử dụng cấu trúc code đơn giản (tags và attributes) để đánh dấu lên trang web. Ví dụ, chúng ta có thể tạo một đoạn văn bằng cách đặt văn bản vào trong cặp tag mở và đóng văn bản <p> và </p>

## 2. CSS là gì?

CSS là chữ viết tắt của Cascading Style Sheets, nó 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). Nói ngắn gọn hơn là ngôn ngữ tạo phong cách cho trang web.

Nếu HTML đóng vai trò định dạng các phần tử trên website như việc tạo ra các đoạn văn bản, các tiêu đề, bảng,…thì CSS sẽ giúp chúng ta có thể thêm style vào các phần tử HTML đó như đổi bố cục, màu sắc trang, đổi màu chữ, font chữ, thay đổi cấu trúc…

Phương thức hoạt động của CSS là nó sẽ tìm dựa vào các vùng chọn, vùng chọn có thể là tên một thẻ HTML, tên một ID, class hay nhiều kiểu khác. Sau đó là nó sẽ áp dụng các thuộc tính cần thay đổi lên vùng chọn đó.

Mối tương quan giữa HTML và CSS rất mật thiết. HTML là ngôn ngữ markup (nền tảng của site) và CSS định hình phong cách (tất cả những gì tạo nên giao diện website), chúng là không thể tách rời.

## 3. JavaScript là gì?

JavaScript là ngôn ngữ lập trình phổ biến nhất trên thế giới trong suốt 20 năm qua. Nó cũng là một trong số 3 ngôn ngữ chính của lập trình web.

JavaScript giúp cải thiện cách hoạt động của trang web. JavaScript thường được nhúng trực tiếp vào một trang web hoặc được tham chiếu qua file .js riêng. Nó là ngôn ngữ phía client, tức là script được tải về máy của khách truy cập và được xử lý tại đó thay vì phía server là xử lý trên server rồi mới đưa kết quả tới khách truy cập.

JavaScript hoặc JS sẽ giúp tăng tính tương tác trên website. Script này chạy trên các trình duyệt của người dùng thay vì trên server và thường sử dụng thư viện của bên thứ 3 nên có thể tăng thêm chức năng cho website mà không phải code từ đầu.

# **C. Quá trình thực hiện xây dựng game Brave**

## I. Sơ lược về sản phẩm

* **Bối cảnh:** Hình ảnh trong game mô tả việc một chú cún tên Brave đang đối đầu với các quái vật trong khu rừng.
* **Thể loại:** Phiêu lưu (Adventure game)
* **Phong cách đồ họa:** 2D
* **Nền tảng phát triển:** nền tảng web
* **Ngôn ngữ:** HTML, CSS và JavaScript

## II. Các đối tượng trong game

* **Chú cún Brave**
* **Các monsters khác.**

**Xác định phương thức chơi:** người chơi sẽ điều khiển chú cún Brave. Chú phải tiêu diệt một lượng nhất định các monster khác bằng cách dùng skill rolling fire trong một khoảng thời gian cho phép để có thể chiến thắng. Nếu chú không sử dụng skill và va chạm với các monster khác, thì sẽ bị mất HP dần, và game sẽ kết thúc khi chú hết sạch HP.

A picture containing text, aircraft, night sky

Description automatically generated

Hình . Chú cún Brave trong game.

## III. Thực hiện xây dựng game

## 1. Thiết lập giao diện game menu

## 1.1. Lên ý tưởng giao diện

Game menu sẽ là trang đầu tiên khi người chơi khởi động game. Bao gồm các button để người chơi có thể tùy chỉnh, chẳng hạn như:

* New game button (bắt đầu trò chơi)
* Guide (xem hướng dẫn điều khiển)

Game menu sẽ được chèn nhạc background để tăng tính hấp dẫn người chơi trước khi vào game.

## 1.2. Cài đặt giao diện

Sử dụng trình biên dịch Visual Studio code, cài đặt một số extensions cần thiết cho quá trình lập trình liên quan đến HTML, CSS và JavaScript.

Tạo một file html với tên gọi “index.html”, thiết đặt các thẻ *<div>* chứa các class liên quan đến thành phần game menu.

Thực hiện chèn nhạc background vào trang bằng thẻ *<embed>,* trỏ đường dẫn đến file nhạc, cài đặt sự lặp lại bài nhạc với *loop=“infinite”,* thiết đặt tự động phát với *autostart=“true”.*

<embed id="myAudio" src="/sounds/menu.mp3" loop="infinite" autostart="true">

Sử dụng sự kiện *onclick=“document.location”* cho button New Game để trỏ đến file html của gameplay, và thiết lập function javascript cho button Guide để trỏ đến thẻ *<div>* hướng dẫn cách điều khiển trong game.

Cài đặt 1 button chức năng đưa người chơi trở lại màn hình chính để sử dụng các chức năng khác. Sử dụng thẻ <div> với class “return”, bên trong thẻ này sẽ chứa một thẻ *<button>* với class “button1” có sự kiện *onclick* để tạo chức năng chuyển người chơi đến trang main menu khi click vào.

    <script>

// khai báo các biến

      var btnguide = document.querySelector('#btnguide');

      var showguide1 = document.querySelector('.guidepage');

      var mainmenu = document.querySelector('.mainmenu');

      var showmenu1 = document.querySelector('.mainmenu');

      var btnmenu = document.querySelector('#btnmenu');

      function showguide() { // hàm để show guide khi click vào button Guide

        showguide1.style.display = "block";

        mainmenu.style.display = "none";

      }

      function showmenu() { // hàm để show mainmenu khi click vào button Main Menu

        showmenu1.style.display = "block";

        showguide1.style.display = "none";

      }

    </script>

Sử dụng CSS để cài giao diện cho class “button1” và apply background cho game menu. Tạo file “style.css” để cài đặt.

Cài đặt ảnh nền cho menu bằng cách set thuộc tính background-image trong phần body của CSS.

