МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОДЕСЬКОГО НАЦІОНАЛЬНОГО ТЕХНОЛОГІЧНОГО УНІВЕРСИТЕТУ»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Розробка програмного забезпечення»

Група: 4РП-05

Дипломний проект

здобувача освіти денної форми навчання
РП.05.10.000.ДП

КЛЕЩОВ
ІЛЛЯ
МАКСИМОВИЧ

м. Одеса
2022 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОДЕСЬКОГО НАЦІОНАЛЬНОГО ТЕХНОЛОГІЧНОГО УНІВЕРСИТЕТУ»

Спеціальність: **121 «Інженерія програмного забезпечення»**
Освітня програма: «**Розробка програмного забезпечення»**
Група: **4РП-05**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи) на тему:

Побудова та аналіз алгоритмів поведінки користувача та штучного інтелекту з використанням засобів JavaScript на базі веб-орієнтованої системи "Шахова гра"

Проектний матеріал складається з пояснювальної записки на 50 сторінках та графічного (презентаційного) матеріалу на 10 аркушах (слайдах).

Дипломник _____ (Клещов І.М.)
Керівник _____ (Медвєдєв А.О.)

Консультанти:
з економічної частини _____ (Копайгородська Т.Г.)
з охорони праці_____ (Чорновол Н.І.)
з дотримання вимог ЄСКД _____ (Петрашова В.І.)
старший консультант _____ (Скорнякова О.В.)

До захисту допущений
Голова циклової комісії _____ (Скорнякова О.В.)
Завідувач відділення _____ (Суліма Ю.Ю.)

Захист «____»_____ 2022 р. Протокол ДКК №_____
Оцінка ДКК _____

Секретар ДКК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОДЕСЬКОГО НАЦІОНАЛЬНОГО ТЕХНОЛОГІЧНОГО УНІВЕРСИТЕТУ»

Відділення **комп'ютерних систем** Комісія **КТ та ІІІ**

Спеціальність **121 «Інженерія програмного забезпечення»**

Освітня програма **«Розробка програмного забезпечення»**

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР_____

_____

"_____"_____2022 р.

З А В Д А Н Н Я

на дипломний проект (роботу)

Здобувачеві (здобувачці) освіти Клещев Ілля Максимович

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Побудова та аналіз флгоритмів поведінки користувача та штучного інтелекту з використанням засобів JavaScript на базі веб-орієнтованої системи "Шахова гра"

затверджена наказом по коледжу від "30" грудня 202 1 р. № 306-А2-ОД

2. Термін здачі закінченого проекту (роботи) _____

3. Вихідні данні до проекту (роботи) JavaScript, HTML, CSS

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

1. Побудова та аналіз алгоритмів. 2. Економічний розрахунок. 3. Охорона праці.

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Презентація (10 слайдів)

6. Консультанти по проекту (роботі), із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний	Медвєдєв А.О.		
Економічний	Копайгородська Т.Г.		
Охорона праці	Чорновол Н.І.		
Нормоконтроль	Петрашова В.І.		
Старший консультант	Скорнякова О.В.		

7. Дата видачі завдання _____

Керівник _____

(підпис)

Завдання прийняв до виконання _____

(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту (роботи)	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	Розділ 1. Побудова та аналіз алгоритмів	Вик.	
2	Розділ 2. Економічний розрахунок	Вик.	
3	Розділ 3. Охорона праці	Вик.	
4	Розробка презентації до дипломної роботи	Вик.	
5	Чистове оформлення пояснювальної записки	Вик.	
6	Підготовка доповіді до захисту	Вик.	
7	Отримання рецензії, відповіді на зауваження	Вик.	
	рецензента	Вик.	
8	Захист роботи	Вик.	

Дипломник _____

(підпис)

Керівник _____

(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			РП 05.15.000 ДП ПЗ	*Дипломний проект*		
А4			*РП 05.15.000 ДП ПЗ*	*Пояснювальна*	*1*	
				записка		
				Презентація		

Ізм.	Арк.	№ докум.	Підпис	Дата	*РП 05.10.000*		
Розробив		Клещов І.М.			Побудова та аналіз флгоритмів поведінки користувача та штучного інтелекту з використанням засобів JavaScript на базі веб-орієнтованої системи “Шахова гра”	Літ.	Аркуш 3 / Аркушів
Перевірив		А.О.					
Н. Контр.		Петрашова В.І				ВСП «ОТФК ОНТУ» гр.РП-05	
Затверд.		Скорнякова О.В.					

АНОТАЦІЯ

Об'єкт дослідження: об'єктом дослідження є процес створення програми гри у шахи.

Предмет дослідження: предметом дослідження є мова програмування, яка дає можливість розробляти ігри на веб сторінці.

Мета роботи: Створення гри на веб сайті за допомогою мови програмування.

Досягнуті результати:

- Вибрані необхідні засоби розробки;
- Визначено алгоритми, які використовуються для обробки зображень.

Ключові слова: шахи, фігури, перевірки, зручність, складність, спрощення.

Обсяг: містить 70 сторінок тексту, рисунків - 25, таблиці - 1, джерел було використано – 13.

ЗМІСТ

ВСТУП

У першу чергу, програмування дозволяє здійснити мрії багатьох людей: знайти друзів, саморозвинутися в будь-якій галузі, почати заробляти та й просто провести вільний час цікаво. Форуми, соціальні мережі, ігри, комп’ютери – це підтримується виключно за допомогою програмістів. В наш час комп'ютери стали настільки поширеними, що використовуються практично скрізь, у тому числі й у сучасній економіці, починаючи від стадії виробництва товару та закінчуючи його продажем. Без використання комп'ютерів та спеціалізованих програм, прямо зараз, досить складно уявити, як усе це могло функціонувати.

Сьогодні налічується кілька сотень мов програмування, але найпоширеніші їх – Java, Python, PHP, C#, JavaScript, C, C++, Objective-C, Swift. Яку мову програмування вибрати, програміст вирішує сам залежно від конкретного завдання та власних знань. Кваліфікований програміст вільно використовує 2-4 мови. За спеціалізацією програмісти діляться на системних, прикладних та веб-програмістів. Прикладні програмісти займаються написанням різних програм і додатків: ігри, офісні програми, наприклад Microsoft Office Word та багато іншого. Системні програмісти займаються створенням операційних систем, а також програм для них, таких як Android або iOS. Веб-програмісти розробляють інтернет-сайти та програми, без яких сайт не зможе нормально функціонувати та вчасно оновлювати або додавати дані. IT-сфера відрізняється молодістю та сучасним підходом до освіти.

При працевлаштуванні цінується не так диплом, як ті навички, які у майбутнього співробітника в голові. Роботодавець поставить багато запитань, дасть кілька тестових завдань із програмування і за його результатами прийме або не прийме шукача на роботу. Саме тому набувати знань та досвіду в інформаційних технологіях можна і вдома за допомогою книг, online-курсів та практикуючись при створенні невеликих програм.

МЕТА ТА ЗАВДАННЯ ПРОЕКТУ

Метою проекта є розробка побудови та аналіз алгоритмів поведінки користувача та штучного інтелекту з використанням засобів JavaScript на базі веб-орієнтованої системи “Шахова гра”

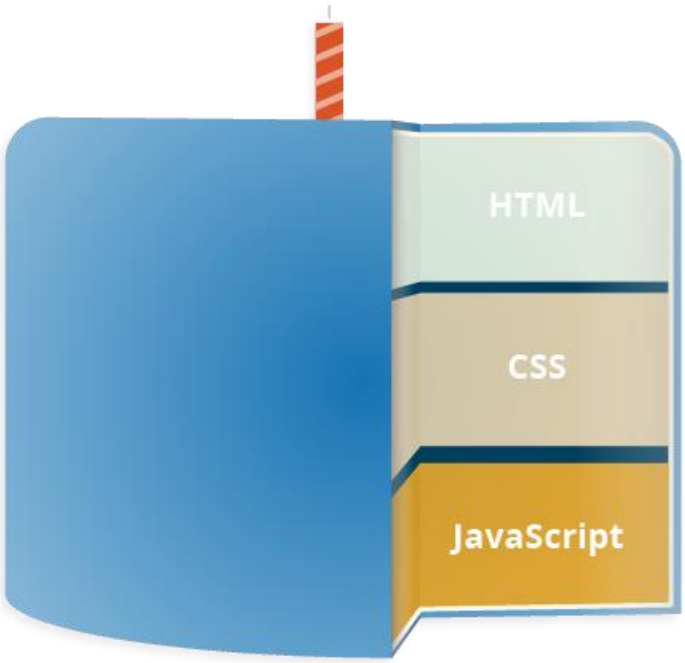
Для досягнення мети необхідно виконати такі завдання:

1. Прости аналіз існуючил алгоритмів поведінки користувача та вибрати основний алгоритм дій.
2. Створення штучного інтелекту за допомогою JavaScript.
3. Розробка веб сторінок за допомогою HTML та CSS.
4. Об’єднання всіх програм для створення одного великого та загального проекту.

1 ПРОЕКТУВАННЯ СИСТЕМИМИ ШАХОВОЇ ГРИ ТА ПОБУДОВА АЛГОРИТМІВ АНАЛІЗУ ПОВЕДІНКИ

1.1 Аналіз існуючих рішень та огляд технологій

JavaScript (часто скорочують до JS) – це повноцінна мова програмування, яка застосовується до HTML та CSS документа, і може забезпечити велику інтерактивність на веб-сайтах. Програми цією мовою називаються скриптами. Вони можуть вбудовуватися в HTML і виконуватися автоматично під час завантаження веб-сторінки. Його розробив Brendan Eich, співзасновник проекту Mozilla, Mozilla Foundation та Mozilla Corporation. Ця мова легка, інтерпретована, об'єктно-орієнтована мова з функціями першого класу, найвідоміша скриптова мова для веб-сторінок, але також використовується в багатьох не браузерних оточеннях. Прототипно-орієнтована, мультипарадигменна мова сценаріїв, яка підтримує динамічний, об'єктно-орієнтований, імперативний та функціональний стилі програмування. JavaScript запускається на стороні клієнта Інтернету, який може використовуватися для створення/програмування того, як веб-сторінки будуть поводитися при настанні будь-яких подій.



Риунок 1.1 – Структура побудови веб-технологій

HTML - це мова розмітки, яку ми використовуємо для візуального та смислового структурування нашого web контенту, наприклад, визначаємо параграфи, заголовки, таблиці даних, або вставляємо зображення та відео на сторінку. CSS - це мова стилів, за допомогою якої ми надаємо стиль відображення нашого HTML контенту, наприклад, надаємо колір фону (background) і шрифту, надаємо контенту багатоколоночний вигляд.

У двох словах, JavaScript - це динамічна скриптова мова, що підтримує прототипне створення об'єктів. Базовий синтаксис навмисно схожий на Java та C++, щоб зменшити кількість нових концепцій, необхідних вивчення мови. Такі мовні конструкції, як if, for, while, switch, try...catch схожі на конструкції цих мов. JavaScript може функціонувати і як процедурна, і як об'єктно-орієнтована мова. Об'єкти можна створювати програмно під час виконання, шляхом приєднання методів та властивостей або порожніх об'єктів під час виконання, на відміну від синтаксичних визначень класів у мовах, що компілюються, таких як C++ або Java. Після того, як об'єкт був створений, він може бути використаний як план (або прототип) для створення схожих об'єктів.

Динамічні можливості JavaScript включають: створення об'єктів під час виконання, змінну кількість параметрів, динамічне створення скриптів (за допомогою eval), перебір об'єктів (за допомогою for ... in), відновлення вихідного коду (програми на JavaScript можуть декомпілювати тіла функцій назад у вихідний код).

