МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»
Циклова комісія *комп'ютерних систем та мереж*
(повна назва циклової комісії)

Допустити до захисту
Голова випускової циклової комісії
комп'ютерних систем та мереж
(повна назва циклової комісії)
(підпис) *Ірина КРАВЧУК*
(ім'я, ПРІЗВИЩЕ)
« 10 » 06 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬО-ПРОФЕСІЙНОГО СТУПЕНЯ
ФАХОВИЙ МОЛОДШИЙ БАКАЛАВР

Тема: *Гра у стилі Rex Chrome Dino з використанням сучасних веб-технологій*

Група: *3-012* Спеціальність: *123 «Комп'ютерна інженерія»*

Здобувач освіти (підпис) *Юрій РЯБОШАПКА* (ім'я, ПРІЗВИЩЕ)

Керівник роботи (підпис) *Тетяна РУБАН* (ім'я, ПРІЗВИЩЕ)

Консультант з оформлення пояснювальної записки (підпис) *Оксана ОСАДЧА* (ім'я, ПРІЗВИЩЕ)

Кривий Ріг 2025 р.

КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

Відділення *комп'ютерної та програмної інженерії*
Циклова комісія *комп'ютерних систем та мереж*
Освітньо-професійний ступінь *фаховий молодший бакалавр*
Спеціальність *123 «Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Голова випускової циклової комісії
комп'ютерних систем та мереж
(повна назва циклової комісії)
(підпис) *Ірина КРАВЧУК* (ім'я, ПРІЗВИЩЕ)
« 01 » 03 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ОСВІТИ

Рябошапка Юрій Сергійович
(прізвище, ім'я, по батькові)

1.Тема роботи *Гра у стилі Rex Chrome Dino з використанням сучасних веб-технологій*

Керівник роботи *Рубан Тетяна Миколаївна, викладач першої категорії*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по коледжу від « *04* » *04* 2025 року № *50-ст*

2. Строк подання здобувачем освіти роботи з 01.03 по 10.06

3. Вихідні дані до роботи *Браузерна гра «Rex Chrome Dino»*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) *обґрунтування вибраного напрямку, розробка програмного додатку мовою JavaScript, що дозволить організовувати людині інтелектуальний і водночас емоційний відпочинок від монотонної роботи, розробка алгоритмів та функціональної схеми.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація Microsoft PowerPoint

6. Консультанти розділів роботи (проекту)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	*Узгодження технічного завдання з керівником кваліфікаційної роботи*	*24.03.2025-27.03.2025*	*виконано*
2	*Підбір та вивчення науково-технічної літератури за темою кваліфікаційної роботи*	*28.03.2025-31.03.2025*	*виконано*
3	*Обґрунтування вибору програмних засобів*	*01.04.2025-04.04.2025*	*виконано*
4	*Опис компонентів. Обґрунтування їх вибору*	*05.04.2025-08.04.2025*	*виконано*
5	*Розробка програмного забезпечення*	*09.04.2025-28.04.2025*	*виконано*
6	*Дослідження ефективності реалізованих методів*	*29.04.2025-04.05.2025*	*виконано*
7	*Написання пояснювальної записки*	*12.05.2025-25.05.2025*	*виконано*
8	*Перевірка на плагіат пояснювальної записки*	*26.05.2025 - 01.06.2025*	*виконано*
9	*Попередній захист кваліфікаційної роботи*	*02.06.2025-06.06.2025*	*виконано*
10	*Захист кваліфікаційної роботи*		

Здобувач освіти _____ *Юрій РЯБОШАПКА*
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ *Тетяна РУБАН*
(підпис) (ім'я, ПРІЗВИЩЕ)

StrikePlagiarism.com

Дата звіту 6/10/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Ukrainian national aviation university
Заголовок
КПІ_2025_123 Рябошапка
Автор Науковий керівник / Експерт
РябошапкаРубан Т.
підрозділ
Криворізький Фаховий коледж

