МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

«УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМ

Реєстраційний №\_\_\_\_\_\_

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**КУРСОВА РОБОТА**

з дисципліни «**Розробка Інтернет клієнт-серверних систем**»

спеціальність 121 Інженерія програмного забезпечення

Тема: «**Розробка он-лайн гри «Вчимося програмувати»**»

Рекомендована до захисту

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_ 2023р.

Робота захищена

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_ 2023р.

з оцінкою

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Підписи членів комісії

Виконав

Студент 3-го курсу

Денної форми навчання

Групи ІПЗ-2

Нападій Олександр Євгенович

Науковий керівник :

к. ф.-м. н., доц. Шпак О. І.

Ужгород – 2023

ЗМІСТ

[ВСТУП 3](#_Toc139469393)

[РОЗДІЛ І. ОГЛЯД СТЕКУ РОЗРОБКИ ТА АНАЛОГІВ 4](#_Toc139469394)

[1.1. Стек розробки 4](#_Toc139469395)

[1.1.1. HTML 4](#_Toc139469396)

[1.1.2. CSS 5](#_Toc139469397)

[1.1.3. JavaScript 6](#_Toc139469398)

[1.2 . Огляд аналогів 7](#_Toc139469399)

[РОЗДІЛ II. РЕАЛІЗАЦІЯ ТА ДЕМОНСТРАЦІЯ ПРОЕКТУ 10](#_Toc139469400)

[2.1. Архітектура проекту 10](#_Toc139469401)

[2.2. Функціонал та реалізація 11](#_Toc139469402)

[ВИСНОВКИ 18](#_Toc139469403)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 19](#_Toc139469404)

[ДОДАТКИ 20](#_Toc139469405)

# ВСТУП

У сучасному інформаційному світі цифрові технології займають важливе місце в житті людей. Програмування стає однією з найпопулярніших галузей в сфері ІТ, яка вимагає від своїх представників не тільки технічного розуміння, а й творчого мислення та логічних навичок. У зв'язку з цим, розвиток відповідних компетенцій у молодого покоління є особливо актуальним завданням.

Одним із шляхів сприяти формуванню навичок програмування серед молоді є використання онлайн-ігор. Вони забезпечують цікаву і взаємодіючу форму навчання, та дозволяють розвивати творчість.

Тому метою курсової роботи стало створення цікавої гри яка б поєднувала в собі веселе та корисне.

Для виконання даної мети – створення он-лайн гри необхідно реалізувати такі етапи:

* Обрати вигляд гри;
* Створити головну сторінку;
* Побудувати ігровий світ та анімації;
* Продумати логіку взаємодій з об’єктами світу;
* Зробити керування головним персонажем;
* Виконати навчальну частину гри.

Об’єктом дослідження курсової роботи є веб-гра, а предметом – процес інтеграції навчання у ігрову активність.

Для реалізації поставленого завдання було використано такий стек технологій:

* Фронтенд – HTML5, CSS, JavaScript.

# РОЗДІЛ І. ОГЛЯД СТЕКУ РОЗРОБКИ ТА АНАЛОГІВ

## Стек розробки

Засоби фронтенд-розробки, такі як HTML, CSS і JavaScript, є основою для створення динамічних та привабливих веб-сторінок і додатків.

### HTML

HTML (HyperText Markup Language) є стандартною мовою розмітки, яка використовується для створення структури та вмісту веб-сторінок. Вона визначає, як браузер повинен відображати різні елементи на сторінці, включаючи текст, зображення, посилання, таблиці, форми та багато іншого.

Основними складовими HTML є теги, які визначають початок і кінець елемента, а також його тип.

* **Теги** (HTML використовує теги для визначення різних елементів на вебсторінці. Кожен тег має початковий тег і закриваючий тег , які обмежують вміст елемента.);
* **Елементи** (елементи HTML складаються з одного або більше тегів, які визначають їх структуру та вигляд.
* **Атрибути** (Атрибути використовуються для надання додаткової інформації про елементи HTML. Вони розміщуються всередині початкового тегу і мають значення, яке вказується в подвійних або одинарних лапках.);
* **Структура сторінки** (HTML дозволяє структурувати веб-сторінку за допомогою основних елементів, визначає кореневий елемент сторінки, містить метадані та посилання на зовнішні файли.).

HTML має велику актуальність як основний засіб фронтенд-розробки, незважаючи на появу нових технологій та фреймворків. Декілька причин чому HTML є актуальним:

* **Основна мова розмітки** (HTML є стандартом для створення структури веб-сторінок. Він забезпечує семантичну розмітку, що допомагає пошуковим системам та іншим програмам краще розуміти вміст.);
* **Інтеграція з технологіями** (HTML чудово працює з CSS та JavaScript, що дозволяє створювати гарні та функціональні сайти.).