Якщо порівнювати з іншими мовами, наприклад Python, то складається враження, що для JS знадобиться більше часу і сил, на вивчення. У свою чергу, JS більш досконала мова, в ній набагато більше функцій та можливостей.

Однією з унікальних переваг JavaScript є його поширеність. Цю мову можна зустріти буквально всюди.

Він підтримується на всіх операційних системах, у всіх видах браузерів, і настільних комп'ютерах, і мобільних пристроях. Дуже важливо ще й те, що

JavaScript - програми працюють без встановлення їх на комп'ютери користувачів. Насправді, вже складно згадати не такі вже й давні часи, коли компанії, розгортаючи клієнт-серверні програми у своїх внутрішніх мережах, витрачали тижні, борючись із проблемами налаштування цих додатків та з несподіваними помилками. Зіткнувшись з подібними кошмарами, треба оцінити привабливість JavaScript. JavaScript, завдяки його поширеності, не обов'язково бути найкращою мовою програмування. Йому лише треба бути досить гарною мовою.

Схожі механізми працюють і тоді, коли йдеться про вивчення програмування. Спочатку проблема поширеності мови не здається особливо важливою для того, чия мета полягає в тому, щоб вивчити перевірені часом практичні прийоми розробки програм. Але безліч тих самих проблем актуальні, як і для професійного програмування, у навчанні програмування. Якщо ви хочете, щоб тим, що ви створили, міг би скористатися будь-хто, то JavaScript - це єдина альтернатива.

1.2 Проблема роботи з типами даних JavaScript

Найважливішою концепцією, яку програмісти-початківці освоюють дуже рано, є ідея змінних, контейнерів, які зберігають інформацію під час роботи програми. Проблема JavaScript полягає в тому, що ця мова надто вільно і неакуратно поводиться зі змінними. Ця мова дозволяє робити те, що не виглядає правильним і ігнорує очевидні нестиковки.

Для JavaScript - розробки потрібні додаткові бібліотеки та фреймворки

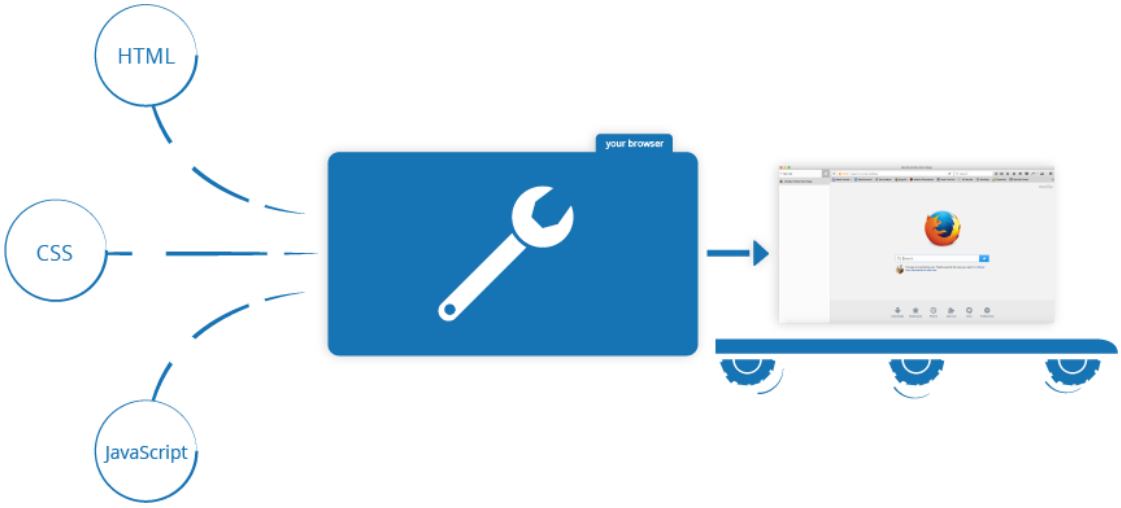
Для того, щоб отримати доступ до більш широкого, ніж є в мові набору функціональних можливостей (і не винаходити велосипеди), JavaScript-програмістам потрібно використовувати бібліотеки та фреймворки сторонніх розробників.

Вибір правильних інгредієнтів, які використовуються під час створення якогось проекту — це не так просто, як може здатися на перший погляд.

Йдеться, зокрема, про те, що обрані додаткові інструменти повинні правильно вирішувати поставлені перед ними завдання, потрібно, щоб у програміста була б впевненість у тому, що вони ще довго користуватимуться підтримкою своїх розробників, потрібно, щоб вони не конфліктували. другом.

Що робить JavaScript на вашій сторінці?

Коли ви завантажуєте сторінку в браузері, ви запускаєте ваш код (HTML, CSS і JavaScript) всередині середовища (всередині вкладки браузера). Це ніби фабрика бере сировину (якийсь код) і видає продукцію (веб-сторінку).



Риунок 1.2 – Структура, що вказує завантаження на веб сторінку

Код JavaScript виконується JavaScript-движком браузера, після того, як код HTML і CSS був оброблений і сформований на веб-сторінку. Це гарантує, що структура та стиль сторінки вже сформовані на момент запуску JavaScript.

Безпека браузера

Кожна вкладка браузера є окремою коробкою для запуску коду (в технічній мові, ці коробки називаються "середовищами виконання") - це означає, що в більшості випадків код на кожній вкладці запускається повністю окремо, а код однієї вкладки не може безпосередньо впливати на код іншої вкладки або на іншому веб-сайті. Це гарний захід безпеки — якби це було

інакше, пірати могли б написати код, який крав інформацію з інших сайтів або робив інші погані речі.

Послідовність виконання

Зазвичай, коли браузер стикається з блоком JavaScript, він запускає його по порядку зверху вниз. Це означає, що потрібно обережно вибирати порядок. Якщо написати блок коду, який не відповідає певному порядку, видасть помилку або нічого не станеться.

Мова, що інтерпретується

У контексті програмування, ви можете почути терміни інтерпретації та компіляції. JavaScript є мовою, що інтерпретується - код запускається зверху вниз і результат запуску негайно повертається. Не потрібно перетворювати код на іншу форму, перед запуском у браузері.

З іншого боку, компілювані мови перетворюються (компілюються) в іншу форму, перш ніж вони будуть запущені комп'ютером. Наприклад, C/C++ компілюються в мову асемблера, який потім запускається комп'ютером.

Область застосування Веб-програми

Веб-додаток JavaScript використовується в клієнтській частині веб-застосунків: клієнт-серверних програм, в якому клієнтом є браузер, а сервером – веб-сервер, що має розподілену між сервером та клієнтом логіку.

Браузерні операційні системи

JavaScript широко використовується у браузерних операційних системах.

Букмарклети

JavaScript використовується для створення невеликих програм, які розміщуються в закладках браузера. При цьому використовуються URL-адреси зі специфікатором JavaScript.

Скрипти користувача у браузері

Скрипти користувача у браузері – це програми, написані на JavaScript, що виконуються у браузері користувача під час завантаження сторінки. Вони дозволяють автоматично заповнювати форми, переформатувати сторінки, приховувати небажаний вміст та вбудовувати бажаний для відображення вміст, змінювати поведінку клієнтської частини веб-застосунків, додавати елементи керування на сторінку тощо.

Для управлінням скриптами користувача Mozilla Firefox використовується розширення Greasemonkey; Opera та Google Chrome надають засоби підтримки користувацьких скриптів та можливості для виконання ряду скриптів Greasemonkey.

Серверні програми

Програми, написані на JavaScript, можуть виконуватись на серверах, що використовують Java 6 і пізніших версій. Ця обставина використовується для побудови серверних програм, що дозволяють обробляти JavaScript на стороні сервера.

Мобільні додатки

Переклад мобільних пристроїв Palm на використання Palm webOS як операційна система з Mojo SDK як комплект засобів розробки дозволяє використовувати JavaScript як мову розробки мобільних додатків.

Прикладне програмне забезпечення

Вихідний код та скріншот JavaScript-програми, що виконується за допомогою Seed

JavaScript використовується для написання програми. Наприклад, 16,4% вихідного коду Mozilla Firefox написано JavaScript.

Google Chrome OS використовує веб-програми як прикладне програмне забезпечення.

Офісні програми

JavaScript використовується в офісних програмах для автоматизації рутинних дій, написання макросів, організації доступу з боку веб-служб.

Бібліотеки

Для забезпечення високого рівня абстракції та досягнення прийнятної міри крос-браузерності при створенні веб-додатків використовуються бібліотеки JavaScript. Вони є набір багаторазово використовуваних об'єктів та функцій. Серед відомих JavaScript бібліотек можна відзначити React.js, Vue.js, Ember.js, Adobe Spry, AngularJS, Svelte[en], Dojo, Extjs, jQuery, Mootools, Prototype, Qooxdoo та Underscore.

Засоби тестування

Більшість фреймворків автоматизованого тестування (англ.) JavaScript-коду передбачають запуск тестів у браузері. Це здійснюється за допомогою HTML-сторінки, яка є контекстом тестування (англ.), яка, у свою чергу, завантажує все необхідне для здійснення тестування. Першими такими фреймворками були JsUnit (англ.) (створений 2001 року), Selenium (створений 2004 року). Альтернатива – запуск тестів із командного рядка. У цьому випадку використовуються оточення, відмінні від браузера, наприклад, Rhino. Одним з перших інструментів такого роду є Crosscheck, що дозволяє тестувати

код, емулюючи поведінку Internet Explorer 6 та Firefox версій 1.0 та 1.5. Інший приклад фреймворку автоматизованого тестування JavaScript-коду, який не використовує браузер для запуску тестів - бібліотека env.js, створена Джоном Резігом. Вона використовує Rhino і при цьому містить емуляцію оточення браузера та DOM.

Blue Ridge, плагін до фреймворку веб-застосунків Ruby on Rails, дозволяє здійснювати модульне тестування JavaScript-коду як у браузері, так і поза ним. Це досягається за рахунок використання фреймворку автоматизованого тестування Screw.Unit та Rhino з env.js.

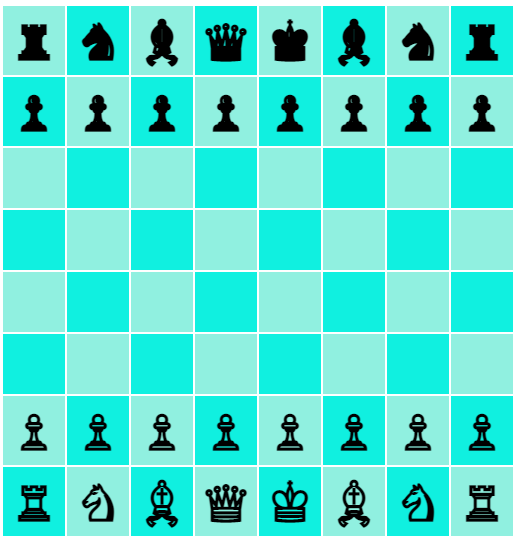
Головна проблема систем тестування, що не використовують браузери, у тому, що вони використовують емуляції, а не реальні оточення, в яких виконується код. Це призводить до того, що успішне проходження тестів не гарантує, що код коректно відпрацює у браузері. Проблемою систем тестування, що використовують браузер, є складність роботи з ними, необхідність здійснення неавтоматизованих рутинних дій.

Для вирішення цього JsTestDriver, фреймворк автоматизованого тестування, що розробляється Google, використовує сервер, що взаємодіє з браузерами для здійснення тестування. Подібним чином веде себе Selenium Remote Control, що входить у фреймворк автоматизованого тестування Selenium: він включає сервер, що запускає і завершує браузери і діє як HTTP-проксі для запитів до них. Крім того, Selenium містить Selenium Grid, що дозволяє здійснювати одночасне тестування JavaScript-коду на різних комп'ютерах з різними оточеннями, зменшуючи час виконання тестів. Testswarm, що має підтримку фреймворків автоматизованого тестування JavaScript-коду QUnit (бібліотека jQuery), UnitTestJS (бібліотека Prototype), JSSpec (бібліотека MooTools), JsUnit, Selenium та Dojo Objective Harness, є розподіленим засобом підтримки.