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

7.74%
7.74%
КП 1

0.90%
0.90%
КЦ

25
Довжина фрази для коефіцієнта подібності 2

6443
Кількість слів

48616
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	1
Інтервали	A→	0
Мікропробіли	·	38
Білі знаки	ß	0
Парафрази (SmartMarks)	a	39

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

Колір тексту

10 найдовших фраз

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Вебсервіс планування та керування завданнями 6/12/2024 State University of Infrastructure and technology (State University of Infrastructure and technology)	38 0.59 %
2	цфцму 6/17/2023 Uzhhorod National University (Department)	33 0.51 %

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Гра у стилі *Rex Chrome Dino* з використанням сучасних веб-технологій»: 42 сторінок основного тексту, 18 рисунки, 2 таблиці, 2 додатки, 16 використаних джерел.

DINO, БРАУЗЕРНА ГРА, ПРОГРАМУВАННЯ, *JAVASCRIPT, HTML5, CSS3*, ГРАФІКА, АНІМАЦІЯ.

Метою кваліфікаційної роботи є дослідження, проєктування та програмування браузерної гри у стилі «*Rex Chrome Dino*» з використанням сучасних веб-технологій, зокрема *JavaScript, HTML5* та *CSS3*.

У процесі виконання кваліфікаційної роботи були реалізовані основні етапи: аналіз і моделювання ігрового процесу, створення графічного інтерфейсу, а також програмування алгоритмів керування персонажем, генерації перешкод і логіки гри. Гра реалізована на базі *HTML5 Canvas* та *JavaScript*, що забезпечує плавну анімацію, кросбраузерну сумісність і динамічну взаємодію з користувачем.

У третьому розділі представлено структурну схему гри, алгоритм руху персонажа та обробки зіткнень з перешкодами, а також описано структуру проєкту та функціональне призначення основних модулів системи. Додатково розглянуто адаптивну зміну складності гри в залежності від прогресу гравця.

У результаті аналізу теми та технічної реалізації було створено повноцінну браузерну гру, яка запускається без встановлення додаткового програмного забезпечення. Управління здійснюється за допомогою клавіатури, а сама гра підтримує базові сценарії перемоги та поразки. Розроблена гра імітує класичний геймплей *Chrome Dino*, використовуючи переваги сучасних веб-технологій для створення швидкого та стабільного ігрового середовища.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА ТЕРМІНІВ

JavaScript – мова програмування, яка використовується для створення інтерактивних елементів на веб-сторінках.

Visual Studio Code (VS Code) – це редактор початкового коду, створений *Microsoft* із *Electron Framework* для *Windows*, *Linux* і *macOS*. Функції включають підтримку налагодження, підсвічування синтаксису, інтелектуальне завершення коду, фрагменти, рефакторинг коду та

вбудований *Git*.

HTML (англ. HyperText Markup Language – мова розмітки гіпертексту) – стандартизована мова розмітки документів для перегляду вебсторінок у браузері. Браузери отримують *HTML* документ від сервера за протоколами *HTTP*/*HTTPS* або відкривають з локального диска, далі інтерпретують код в інтерфейс, який відображатиметься на екрані монітора.

CSS (англ. Cascading Style Sheets, укр. Каскадні таблиці стилів) – це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних. *CSS* є основною технологією всесвітньої павутини, поряд із *HTML* та *JavaScript*.

ВСТУП

У сучасних умовах стрімкого розвитку веб-технологій браузерні ігри знову набувають популярності як один із найзручніших способів інтерактивної розваги. Завдяки широким можливостям мови програмування *JavaScript,* створення динамічних ігрових застосунків стало простішим і ефективнішим: достатньо лише сучасного веб-браузера, без необхідності встановлення додаткового програмного забезпечення.

Особливої актуальності набуває відтворення популярних аркадних мініігор, які стали частиною цифрової культури. Однією з таких ігор є *Rex Chrome Dino* – проста, але водночас захоплива гра, вбудована у браузер *Google Chrome*, яка стала відомою завдяки своїй доступності в режимі офлайн. Римейк цієї гри дозволяє реалізувати класичний геймплей із сучасним підходом до структурування коду, графіки та *UI*/*UX*-дизайну.