### CSS

CSS (Cascading Style Sheets) - це мова стилів, яка використовується для оформлення та візуального оформлення веб-сторінок, які були створені за допомогою HTML. CSS визначає зовнішній вигляд елементів на сторінці, включаючи кольори, шрифти, розміри, відступи, позиціонування та анімацію.

Основні концепції CSS:

* **Селектори:** (CSS використовує селектори для вибору елементів, до яких будуть застосовані стилі. Наприклад, ви можете вибрати всі елементи <p> на сторінці, використовуючи селектор p, або елемент з ідентифікатором #element, використовуючи селектор #element.);
* **Властивості:** (Кожна CSS-властивість визначає певний аспект вигляду елемента. Наприклад, color визначає колір тексту, font-size встановлює розмір шрифту. Існує велика кількість властивостей CSS, які можна використовувати для керування виглядом елементів.);
* **Каскадування та спадковість:** (CSS дозволяє визначати стилі на різних рівнях, і вони можуть успадковуватися від батьківських елементів до дочірніх.);
* **Зовнішні файли CSS:** (CSS може бути вкладений безпосередньо в HTML-документ за допомогою тегу <style>, але також зазвичай використовується як зовнішній файл з розширенням .css, який підключається до HTML-документу за допомогою тегу <link>.);
* **Респонсивний дизайн:** (CSS також дозволяє створювати респонсивні веб-сторінки, які адаптуються до різних розмірів екранів.).

CSS є потужним інструментом для стилізації та оформлення веб-сторінок. Використовуючи CSS разом з HTML і JavaScript, можна створити привабливі веб сторінки насичені функціональністю.

### JavaScript

JavaScript (JS) - це високорівнева, МП, яка використовується для розробки динамічних веб-сторінок і додатків. Вона виконується безпосередньо в браузері користувача і дозволяє забезпечити інтерактивність, маніпулювання елементами сторінки.

Основні концепції та можливості JavaScript:

* **Змінні та типи даних:** (JavaScript має динамічну типізацію, що означає, що змінні можуть змінювати свій тип під час виконання програми.);
* **Функції:** (Функції в JavaScript дозволяють групувати код у логічні блоки, які можна викликати з інших частин програми. Вони можуть бути оголошеними, анонімними, приймати аргументи та повертати значення.);
* **Об'єкти:** (JavaScript є об'єктно-орієнтованою мовою програмування, що означає, що вона працює з об'єктами і класами. Об'єкти в JavaScript мають властивості, які можуть бути рядками або значеннями будь-якого типу, і методи, які є функціями, пов'язаними з об'єктом.);
* **Маніпулювання DOM:** (JavaScript дозволяє отримувати доступ до елементів HTML-сторінки та змінювати їх вміст, стилі, атрибути та структуру. Це дозволяє динамічно оновлювати вміст сторінки під час взаємодії з користувачем.);
* **Модулі та пакети:** (JavaScript підтримує модульну структуру, яка дозволяє розділити код на невеликі, самостійні модулі. Це полегшує організацію та підтримку великих проектів.);

JavaScript є однією з найпопулярніших мов програмування, що забезпечує широкий спектр можливостей для створення веб-додатків, ігор, мобільних додатків та багато іншого.

## . Огляд аналогів

На даний момент існує багато ігор які можуть допомогти вивчити програмування або ж просто покращити свої навички.

Ось деякі з них:

* **CodinGame** — це технологічна компанія, яка редагує онлайн-платформу для розробників, що дозволяє їм грати з програмуванням із дедалі складнішими головоломками, навчитися краще кодувати за допомогою програми онлайн-програмування, що підтримує двадцять п’ять мов програмування, і брати участь у змаганнях із програмування для кількох гравців із залученням штучного інтелекту;