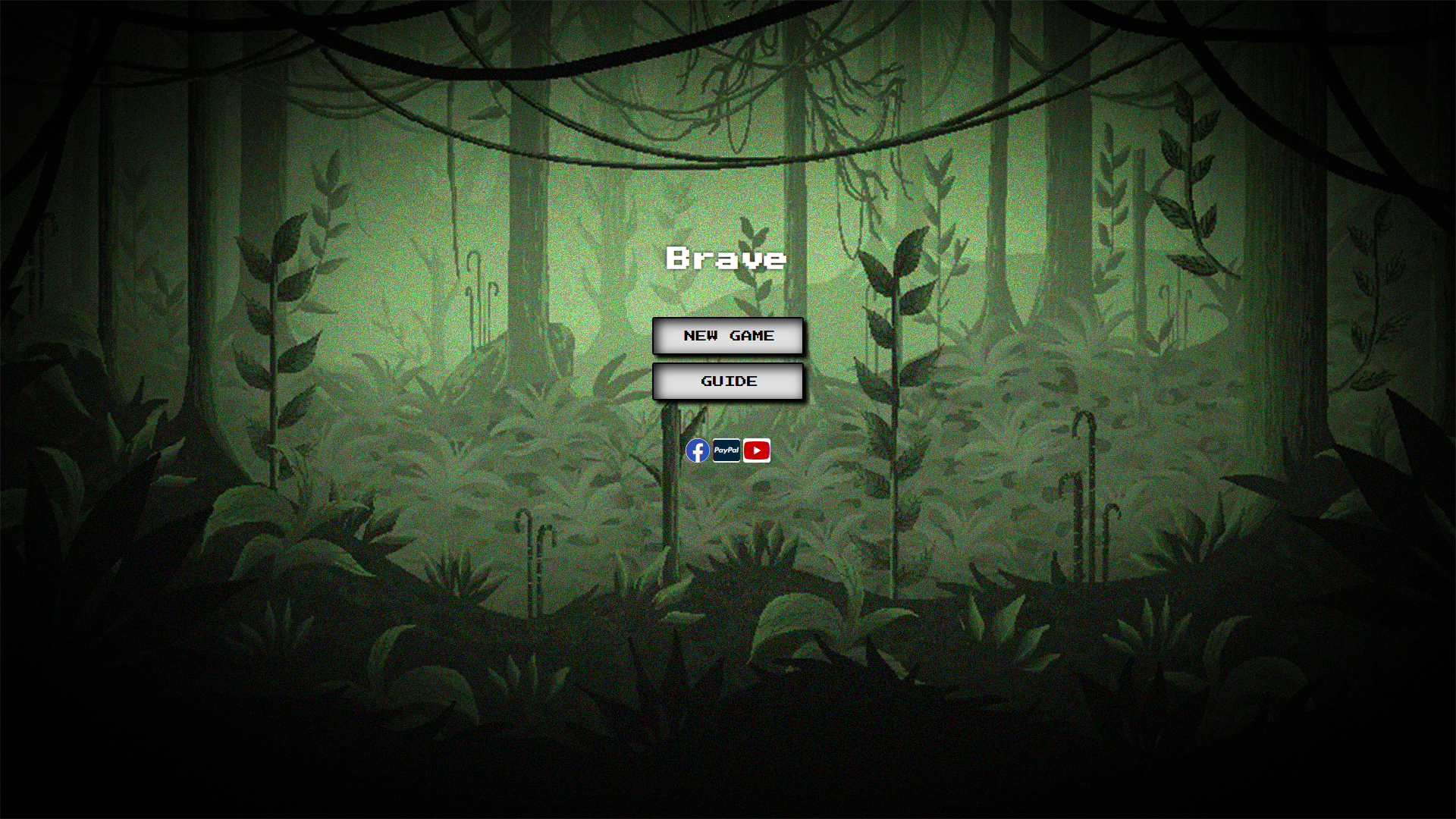
body {

    background-image: url("/img/wallpaper.png");

    text-align: center;

    position: relative; }

Cài đặt các thuộc tính trong CSS cho button1 để tạo các button hoàn chỉnh. Sau đó, được kết quả giao diện như sau:



Hình . Giao diện Main menu

## 2. Xây dựng cơ chế game

## 2.1. Lên ý tưởng

Game sẽ được đặt trên một nền canvas với một kích thước nhất định theo tỷ lệ khung hình phổ biến nhất của hầu hết các màn hình máy tính hiện nay là 16:9, trong trường hợp này, vì game sẽ sử dụng các sprite background có kích thước nhỏ, nên kích thước canvas cho game phù hợp nhất đối với các sprite đó sẽ là 1667x500 pixel.

Nhân vật chú chó Brave sẽ được tạo nên từ hình chữ nhật (Red Rectangle). Người chơi sẽ phải điều khiển Brave để tiêu diệt các quái vật và dành chiến thắng.

## 2.2. Thực hiện xây dựng project

## 2.2.1. Cài đặt project:

Tạo file gameon.html để thiết đặt các khung sườn cho game, và main.js để thiết đặt các cơ chế chính cho game.

Như đã lên ý tưởng, đầu tiên ta sẽ khởi tạo một canvas để làm khu vực game hoạt động, đồng thời sẽ là nơi chịu trách nhiệm vẽ ra các hình ảnh trong game. Sử dụng thẻ *<canvas></canvas>* thuộc phần *<body>* trong file html để khởi tạo một canvas element.

Để định dạng kích thước cho canvas, tiến hành khởi tạo một hằng số (const) trong main.js cho element canvas.

Tạo thêm một hằng số (const) và gọi hằng số đó là c, gán hằng canvas vào hằng c vừa khởi tạo. Vì game sẽ được xây dựng theo mô hình 2D nên ta sử dụng phương thức *getContext(‘2d’)* cho canvas.