Негативна властивість, якою може мати фреймворк автоматизованого тестування JavaScript-коду - наявність залежностей. Це створює ризик відмови у роботі тестованого коду, успішно проходить тести, серед із відсутністю цих залежностей. Наприклад, вихідна версія JsUnitTest, фреймворку, створеного та використовуваного для тестування бібліотеки Prototype, залежала від самої Prototype, що змінює властивості об'єктів у глобальній області видимості. Включення до бібліотеки JavaScript інструменту тестування – поширена практика. Так YUI Test 3 є частиною Yahoo! UI Library та може бути безпечно використаний для тестування довільного JavaScript-коду. QUnit – фреймворк автоматизованого тестування, створений розробниками JQuery.

1.3 Створення шахів за допомогою мов програмування

JavaScript в роботі використовує майже кожен веб-розробник. Кожен з розробників, колись створював або починав створювати шахи, або калькулятор, адже це неймовірно цікава робота, яка закріпить всі твої навички або навчить чогось нового. Сам проект створення шахів або калькулятора неймовірно об'ємна, тому вона забирає багато часу на роботу, а також багато важить, від чого може підвисати сайт на секунду або більше, що покаже некоректну роботу сайту. Поки тільки мексиканцю Оскару Толедо вдалося запхати в один кілобайт коду на JavaScript, справжнісінький шаховий алгоритм.



Риунок 1.2 – Шахова гра Оскара Толедо

Тобто зараз ми бачимо додаток, код якого важить 1 кілобайт, це неймовірно. Після створення Javascript Chess Game, Толедо сказав: “Приблизно в січні 2007 року я згадав подібність між C та Javascript, і мені було цікаво, чи можна зробити переклад моєї нещодавньої перемогли роботи, я зробив це вдень, і результат найменша шахівниця у світі на Javascript, подивіться також мої Tiny Chess для JS1K.” Під час гри, бот діє швидко, мінімалізм, який приємний для ока, як приємна і сама шахівниця. Цю роботу допрацьовували дуже багато людей, вони додавали або 3-D дошку, або поліпшували CSS. “Більшість коду схожа на вихідний код, за винятком вказівників, переведених на доступ до масиву.”

```
<script>//(c)2009 Oscar Toledo G.
var B,i,y,u,b,I=[],G=120,x=10,z=15,M=1e4,l=[5,3,4,6,2,4,3,5,1,1,1,1,1,1,1,1,9,9
,9,9,9,9,9,9,13,11,12,14,10,12,11,13,0,99,0,306,297,495,846,-1,0,1,2,2,1,0,-1,-
1,1,-10,10,-11,-9,9,11,10,20,-9,-11,-10,-20,-21,-19,-12,-8,8,12,19,21];function
X(w,c,h,e,S,s){var t,o,L,E,d,O=e,N=-M*M,K=78-h<<x,p,g,n,m,A,q,r,C,J,a=y?-x:x;
y^=8;G++;d=w||s&&s>=h&&X(0,0,0,21,0,0)>M;do{if(o=I[p=O]){q=o&z^y;if(q<7){A=q--&
2?8:4;C=o-9&z?[53,47,61,51,47,47][q]:57;do{r=I[p+=l[C]];if(!w|p==w){g=q|p+a-S?0
:S;if(!r&(!!q|A<3||!!g)||(r+1&z^y)>9&&q|A>2){if(m=!(r-2&7))return y^=8,I[G--]=
O,K;J=n=o&z;E=I[p-a]&z;t=q|E-7?n:(n+=2,6^y);while(n<=t){L=r?l[r&7|32]-h-q:0;if(
s)L+=(1-q?l[(p-p%x)/x+37]-l[(O-O%x)/x+37]+l[p%x+38]*(q?1:2)-l[O%x+38]+(o&16)/2:
!!m*9)+(!q?!(I[p-1]^n)+!(I[p+1]^n)+l[n&7|32]-99+!!g*99+(A<2):0)+!(E^y^9);if(s>h
||1<s&s==h&&L>z|d){I[p]=n,I[O]=m?(I[g]=I[m],I[m]=0):g?I[g]=0:0;L-=X(s>h|d?0:p,L
-N,h+1,I[G+1],J=q|A>1?0:p,s);if(!(h||s-1|B-O|i-n|p-b|L<-M))return W(),G--,u=J;
J=q-1|A<7||m||!s|d|r|o<z||X(0,0,0,21,0,0)>M;I[O]=o;I[p]=r;m?(I[m]=I[g],I[g]=0):
g?I[g]=9^y:0;}if(L>N||s>1&&L==N&&!h&&Math.random()<.5){I[G]=0;if(s>1){if(h&&c-L
<O)return y^=8,G--,L;if(!h)i=n,B=O,b=p;}N=L;}n+=J||(g=p,m=p<O?g-3:g+2,I[m]<z|I[
m+O-p]||I[p+=p-O])?1:0;}}}}while(!r&q>2||(p=O,q|A>2|o>z&!r&&++C*--A));}}}while(
++O>98?O=20:e-O);return y^=8,G--,N+M*M&&N>-K+1924|d?N:0;}B=i=y=u=0;while(B++<
120)I[B-1]=B%x?B/x%x<2|B%x<2?7:B/x&4?0:l[i++]|16:7;for(a=
"<table cellspacing=0 align=center>",i=18;i<100;a+=++i%10-9?
"<th width=40 height=40 onclick=Y("+i+") style='border:2px solid #aae' id=o"+i+
" bgcolor=#"+(i*.9&1?"9090d0>":"c0c0ff>"):(i++,"<tr>"));
a+="<th colspan=8><select id=t><option>Q<option>R<option>B";
document.write(a+"<option>N</select></table>");
function W(){B=b;for(p=21;p<99;++p)if(q=document.getElementById("o"+p)){q.
innerHTML="<img width=40 src="+(I[p]&z)+".gif>";q.
style.borderColor=p==B?"#ff0":"#aae";}}W();
function Y(s){i=(I[s]^y)&z;if(i>8){b=s;W();}else if(B&&i<9){b=s;i=I[B]&z;if((i&
7)==1&(b<29|b>90))i=14-document.getElementById("t").selectedIndex^y;X(0,0,0,21,
u,1);if(y)setTimeout("X(0,0,0,21,u,2/*ply*/),X(0,0,0,21,u,1)",250);}}
</script>
```

Рисунок 1.3 – Шаховий код Оскара Толедо

Конкурс JS1K розпочався у неділю, 1 серпня 2010 року. Він був організований Пітером ван дер Зі з Томасом Фуксом, Ремі Шарпом, Патріком Х. Лауком та Крістофером Вільямсом як судді, і всі вони були досвідченими програмістами Javascript.

1.4 Коротко про створення шахів

На початку, ми повинні створити візуалізацію шахівниці. Тобто квадрати рівної вилікувані, що має певний порядок і колір своєї клітини, без жодних аргументів.

```css
/* clearfix */
.clearfix-7da63 {
  clear: both;
}

/* board */
.board-b72b1 {
  border: 2px solid #404040;
  -moz-box-sizing: content-box;
  box-sizing: content-box;
}

/* square */
.square-55d63 {
  float: left;
  position: relative;

  /* disable any native browser highlighting */
  -webkit-touch-callout: none;
    -webkit-user-select: none;
     -khtml-user-select: none;
       -moz-user-select: none;
        -ms-user-select: none;
            user-select: none;
}

/* white square */
.white-1e1d7 {
  background-color: #f0d9b5;
  color: #b58863;
}

/* black square */
.black-3c85d {
  background-color: #b58863;
  color: #f0d9b5;
}

/* highlighted square */
.highlight1-32417, .highlight2-9c5d2 {
  -webkit-box-shadow: inset 0 0 3px 3px yellow;
  -moz-box-shadow: inset 0 0 3px 3px yellow;
  box-shadow: inset 0 0 3px 3px yellow;
}



/* notation */
.notation-322f9 {
  cursor: default;
  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
  font-size: 14px;
  position: absolute;
}
.alpha-d2270 {
  bottom: 1px;
  right: 3px;
}
.numeric-fc462 {
  top: 2px;
  left: 2px;
}
```

Рисунок 1.4 – CSS код для шахів

Для появи самої шахівниці ми повинні підключити CSS. Так само дана частина коду регулюватиме колір ваших ходів, а також показуватиме, звідки користувач завдавав удару, що буде дуже зручно, при грі, коли користувач

відволікався. Кольори можна зробити будь-якими, ми вибрали найзручніші, щоб вони не зливалися з кольором дошки та фігур.

На цьому малюнку зображено основні функції, без яких не буде візуального стопорника. Оскільки це найголовніша частина, ми докладніше її розберемо. "square" дає нам розосередити квадрати у правильному порядку, тобто у центрі гри, а не інакше. /* black square */ та /* white square */ це кольори самої нашої дошки, на якій ми граємо. Ці кольори так само можна поміняти, як чорний і білий, як в інших ша-хах, але розмальовка в цьому випадку сильно впадає в очі і приємна для сприйняття користувача, так само колір дошки не зливається з фігурами, цифрами і буквами, що знаходяться з боків нашої дошки. highlighted square – дає можливість відзначати ходи користувача, забарвлюючи їх у колір, у нашому випадку жовтий. Це дозволяє бачити користувачеві, куди користувач збирається ходити і звідки це відбувається, це дуже зручно, адже сам колір не зливається з дошкою і досить помітний, щоб користувач може швидко зосередитись. /* notation */ - це розміри вже самих чисел і букв на нашій шахівниці, обравши правильний розмір і шрифт, отже отримуємо зручну функцію, щоб бачити, як ходить суперник і користувач.

На наступному малюнку ми можемо файл style.css, який відповідає за розміри шахівниці по центру екрана вікна браузера. Тобто, можно як збільшувати, так і зменшувати розмір шахівниці і так само зробити її посередині або розмістити її в певній точці на нашій сторінці. Ця функція дозволяє нам при зменшенні вікна браузера тримати в центрі всю дошку разом із фігурами, літерами та цифрами.

```
.board {
    width: 500px;
    margin: auto
}

.info {
    width: 500px;
    margin: auto;
}

.move-history {
    max-height: 100px;
    overflow-y: scroll;
}
```

Рисунок 1.5 – Налаштування розміру шахів

Насамперед треба знайти фігури в інтернеті або малюємо самі. Далі треба передати start як аргумент, щоб ініціалізувати плату в початкову позицію і розставити всі фігури в потрібні клітини.

Рисунок 1.6 – Налаштування розміру шахів

Тепер треба додавати пару кнопок для можливості очищення фігур та повернення їх на початкову позицію.

Далі треба дати можливість переміщати фігури у вільному порядку для початкового розуміння в управлінні. Наступним кроком ми призначимо кожній фігурі її можливі переміщення за правилами шахів.

Рисунок 1.7 – Переміщення фігур у вільному порядку

Рисунок 1.7 демонструє ходи, що стають непотрібними у процесі використання альфа-бета-відсікання. Як бачите, з альфа-бета-відсіканням мінімакс оптимізується, і дуже значно.

Рисунок 1.8 – Ходи, що не потрібні у процесі використання

Тепер є можливість додати штучний інтелект.