JavaScript, *HTML5* і *CSS3* – як основні інструменти фронтенд-розробки – відкривають широкі можливості для створення двовимірних ігор, що можуть функціонувати на будь-якому пристрої з підтримкою браузера. Завдяки елементу <*canvas*> та подієвій моделі *JavaScript* стає можливим реалізувати плавну анімацію, реакцію на дії користувача та опрацювання колізій між ігровими об'єктами.

Метою даної роботи є створення інтерактивної браузерної гри у стилі «*Rex Chrome Dino*» із застосуванням сучасних веб-технологій, що відтворює

основні принципи геймплею класичної аркадної гри, забезпечуючи зручність використання, адаптивність та цікавість для користувача.

З мети випливають наступні основні завдання, які необхідно реалізувати в процесі розробки гри:

- дослідити ігрову механіку *Dino* та особливості реалізації таких ігор у веб-середовищі;
- проаналізувати сучасні підходи до створення 2D-ігор за допомогою *JavaScript*;

- розробити структуру гри, включаючи основні компоненти: персонаж, перешкоди, лічильник очок, обробка зіткнень;
- реалізувати управління персонажем через клавіатуру та інтерактивні події;
- створити ігрову логіку: підрахунок балів, зміну складності з часом, умови завершення гри;
- забезпечити візуальну частину гри з базовою графікою та анімацією; - оптимізувати гру для коректної роботи у сучасних браузерах. Об’єктом дослідження є процес розробки браузерної гри, що працює з використанням веб-технологій, зокрема *JavaScript*.

Предметом дослідження є методи реалізації *2D*-ігор у браузері, а саме: техніки створення анімацій, обробки зіткнень, інтерактивності та побудови зручного інтерфейсу користувача з урахуванням сучасних принципів *UI*/*UX*.

Таким чином, розробка браузерної гри «*Rex Chrome Dino*» із застосуванням сучасних веб-технологій є актуальним прикладом практичного використання навичок у сфері веб-програмування. Цей проєкт дозволяє поєднати вивчення базових алгоритмів геймдизайну з поглибленням знань у галузі фронтенд-розробки та створенням кросплатформеного продукту, доступного широкому колу користувачів.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналітичний огляд та історія створення комп'ютерних ігор

Комп'ютерні ігри є одним із найдинамічніших і найвпливовіших напрямів сучасних інформаційних технологій. Їх розвиток розпочався ще у середині *XX* століття і пройшов шлях від простих текстових ігор до високотехнологічних інтерактивних середовищ з розширеною реальністю. У 1950–60-х роках з'явилися перші експерименти з іграми як симуляціями: зокрема, у 1958 році Вільям Гігінботам створив гру *Tennis for Two* на осцилографі, а у 1962 році була представлена *Spacewar!* – одна з перших аркадних ігор для комп'ютера *PDP*-1.

У 1970–80-х роках відбулося становлення комерційних аркадних ігор, таких як *Pong*, *Pac-Man* і *Donkey Kong*, а також перших домашніх ігрових консолей, що започаткувало «золоту еру» аркад. Протягом 1990–2000-х років спостерігався стрімкий розвиток персональних комп'ютерів, 3D-графіки та інтернет-сервісів, що дало поштовх до появи жанрів *FPS*, *RTS* та *MMORPG*. Починаючи з 2010-х років, активно розвиваються мобільні, хмарні та браузерні ігри, для створення яких дедалі частіше використовуються вебтехнології, зокрема HTML5, *JavaScript* і *WebGL*, що дає змогу реалізовувати повноцінні ігрові продукти без необхідності їх встановлення на пристрій користувача.

Комп'ютерні ігри класифікують за різними критеріями. За жанром виділяють аркадні ігри, такі як *Tetris* або *Flappy Bird*, платформери на кшталт *Super Mario* чи *Rex Chrome Dino*, стратегії, рольові ігри (*RPG*), симулятори, логічні та пригодницькі проєкти.