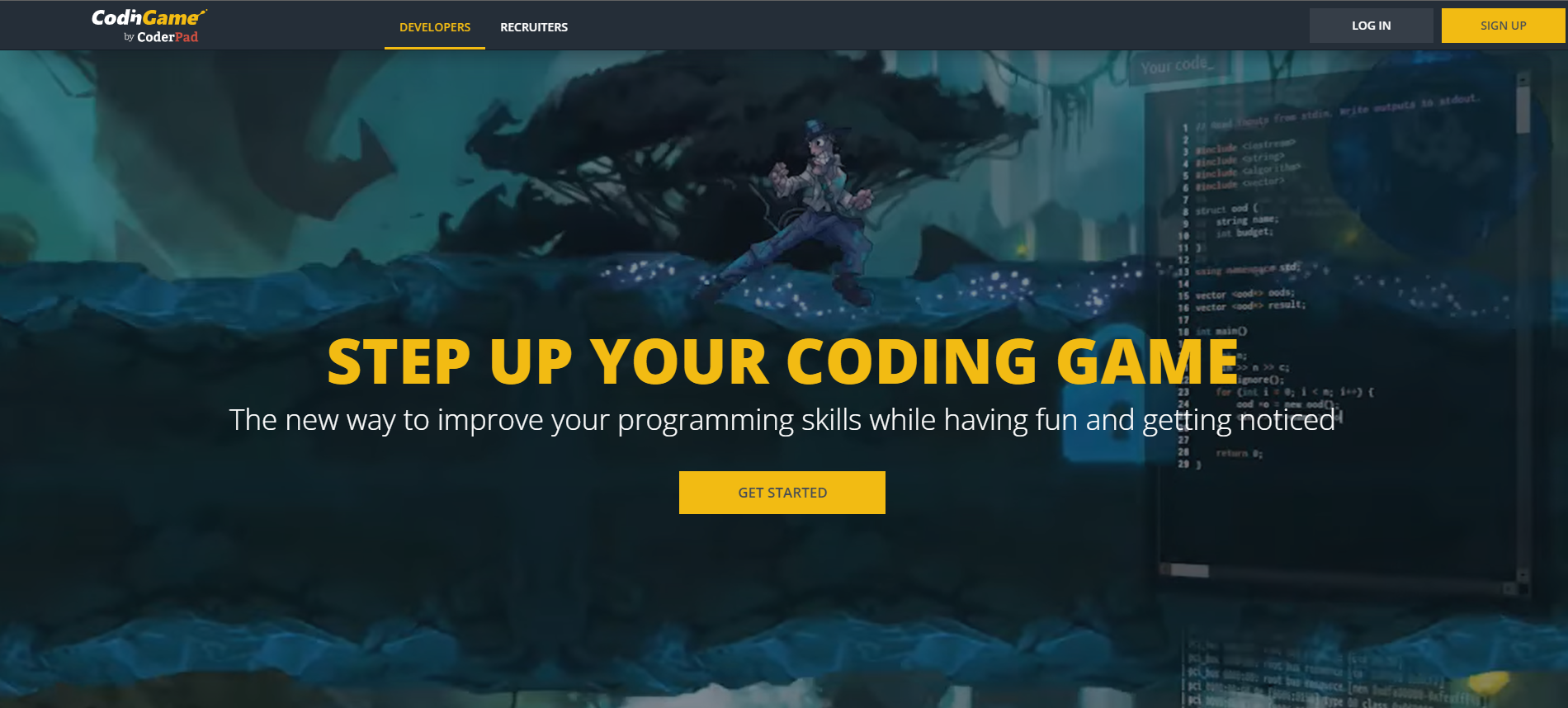


Рис. 1.1 – CodinGame

* **CodeCombat** — навчальна відеогра для вивчення концепцій і мов програмування. Ця гра рекомендована для учнів 9–16 років. Студенти вчаться друкувати такими мовами кодування, як JavaScript, Python, HTML і CoffeeScript, а також вивчають основи інформатики. CodeCombat має 11 підрозділів - три підрозділи розробки ігор, два підрозділи веб-розробки та шість підрозділів інформатики;



Рис. 1.2 – Code Combat

* **CheckIO** – ресурс для вивчення та практики мови програмування Python. Абсолютно будь-хто може зареєструватися на майданчику і почати навчання або, вже знаючи мову, відшліфувати свої навички. Навчання представлено у вигляді гри, в якій кожному користувачеві необхідно тією чи іншою мірою використовувати свої знання. Наприклад, перший етап навчання «Learning» - це ланцюжок завдань від легкого до складного. Причому в описі самої задачі є всі довідкові дані для її вирішення.



Рис. 1.3 – CheckIO

# РОЗДІЛ II. РЕАЛІЗАЦІЯ ТА ДЕМОНСТРАЦІЯ ПРОЕКТУ

## Архітектура проекту

Весь функціонал проекту був виконаний за допомогою HTML, CSS та JS. HTML – за його допомогою виконано структуру сторінки. CSS – оформив та стилізував її. JS – використовувався для прописування логіки гри та скриптів на сторінці.

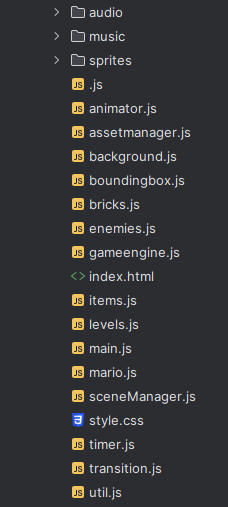


Рис. 2.1 – Структура проекту

audio/ - файл де містяться аудіозаписи, які будуть використовуватися при анімаціях.

music/ - файл в якому міститься музика яка буде грати при проходженні рівня.

sprites/ - папка з різними листами спрайтів, які потім використовуються в анімаціях та створенні моделей та рівнів.