Tiếp đến, ta cần phải sửa kích thước của canvas thành 1667x500px như đã lên ý tưởng.

window.addEventListener('load', function(){

  const canvas = document.getElementById('canvas1');

  const ctx = canvas.getContext('2d');

  canvas.width = 1667;

  canvas.height = 500;

Tạo class Game, tất cả các logic trong project này sẽ thực thi tại đây. Với constructor bao gồm các arguments là width (độ rộng) và height (độ cao) của canvas. Bên trong đó, chúng ta hoán đổi chúng sang các thuộc tính của class. Bên ngoài constructor, sẽ có một method *update()* để thực thi tất cả các animation frame và kích hoạt tất cả các sự tính toán cần xảy ra. Một method *draw()* dẽ giúp ta thể hiện được các hình ảnh game, score (điểm) và nhiều thứ khác.

import { Player } from './player.js';

class Game {

    constructor(width, height) {

      this.width = width;

      this.height = height;

this.player = new Player(this); //các properties link với class Player

this.debug = false; //debug mode để show hitbox (khung nhân vật)

      this.gameOver = false; //sự kiện game over mặc định là false

      …

    }

    update() {

…

    }

    draw() {

…

    }

  }

## 2.2.2. Khởi tạo các nhân vật

Ta sẽ xác định sử dụng lập trình hướng đối tượng vì các nhân vật trong game cần phải tương tác với nhau. Chúng sẽ có các thuộc tính riêng lẻ hoạt động độc lập với nhau để tạo ra một đối tượng hoàn chỉnh.

Tạo một file javascript có tên *player.js* để cài đặt các thuộc tính và chức năng cho chú chó Brave. Class Player sẽ có nhiệm vụ tạo hình và update nhân vật Brave của chúng ta. Constructor sẽ lấy tất cả đối tượng game như là một argument.

Trong method *draw(),* đối số *context* để chỉ định đến canvas element mà ta cần vẽ lên.

export class Player {

    constructor(game) {

        this.game = game;

//mỗi frame trong sprite sheet có kích thước width và height lần lượt là

        this.width = 100;

        this.height = 91.3;

//vị trí x và y của nhân vật trên canvas

//với this.game.height là chiều cao của canvas, this.height là chiều cao

của nhân vật, this.game.groundMargin là khoảng cách giữa mặt đất và khung canvas.

        this.x = 0;

        this.y = this.game.height - this.height - this.game.groundMargin;

    }

    update() {

        …

    }

    draw(context) {

        …

    }

Để thấy được nhân vật của chúng ta trông như thế nào, chúng ta tiến hành tạo một thẻ div trong *gameon.html* và dẫn đường dẫn đến file hình ảnh của nhân vật. Các hình ảnh khác sử dụng trong game cũng được thêm vào tương tự. set style của các id hình ảnh đó trong css với *display: none* để tất cả được giấu đi, chỉ khi được gọi đến mới hiển thị lên màn hình game.

<body>

    <div class="game">

        <canvas id="canvas1"></canvas>

        <img id="player" src="./img/sprite/player.png" alt="">

        <img id="layer4" src="./img/sprite/layer-4.png" alt="">

        <img id="layer5" src="./img/sprite/layer-5.png" alt="">

        <img id="enemy\_fly" src="./img/sprite/enemy\_fly.png" alt="">

        <img id="enemy\_plant" src="./img/sprite/enemy\_plant.png" alt="">

        <img id="enemy\_spider\_big" src="./img/sprite/enemy\_spider\_big.png" alt="">

        <img id="fire" src="./img/sprite/fire.png" alt="">

        <img id="collisionAnimation" src="./img/sprite/boom.png" alt="">

        <img id="lives" src="./img/sprite/lives.png" alt="">

    </div>

Đồng thời tại *draw()* method của class *Player*, sử dụng *strokeRect()* để vẽ khung nhân vật lên màn hình canvas. Nếu debug mode có giá trị là true, thì khung nhân vật sẽ hiển thị.

export class Player {

    constructor(game) {

…

    }

    update() {

        …

    }

    draw(context) {

        if (this.game.debug) context.strokeRect(this.x, this.y, this.width, this.height);

        context.drawImage(this.image, this.frameX \* this.width, this.frameY \* this.height, this.width, this.height, this.x, this.y, this.width, this.height);

    }

Tại *draw()* method của *main.js*, ta cũng gọi đến *draw()* method của *player.js* để xuất hình ảnh của khung hitbox lên màn hình.

Hằng *const game* của *Game* class để kích hoạt class constructor, và nó sẽ tự động tạo ra nhân vật.

Để game được update và draw liên tục với 60fps, tạo một function animate() gọi đến các method update() và draw(context) và sau đó thêm một điều kiện nếu trò chơi không phải trạng thái gameOver thì các animation vẫn được hoạt động bình thường. Trong đó, *clearRect()* sẽ có nhiệm vụ xóa canvas trong mỗi khoảng thời gian được update và draw.

draw(context) {

      this.player.draw(context);

    …

    }

}

const game = new Game(canvas.width, canvas.height);

function animate(timeStamp) {

    ctx.clearRect(0,0,canvas.width, canvas.height);

    game.update(deltaTime);

    game.draw(ctx);

    if (!game.gameOver) requestAnimationFrame(animate);

  }

  animate(0);

});

A picture containing text, screenshot

Description automatically generated

## 2.2.3. Di chuyển nhân vật bằng Event Listeners (keys)

Các key điều khiển cần được thiết lập cho 2 nhân vật là:

* Đối với player:
* W, A, D lần lượt là Nhảy, sang trái, sang phải
* Đối với enemy:
* Phím ArrowUp, ArrowLeft, ArrowRight lần lượt là Nhảy, sang trái, sang phải.