За платформою ігри поділяють на ПК-ігри, мобільні, консольні та браузерні. За типом геймплею виділяють однокористувацькі, мережеві та кросплатформні ігри. Сучасні веб-технології суттєво розширили можливості розробників, дозволяючи створювати браузерні ігри, які вже не обмежуються простими анімаціями. Завдяки таким технологіям, як *JavaScript* – основна мова

програмування логіки гри, *HTML5 Canvas* і *WebGL* для рендерингу графіки, *CSS3* для стилізації інтерфейсу, а також *WebSockets* і *API* для забезпечення онлайн-взаємодії в реальному часі – сучасні вебігри можуть конкурувати з класичними програмними продуктами. При цьому користувачам не потрібно завантажувати чи встановлювати ігри, достатньо мати браузер і доступ до Інтернету.

Аркадні ігри, попри простоту, зберігають популярність завдяки швидкому темпу, інтуїтивному керуванню та доступності. Класичні ігри типу *Tetris, Pac-Man, Chrome Dino* отримують нове життя у вигляді браузерних римейків. Вони стають частиною як навчальних курсів з програмування, так і комерційних платформ для розваг.

Гра *Chrome Dino* (або *Dino Runner, T-Rex Game*) – це проста аркадна мінігра, створена компанією *Google* у 2014 році. Вона вбудована у браузер *Google Chrome* і автоматично активується у випадку втрати підключення до Інтернету. Головним героєм є піксельний динозавр *T-Rex*, який біжить по пустелі, уникаючи перешкод у вигляді кактусів та птеродактилів.

Розробка гри була ініційована розробниками *Chrome* як гумористичний спосіб зробити сторінку з повідомленням «*No Internet*» більш цікавою. Дизайн гри навмисно мінімалістичний і натякає на «доісторичний» період – своєрідну алюзію на відсутність Інтернету, як на еру до цифрової еволюції.

Спочатку гра з’явилася у вересні 2014 року у версії *Chrome* 39, однак через технічні недоліки її тимчасово відключили. У грудні того ж року вона була випущена знову у стабільному вигляді. Відтоді вона стала однією з найбільш впізнаваних ігор *Google* і була адаптована для мобільних пристроїв та планшетів.

Основні особливості гри:

1. Простота управління. Гравець керує динозавром за допомогою клавіш *Space*, ↑ або дотику (на мобільних пристроях). Основна механіка – стрибки через кактуси та ухилення від птеродактилів.

2. Поступове ускладнення. З часом швидкість бігу динозавра збільшується, що ускладнює гру. Зі збільшенням відстані також

змінюється колір фону (імітація дня та ночі).

3. Відсутність кінця. Гра не має фіксованого завершення – вона триває, поки гравець не зробить помилку. Такий нескінченний геймплей стимулює змагання з самим собою та покращення результату.
4. Відсутність Інтернету не є обов'язковою. Хоча спочатку гру можна було побачити лише при відключеному Інтернеті, сьогодні її легко запустити вручну, перейшовши за посиланням *chrome://dino/* у браузері *Chrome*.
5. Легка адаптація до будь-яких пристроїв. Завдяки простому дизайну та мінімальному використанню ресурсів гра однаково добре працює на настільних комп'ютерах, ноутбуках, планшетах та смартфонах.

Оригінальна гра реалізована на *JavaScript* з використанням *HTML5 Canvas*, що забезпечує плавну анімацію та обробку подій клавіатури/сенсорного екрана. Її код є частиною внутрішнього програмного середовища *Chrome*, але існує велика кількість римейків на відкритому коді, які імітують геймплей *Dino* у браузерах поза *Chrome*.

Історія комп'ютерних ігор демонструє постійний розвиток технологій, зростання вимог користувачів та креативність розробників. Сьогодні створення гри – це не лише програмування, а й моделювання поведінки, *UI*/*UX*-дизайн, графічне оформлення, оптимізація продуктивності. Зокрема, розробка аркадної гри у стилі *Chrome Dino* на основі *JavaScript* – це актуальний, пізнавальний і практично орієнтований напрямок для вивчення сучасних веб-технологій.