.js – файл в якому містяться створені всі блоки, вороги, фон, ігровий персонаж та інше.

animator.js – виконує створення анімацій.

assetmanager.js – виконує дію завантаження спрайт-листів.

background.js – містить в собі класи всіх спрайтів, які знаходяться на фоні.

boundingbox.js – створений для того, щоб містити в собі хіт-бокси всіх спрайтів.

bricks.js – містить в собі класи всіх блоків.

enemies.js – тут розміщені класи ворожих персонажів.

gameengine.js – забезпечує зчитування клавіш та пропис ігрового циклу.

index.html – файл в якому містяться теги якими створена сторінка.

items.js – файл в якому знаходяться класи для монет та грибів.

level.js – файл де розміщені координати всіх спрайтів на рівні.

main.js – тут завантажені всі спрайт-листи, музика та запущено ігровий рушій.

sceneManager.js – файл за допомогою якого промальовується головне меню гри.

style.css – за допомогою нього стилізовано сторінку.

timer.js – створено ігровий таймер який використовується в анімаціях, ігровому тіку і так далі.

transition.js – за допомогою його функціоналу створені переходи при смерті та завантаженні рівня.

util.js – деякі базові параметри які не можна було приписати до інших файлів.

## Функціонал та реалізація

Наразі мій проект не є повністю запрограмований, тому частина функціоналу не працює, а також він не знаходиться на хостингу, тому необхідно його запускати локально.

Початкова сторінка. Тут ми відразу можемо побачити назву гри “Super Mario Proger”. Я обрав за основу своєї гри старий “Super Mario Bros”, звідки взяв музику, спрайт-листи та звуки. Відразу під назвою знаходиться сама гра. Відразу під грою можна побачити налаштування звуку.



Рис. 2.2 – вигляд сторінки при загрузці

Перші дві кнопки “MARIO” та “LUIGI” дають нам можливість вибрати одного з культових персонажів, за яких можна грати (рис. 2.3 – рис. 2.4). Третя ж клавіша відповідає за показ таблиці де мають бути прописані використані джерела (рис. 2.5).



Рис. 2.3 – початок за Маріо

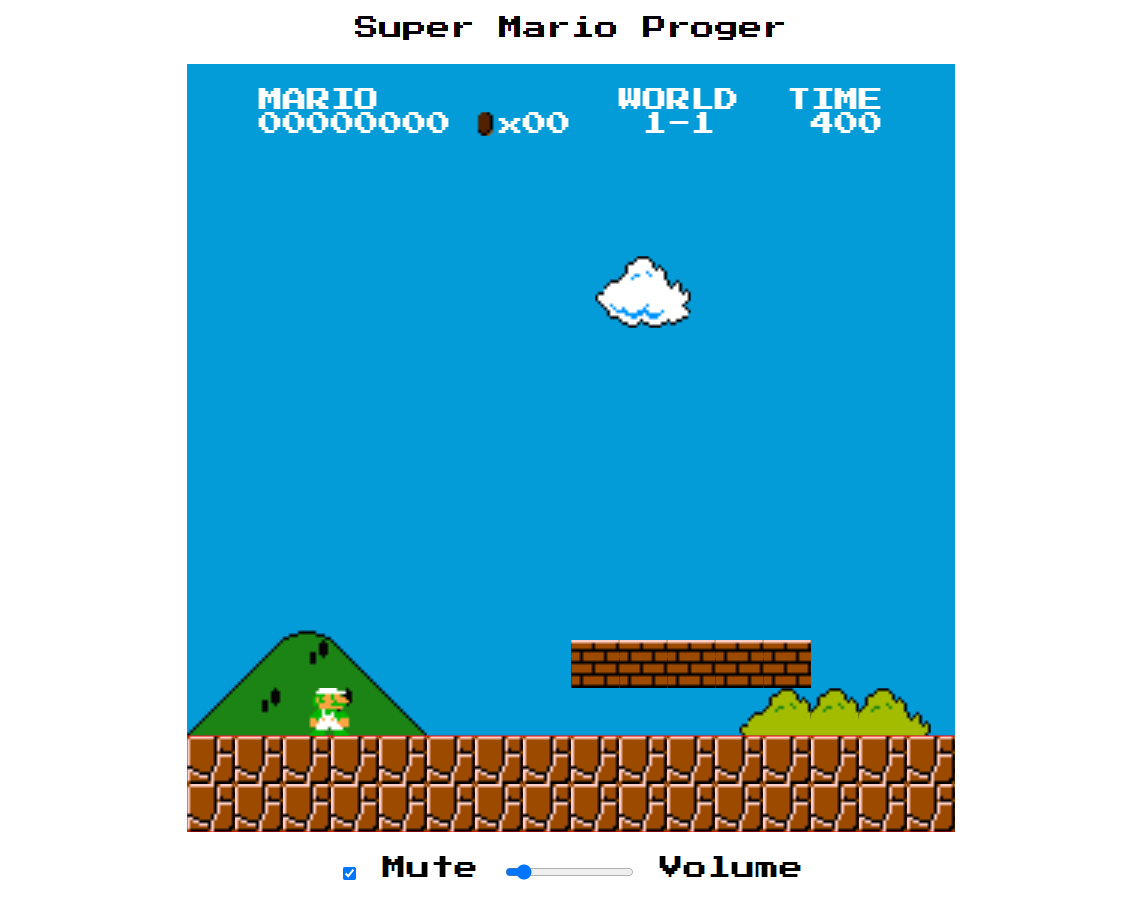


Рис. 2.4 – Початок за Луіджі



Рис. 2.5 – вигляд таблиці з джерелами.