Щоб створити простий штучний інтелект, який вміє грати в шахи, можна використати п’ять базових концепції:

1. Переміщення;
2. Оцінка дошки;
3. Мінімакс;
4. Альфа-бета-відсікання.
5. Поліпшена функція оцінки

Для створення шахів треба дивитися, як правильно налаштувати бота. Вибір пав на варіант knowledge-based, щоб користувачеві було цікавіше грати, але неймовірно багато перевірок, які навантажують систему.

Fast searchers	Спростивши до краю оцінну функцію. Відповідно, вона проста, як сокира – перед початком перебору програма будує таблиці, де на кожній клітці намальовано, яка фігура добре, а яка – погано.
knowledge-based	Тут всі сили кинуті написання складної оцінної функції. Враховується і взаємодія фігур друг з одним, і прикритість короля, і контроль позицій, і майже фаза місяця.

На першому етапі використовується chess.js для створення ходів та chessboard.js для візуалізації дошки. Файл, який відповідає за генерацію ходів, дозволяє застосовувати всі шахові правила, тому ми можемо розраховувати кожну дію для конкретного розташування фігур. Робота з цими файлами дозволяє сконцентруватися на головному завданні - пошуку та створенні алгоритму, що дозволяє знайти оптимальний хід. Роботу починаємо з написання функції, що повертає випадковий перебіг зі списку всіх можливих.

При використанні файлу chess.js починається перегляд можливості атак суперника та користувача, та робить перевірки цілої гри.

Для отримання кращого ходу ми маємо перебрати всі ці позиції та отримати їх оцінки з погляду супротивника. Оцінки беруться зі знаком мінус, бо це оцінки з погляду супротивника. Для отримання оцінки з погляду супротивника користувач рекурсивно викликає функцію прорахунку для супротивника та передаємо їй як аргумент, позицію, змінену після нашого ходу. Функція противника теж отримує набір позицій противника і намагається їх оцінити, своєю чергою, рекурсивно викликаючи функцію прорахунку кольору фігур, але вже після свого ходу. Цей процес триває рекурсивно, поки вичерпано глибину прорахунку (ми можемо рекурсивно заглиблюватися до нескінченності) або доки виконається певна умова дострокового припинення перебору.

Перевірок неймовірно багато, адже кожна фігура має свої особливості як при ходьбі, так і при попаданні пішака на кінець території, за якої може стати королевою. Також тут відбувається перевірка ходу короля при його захопленні. Для цього використовується legal_moves, який перевіряє можливість здійснити хід у місце, де не буде ворожого пішака, якщо такої можливості немає, то відбувається інша перевірка.

```
var ATTACKS = [
    20, 0, 0, 0, 0, 0, 0, 24,  0, 0, 0, 0, 0, 0,20, 0,
    0,20, 0, 0, 0, 0, 0, 24,  0, 0, 0, 0, 0,20, 0, 0,
    0, 0,20, 0, 0, 0, 0, 24,  0, 0, 0, 0,20, 0, 0, 0,
    0, 0, 0,20, 0, 0, 0, 24,  0, 0, 0,20, 0, 0, 0, 0,
    0, 0, 0, 0,20, 0, 0, 24,  0, 0,20, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0,20, 2, 24,  2,20, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 2,53, 56, 53, 2, 0, 0, 0, 0, 0, 0,
    24,24,24,24,24,24,56,  0, 56,24,24,24,24,24,24, 0,
    0, 0, 0, 0, 0, 2,53, 56, 53, 2, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0,20, 2, 24,  2,20, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0,20, 0, 0, 24,  0, 0,20, 0, 0, 0, 0, 0,
    0, 0, 0,20, 0, 0, 0, 24,  0, 0, 0,20, 0, 0, 0, 0,
    0, 0,20, 0, 0, 0, 0, 24,  0, 0, 0, 0,20, 0, 0, 0,
    0,20, 0, 0, 0, 0, 0, 24,  0, 0, 0, 0, 0,20, 0, 0,
    20, 0, 0, 0, 0, 0, 0, 24,  0, 0, 0, 0, 0, 0,20
];

var RAYS = [
    17,  0,  0,  0,  0,  0,  0, 16,  0,  0,  0,  0,  0,  0, 15, 0,
    0, 17,  0,  0,  0,  0,  0, 16,  0,  0,  0,  0,  0, 15,  0, 0,
    0,  0, 17,  0,  0,  0,  0, 16,  0,  0,  0,  0, 15,  0,  0, 0,
    0,  0,  0, 17,  0,  0,  0, 16,  0,  0,  0, 15,  0,  0,  0, 0,
    0,  0,  0,  0, 17,  0,  0, 16,  0,  0, 15,  0,  0,  0,  0, 0,
    0,  0,  0,  0,  0, 17,  0, 16,  0, 15,  0,  0,  0,  0,  0, 0,
    0,  0,  0,  0,  0,  0, 17, 16, 15,  0,  0,  0,  0,  0,  0, 0,
    1,  1,  1,  1,  1,  1,  1,  0, -1, -1, -1,-1, -1, -1, -1, 0,
    0,  0,  0,  0,  0,  0,-15,-16,-17,  0,  0,  0,  0,  0,  0, 0,
    0,  0,  0,  0,  0,-15,  0,-16,  0,-17,  0,  0,  0,  0,  0, 0,
    0,  0,  0,  0,-15,  0,  0,-16,  0,  0,-17,  0,  0,  0,  0, 0,
    0,  0,  0,-15,  0,  0,  0,-16,  0,  0,  0,-17,  0,  0,  0, 0,
    0,  0,-15,  0,  0,  0,  0,-16,  0,  0,  0,  0,-17,  0,  0, 0,
    0,-15,  0,  0,  0,  0,  0,-16,  0,  0,  0,  0,  0,-17,  0, 0,
    -15,  0,  0,  0,  0,  0,  0,-16,  0,  0,  0,  0,  0,  0,-17
];
```

Рисунок 1.9 – Перевірки ходу

Також внутрішнє уявлення шахового ходу має формат 0x88 та не призначене для читання людиною. Тому Код перетворює 0x88 квадратних координат на координати алгебри. Він також обрізає непотрібні кнопки переміщення через багатослівний виклик.

Але також є невелика проблема, яку слід вирішити. Тип даних "безособово" доводиться конструювати – конструювати самим. Це дуже просто. Рішення досить цікаве. Множина представлена масивом 16 байт.

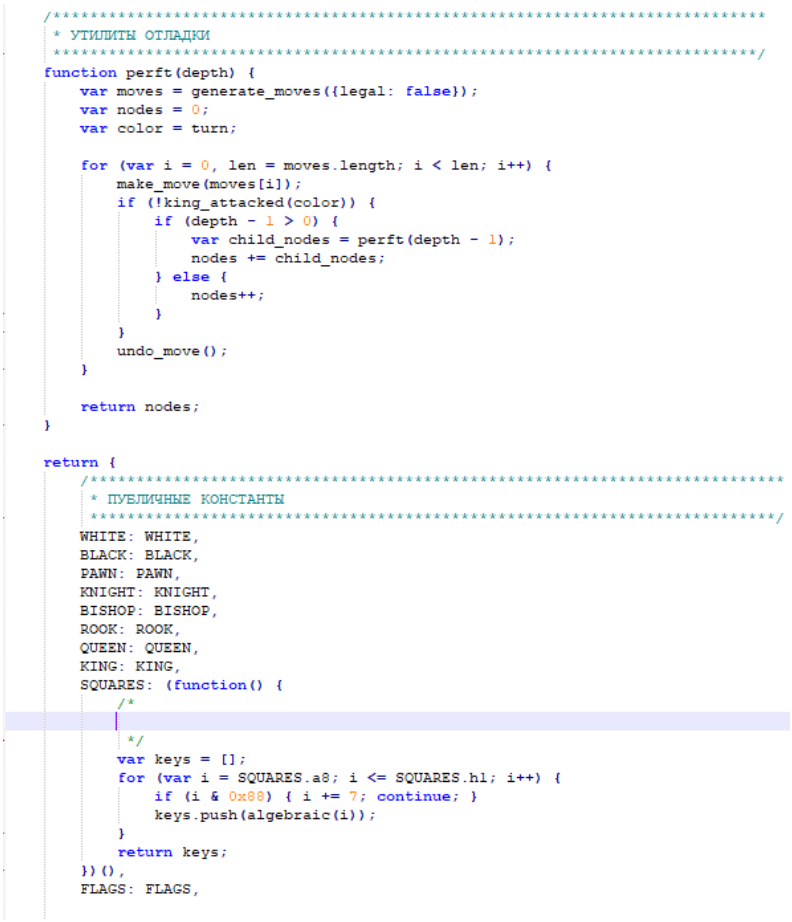
У кожному байті - 8 біт, отже, у множині може бути запам'ятовується 16 x 8 = 128 елементів (значень від 0 до 127).

Якщо елемент (наприклад, 10) є, безліч відповідний біт встановлений в 1. Якщо ні – в 0. Для визначення біта, відповідного даному числу, треба знайти номер байта і номер біта в цьому біті..

```
var board = new Array(128);
var kings = {w: EMPTY, b: EMPTY};
var turn = WHITE;
var castling = {w: 0, b: 0};
var ep_square = EMPTY;
var half_moves = 0;
var move_number = 1;
var history = [];
var header = {};
```

Рисунок 1.10 – Номер байта і біта

Наприкінці коду, так само треба зробити налагодження та перевірку ходів, білих і чорних фігур, для того, щоб не сталося найменшої помилки і користувач міг без проблем насолоджуватися грою з розумним, а не дурним ботом, який дасть користувачеву обіграти його за пару кроків.



```
/*****************************************************************************
 * УТИЛИТЫ ОТЛАДКИ
 *****************************************************************************/
function perft(depth) {
    var moves = generate_moves({legal: false});
    var nodes = 0;
    var color = turn;

    for (var i = 0, len = moves.length; i < len; i++) {
        make_move(moves[i]);
        if (!king_attacked(color)) {
            if (depth - 1 > 0) {
                var child_nodes = perft(depth - 1);
                nodes += child_nodes;
            } else {
                nodes++;
            }
        }
        undo_move();
    }

    return nodes;
}

return {
    /*****************************************************************************
     * ПУБЛИЧНЫЕ КОНСТАНТЫ
     *****************************************************************************/
    WHITE: WHITE,
    BLACK: BLACK,
    PAWN: PAWN,
    KNIGHT: KNIGHT,
    BISHOP: BISHOP,
    ROOK: ROOK,
    QUEEN: QUEEN,
    KING: KING,
    SQUARES: (function() {
        /*

         */
        var keys = [];
        for (var i = SQUARES.a8; i <= SQUARES.h1; i++) {
            if (i & 0x88) { i += 7; continue; }
            keys.push(algebraic(i));
        }
        return keys;
    })(),
    FLAGS: FLAGS,
```

Рисунок 1.11 – Перевірка ходів

Тепер перейдемо до вивчення chessboard.js, адже це теж один із найважливіших моментів у нашій роботі.

Спочатку треба додати функції шахів. Найчастіше треба повторювати одну й ту саму дію в багатьох частинах програми, щоб не повторювати один і той же код у багатьох місцях, придумані функції. Функції є основними "будівельними блоками" програми, тому функції стежитимуть за ходами, і обрізають з кінця інформацію про ходи, рокіровки тощо, показуючи лише інформацію про позиції. Так само в цій функції ми створюємо 8 розділів, розділених косою межею, це потрібно для самої шахівниці.

Далі треба додати константи. Константа це коли, у всьому модулі переважна більшість прикладів коду використовувало змінні як імена (псевдоніми) конкретних значень, а не як змінні, які змінюють своє значення з часом.