Tạo *const keys* và định nghĩa cho các key với giá trị được nhấn (pressed) là false để game tự động nhận diện là chưa có sự tác động vào các phím khi game vừa khởi chạy.

const keys = {

  a: {

    pressed: false

  },

  d: {

    pressed: false

  },

  ArrowRight: {

    pressed: false

  },

  ArrowLeft: {

    pressed: false

  }}

Các vòng lặp if để khởi tạo điều kiện cho các phím bấm tương tác với di chuyển nhân vật.

  if (keys.a.pressed && player.lastKey === 'a') {

    player.velocity.x = -10

  } else if (keys.d.pressed && player.lastKey === 'd') {

    player.velocity.x = 10

  }

  //enemy movement

  if (keys.ArrowLeft.pressed && enemy.lastKey === 'ArrowLeft') {

    enemy.velocity.x = -10

  } else if (keys.ArrowRight.pressed && enemy.lastKey === 'ArrowRight') {

    enemy.velocity.x = 10

  }

Để sử dụng event listener, tiến hành thêm method *addEventListener()* cho các sự kiện key down và key up vào file gameon.js. Để mỗi khi ta nhấn phím, nó sẽ bắt sự kiện từ đây. Trong sự kiện keydown và keyup, ta thực hiện gán giá trị true và false tương ứng khi người chơi nhấn và thả phím. Đây là các vòng lặp switch case dành cho các phím bấm trong sự kiện. Nếu như người chơi bấm phím, sự kiện sẽ được kích hoạt và nhân vật sẽ di chuyển một khoảng bao nhiêu pixel tùy chỉnh

window.addEventListener('keydown', (event) => {

  //phím của player

  switch (event.key) {

    case 'd':

      keys.d.pressed = true

      player.lastKey = 'd'

    break

    case 'a':

      keys.a.pressed = true

      player.lastKey = 'a'

    break

    case 'w':

      player.velocity.y = -20

      player.lastKey = 'w'

    break

//phím của enemy

    case 'ArrowRight':

      keys.ArrowRight.pressed = true

      enemy.lastKey = 'ArrowRight'

    break

    case 'ArrowLeft':

      keys.ArrowLeft.pressed = true

      enemy.lastKey = 'ArrowLeft'

    break

    case 'ArrowUp':

      enemy.velocity.y = -20

    break

  }

  console.log(event.key)

})

Nhân vật cũng cần vận tốc di chuyển. Toán tử *this.position.x += this.velocity.x* tại hàm *update()* sẽ thực hiện điều này*.*

Sự kiện keyup và set cho *velocity.x = 0* tại *function animate()* để nhân vật có thể dừng lại khi người chơi thả phím sau khi nhấn phím kích hoạt di chuyển.

function animate(){

  …

  player.velocity.x = 0

enemy.velocity.x = 0

  …

}

window.addEventListener('keyup', (event) => {

  //phím của player

  switch (event.key) {

    case 'd':

      keys.d.pressed = false

    break

    case 'a':

      keys.a.pressed = false

    break

  }

  //phím của enemy

  switch (event.key) {

    case 'ArrowRight':

      keys.ArrowRight.pressed = false

    break

    case 'ArrowLeft':

      keys.ArrowLeft.pressed = false

    break

  }

  console.log(event.key)

})

## 2.2.4. Cơ chế tấn công (gây sát thương)

Các nhân vật đều có một phạm vi tấn công nhất định. Nếu một trong 2 nhân vật đứng trong phạm vi tấn công của đối phương thì sẽ bị trừ một lượng HP.

Như vậy, để thiết lập một phạm vi tấn công cho nhân vật, ta có thể tạo cho mỗi nhân vật một hình chữ nhật mới nằm ngang (attackBox), gắn liền với nhân vật khi nhân vật di chuyển. Khi đối phương chạm vào attackBox thì sẽ bị trừ HP.

Tạo thuộc tính attackBox cho *class Fighter*, đồng thời gán vị trí cho attackBox luôn đi theo nhân vật bằng cách gọi *position: this.position*

Tạo thuộc tính *isAttacking* cho *class Fighter*, để cài đặt nhận diện nhân vật đang tấn công.

this.attackBox = {

      position: this.position,

      width: 100,

      height: 50

    }

this.isAttacking

Fill attackBox bằng cách gọi *fillRect()* trong method *draw()*.

    //attack box

    c.fillStyle = 'green'

    c.fillRect (

      this.attackBox.position.x,

      this.attackBox.position.y,

      this.attackBox.width,

      this.attackBox.height

)

Chart

Description automatically generated

Hình . Attack box của 2 nhân vật

Tiếp theo, để thiết lập nhận diện va chạm. Cần phân tích một số điểm như sau:

* Khi nhân vật di chuyển trái phải:
* Vị trí x của cạnh bên phải attackBox của player lớn hơn hoặc bằng vị trí x của enemy,
* Vị trí x của attackBox của player bé hơn hoặc bằng vị trí x của cạnh bên phải của enemy.
* Khi nhân vật nhảy:
* Vị trí y của cạnh bên trên attackBox của player lớn hơn hoặc bằng vị trí y của enemy,
* Vị trí y của attackBox của player bé hơn hoặc bằng vị trí y của cạnh bên trên attackBox của enemy.

Nếu các nhân vật thỏa mãn các điều kiện như trên thì sự va chạm sẽ xảy ra. Như vậy*.* Trước tiên ta tạo một *function rectangularCollision()* mới ngay trên *function animate()*, khai báo 2 đối tượng là rectangular1 và rectangular2 thay thế lần lượt cho player và enemy. Cho return về tất cả các điều kiện như phân tích ở trên

function rectangularCollision({ rectangle1, rectangle2 }) {

  return (

    rectangle1.position.x + rectangle1.attackBox.width >= rectangle2.position.x &&

    rectangle1.attackBox.position.x <= rectangle2.position.x + rectangle2.width &&

    rectangle1.attackBox.position.y + rectangle1.attackBox.height >= rectangle2.position.y &&

    rectangle1.attackBox.position.y <= rectangle2.position.y + rectangle2.height

  )

}

Tại 2 vòng lặp if để nhận diện va chạm, hàm *rectangularCollision()* được gọi và gán player cho rectangular1 và enemy cho rectangular2. Các điều kiện *player.isAttacking* và *enemy.isAttacking* ở cuối vòng lặp để nhận diện điều kiện nhân vật đang tấn công.