Також при запуску гри відбувається анімація де показано кількість здоров’я у персонажа.



Рис. 2.6 – Анімація запуску гри

Після запуску ми маємо змогу керувати персонажем та взаємодіяти з середовищем. До прикладу:

* Вибивати монетки та бонуси з блоків.



Рис. 2.7 – Вибиття монеток з блоків

* Боротися з ворогами ( перемога при наступі на ворога та програш при простому доторку)



Рис. 2.8 – Перемога над ворогом

* Виконувати завдання від персонажів



Рис. 2.9 – Питання від персонажа

При русі помітно анімації у ігрового персонажа, ворогів та інших персонажів. Також є анімації взаємодії з блоками. При смерті показується кількість життів що залишилися та кількість набраних балів.



Рис. 2.10 – Екран при смерті

При проходженні гри відбувається анімація спуску по прапорцю та захід в замок як і в оригінальній грі.

При підбиранні магічного гриба персонаж стає більшим.

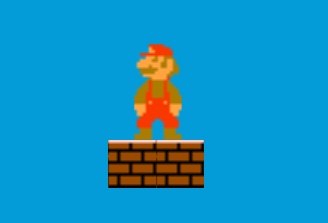


Рис. 2.11 – Зміна вигляду персонажа після підбора гриба

# ВИСНОВКИ

У цій курсовій роботі була успішно розроблена веб-гра, яка має потенціал вчити дітей програмуванню не набридаючи їм. Завдяки їй, учні зможуть посилити свої знання у програмуванні, та просто відпочити.

При створенні проекту, я використовував різні технології для досягнення мети. Я використав HTML5, CSS3 для верстки сторінки, адже вони є найбільш поширеними та найкращими інструментами наразі. Також для логіки гри та сайту я використав JavaScript – мову програмування для створення сайтів. В цілому цей стек технологій був достатнім для того аби досягти мети проекту, та зробити його достатньо якісним аби презентувати. Також я завантажив спрайт-листи та звуки з оригінальної гри “Super Mario Bros”, щоб надати моїй грі доречного вигляду.

Взагалом, даний додаток відповідає вимогам та меті курсової роботи. Але звичайно ж, його можна вдосконалити багатьма способами (додати нові рівні, нових ворогів та функціонал до ігрового персонажа, зробити більш цікавою частину, пов’язану з програмуванням.).

Повний програмний код додатку можна знайти на GitHub:

<https://github.com/napjo1n/SuperMarioProg>

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. [Електронний ресурс] – HTML – Вікіпедія - <https://uk.wikipedia.org/wiki/HTML>
2. [Електронний ресурс] – CSS – Вікіпедія - <https://uk.wikipedia.org/wiki/CSS>
3. [Електронний ресурс] – JavaScript – Вікіпедія - <https://uk.wikipedia.org/wiki/JavaScript>
4. [Електронний ресурс] – codingame – <https://www.codingame.com/start>
5. [Електронний ресурс] – CodeCombat - <https://codecombat.com/play>
6. [Електронний ресурс] – CodeCombat - <https://checkio.org/>
7. [Електронний ресурс] – Seth Ladd’s Blog - <http://blog.sethladd.com/2011/05/source-code-slides-and-video-for-html5.html>
8. Ерік Фрімен HEAD FIRST. Програмування на JavasScript/ Видавництво «Фабула», 2022 - 672с.
9. Васильєв А. Програмування на JavaScript у прикладах і задачах./ Україна, 2017. – 720 с.

# ДОДАТКИ

Завантаження ассетів **assetmanager.js**

class AssetManager {  
 constructor() {  
 this.successCount = 0;  
 this.errorCount = 0;  
 this.cache = [];  
 this.downloadQueue = [];  
 };  
  
 queueDownload(path) {  
 *console*.log("Queueing " + path);  
 this.downloadQueue.push(path);  
 };  
  
 isDone() {  
 return this.downloadQueue.length === this.successCount + this.errorCount;  
 };  
  
 downloadAll(callback) {  
 if (this.downloadQueue.length === 0) setTimeout(callback, 10);  
 for (let i = 0; i < this.downloadQueue.length; i++) {  
 const that = this;  
  
 const path = this.downloadQueue[i];  
 *console*.log(path);  
 const ext = path.substring(path.length - 3);  
  