Початкова позиція шахової партії:
rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1
rnbqkbnr - розташування фігур на 8-й горизонталі зліва направо, / - роздільник, pppppppp - розташування фігур на 7-й горизонталі,
8/8/8/8 - порожні 6-5-4-3-і горизонталі,
PPPPPPPP - розташування фігур на 2-й горизонталі,
RNBQKBNR - розташування фігур на 1-й горизонталі,
w — чекає хід білих,
KQkq — можливі короткі та довгі рокіровки білих і чорних, не було попереднього ходу пішки на два поля (-), останніх ходів без взяття або руху пішаків не було (0), чекає перший хід (1)

Рисунок 1.12 – Констинта

Тобто, ми використовуємо унікальні імена класів, щоб запобігти конфліктам із чимось ще на сторінці та спростити селектори. JavaScript, як і CSS, має функціонал, що дозволяє звертатися до HTML елементів для перетворення контенту сторінок. У CSS це досягається шляхом написання селекторів.

```
var CSS = {
  alpha: 'alpha-d2270',
  black: 'black-3c85d',
  board: 'board-b72b1',
  chessboard: 'chessboard-63f37',
  clearfix: 'clearfix-7da63',
  highlight1: 'highlight1-32417',
  highlight2: 'highlight2-9c5d2',
  notation: 'notation-322f9',
  numeric: 'numeric-fc462',
  piece: 'piece-417db',
  row: 'row-5277c',
  sparePieces: 'spare-pieces-7492f',
  sparePiecesBottom: 'spare-pieces-bottom-ae33d',
  sparePiecesTop: 'spare-pieces-top-4028b',
  square: 'square-55d63',
  white: 'white-1e1d7'
};
```

Рисунок 1.13 – Функціонал CSS

Наступним зробимо змінні області модуля, що складаються з DOM - елементів, і додамо об'єкт конструктора, що повертається.

Як би не було дивно, але треба зробити стан, де буде показуватися анімація, розміри межі дошки, поточні позиції і поточні орієнтації, перетягнуті частини, запасні частини та інше.

Потім, знову ж таки треба додати функції, але цього разу ми додамо JS-функції. До цієї функції ми додаємо глобально унікальні індикатори. Це ідентифікатори, які призначені для забезпечення певних гарантій унікальності. Це підпрограми, доступні у всіх браузерах, створює генерацію чисел, необхідних гри.

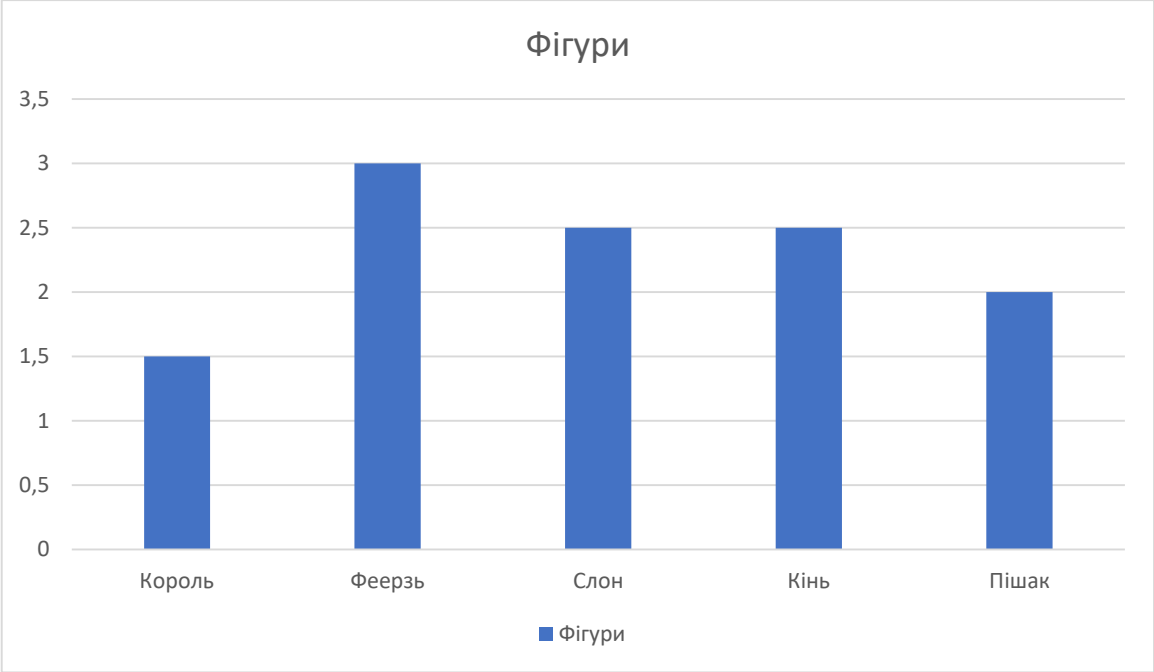
Без перевірок на помилки нікуди, особливо в такій великій і важкій грі, як шахи. Тому наступним кроком буде додавання перевірок/помилки. Ця частина коду робить велику перевірку ходів. Також, тут привласнюється значення орієнтації за умовчанням. Від цього залежить, з якого боку відбуватиметься хід. Також тут додається фігури, у вигляді .png картинки.

Наприклад, пішак ходить вперед на одну клітку. Б'є вправо та вліво по діагоналі на одну клітку. Якщо пішак робить свій перший хід, і клітина перед нею вільна, вона може піти через клітину. Якщо пішака пішла через клітку і перетнула поле, бите пішаком супротивника, пішака супротивника може взяти її, при цьому вона стає в полі, яке б'є, тобто хоча пішака пішла через клітку, її взяли так, ніби вона пішла на одну клітку. Якщо пішак дійшов до кінця дошки, він може перетворитися на будь-яку фігуру, крім короля. Король ходить на одну клітку у всіх напрямках. Взяти пішки через бите поле. Ходи короля. Якщо король і тура не переміщалися з початку гри, вони можуть зробити рокіровку. При цьому король під шахом рокірування не робить і не повинен ресікати і ставати на бите поле. Кінь ходить буквою Г. Ходи інших постатей досить прості.

Слон ходить по діагоналі на будь-яку кількість клітин. Ладья ходить по вертикалі та горизонталі на будь-яку кількість клітин. Королева (ферзь) -

найсильніша постать. Вона ходить у всіх напрямках, і мат, як правило, ставиться за її допомогою.

Ефективність фігур у грі



Доцільно згадати ще кілька правил:

1. Всі фігури, крім пішаків, б'ють так само, як ходять.
2. Мета гри - поставити мат супротивнику. Мат – це коли король підбоєм, і піти йому нікуди, він скрізь потрапляє під бій.
3. Пат - це коли ходити може тільки король, а піти йому нікуди, він скрізь потрапляє під бій. Король при паті не під шахом. Пат - це нічия. коли будь-яка позиція повторюється тричі.
4. Нічим може бути деяке безперспективне закінчення, коли немає коштів, щоб поставити мат, або противники погоджуються на нічию.

Далі треба додати швидкість анімації, ставимо середнє значення, щоб око людини могло без проблем встежити за рухом анімації.

Потім треба зробити DOM Misc, ця частина коду обчислює розмір квадрата на основі ширини контейнера. тут трохи чорної магії CSS, тому дозвольте мені пояснити: отримати ширину елемента контейнера (може бути будь-що), зменшити на 1 для коефіцієнта вигадки, а потім продовжувати зменшувати, поки ми не знайдемо точний мод 8 для нашого квадратного розміру.

Тут робиться перевірка на запобігання нескінченному циклу, додаємо один піксель, далі створюємо випадкові ідентифікатори для елементів, потім знаходимо точний мод, для квадратів на нашій дошці і додаємо запасні частини.

Далі створюється розмітка за допомогою CSS. Тут буде використовуватися алгебраїчне позначення, альфа-нотації, числові позначення, і робимо перевірки, того, чого ніколи не повинно відбуватися, при розмітці. Тепер ми робимо анімації в повному обсязі. Насамперед даємо можливість анімації отримувати інформацію про вихідні та цільові квадрати.

Далі створюється анімований фрагмент і абсолютно позиціонуємо його над вихідним квадратом, потім займаємося анімацією самих фігур та вихідних квадратів. Тут треба додати фігури в цільову клітину, тобто вільну, і анімуємо переходи, після чого видаляємо анімацію, по завершенню перенесення фігури в потрібний квадрат. Наступним кроком буде пошук за позицією в порядку відстані для фігури, обчислюємо масив анімації, який має статися, щоб потрапити з однієї точки в іншу.

Потік управління – керує ходом фігур. Для початку треба почистити дошку і потім додаємо шматочки дошки. Потім надають можливість враховувати позицію та набір ходів, повертаємо нову позицію з виконаними ходами. Далі пропускаємо хід, якщо в позиції немає фігури на вихідному полі, або нічого не робимо якщо немає змін у положенні фігур. Потім отримується позиція вихідного квадрата, анімуємо шматок в цільовий квадрат і видаляється вихідний фрагмент, тому що він нам уже не знадобиться.

Наступним додається можливість запустити свою функцію onDragStart, так само їх функція onDragStart може скасувати початок перетягування, що не дасть користувачеві зробити зайвий хід і помилитися. Так само створюємо шматок, що перетягується, виділяємо вихідний квадрат і приховуємо шматок, який збираємося перетягнути, поміщаємо перетягується фрагмент на курсор миші, щоб ми могли керувати фігурою, як захочемо, але так само додамо перевірку, яка перевірить, якщо місцезнаходження не змінилося, то нічого не робимо. Це має на увазі можливість гравця передумати ходити фігурою, яку він уже вибрав, і ми повертаємо її на вихідну позицію.

Давньою мрією шахових програмістів є генерація взяття далекобійних фігур, не будуючи лінії. Генерація взяття є найкритичним за часом виконання фрагментом програми. Справа в тому, що більшість позицій можна відсікти лише взяттями, і навіть не обов'язково мати весь перелік переміщень.

Крім того, взяття у форсованому пошуку розглядаються до кінця. Генерація переміщень на глибині 20 набагато більше впливає швидкість виконання програми, ніж на глибині 5.

Як писати шахи, не маючи потоку управління? Навіщо він нам потрібний? Відомо що потом управління дає можливість ходити у всіх напрямках, по всіх діагоналях, вертикалях та горизонталь. З вертикалями та горизонталями все зрозуміло: якщо був хоч один ферзь з координатою X чи Y такий, як у нашого ферзя, то це поле б'ється. Як бути із діагоналями? Якщо діагональ висхідна (/), то сума X і Y всім клітин діагоналі одна. Якщо низхідна (\), то раз- різниця X - Y для всіх клітин однакова. Ми можемо мати деякі множини, обнулені на початку обчислення: 1. Безліч X. 2. Y. 3. X+Y. 4. X-Y. Щоб дізнатися, чи б'ється це поле будь-яким ферзем, ми повинні перевірити клітини масивів (множин) 1, 2, 3, 4. Для масиву 1 це буде клітина з індексом X, для масиву 2 - Y, для 3 - (X+ Y), для 4 - (X-Y +8). Всі ці клітини мають бути нульовими. Якщо хоч одна клітина 1, поле б'ється. Якщо ми поставили ферзя в якусь клітку, то у відповідні осередки множин маємо записати 1.