  //nhận diện va chạm

  if (

    rectangularCollision({

      rectangle1: player,

      rectangle2: enemy

    }) &&

    player.isAttacking

  ) {

    player.isAttacking = false

    console.log('go')

  }

  if (

    rectangularCollision({

      rectangle1: enemy,

      rectangle2: player

    }) &&

    enemy.isAttacking

  ) {

    enemy.isAttacking = false

  }

Khi method *attack()* được gọi thì nhân vật sẽ được set trạng thái *isAttacking = true*, đồng thời trạng thái tấn công sẽ chỉ được diễn ra trong một khoảng thời gian nhất định (set 1000 mili giây), sau đó sẽ trở về trạng thái *isAttacking = false*.

  attack() {

    this.isAttacking = true

    setTimeout(() => {

      this.isAttacking = false

    }, 1000)

  }

Để kích hoạt tính năng tấn công, cần cài đặt một phím để kích hoạt. Với player, cài đặt phím Spacebar. Với enemy, cài đặt phím ArrowDown.

window.addEventListener('keydown', (event) => {

  switch (event.key) {

…

    case ' ':

      player.attack()

    break

…

case 'ArrowDown':

      enemy.attack()

    break

  }

  console.log(event.key)

})

attackBox của enemy đang cùng chiều so với attackBox của player. Nhưng game đối kháng thì 2 nhân vật cần đối mặt vào nhau để gây sát thương lên nhau. Vậy, ta cần đổi chiều của attackBox cho enemy. Thêm thuộc tính offset và chỉnh sửa một chút ở *class Fighter*.

class Fighter {

 constructor({position, velocity, color = 'red', offset}) {

…

      position: {

        x: this.position.x,

        y: this.position.y

      },

offset,

      …

  }

Thêm thuộc tính offset vào *const playe*r và *const enemy*. Set cho offset x của enemy là -50. Công dụng là để bù trừ một khoảng -50 pixel cho attackBox.

const player = new Fighter({

  …

  offset: {

    x: 0,

    y: 0

  },

})

const enemy = new Fighter({

  …

  offset: {

    x: -50,

    y: 0

  },

})

Tại method *update()*, toán tử này nhằm set lại vị trí cho attackBox của enemy.

  update() {

    this.draw()

    this.attackBox.position.x = this.position.x + this.attackBox.offset.x

    … }

Chart, waterfall chart

Description automatically generated

Hình . Hoàn thiện attack box của 2 nhân vật

## 2.2.5. Thanh HP (healthbar) và giao diện đồ họa của game

Về thanh healthbar, ta cần tạo một element mới trong html để bắt đầu cài đặt. Tạo 3 cặp thẻ *<div>* tương ứng với thanh *player health, timer (bộ đếm thời gian), enemy health.* Hình phát thảo minh họa ý tưởng như bên dưới.

<body>

  <div id="containerbase" style="position: relative; display: inline-block">

    <div style="position: absolute; display: flex; width: 100%; align-items: center; padding: 20px">

      <!-- player health -->

      <div></div>

      <!-- timer -->

      <div></div>

      <!-- enemyhealth -->

      <div></div>

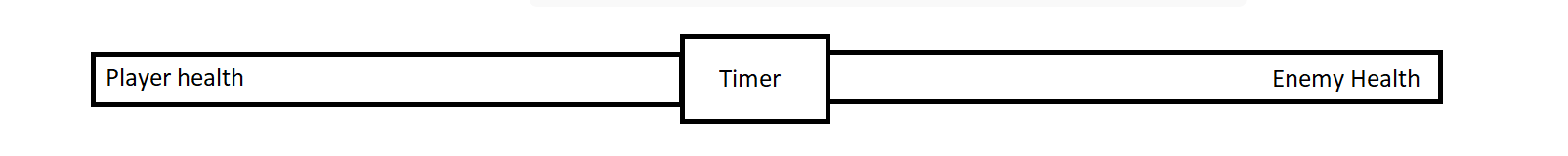
    </div>

    <canvas></canvas>

  </div>

  <script src="index.js"></script>

</body>



Hình . Lên ý tưởng cho thanh healthbar

Thẻ *<div>* cha có thuộc tính *position: relative*; *display: inline-block* có công dụng tạo một container, giúp cho thanh healthbar luôn được nằm bên trên va trong canvas. Thẻ *<div>* con có thuộc tính *position: absolute* có nhiệm vụ tạo thêm một container nữa làm cho thanh healthbar nằm chồng lên canvas. Thêm thuộc tính *display: flex; width: 100%* vào lớp cha của 3 phần tử trên thanh healthbar để 3 phần tử này được xếp thành một hàng ngang ở phía trên của canvas, đồng thời phần healthbar sẽ được trải dài 100%.

Thiết lập các thuộc tính styles cho 3 lớp con để định dạng thanh healthbar.

A picture containing chart

Description automatically generated

Hình . Thanh healthbar

Khi 2 nhân vật gây sát thương lên nhau và thanh HP của nhân vật bị sát thương trừ một lượng HP tương ứng, trước hết, thuộc tính *this.health = 100* vào *class Fighter* để xác định lượng HP của mỗi nhân vật có giá trị là 100.

Câu lệnh if nhận diện va chạm:

  if (

    …

    player.isAttacking = false

    enemy.health -= 10

    document.querySelector('#enemyHealth').style.width = enemy.health + '%'

  }

Hai toán tử *enemy.health -= 10* và *player.health -= 10* có nhiệm vụ thiết đặt cho cả 2 nhân vật mỗi khi nhận sát thương đều bị khấu trừ một lượng HP là 10%.