 switch (ext) {  
 case 'jpg':  
 case 'png':  
 const img = new Image();  
 img.addEventListener("load", function () {  
 *console*.log("Loaded " + this.src);  
 that.successCount++;  
 if (that.isDone()) callback();  
 });  
  
 img.addEventListener("error", function () {  
 *console*.log("Error loading " + this.src);  
 that.errorCount++;  
 if (that.isDone()) callback();  
 });  
  
 img.src = path;  
 this.cache[path] = img;  
 break;  
 case 'wav':  
 case 'mp3':  
 case 'mp4':  
 const aud = new Audio();  
 aud.addEventListener("loadeddata", function () {  
 *console*.log("Loaded " + this.src);  
 that.successCount++;  
 if (that.isDone()) callback();  
 });  
  
 aud.addEventListener("error", function () {  
 *console*.log("Error loading " + this.src);  
 that.errorCount++;  
 if (that.isDone()) callback();  
 });  
  
 aud.addEventListener("ended", function () {  
 aud.pause();  
 aud.currentTime = 0;  
 });  
  
 aud.src = path;  
 aud.load();  
  
 this.cache[path] = aud;  
 break;  
 }  
 }  
 };  
  
 getAsset(path) {  
 return this.cache[path];  
 };  
  
 playAsset(path) {  
 let audio = this.cache[path];  
 audio.currentTime = 0;  
 audio.play();  
 };  
  
 muteAudio(mute) {  
 for (let key in this.cache) {  
 let asset = this.cache[key];  
 if (asset instanceof *Audio*) {  
 asset.muted = mute;  
 }  
 }  
 };  
  
 adjustVolume(volume) {  
 for (let key in this.cache) {  
 let asset = this.cache[key];  
 if (asset instanceof *Audio*) {  
 asset.volume = volume;  
 }  
 }  
 };  
  
 pauseBackgroundMusic() {  
 for (let key in this.cache) {  
 let asset = this.cache[key];  
 if (asset instanceof *Audio*) {  
 asset.pause();  
 asset.currentTime = 0;  
 }  
 }  
 };  
  
 autoRepeat(path) {  
 const aud = this.cache[path];  
 aud.addEventListener("ended", function () {  
 aud.play();  
 });  
 };  
}

Створення анімацій **animator.js**

class Animator {  
 constructor(spritesheet, xStart, yStart, width, height, frameCount, frameDuration, framePadding, reverse, loop) {  
 *Object*.assign(this, { spritesheet, xStart, yStart, height, width, frameCount, frameDuration, framePadding, reverse, loop });  
  
 this.elapsedTime = 0;  
 this.totalTime = this.frameCount \* this.frameDuration;  
  
 };  
  
 drawFrame(tick, ctx, x, y, scale) {  
 this.elapsedTime += tick;  
  
 if (this.isDone()) {  
 if (this.loop) {  
 this.elapsedTime -= this.totalTime;  
 } else {  
 return;  
 }  
 }  
  
 let frame = this.currentFrame();  
 if (this.reverse) frame = this.frameCount - frame - 1;  
   
 ctx.drawImage(this.spritesheet,  
 this.xStart + frame \* (this.width + this.framePadding), this.yStart, //source from sheet  
 this.width, this.height,  
 x, y,  
 this.width \* scale,  
 this.height \* scale);  
  
 };  
  
 currentFrame() {  
 return *Math*.floor(this.elapsedTime / this.frameDuration);  
 };  
  
 isDone() {  
 return (this.elapsedTime >= this.totalTime);  
 };  
}

Створення логіки ворогів **enemies.js**

class Goomba {  
 constructor(game, x, y) {  
 *Object*.assign(this, { game, x, y });  
 this.velocity = { x: -*PARAMS*.BITWIDTH, y: 0 };  
 this.spritesheet = *ASSET\_MANAGER*.getAsset("./sprites/enemies.png");  
 this.animation = new Animator(this.spritesheet, 0, 4, 16, 16, 2, 0.2, 14, false, true);  
 this.paused = true;  
 this.dead = false;  
 this.deadCounter = 0;  
 this.flickerFlag = true;  
 this.updateBB();  
 };  
  
 updateBB() {  
 this.lastBB = this.BB;  
 this.BB = new BoundingBox(this.x, this.y, *PARAMS*.BLOCKWIDTH, *PARAMS*.BLOCKWIDTH);  
 };  
  
 update() {  
 const FALL\_ACC = 1800;  
  
 if (this.dead) {  
 if (this.deadCounter === 0) this.game.addEntity(new Score(this.game, this.x, this.y, 100));  
 this.deadCounter += this.game.clockTick;  
 }  
 if (this.paused && this.game.camera.x > this.x - *PARAMS*.CANVAS\_WIDTH) {  
 this.paused = false;  
 }  
 if (!this.paused && !this.dead) {  
 this.velocity.y += FALL\_ACC \* this.game.clockTick;  
 this.x += this.game.clockTick \* this.velocity.x \* *PARAMS*.SCALE;  
 this.y += this.game.clockTick \* this.velocity.y \* *PARAMS*.SCALE;  
 this.updateBB();  
  