Якщо скасували хід — має відновитися старе значення осередків. Масив 4 має додатковий індекс 8 тому, що X-Y не може бути менше 0, а в мові С індексація масивів від 0. Якщо ми поставили останнього ферзя, то кінець перебору та виведення результату. Так як ми маємо справу з окремим виродженим випадком, то замість 4 множин можна використовувати 3, а координату X збільшувати покроково. Це гарантує, що не буде двох ферз з однаковою координатою X. Старі значення множин можна також не зберігати і не відновлювати, тому що якщо ми поставили ферзя на клітку, то це гарантія того, що у відповідних осередках масивів були нулі.

```
function isXYOnSquare(x, y) {
  for (var i in SQUARE_ELS_OFFSETS) {
    if (SQUARE_ELS_OFFSETS.hasOwnProperty(i) !== true) continue;

    var s = SQUARE_ELS_OFFSETS[i];
    if (x >= s.left && x < s.left + SQUARE_SIZE &&
        y >= s.top && y < s.top + SQUARE_SIZE) {
      return i;
    }
  }

  return 'offboard';
}


function captureSquareOffsets() {
  SQUARE_ELS_OFFSETS = {};

  for (var i in SQUARE_ELS_IDS) {
    if (SQUARE_ELS_IDS.hasOwnProperty(i) !== true) continue;

    SQUARE_ELS_OFFSETS[i] = $('#' + SQUARE_ELS_IDS[i]).offset();
  }
}
```

Рисунок 1.14 – Генерація взяття фігур

Якщо потрібно отримати взяття короля або коня, просто робиться операція порозрядного І маски переміщень цієї фігури для конкретної клітини та маски всіх фігур супротивника. Для отримання переміщень далекобійних фігур використовуються повернені (у прямому та переносному сенсі) бітові дошки. У цьому прикладі використовується спрощений прийом. Перед початком обчислення ініціалізовано 4 масиви для кожного кольору фігур.

У них міститься інформація, скільки фігур у кожній вертикалі, горизонталі та діагоналях. Якщо діагональ висхідна, сума координат всіх клітин однакова. Якщо низхідна - різниця XY одна для всіх клітин діагоналі. Масиви коригуються покроково. Наприклад, пішов білий пішак e2-e4. Її цільові координати X = 4, Y = 4. У першому масиві в клітці з індексом X збільшимо значення на 1. У другому масиві-в клітці з індексом Y. Інші два масиви потрібно коригувати в клітинах (X+Y) і (X- Y+8). Відповідним чином потрібно коректувати. коригувати масиви клітин, звідки перемістилася постать. Далі, наприклад, береться взяття королеви, сканхуємо вертикаль, тільки якщо в першому масиві в клітці X не дорівнює 0. Всі інші промені беремо, якщо у відповідних масивах у клітинах Y, X+Y, XY+8 не нули. Це просто і досить ефективно.

Далі додається Public Methods. Всі публічні методи повинні бути оголошені як властивості, а не змінні/функції. У нашому випадку ми так само робимо перевірку ходів та анімації, адже це дуже потрібна та важлива частина шахів.

Виявлення браузера (Browser Events). Корисною, але часто переоцінюється функцією JavaScript є визначення браузера. Іноді ви хочете дати конкретні інструкції або завантажити нову сторінку, якщо користувач використовує, наприклад Safari. Якщо користувач новачок у JavaScript, не використовується визначення браузера. Вони не потрібні. Використовується браузер WhatBrowser, бо ця сторінка допомагає знайти браузер. Він актуальний і містить джерело мобільної інформації. Нижче можно побачити об'єкти, що містяться у навігаторі об'єктів.

```
navigator.vendorSub =
navigator.productSub = 20030107
navigator.vendor = Google Inc.
navigator.maxTouchPoints = 0
navigator.userActivation = [object UserActivation]
navigator.doNotTrack = null
navigator.geolocation = [object Geolocation]
navigator.connection = [object NetworkInformation]
navigator.plugins = [object PluginArray]
navigator.mimeTypes = [object MimeTypeArray]
navigator.pdfViewerEnabled = true
navigator.webkitTemporaryStorage = [object DeprecatedStorageQuota]
navigator.webkitPersistentStorage = [object DeprecatedStorageQuota]
navigator.hardwareConcurrency = 4
navigator.cookieEnabled = true
navigator.appCodeName = Mozilla
navigator.appName = Netscape
navigator.appVersion = 5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.127 Safari/537.36 OPR/86.0.4363.59
navigator.platform = Win32
navigator.product = Gecko
navigator.userAgent = Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.127 Safari/537.36 OPR/86.0.4363.59
navigator.language = ru-RU
navigator.languages = ru-RU,ru,en-US,en
navigator.onLine = true
navigator.webdriver = false
navigator.getBattery = function getBattery() { [native code] }
navigator.getGamepads = function getGamepads() { [native code] }
navigator.javaEnabled = function javaEnabled() { [native code] }
navigator.sendBeacon = function sendBeacon() { [native code] }
navigator.vibrate = function vibrate() { [native code] }
navigator.scheduling = [object Scheduling]
navigator.bluetooth = [object Bluetooth]
navigator.clipboard = [object Clipboard]
navigator.credentials = [object CredentialsContainer]
navigator.keyboard = [object Keyboard]
navigator.managed = [object NavigatorManagedData]
navigator.mediaDevices = [object MediaDevices]
navigator.storage = [object StorageManager]
navigator.serviceWorker = [object ServiceWorkerContainer]
navigator.wakeLock = [object WakeLock]
navigator.deviceMemory = 8
navigator.ink = [object Ink]
navigator.hid = [object HID]
navigator.locks = [object LockManager]
navigator.mediaCapabilities = [object MediaCapabilities]
navigator.mediaSession = [object MediaSession]
navigator.permissions = [object Permissions]
navigator.presentation = [object Presentation]
navigator.serial = [object Serial]
navigator.virtualKeyboard = [object VirtualKeyboard]
navigator.usb = [object USB]
navigator.xr = [object XRSystem]
navigator.userAgentData = [object NavigatorUAData]
navigator.canShare = function canShare() { [native code] }
navigator.share = function share() { [native code] }
navigator.getInstalledRelatedApps = function getInstalledRelatedApps() { [native code] }
navigator.getUserMedia = function getUserMedia() { [native code] }
navigator.requestMIDIAccess = function requestMIDIAccess() { [native code] }
navigator.requestMediaKeySystemAccess = function requestMediaKeySystemAccess() { [native code] }
navigator.webkitGetUserMedia = function webkitGetUserMedia() { [native code] }
navigator.registerProtocolHandler = function registerProtocolHandler() { [native code] }
navigator.unregisterProtocolHandler = function unregisterProtocolHandler() { [native code] }
```

Рисунок 1.15 – Навігатор об’єктів

Ці змінні можна зчитувати та надавати інформацію про браузер та комп'ютер ваших користувачів, що дозволить швидко адаптуватися до користувача та дати йому можливість чудово провести час у грі, а творцям, спростити можливі проблеми з браузерами, адже кожен використовує свій браузер, що не дає можливості все спростити, а так як гра робиться так, щоб, як найбільш людей грали в неї, то доводиться витрачати час на додавання додаткових функцій. У цьому випадку треба дивитися, що може і не може робити. Наприклад, нічого не робити якщо обрану користувачам рандомну фігуру не можна перетягувати. Також встановлюємо гарантію, щоб не сталося проблем із запасними частинами.

Останнім пунктом додається ініціалізацію. Ініціалізатор об'єкта це розділений комами список нуль або більше пар імен властивостей та значень, що асоціюються з ними. Тут треба запобігати "перетягування зображення" браузером шматочки перетягування мишею, щоб дошка або фігури не змогли вийти зі свого "вимірювання", до якого приписані.

На другому етапі найпростіший шлях - розрахувати відносну силу фігур на дошці, це можна зробити за допомогою таблиці.

Рисунок 1.16 – Відносна сила фігур

Використовуючи функцію оцінки, мається можливість створити алгоритм, який вибирає хід із максимальною оцінкою.

Після другого етапу, починається третій. Треба створити дерево пошуку. Тепер програма може вибрати із нього найкращий хід. Це робиться за допомогою мінімакс-алгоритму. Тут рекурсивне дерево з відображенням усіх можливих ходів аналізується до заданої глибини. Позиція ж оцінюється на листі нашого дерева. Далі повертається мінімальне чи максимальне значення нащадка у батьківський вузол. Все залежить від того, хід якої сторони зараз прораховується. Іншими словами, результат максимізується або мінімізується на кожному рівні.

Четвертий етап оптимізації мінімакс-алгоритму, що дозволяє ігнорувати деякі гілки в дереві пошуку. І це дозволяє збільшити глибину пошуку, витрачаючи колишній обсяг ресурсів. Альфа-бета-відсікання ґрунтується на ситуації, коли ми можемо зупинити оцінку певної гілки, якщо виявляється, що новий хід призведе до гіршої ситуації, ніж та, яку ми бачили при оцінці попереднього. На результат мінімаксу оптимізація не впливає, але все починає працювати швидше. Цей алгоритм набагато ефективніший у тому випадку, якщо спочатку перевірити шляхи, що ведуть до хороших ходів.

Переходимо до останнього етапу. Початкова функція оцінки досить проста, оскільки вона просто вважає окуляри фігур, що знаходяться на дошці. Для її оптимізації можна враховувати становище фігур. Наприклад, якщо розмістити коня у центрі дошки, він стає дорожче — спектр доступних ходів цієї фігури розшириться.

Рисунок 1.17 – Хід коня

Після створення шахів, обов’язково додається меню для зручності переміщення між режимами гри. Так само це буде перша сторінка, на яку буде потрапляти користувач, тому треба зробити кольори приємнішими і помітнішими, щоб вони не зливалися з картинкою та один одним. Для початку треба створити п'ять квадратів, які розмістимо наші посилання для входу в гру. Щоб користувачеві було зручніше відразу ж почати орентуватися в основному меню, ми додали загоряння посилання, при наведенні курсору мишки на них, так само їх підчекуємо, і сам курсор змінює свою іконку при наведенні до посилання. Це все дозволяє дуже швидко вибрати, що потрібно користувачеві і не дати йому заплутатися, та й це просто естетично, приносить яскравіших фарб в похмуре меню користувача.

На малюнку нижче робиться можливість забарвлювати наші посилання. При наведенні посилання світитися одним кольором, а при затисканні іншим це проходить все дуже швидко, але має гарний ефект, що показує, що ви натиснули на ту або іншу кнопку. Так само все підкреслюється, що точно дасть користувачеві можливість не заплутатися.

```
body {
  width: 300px;
  margin: 0 auto;
  font-family: sans-serif;
}
p {
  line-height: 1.4;
}
a {
  outline: none;
  text-decoration: none;
  padding: 2px 1px 0;
}
a:link {
  color: blue;
}
a:visited {
  color: purple;
}
a:focus, a:hover {
  border-bottom: 1px solid;
}
a:active {
  color: red;
}
a[href*="http"] {
  background: url('external-link-52.png') no-repeat 100% 0;
  background-size: 16px 16px;
  padding-right: 19px;
}
body,html {
  margin: 0;
  font-family: sans-serif;
}
ul {
  padding: 0;
  width: 100%;
}
li {
  display: inline;
}
li:last-child a {
  margin-right: 0;
}

a:hover {
  background: orange;
}

a:active {
  background: red;
  color: white;
}
```

Рисунок 1.18 – Окрас посилання

На наступному малюнку побачемо створення самих іконок, в яких перебувати переходи до гри, так само тут надається колір цим кнопкам. Також тут відбувається функція розміщення їх по центру, для того, щоб користувачеві було зручніше переміщатися. Треба було прийти до такого рішення, тому що треба було вирішувати, що кнопки з боку не дадуть можливості швидше і зручніше переміщатися по меню, а також це щось незвичайне, адже більшість ігор створитися з бічним меню, не дуже зручно, змушує заплутатися за кілька секунд.

```
body{
    color:black;
    font-family:arial;
    background-size:cover;
}
#story_box{
    width:400px;
    padding:20px;
    position:absolute;
    top:25%;
    text-align:center;
    left:calc(50% - 200px);
    background-color:rgba(139, 0, 255, 0.2);
    border-radius:15px;
}
#story{
    background-color: rgba(80, 200, 120, 0.8);
    padding: 20px;
    border-radius:15px;
}

#btn1,
#btn2,
#btn3{

    display:inline-block;
    width:200px;
    padding:20px;
    margin-top:20px;
    border-radius:15px;
}
#btn1{

    color:black;
    background-color:white;
}
#btn2{
    background-color:white;
}
#btn3{
    background-color:white;
}
```

Рисунок 1.19 – Іконки

Також треба додати додамо js файл, який буде підключений до головного меню, щоб привласнити нашому меню фонову картинку, для більш гарної візуалізації.

```
function lvl1()
{
    body.style.backgroundImage = 'url(bd1.jpg)'
}
lvl1();
```

Рисунок 1.21 – Фон

Також, для більшого реалізму та розуміння того, що відбувається в нашій грі, додається показ ходів бота та гравця, де пишеться, хто і куди зміг походити за номером та буквою, це можливість дає нам більш правдоподібне розуміння гри, а так само ви зможете вивчити свої ходи в історії, щоб зрозуміти, де і як ви могли помилятися. Ця фіча дуже корисна, не тільки для вивчення, а так само для розуміння, хто і куди був схожим, якщо ви відволіклися, це може бути все що завгодно.