1.2 Огляд існуючих рішень гри *Chrome Dino*

Гра *Chrome Dino*, або *T-Rex Runner*, з моменту свого створення у 2014 році набула великої популярності не лише серед користувачів браузера *Google Chrome*, а й серед веброзробників, які створюють її римейки та модифікації з

метою вдосконалення навичок програмування або дослідження геймплейних механік. У цьому розділі розглянуто найбільш відомі реалізації гри, їх особливості та застосовані технології.

Оригінальна гра *Chrome Dino* від *Google* є вбудованою у браузер *Google Chrome* і доступна за адресою *chrome://dino/*. Вона створена з використанням технологій *JavaScript* та *HTML5 Canvas* і автоматично активується при втраті підключення до Інтернету. Гра має мінімалістичну піксельну графіку (рисунок 1.1) та є нескінченним раннером з поступовим ускладненням геймплею, де головними перешкодами виступають кактуси та птеродактилі. У грі відсутня музика, є лише звукові ефекти, а фон змінюється залежно від часу доби (день/ніч). Ця версія стала еталонною реалізацією, яка надихнула створення численних клонів і модифікацій.

Рисунок 1.1 – Інтерфейс оригінальної гри *Chrome Dino*

Dino Runner – Trex Christmas Game Chrome є тематичним різдвяним римейком класичної гри *Chrome Dino*. Вона створена як розширення для браузера *Google Chrome* і орієнтована на святкову атмосферу, поєднуючи знайомий геймплей з оновленим візуальним оформленням. Гра зберігає класичну механіку нескінченного раннера, у якому користувач керує динозавром, що стрибає через перешкоди, однак додає святкові елементи — замість пустелі тут снігові пейзажі, а замість кактусів і птеродактилів з’являються ялинки, подарунки, сніговики та інші різдвяні атрибути.

Цей варіант також використовує *HTML5*, *JavaScript* і *Canvas API*, що забезпечує плавну графіку та швидку взаємодію прямо в браузері без потреби у встановленні додаткового програмного забезпечення. Візуальний стиль має яскраві кольори, святкові шапки та декоративні елементи, що робить гру привабливою для користувачів усіх вікових категорій, особливо у зимовий період.

Гра не потребує підключення до Інтернету, її можна запускати навіть офлайн через встановлене розширення. *Dino Runner – Trex Christmas Game* є прикладом успішного поєднання ностальгії за класичним геймплеєм із сучасною адаптацією та креативним оформленням, що додає грі унікальності й святкового настрою, що представлено на рисунку 1.2.

Рисунок 1.2 – Інтерфейс оригінальної гри *Trex Christmas Game Chrome*

Однією з популярних адаптацій класичної гри *Chrome Dino* є мобільна гра *T-Rex: Dino Endless Run*, розроблена *KC Creations*™ для платформи *Android* (рисунок 1.3). Це нескінченний раннер, у якому гравець керує динозавром, що автоматично біжить уперед, долаючи перешкоди, зокрема кактуси, за допомогою стрибків. Окрім звичного нескінченного режиму, гра також містить набір рівнів з різними ігровими локаціями, серед яких джунглі, пустеля, місто, Марс та Арктика.

Користувачам доступна система розблокування нових персонажів за внутрішньоігрову валюту, а також змагання в таблиці лідерів. Гра працює в офлайн-режимі та поширюється безкоштовно з підтримкою внутрішньоігрових покупок. Завдяки розширеному функціоналу, додатковим персонажам і системі

досягнень ця версія гри стала привабливою для широкої аудиторії користувачів мобільних пристроїв.

Рисунок 1.3 – Інтерфейс гри *T-Rex: Dino Endless Run*

У таблиці 1.1 представлено характеристики існуючих рішень гри *Chrome Dino*, що демонструють відмінності у графічному оформленні, функціональності, платформах та особливостях геймплею кожної версії.