 let that = this;  
 this.game.entities.forEach(function (entity) {  
 if (entity.BB && that.BB.collide(entity.BB)) {  
 if (entity instanceof Mario) {  
 entity.die();  
 } else if ((entity instanceof Ground || entity instanceof Brick || entity instanceof Block)  
 && that.lastBB.bottom <= entity.BB.top) {  
 that.y = entity.BB.top - *PARAMS*.BLOCKWIDTH;  
 that.velocity.y = 0;  
 that.updateBB();  
 } else if (entity !== that) {  
 that.velocity.x = -that.velocity.x;  
 }  
 }  
 });  
 }  
 };  
  
 draw(ctx) {  
 if (this.dead) {  
 if (this.flickerFlag) {  
 ctx.drawImage(this.spritesheet,  
 0, 4, //source from sheet  
 16, 16,  
 this.x - this.game.camera.x, this.y + *PARAMS*.BLOCKWIDTH \* 3 / 4,  
 *PARAMS*.BLOCKWIDTH,  
 *PARAMS*.BLOCKWIDTH / 4);  
 }  
 this.flickerFlag = !this.flickerFlag;  
 } else {  
 this.animation.drawFrame(this.game.clockTick, ctx, this.x - this.game.camera.x, this.y, *PARAMS*.SCALE)  
  
 }  
 };  
}  
  
class Koopa {  
 constructor(game, x, y, facing, color) {  
 *Object*.assign(this, { game, x, y, facing, color});  
 this.spritesheet = *ASSET\_MANAGER*.getAsset("./sprites/enemies.png");  
 this.animations = [];  
 this.animations.push(new Animator(this.spritesheet, 210, 30, 16, 24, 2, 0.2, 14, false, true));  
 this.animations.push(new Animator(this.spritesheet, 150, 30, 16, 24, 2, 0.2, 14, false, true));  
 this.paused = true;  
 this.dead = false;  
 this.deadCounter = 0;  
 this.updateBB();  
 };  
  
 updateBB() {  
 this.BB = new BoundingBox(this.x, this.y, *PARAMS*.BLOCKWIDTH, (1 + 7/16) \* *PARAMS*.BLOCKWIDTH);  
 };  
  
 update() {  
  
 if (this.dead) {  
 if (this.deadCounter === 0) this.game.addEntity(new Score(this.game, this.x, this.y, 100));  
 this.deadCounter += this.game.clockTick;  
 this.removeFromWorld = true;  
 }  
 if (this.paused && this.game.camera.x > this.x - *PARAMS*.CANVAS\_WIDTH) {  
 this.paused = false;  
 }  
 if (!this.paused && !this.dead) {  
  
 }  
 };  
 draw(ctx) {  
 if (this.dead) {  
   
 } else {  
 this.animations[this.facing].drawFrame(this.game.clockTick, ctx, this.x - this.game.camera.x, this.y, *PARAMS*.SCALE)  
 let that = this;  
 this.game.entities.forEach(function (entity) {  
 if (entity.BB && that.BB.collide(entity.BB)) {  
 if (entity instanceof Mario) {  
 ctx.font = *PARAMS*.BLOCKWIDTH / 3 + 'px "Press Start 2P"';  
  
 ctx.fillText("Що таке цикл?", 6 \* *PARAMS*.BLOCKWIDTH, 5.5 \* *PARAMS*.BLOCKWIDTH);  
 ctx.fillText("повторення дій до виконання умови", 1 \* *PARAMS*.BLOCKWIDTH, 8.5 \* *PARAMS*.BLOCKWIDTH);  
 ctx.fillText("перевірка умови на правдивість", 1 \* *PARAMS*.BLOCKWIDTH, 9.5 \* *PARAMS*.BLOCKWIDTH);  
 ctx.fillText("не знаю", 1 \* *PARAMS*.BLOCKWIDTH, 10.5 \* *PARAMS*.BLOCKWIDTH);  
 }  
 }  
 });  
 }  
 };  
}

Створення ігрового рушія **gameengine.js**

class GameEngine {  
 constructor() {  
 this.entities = [];  
 this.ctx = null;  
  
  
 this.left = false;  
 this.right = false;  
 this.up = false;  
 this.down = false;  
 this.A = false;  
 this.B = false;  
  
 this.gamepad = null;  
 };  
  
 init(ctx) {  
 this.ctx = ctx;  
 this.startInput();  
 this.timer = new Timer();  
 };  
  
 start() {  
 const that = this;  
 (function gameLoop() {  
 that.loop();  
 requestAnimFrame(gameLoop, that.ctx.canvas);  
 })();  
 };  
  
 startInput() {  
 this.keyboardActive = false;  
 const that = this;  
  