Наприклад у маршрутці користувача штовхнули або хтось відволік на роботі, і користувач не встиг побачити, як походив суперник, хоча для цього є колірні позначки ходів.

Далі було прийнято рішення додати годинник, щоб можно було бачити скільки користувач грає, це дозволить користувачеві повністю зануритися в гру і не дивитися на годинник, щоб засікти час вашого проводження.

```
<form name=MyForm>
  <input name=stopwatch size=10 value="00:00:00.00" disabled>
</form>
```

Рисунок 1.21 – Годинник

Разом із цим було ухвалено рішення додати можливість вибирати швидкість ходів. Якщо користувач любить грати повільно і любить відволікатися, то можно вибрати можливість довгого ходу товариша, або є можливість вибрати неймовірну швидкість, якщо користувач любить дуже швидко грати і може читати супротивника.

```
<div class="info">
    Search depth:
    <select id="search-depth">
        <option value="1">1</option>
        <option value="2">2</option>
        <option value="3"selected>3</option>
        <option value="4">4</option>
        <option value="5">5</option>
    </select>
```

Рисунок 1.22 – Швидкість ходу бота

Після перегляду мене та вивчення візуального аспекту нашого проекту, ми перейдемо до самого проекту, який реалізує можливість ходити гравцю. Через складність гри та велику кількість фігур, які мають свої можливості, доводиться робити неймовірно багато перевірок, через що більша частина програми лише перевіряє можливість ходити та перевіряти правильність ходу. У цьому файлі так само ми встановили можливість прискорювати або уповільнювати хід робота, додаємо фігури, і пишемо, програв або ж виграв гравець.

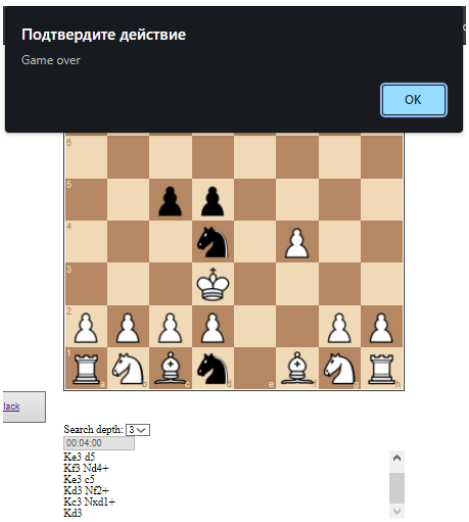


Рисунок 1.23 – Можливість бачити, коли гравець програв

2 ОХОРОНА ПРАЦІ

Однією із характерних особливостей сучасного розвитку суспільства є зростання сфер діяльності людини, в яких використовуються інформаційні технології. Широке розповсюдження отримали персональні комп'ютери. Однак їх використання загострило проблеми збереження власного та суспільного здоров'я, вимагає удосконалення існуючих та розробки нових підходів до організації робочих місць, проведення профілактичних заходів для запобігання розвитку негативних наслідків впливу ПК на здоров'я користувачів.

Слід зважити, що безпечні умови праці – не тільки запорука комфортного існування працівників у межах підприємства, а в першу чергу – їх здоров'я та працездатності, а відтак і прибутковості підприємства.

В розділі дипломного проекту розглядається питання охорони праці програміста на стадії вирішення ним питань побудови і аналізу алгоритмів поведінки користувача і штучного інтелекту на базі веб-орієнтованої системи «Шахова гра».

3.1 Аналіз небезпечних та шкідливих чинників, що впливають на працівника.

Оператори ПК і програмісти зіштовхуються із впливом таких фізично небезпечних і шкідливих виробничих факторів, як підвищений рівень шуму, підвищена температура зовнішнього середовища, недостатня освітленість робочої зони, електричний струм та інші. Тому на робочому місці програміста повинні бути створені умови для високопродуктивної праці.

3.2 Розробка заходів з охорони праці

3.2.1 Виробничі приміщення

Розміщення робочих місць з ВДТ заборонено у підвальних приміщеннях та на цокольних поверхах заборонено. Для приміщень, які призначені для роботи з ВДТ, доцільно обрати орієнтацію вікон на північ або на північний схід. На вікнах повинні бути жалюзі, що регулюються, або штори, що дають можливість їх повністю закривати.

Приміщення відповідно до ДБН В.2.5-28-2006 «Природне і штучне освітлення» повинні мати природне та штучне освітлення. При приміщеннях з ВДТ мають бути обладнані побутові приміщення для відпочинку, психологічного розвантаження тощо.

Площа на одне робоче місце для користувачів повинна складати не менше 6 кв.м, а об'єм – не менше 20,0 куб.м.

3.2.2 Мікроклімат робочої зони працівників, вентиляція.

Основними нормативними документами, що регламентують параметри мікроклімату виробничих приміщень, є ДСН 3.3.6.042-99 та ГОСТ 12.1.005-88. Ці параметри нормуються для робочої зони - визначеного простору, в якому знаходяться робочі місця. У нормативному документі ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» параметри мікроклімату повинні відповідти данним таблиці 1.

Таблиця 1 – Параметри мікроклімату приміщення.

Перід року	Категорія робіт	Температура, С		Відносить. Вологість, %
		оптимальна	допустима	
Холодний	Легка – Іа	22-24	21-25	40-60
Теплий	Легка – Іа	23-25	22-26	40-60

Примітка: Іа – категорія легких фізичних робіт при яких витрата енергії дорівнює 105-140Вт (90-120 ккал/год)

У приміщенні де працює оператор параметри мікроклімату відповідають нормам з таблиці 1.

Для створення в приміщенні нормальних умов мікроклімату для працівника і видалення шкідливих забруднень, була спроектована і належним чином встановлена вентиляційна система – загально обмінна, припливно-витяжна по нормам ДСТУ Б А. 3.2-12:2009 ССБП.

Вентиляція створює на робочому місці, метеорологічні умови і чистоту повітряного середовища, що відповідаюсь чинним санітарним нормам. Разом з тим вентиляція забезпечує умови, що відповідають вимогам технологічного процессу.

3.2.3.Освітлення робочого місця, шум, вібрація

Приміщення, в яких встановлені персональні комп'ютери, повинні мати природне та штучне освітлення. Природне освітлення здійснюється через світові прорізи (вікна), орієнтовані переважно на північ чи північний схід. Штучне освітлення в приміщені здійснюється системою загального рівномірного освітлення. На поверхні столу в зоні розміщення документів штучне освітлення має становити 300-500лк.

Так як шум має 35Дб, сприйняття шуму людським вухом межується від 20Дб до 120 дб, це означає, що при роботі за ЕОМ шум не заважає, працівнику працювати.

Для запобігання виникнення інших шумів у відповідності з ГОСТ 12.1.029-80 зниження шуму й вібрації в приміщенні дипломним проектом передбаченні звукоізоляція вікон та дверей.

3.2.4 Електробезпека.

На відмінну від інших джерел небезпеки електричний струм не можна виявити без спеціального устаткування й приладів, тому вплив його на людину найчастіше зненацький.

Проходячи через організм людини електричний струм робить термічну, електролітичну і біологічну дію. У результаті термічного впливу викликається розігрів організму й виникають опіки ділянок тіла, у результаті електролітичного впливу розкладається кров і інші органічні рідини в організмі. Біологічний вплив проявляється в порушенні й роздратуванні тканин і мимовільному судорожному скороченні м'язів.

Для попередження поразок електричним струмом необхідно:

- У повному обсязі виконувати правила провадження робіт і правил технічної експлуатації;
- Виключати можливість доступу працівника до частин устаткування, що працює під небезпечною напругою, неізольованим частинам, призначеним для роботи при малій напрузі й не підключеним до захисного заземлення;
- Застосовувати ізоляцію, що служить для захисту від поразки електричним струмом.
- Заземлені конструкції, що знаходяться в приміщеннях, де розміщені робочі місця операторів (батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном) мають бути надійно захищені діелектричними щитками або сітками з метою недопущення потрапляння працівника під напругу.

3.2.5 Організація робочого місця користувача ПК

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. Власник або уповноважений ним орган повинен впроваджувати сучасні засоби техніки безпеки, які запобігають виробничому травматизму, і забезпечувати санітарно-гігієнічні умови, що запобігають виникненню професійних захворювань працівників

Робочі місця слід розташовувати відносно світових прорізів так, щоб природне світло падало переважно з лівого боку.

Конструкція робочого місця користувача персонального комп’ютера має забезпечити підтримання оптимальної робочої пози офісного працівника.

Конструкція робочого столу має відповідати сучасним вимогам ергономіки і забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання (дисплея, клавіатури, принтера) і документів.

Робочий стілець має бути підйомно-поворотним, регульованим за висотою, з кутом і нахилу сидіння та спинки і за відстанню від спинки до переднього краю сидіння поверхня сидіння має бути плоскою, передній край – заокругленим. Поверхня сидіння і спинки стільця має бути напівм'якою з нековзним, повітронепроникним покриттям, що легко чиститься і не електризується.

Робоче місце з персональним комп'ютером слід обладнати пюпітром для документів, що легкорегулюються за висотою над сидінням у межах 230-260мм і відстанню між підлокітниками в межах 350-500мм.

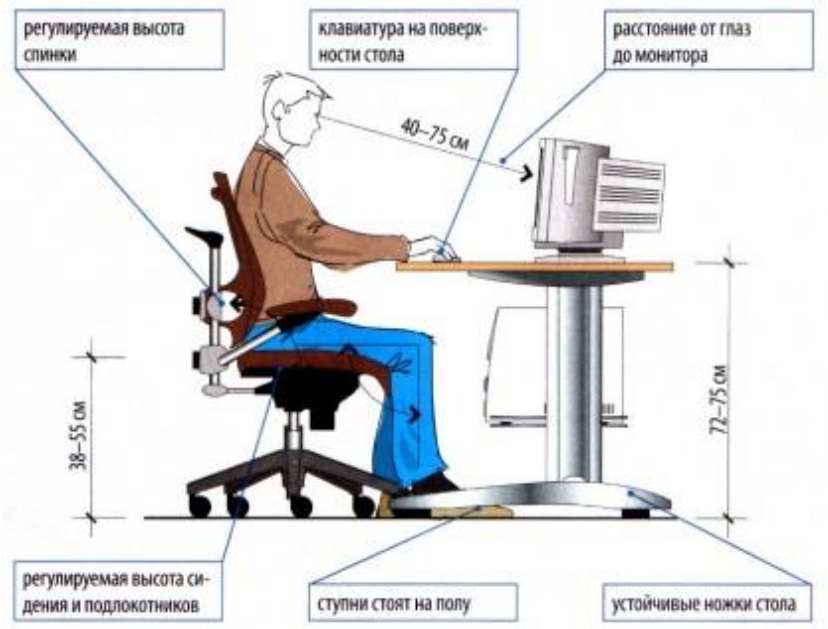


Рисунок 2.1 – Правильне робоче місце

Для забезпечення захисту і досягнення нормованих рівнів комп'ютерних випромінювань необхідно застосування приекранних фільтрів, локальних світлофільтрів (засобів індивідуального захисту очей) та інших засобів захисту, що пройшли випробування в акредитованих лабораторіях і мають щорічний гігієнічний сертифікат.