## 2.2.6. Bộ đếm ngược thời gian và sự kiện để trò chơi kết thúc

Function *decreaseTimer()* tại file gameon.js có chức năng bất kể khi nào ta gọi đến function này, thì bộ đếm thời gian sẽ giảm từng giây. Để thực hiện điều này, ta cần set cho bộ đếm thời gian một khoảng thời lượng là 60 giây. Khởi tạo biến giá trị *let timer* và set cho nó bằng 60. (*timer* là id thẻ *<div>* bộ đếm thời gian trong gameon.html). *let timerId* để sau này thiết lập cho thời gian dừng lại khi trò chơi kết thúc.

Để set cho thời gian giảm từng giây, timer sẽ phải tự trừ đi một đơn vị giá trị của chính nó, khi và chỉ khi timer có giá trị lớn hơn 0 giây, vòng lặp if với điều kiện *timer > 0* rồi sử dụng toán tử *timer--*  sẽ thực hiện chức năng này.

Để timer giảm dần giá trị với tốc độ tương đương với tốc độ của 1 giây. *setTimeout()* function sẽ thực hiện việc này, gọi lại hàm *decreaseTimer* lồng vào trong và set một giá trị bằng 1000 mili giây.

Dùng method *document.querrySelector()*, gọi id *timer* trong file gameon.html vào method với *.innerHTML*, và set cho nó bằng *timer*. Bộ đếm thời gian sẽ bắt đầu đếm ngược.

let timer = 60

let timerId

function decreaseTimer() {

  if (timer > 0)

  {

    timerId = setTimeout(decreaseTimer, 1000)

    timer--

    document.querySelector('#timer').innerHTML = timer

  }

}

Khi đã có bộ đếm ngược thời gian, xét các sự kiện để trò chơi kết thúc. Có 3 trường hợp sẽ xảy ra:

* Khi timer = 0:
* HP của player bằng với HP của enemy thì hòa.
* HP của player nhiều hơn HP của enemy thì player thắng.
* HP của enemy nhiều hơn HP của player thì enemy thắng.
* Khi timer > 0:
* HP của player bằng 0 thì enemy thắng.
* HP của enemy bằng 0 thì player thắng.

Cũng tiếp tục sử dụng vòng lặp if và method document.querySelector để gọi id displayText từ HTML, tạo function *determineWinner()* để xác định người chiến thắng cho player và enemy. timerId và clearTimeout(timerId) có chức năng dùng để dừng đồng hồ khi game kết thúc.

function determineWinner({player, enemy, timerId}) {

clearTimeout(timerId)

document.querySelector('#displayText').style.display = 'flex'

  if (player.health === enemy.health) {

    document.querySelector('#displayText').innerHTML = 'Tie'

  } else if (player.health > enemy.health) {

    document.querySelector('#displayText').innerHTML = 'Player 1 Wins'

  } else if (player.health < enemy.health) {

    document.querySelector('#displayText').innerHTML = 'Player 2 Wins'

  }

}

Tại function decreaseTimer(), vòng lặp if để xác định người chiến thắng khi timer = 0. Và một vòng lặp if để xác định người chiến thắng khi 1 trong 2 nhân vật hết HP.

  if(timer === 0) {

    determineWinner({player, enemy, timerId})

  }

  if (enemy.health <= 0 || player.health <= 0) {

    determineWinner({player, enemy, timerId})

  }

Graphical user interface, application

Description automatically generated

Hình . Trường hợp hòa

A picture containing graphical user interface

Description automatically generated

Hình . Trường hợp player 2 thắng

A picture containing graphical user interface

Description automatically generated

Hình . Trường hợp player 1 thắng

A picture containing graphical user interface

Description automatically generated

Hình . Trường hợp player 2 hết sạch HP và player 1 thắng

## 2.3. Cài đặt các Sprite và animation cho game

## 2.3.1. Background Sprite

Folder img có nhiệm vụ lưu trữ tất cả các file hình ảnh tại folder của project. 3 file hình ảnh của 3 background màn chơi sẽ được lưu tại đây.

Folder js dùng để lưu các file javascript, tại đây có file *classes.js* chứa tất cả các cài đặt của các class. Mục đích là để các file code chức năng được phân chia rõ ràng, khoa học và không bị lằng nhằng.

Graphical user interface, application

Description automatically generated

Hình . Sắp xếp folder project

Tại *classes.js*, tạo *Sprite class*, gọi các thuộc tính tương tự như *class Fighter*, loại bỏ các thuộc tính như velocity, lastKey, các thuộc tính liên quan đến attack và health*.* Nhưng có một điều, tại class Sprite này, ta tiến hành làm việc với các sprite sheet và hình ảnh trong game, cho nên ta sẽ gọi các thuộc tính image và imageSrc để lấy nguồn hình ảnh từ trong thư mục project.

class Sprite {

  constructor({position, imageSrc }) {

    this.position = position

    this.width = 50

    this.height = 150

    this.image = new Image()

    this.image.src = imageSrc

  }

  draw() {

    c.drawImage(this.image, this.position.x, this.position.y)

  }

  update() {

    this.draw()

  }

}

A picture containing website

Description automatically generated

Hình . Apply background thành công

Backgroud có phần mặt đất, ta cần phải điều chỉnh vị trí của các nhân vật cho phù hợp. Tại điều kiện if của hàm update() trong classes.js, lấy chiều cao của canvas trừ đi một khoảng 97 pixel, các nhân vật sẽ được đặt về đúng vị trí.

  update() {

    …

    if(this.position.y + this.height >= canvas.height - 97) {

      this.velocity.y = 0

    }

    …

  }

## 2.3.2. Sprite trạng thái idle (đứng yên tại chỗ)

Sprite sheet của trạng thái idle của nhân vật SamuraiMack (player đỏ) sẽ có 8 frame (khung hình chuyển động), và của Kenji (enemy xanh) là 4 frame.