 const getXandY = function (e) {  
 const x = e.clientX - that.ctx.canvas.getBoundingClientRect().left;  
 const y = e.clientY - that.ctx.canvas.getBoundingClientRect().top;  
  
 return {x: x, y: y, radius: 0};  
 };  
  
 function mouseListener (e) {  
 that.mouse = getXandY(e);  
 }  
 function mouseClickListener (e) {  
 that.click = getXandY(e);  
 if (*PARAMS*.DEBUG) *console*.log(that.click);  
 }  
 function wheelListener (e) {  
 e.preventDefault();  
 that.wheel = e.deltaY;  
 }  
 function keydownListener (e) {  
 that.keyboardActive = true;  
 switch (e.code) {  
 case "ArrowLeft":  
 case "KeyA":  
 that.left = true;  
 break;  
 case "ArrowRight":  
 case "KeyD":  
 that.right = true;  
 break;  
 case "ArrowUp":  
 case "KeyW":  
 that.up = true;  
 break;  
 case "ArrowDown":  
 case "KeyS":  
 that.down = true;  
 break;  
 case "KeyZ":  
 case "Comma":  
 that.B = true;  
 break;  
 case "KeyX":  
 case "Period":  
 that.A = true;  
 break;  
 }  
 }  
 function keyUpListener (e) {  
 that.keyboardActive = false;  
 switch (e.code) {  
 case "ArrowLeft":  
 case "KeyA":  
 that.left = false;  
 break;  
 case "ArrowRight":  
 case "KeyD":  
 that.right = false;  
 break;  
 case "ArrowUp":  
 case "KeyW":  
 that.up = false;  
 break;  
 case "ArrowDown":  
 case "KeyS":  
 that.down = false;  
 break;  
 case "KeyZ":  
 case "Comma":  
 that.B = false;  
 break;  
 case "KeyX":  
 case "Period":  
 that.A = false;  
 break;  
 }  
 }  
  
 that.mousemove = mouseListener;  
 that.leftclick = mouseClickListener;  
 that.wheelscroll = wheelListener;  
 that.keydown = keydownListener;  
 that.keyup = keyUpListener;  
  
 this.ctx.canvas.addEventListener("mousemove", that.mousemove, false);  
  
 this.ctx.canvas.addEventListener("click", that.leftclick, false);  
  
 this.ctx.canvas.addEventListener("wheel", that.wheelscroll, false);  
  
 this.ctx.canvas.addEventListener("keydown", that.keydown, false);  
  
 this.ctx.canvas.addEventListener("keyup", that.keyup, false);  
 };  
  
 disableInput() {  
 const that = this;  
 that.ctx.canvas.removeEventListener("mousemove", that.mousemove);  
 that.ctx.canvas.removeEventListener("click", that.leftclick);  
 that.ctx.canvas.removeEventListener("wheel", that.wheelscroll);  
 that.ctx.canvas.removeEventListener("keyup", that.keyup);  
 that.ctx.canvas.removeEventListener("keydown", that.keydown);  
  
 that.left = false;  
 that.right = false;  
 that.up = false;  
 that.down = false;  
 that.A = false;  
 that.B = false;  
 }  
  
 addEntity(entity) {  
 this.entities.push(entity);  
 };  
  
 draw() {  
 this.ctx.clearRect(0, 0, this.ctx.canvas.width, this.ctx.canvas.height);  
 for (let i = 0; i < this.entities.length; i++) {  
 this.entities[i].draw(this.ctx);  
 }  
 this.camera.draw(this.ctx);  
 };  
  
 gamepadUpdate() {  
 this.gamepad = *navigator*.getGamepads()[0];  
 let gamepad = this.gamepad;  
 if (gamepad != null && !this.keyboardActive) {  
 this.A = gamepad.buttons[0].pressed;  
 this.B = gamepad.buttons[1].pressed;  
 this.left = gamepad.buttons[14].pressed || gamepad.axes[0] < -0.3;  
 this.right = gamepad.buttons[15].pressed || gamepad.axes[0] > 0.3;  
 this.up = gamepad.buttons[12].pressed || gamepad.axes[1] < -0.3;  
 this.down = gamepad.buttons[13].pressed || gamepad.axes[1] > 0.3;  
 }  
 }  
  
 update() {  
 let i;  
 const entitiesCount = this.entities.length;  
  
 this.gamepadUpdate();  
   
 for (i = 0; i < entitiesCount; i++) {  
 const entity = this.entities[i];  
  
 if (!entity.removeFromWorld) {  
 entity.update();  
 }  
 }  
  
 this.camera.update();  
   
 for (i = this.entities.length - 1; i >= 0; --i) {  
 if (this.entities[i].removeFromWorld) {  
 this.entities.splice(i, 1);  
 }  
 }  
 this.wheel = 0;  
 };  
  
 loop() {  
 this.clockTick = this.timer.tick();  
 this.update();  
 this.draw();  
  
 this.click = null;  
 };  
}