3.3 Пожежна безпека

Всі приміщення повинні бути забезпечені первинними засобами пожежогасіння: пожежним водопостачанням (пожежні крани ПК), пожежні щити з набором пожежного інструменту, Пожежна безпека приміщень, що мають електричні мережі, регламентується ГОСТ 12.1.033-81, ГОСТ 12.1.004-85. Робота оператора ЕОМ повинна вестися в приміщенні, що відповідає категорії Д пожежної безпеки (негорючі речовини й матеріали в холодному стані).

Протипожежний захист приміщення забезпечується застосуванням установки автоматичної пожежної сигналізації, наявністю засобів пожежогасіння, організацією своєчасної евакуації людей.

Приміщення оснащені вуглекислотними або порошковими вогнегасниками. У випадку виникнення пожежі необхідно відключити електроживлення, викликати по телефону 101 пожежну команду, евакуювати людей із приміщення відповідно до плану евакуації і приступити до ліквідації пожежі.

Будівлі укомплектовані пожежними щитами з набором інструментів, біля щитів – бочки з водою, ящики з піском.

Рисунок 2.2 – Набор інструментів

Виробничі приміщення мають запасні виходи. Двері повинні мати освітлений надпис « Запасний вихід». План евакуації вивішується на видному місці у основного виходу із приміщення.

3 ЕКОНОМІЧНА ЧАСТИНА

Метою даних розрахунків є обчислення вартості виконання науково-дослідної роботи «Побудова і аналіз алгоритмів поведінки користувача і штучного інтелекту на базі веб-орієнтованої системи "Шахова гра». Розроблений проект призначений для саморозвитку користувача. Шахи дадуть людині розвинути мислення та будувати стратегію. Даний вид проекту відноситься до науково-дослідницької розробки. Оцінка якості розробленого проекту включає визначення трудомісткості і вартості його створення.

Розрахунок трудомісткості НДР здійснений в наступній послідовності:

1) Складений перелік всіх етапів і видів робіт, які необхідно виконати в ході даної НДР. Після узгодження з керівником проекту допущено виключення, доповнення, об'єднання окремих етапів і видів робіт;

2) По кожному виду робіт визначений кваліфікаційний рівень виконавців. Перелік етапів і робіт, що виконуються при проведенні НДР, приведений в таблиці 4.1.

Розподіл робіт по етапах і видах виконавців.

Таблиця4.1. План роботи

Етап проведення НДР	**Вигляд робіт**	**Посада виконавця**
Розробка технічного завдання (ТЗ)	1.Складання і затвердження ТЗ для НДР по розробці «Побудова і аналіз алгоритмів поведінки користувача і штучного інтелекту на базі веб-орієнтованої системи "Шахова гра»	Дипломник, керівник
Вибір напряму дослідження	1. Збір і вивчення науково-технічної літератури. 2. Формулювання можливих напрямів вирішення завдань, поставлених в технічному завданні НДР і їх порівняльна оцінка. 3. Вибір напряму проведення досліджень 4. Розробка плану проведення досліджень для подальшої розробки.	Дипломник керівник
Теоретичні і	Аналітичний розділ дипломного проекту.	Дипломник

експериментальні дослідження	1.Огляд існуючих рішень 2. Мова JavaScript 3 Створення шахів за допомогою мов програмування 11	керівник консультанти
Узагальнення і оцінка результатів досліджень	1. Узагальнення результатів 2. Оцінка повноти вирішення поставлених завдань. 3.Складання і оформлення звіту. Розгляд результатів проведеною НДР і прийняття результатів в цілому.	Дипломник керівник консультанти

Оцінка тривалості виконання робіт розраховується на основі вірогідних оцінок робіт, що задаються виконавцями.

Очікувана трудомісткість робіт.

Таблиця 4.2. Очікуванна вартість

Вигляд роботи	Очікуваний час виконання (дні)
1. Складання і затвердження ТЗ для НДР «Побудова і аналіз алгоритмів поведінки користувача і штучного інтелекту на базі веб-орієнтованої системи "Шахова гра»	1

2. Збір і вивчення науково – технічної літератури, технічної документації і інших матеріалів.	2
3. Формулювання можливих напрямів вирішення завдань, поставлених в технічному завданні НДР і їх порівняльна оцінка.	2
4. Розробка плану проведення досліджень для подальшої розробки.	1
5. Аналітичний розділ. 1.Огляд існуючих рішень 2. Мова JavaScript 3 Створення шахів за допомогою мов программування 11	14
Всього:	20

Розрахунок собівартості і ціни виконання НДР. Виходячи з особливостей створення науково – технічної продукції і її залежності від інтелектуальної праці, розрахунок собівартості і ціни виконання НДР включає наступні статті витрат: витрати на матеріали, основна і додаткова заробітна плата, відрахування до єдиного соціального фонду страхування, витрати на роботи, що виконуються сторонніми організаціями, і деякі інші.

1) Витрати на матеріали складають 160 грн.

2) До витрат «Основна заробітна плата» відносяться оплата праці виконавців, безпосередньо притягнених до її виконання. Розмір основної зарплати встановлюється виходячи з чисельності різних категорій виконавців, трудомісткості, що витрачається ними на виконання різних видів робіт, а також їх середньої заробітної плати (ставки) за один робочий день. Відповідно до статті 8 «Закону про Державний бюджет України на 2021» встановлено мінімальну заробітну плату у місячному розмірі з з 1 січня 2022 року - 6500 гривень; мінімальну погодинну тарифну ставку – 39,26 грн.

Середня зарплата за один робочий день для кожного виконавця визначена по формулі:

Зден= п.т.с. * 8;

де п.т.с – погодинна тарифна ставка, грн..;

8 – тривалість робочого дня, год.

Зден дипломника = 39.26*8 = 314,08 грн.

Зден керівника = 60 *8 = 480 грн.

Зден консультантів = 55*8 =440 грн.

Витрати на основну заробітну плату, НДР, що включаються в собівартість, приведені в таблиці 4.4.

Витрати на основну заробітну плату.

Таблиця 4.3. Витрати на заробітну плату

Виконавець	**Погодинна тарифна ставка, грн**	**Денна ставка, грн**	**Трудомісткість робочих днів**	**Сума основної зарплати, грн**
Дипломник	39,26	314,08	20	6281,60
Керівник	60,00	480	1	480
Консультант по економічній частині	55,00	440	0,25	110
Консультант по охороні праці	55,00	440	0,25	110
Нормоконтроль	55,00	440	0,25	110
Всього (Зо)				7091,60

3) Витрати на додаткову заробітну плату визначаються у відсотках від основної. У наукових закладах додаткова заробітна плата складає 0,1- 0,5 від основної заробітної плати.

Зд=0,1*Зо= 0,1*7091,60 = 709,16 грн

4)До складу собівартості НДР включаються податки, збори і інші обов'язкові платежі, встановлені системою оподаткування що діє. Відрахування до єдиного соціального внеску складає:

Зєсв=0,22*(Зо+Зд) =0,22 * (7091,60 +709,16) = 1716,17 грн

5) До накладних витрат відносять витрати на управління і господарське обслуговування, що відноситься до всіх виконуваних НДР. У наукових закладах накладні витрати складають 40 -120% від основної і додаткової заробітної плати.

Рнакл= (З0+Зд)*0,4 = (7091,60 +709,16)*0,4 =3120,30 грн

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому НДР за формою, приведеною в таблиці 4.4.

Калькуляція планової собівартості

Таблиця 4.4. Стаття витрат

Статті витрат	Сума, грн.
1. Матеріали	160,00
2. Основна заробітна плата	7091,60
3. Додаткова заробітна плата	709,16
4. Відрахування до єдиного соціального внеску	1716,17
5. Накладні витрати	3120,30
Планова собівартість (Спл)	12797,23

Плановий прибуток визначений по формулі:

Ппл = 0,1*Спл =0,1 * 12797,23= 1279,72 грн

Де 0,1 – норматив, який враховує граничний рівень рентабельності, встановлений чинним законодавством для науково-технічної продукції.

Договірна ціна визначається по формулі

Цнір = Спл + Ппл = 12797,23 +1279,72= 14076,96 грн.

Ціну реалізації встановлюємо з урахуванням ПДВ

Цр = Цнір + ПДВ = 14076,96 + 14076,96 * 0,2= 16892,35 грн

ВИСНОВКИ

У процесі роботи над дипломним проектом було поставлено мету розробки гру на веб сайті.

Приступаючи до роботи було проведено передпроектне обстеження для врахування особливостей та вимог до управління даними.

1. Розглянуто існуючу технологію обробки інформації та виявлено її недоліки;
2. Сформульовані цілі та призначення автоматизованого варіанту вирішення комплексу завдань;
3. Формалізовані розрахунки;
4. Зроблено вибір технології проектування та технічного, програмного, інформаційного, а також технологічного забезпечення.
5. Було розроблено інформаційне та програмне забезпечення комплексу завдань, які дозволяють скоротити трудомісткість виконуваних робіт та вартісні витрати на їх виконання.

Систма була розроблена на основі JavaScript з використанням CSS і HTML.

У класичній моделі веб-програми:

1. Користувач заходить на веб-сторінку і натискає на якийсь її елемент;
2. Браузер формує та надсилає запит серверу;
3. У відповідь сервер генерує абсолютно нову веб-сторінку та відправляє її браузеру і т.д., після чого браузер повністю перезавантажує всю сторінку.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Wikipedia. мова JavaScript [Електронний ресурс] / Wikipedia – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/JavaScript.

2. github. мова JavaScript [Електронний ресурс] / github – Режим доступу до ресурсу: http://caustique.github.io/chessboard-js/#/examples/methods/setposition.

3. розробник.mozilla. мова JavaScript [Електронний ресурс] / розробник.mozilla – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/JavaScript/....

4. chessboardjs. мова JavaScript [Електронний ресурс] / chessboardjs – Режим доступу до ресурсу: https://chessboardjs.com/examples.

5. nanoches. мова JavaScript [Електронний ресурс] / nanoches – Режим доступу до ресурсу: https://www.nanochess.org.

6. overcode. мова JavaScript [Електронний ресурс] / overcode – Режим доступу до ресурсу: https://overcoder.net/q/362656/проста-шахова-дошка-javascript.

7. розробник.mozilla. мова JavaScript [Електронний ресурс] / розробник.mozilla – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Learn/CSS/Styling_text/Styling_links.

8. розробник.mozilla. мова JavaScript [Електронний ресурс] / розробник.mozilla – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Learn/JavaScript/First_steps/What_is_JavaScript.

9. habr. мова JavaScript [Електронний ресурс] / habr – Режим доступу до ресурсу: https://qna.habr.com/q/180849.

10. Бойчик І.М Економіка підприємства: підручник. / І.М.Бойчик. – К.: Кондор -Видавництво, 2016.

11. Азарова А. О., Нікіфорова Л.О. Економіка підприємства / А. О. Азарова, Л. О. Нікіфорова. – Вінниця: ВНТУ, 2016.

12. Бедрій Я. І. Охорона праці та пожежна безпека / Я. І. Бедрій. – Тернопіль: Богдан, 2013.

13. Бедрій Я. І. Основи охорони праці / Я. І. Бедрій. – Тернопіль: Богдан, 2014.