A picture containing dark

Description automatically generated

Hình . Sprite idle của nhân vật kenji

A picture containing night sky

Description automatically generated

Hình . Sprite idle của nhân vật samuraiMack

Để thêm sprite animation cho các nhân vật, ta cần lấy tất cả các thuộc tính từ *class Sprite* cho *class Fighter*, sử dụng *extends* để *class Fighter* được kế thừa tất cả các thuộc tính từ *class Sprite*. Nhưng khi lấy các thuộc tính, ta chỉ cần chọn lựa những thuộc tính cần thiết, cái nào cần giữ lại cho *class Fighter*: position, imageSrc, scale, framesMax, offset.

class Fighter extends Sprite {

  constructor({position, velocity, color = 'red', offset, imageSrc, scale = 1, framesMax = 1, offset = { x: 0, y: 0 }}) {

    super({

      position,

      imageSrc,

      scale,

      framesMax,

offset

    })

    …

    }

    …

    this.frameCurrent = 0

    this.framesElapsed = 0

    this.framesHold = 10

  }

Gọi imageSrc cho const player và const enemy. Player có 8 frame, và enemy có 4 frame nên ta set framesMax: 8 và framesMax: 4 cho lần lượt 2 nhân vật. Tiến hành thêm sprite vào cho nhân vật

Để điều chỉnh kích cỡ cho sprite, set scale bằng 2,5 cho cả 2 nhân vật, đồng thời set vị trí offset cho sprite về đúng vị trí trên màn hình.

const player = new Fighter({

  …

  imageSrc: './img/samuraiMack/Idle.png',

  framesMax: 8,

scale: 2.5,

  offset: {

    x: 215,

    y: 157

  },

})

const enemy = new Fighter({

…

  imageSrc: './img/kenji/Idle.png',

  framesMax: 4

scale: 2.5,

  offset: {

    x: 215,

    y: 187

  },

})

## 2.3.3. Sprite trạng thái run (chạy)

Thuộc tính *sprites* của *class Fighter* là một vật thể chứa tất cả các sprites của mỗi nhân vật riêng biệt.



Hình . Sprite run

Thuộc tính *sprites* được gọi vào *const player* để ta có thể thiết đặt các thông số cho sprites. Trong đó bao gồm idle sprite và run sprite,…

class Fighter extends Sprite {

  constructor({

…

    sprites

  sprites: {

    idle: {

      imageSrc: './img/samuraiMack/Idle.png',

      framesMax: 8

    },

    run: {

      imageSrc: './img/samuraiMack/Run1.png',

      framesMax: 8

    }

  }

Vòng lặp *for* tại *class Fighter* với *const sprite in sprites*. Vòng lặp này có tác dụng lặp đi lặp lại các sprite thay phiên nhau, chuyển trạng thái từ idle sang run, hoặc từ run sang idle. *sprite* sẽ là đối tượng mấu chốt để ta tham chiếu sprite idle hay run.

for (const sprite in sprites) {

      sprites[sprite].image = new Image()

      sprites[sprite].image.src = sprites[sprite].imageSrc

    }

Tiến hành thiết đặt khi người chơi nhấn phím di chuyển thì nhân vật sẽ kích hoạt sprite run. Tại phần player movement, ta thêm vào các vòng lặp if nếu người chơi nhấn phím thì game sẽ gọi đến thuộc tính image và sprites của nhân vật. Nếu không nhấn nút gì thì nhân vật ở trạng thái idle.

  //player movement

  if (keys.a.pressed && player.lastKey === 'a') {

    player.velocity.x = -10

    player.switchSprite('run')

  } else if (keys.d.pressed && player.lastKey === 'd') {

    player.velocity.x = 10

    player.switchSprite('run')

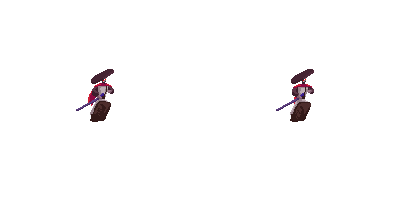
  } else {

    player.switchSprite('idle')

  }

//switchSprite sẽ được giải thích trong phần sau.

## 2.3.3. Sprite trạng thái jump (nhảy)



Hình . Sprite jump

Tương tự như trạng thái run, Thuộc tính *jump* cũng được gọi vào *const player* để ta có thể thiết đặt các thông số cho sprites. Sprite của trạng thái jump chỉ có 2 frame nên ta đặt framesMax là 2.

    jump: {

      imageSrc: './img/samuraiMack/Jump.png',

      framesMax: 2

    }

Function switchSprite sẽ đảm nhiệm việc thay đổi trạng thái giữa các sprite với nhau. Mệnh đề switch case sẽ giúp thay đổi sang sprite khác khi chuyển trạng thái. Các case sẽ xác định các trường hợp để thay đổi sprite sang trạng thái được thiết đặt.

switchSprite(sprite) {

    switch (sprite) {

      case 'idle':

        if (this.image !== this.sprites.idle.image) {

          this.image = this.sprites.idle.image

          this.framesMax = this.sprites.idle.framesMax

          this.framesCurrent = 0

        }

        break

      case 'run':

        …

        }

        break

      case 'jump':

        …

        }

        break

    }

  }

Khi nhân vật nhảy lên, thì sprite của nhân vật sẽ chuyển từ trạng thái idle hoặc run sang trạng thái jump. Như vậy, ta cần 1 vòng lặp if với điều kiện khi velocity theo trục y của nhân vật bé hơn 0, thì sprite sẽ được chuyển sang trạng thái jump. Vòng lặp if này thuộc về cơ chế chuyển động của nhân vật nên sẽ được đặt tại phần cơ chế player movement.

if (player.velocity.y < 0) {

    player.switchSprite('jump')

  }

Nhân vật sau khi nhảy thì sẽ rơi xuống, tương tự jump, cài đặt chuyển sprite cho nhân vật sang trạng thái rơi cũng với switch case và điều kiện if. Và khi velocity theo trục y của nhân vật lớn hơn 0, thì sprite sẽ được chuyển sang trạng thái fall.

case 'fall':

        if (this.image !== this.sprites.fall.image) {

          this.image = this.sprites.fall.image

          this.framesMax = this.sprites.fall.framesMax

          this.framesCurrent = 0

        }

        break

  if (player.velocity.y < 0) {

    player.switchSprite('jump')

  } else if (player.velocity.y > 0) {

    player.switchSprite('fall')

  }

## 2.3.4. Sprite trạng thái attack (tấn công)

A picture containing night sky

Description automatically generated

Hình . Sprite attack

Tương tự như các sprite trước, gọi source hình ảnh vào *const player* và cài đặt switch case cho trạng thái.