Таблиця 1.1 – Порівняльні характеристики існуючих рішень гри

Назва гри	**Платформа**	**Технології**	**Особливості геймплею**	**Графіка і звук**	**Додаткові можливості**
Chrome Dino	Вбудована в браузер *Google Chrome*	*HTML5, JavaScript, Canvas*	Нескінченний раннер, поступове ускладнення, кактуси і птеродактилі як перешкоди	Піксельна графіка, звукові ефекти, зміна фону день/ніч	Автоматичне вмикання при втраті Інтернету, натхнення для клонів
Dino Runner – Trex Christma s Game	Розширення для *Google Chrome*	*HTML5, JavaScript, Canvas API*	Нескінченний раннер зі святковими елементами	Яскраві кольори, снігові пейзажі, святкові атрибути	Офлайн-режим, святковий стиль, підходить для всіх вікових груп

T-Rex: *Dino Endless Run*	*Android* (мобільні пристрої)	*HTML5, JavaScript* (опосередковано)	Нескінченний раннер + рівні, різні локації, розблокування персонажів	Багатокольорове оформлення, адаптоване під локації	Система досягнень, таблиця лідерів, внутрішньоігрові покупки

Існуючі реалізації гри *Chrome Dino* демонструють її гнучкість, універсальність та популярність серед користувачів і розробників. Її проста механіка та легка реалізація на основі *JavaScript* і *HTML5* дозволяють ефективно використовувати гру як навчальний приклад або базу для створення унікальних варіацій. Більшість реалізацій фокусуються на оптимізації графіки, удосконаленні геймплею або додаванні штучного інтелекту, що відкриває широкі можливості для експериментів у галузі веброзробки.

1.3 Опис предметної області

Предметна область даної кваліфікаційної роботи – браузерна гра «*Rex Chrome Dino*».

Гра *Chrome Dino* – це браузерна аркада у жанрі нескінченного раннера, розроблена компанією *Google* як частина браузера *Google Chrome*, що представлена на рисунку 1.4. Вона активується автоматично при втраті підключення до Інтернету, слугуючи своєрідним «пасхальним яйцем» та розвагою для користувача під час відсутності мережі.

Основним героєм гри є піксельний динозавр *T-Rex*, який біжить по пустелі, уникаючи перешкод, таких як кактуси та птеродактилі. Гравець керує стрибками динозавра за допомогою клавіші пробілу або натискання на сенсорному екрані, щоб уникнути зіткнення з об'єктами.

Рисунок 1.4 – Основний герой гри

Ігровий процес нескінченний і поступово ускладнюється, збільшуючи швидкість руху. Оформлення гри мінімалістичне, виконане у стилі монохромної піксельної графіки, з плавним переходом між режимами дня і ночі. Завдяки простоті геймплею та доступності, *Chrome Dino* стала однією з

найвідоміших ігор, що демонструють можливості *HTML5* та *JavaScript* у сучасних веббраузерах, а також слугувала основою для створення численних клонів і модифікацій на різних платформах.

1.4 Вибір технологій для розробки

Для створення гри, аналогічної *Chrome Dino*, було обрано сучасні вебтехнології, які забезпечують високу продуктивність, кросплатформеність та зручність розробки. Основними критеріями при виборі технологій стали: простота реалізації, підтримка анімації та взаємодії з користувачем, а також можливість запуску гри без встановлення додаткового програмного забезпечення.

У розробці гри планується використання таких технологій:

- *HTML5* – для структурування сторінки та створення полотна (*Canvas*), на якому відображається графіка гри;
- CSS3 – для стилізації елементів сторінки та базових візуальних ефектів;
- *JavaScript* – як основна мова програмування для реалізації логіки гри, обробки подій, анімації та взаємодії з елементами *Canvas*; - *Canvas API* – для відображення ігрових об’єктів, анімації та реалізації динамічної графіки;
- *Visual Studio Code* – як основне середовище розробки, яке забезпечує

зручну роботу з кодом, розширеннями та відлагодженням.

Обрані технології є універсальними для розробки браузерних *2D*-ігор та дозволяють створити гру, яка буде доступна для запуску на будь-якому сучасному пристрої з веббраузером, без потреби в додаткових інсталяціях.