Thiết lập chức năng mỗi lần nhấn nút tấn công, nhân vật sẽ chuyển sang attack sprite tại hàm *attack().*

  attack() {

    this.switchSprite('attack1')

    …

  }

Tại *switchSprite()*, vòng lặp if với điều kiện nếu hình ảnh được gọi là hình ảnh của sprite attack, và số *frameCurrent* (hiện tại) nhỏ hơn số *framesMax* (tối đa) của sprite sheet, thì ta cho return và không gọi đến bất kỳ một case sprite nào khác. Chỉ khi nào *frameCurrent* vượt quá *framesMax*, thì ta mới được phép tiếp tục gọi đến switch case của các sprite khác, và thậm chí là gọi đến case attack nếu nhấn phím tấn công thêm một lần nữa. Vòng lặp if này cũng còn được gọi là dùng để ghi đè lên tất cả các animation khác bằng attack animation.

  switchSprite(sprite) {

    if (

      this.image === this.sprites.attack1.image &&

      this.frameCurrent < this.sprites.attack1.framesMax - 1

    )

    return

Khi bị tấn công, nhân vật sẽ trừ HP, nhưng để nhân vật chỉ bị trừ HP khi bị sát thương trúng, thì ta cần set một vòng lặp if với điều kiện nếu player đang tấn công và *frameCurrent* đang bằng 4 (đang ở cuối sprite trạng thái tấn công) thì sát thương của player sẽ là giá trị false (không gây sát thương). Tương tự với khi enemy đánh trượt.

  // nếu player đánh trượt

  if (player.isAttacking && player.frameCurrent === 4) {

    player.isAttacking = false

  }

## 2.3.4. Sprite trạng thái receive hit (bị sát thương)

Tương tự như các sprite trước, gọi source hình ảnh vào *const player*, set framesMax và cài đặt switch case cho trạng thái, đặt tên là *takeHit*.

Tại *switchSprite()*, thực hiện vòng lặp if để ghi đè lên các animation khác bằng animation takeHit khi người chơi bị tấn công.

Sử dụng method *takeHit()*, để mỗi khi người chơi bị tấn công, sprite của takeHit sẽ được gọi đến tại sự kiện nhận diện va chạm, đồng thời, HP sẽ bị khấu trừ đi một lượng giá trị là 10.

  takeHit() {

    this.switchSprite ('takeHit')

    this.health -= 10

  }

## 2.3.4. Sprite trạng thái death (chết)

Tương tự như các sprite trước, gọi source hình ảnh vào *const player,* set framesMax và cài đặt switch case cho trạng thái, đặt tên là *death*.

Gọi thuộc tính *this.dead = false* ở *class Fighter*. Tại *switchSprite(),* lại thực hiện vòng lặp if để ghi đè lên các animation khác bằng sprite death.

    if (this.image === this.sprites.death.image) {

      if (this.framesCurrent === this.sprites.death.framesMax - 1)

        this.dead = true

      return

    }

Tại method *takeHit(),* thực hiện vòng lặp if với điều kiện, nếu health bé hơn hoặc bằng 0 thì sẽ chuyển từ các sprite khác sang sprite trạng thái death. Nếu trường hợp khác thì chuyển sang sprite *takeHit* như bình thường.

  takeHit() {

    …

    if (this.health <= 0) {

      this.switchSprite('death')

    } else this.switchSprite('takeHit')

  }

Trong *addEventListener* của keydown, thực hiện vòng lặp if bao gọn các switch case với điều kiện nếu nhân vật còn sống, chưa chết (*!player.dead*), thì các key chức năng sẽ được thực hiện bình thường.

window.addEventListener('keydown', (event) => {

  if (!player.dead) {

    …

    }

  }

  if (!enemy.dead) {

    …

    }

  }

})

## 3. Push game lên live server

Thực hiện push project game lên Github. Tạo một Repository mới và đặt tên cho nó. Tiến hành push tất cả các file thuộc project lên repository vừa tạo.

Sử dụng trang netlify.com để tạo live server. Đăng nhập vào trang web bằng tài khoản Github. Connect với tài khoản Github, và chọn project muốn push live. Sau đó nhấn nút Deploy là đã hoàn thành push live game lên web.

# **D. Demo sản phẩm**

Đường dẫn game: *https://thelastsamurai.netlify.app/index.html*

# **TÀI LIỆU THAM KHẢO**

[1] <https://youtu.be/vyqbNFMDRGQ>

[2] <https://luizmelo.itch.io/martial-hero-2>

[3] <https://luizmelo.itch.io/martial-hero>

[4] <https://www.w3schools.com/html/html_scripts.asp>

[5] <https://www.w3schools.com/html/>

[6] <https://www.w3schools.com/html/html_css.asp>

[7] <https://vietnix.vn/javascript-la-gi>

[8] <https://vietnix.vn/html-la-gi>

[9] <https://topdev.vn/blog/css-la-gi>

[10] <https://www.w3schools.com/jsref/met_element_addeventlistener.asp>

[11] <https://www.quackit.com/html/codes/html_background_music_codes.cfm>

[12] <https://www.w3schools.com/graphics/game_intro.asp>

[13] <https://viblo.asia/p/tao-1-game-don-gian-voi-canvas-trong-html5-n157G543MAje>

[14] <https://www.w3schools.com/graphics/game_canvas.asp>

[15] <https://www.freecodecamp.org/news/how-javascript-implements-oop>

[16] <https://www.freecodecamp.org/news/publish-your-website-netlify-github/>