На рисунку 1.5 представлено схему, яка демонструє взаємодію трьох основних веб-технологій у процесі розробки браузерної гри *Chrome Dino*. У верхній частині схеми розміщено заголовок «*Web Game Development* (*Chrome Dino*)», під яким горизонтально розташовані три ключові компоненти: *HTML5*

(*Canvas*), *CSS* (*Styles*) та *JavaScript* (*Logic*). Кожен з них виконує окрему функцію у створенні гри: *HTML5* відповідає за структуру та графічний вивід через елемент <*canvas*>, *CSS* – за стилізацію та візуальні ефекти, а *JavaScript* реалізує ігрову логіку, взаємодію з користувачем та динамічну поведінку об’єктів. Від кожного блоку відходять стрілки до нижнього елемента з написом «*Output*: *Web Game*», що позначає фінальний результат – інтерактивну гру, яка запускається безпосередньо в браузері. Схема виконана в сучасному стилі з використанням м’якої кольорової палітри (синій, зелений, жовтий), що робить її наочною та зручною для демонстрацій у навчальних матеріалах або презентаціях**.**

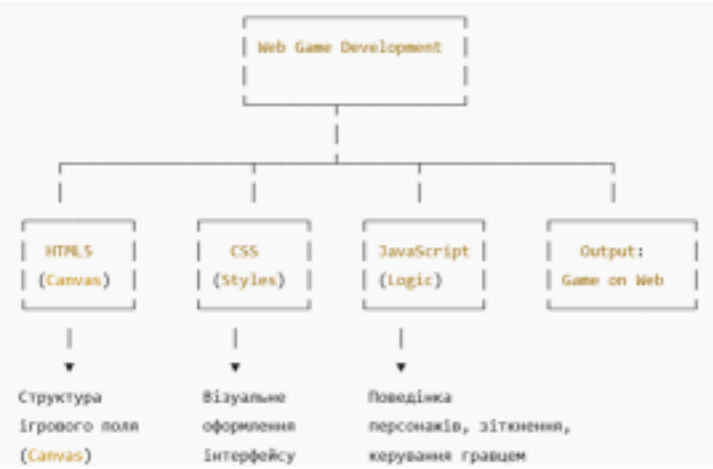

Рисунок 1.5 – Взаємодія ключових веб-технологій у процесі створення браузерної гри

РОЗДІЛ 2

ПРОЕКТУВАННЯ БРАУЗЕРНОЇ ГРИ

2.1 Архітектура гри

Гра *Rex Chrome Dino* має просту, але ефективну архітектуру, яка базується на використанні класичних веб-технологій: *HTML5*, *CSS* та *JavaScript*. Її побудова відповідає принципам клієнтської веб-архітектури, де весь функціонал виконується безпосередньо в браузері користувача. Основні компоненти архітектури описано нижче:

1. *HTML5* (*Canvas*) – основа структури гри. Елемент <*canvas*> використовується для відображення всіх графічних об’єктів – динозавра, перешкод, фону та інших елементів. Полотно оновлюється з кожним кадром, забезпечуючи плавну анімацію.

index.html > ...

```
<!doctype html>
<!-- Вказує, що це документ HTML5 -->
<html>
<head>
    <!-- Встановлення кодування сторінки -->
    <meta charset="utf-8">

    <!-- Адаптивність: ширина екрана, масштаб 1.0, без можливості масштабування -->
    <meta name="viewport" content="width=device-width, initial-scale=1.0,maximum-scale=1.0, user-scal

    <!-- Назва сторінки у вкладці браузера -->
    <title>Chrome Dino – Run Rex!</title>

    <!-- Підключення зовнішнього CSS-файлу для стилів -->
    <link rel="stylesheet" href="index.css">

    <!-- Підключення зовнішнього JS-файлу для логіки гри -->
    <script src="index.js"></script>
</head>
```

Рисунок 2.1 – Фрагмент програмного коду сторінки *index.html*

2. *CSS* – відповідає за базове стилізування інтерфейсу гри (наприклад, фону, положення *canvas*, оформлення кнопок перезапуску), а також може використовуватися для анімацій простих *UI*-елементів.

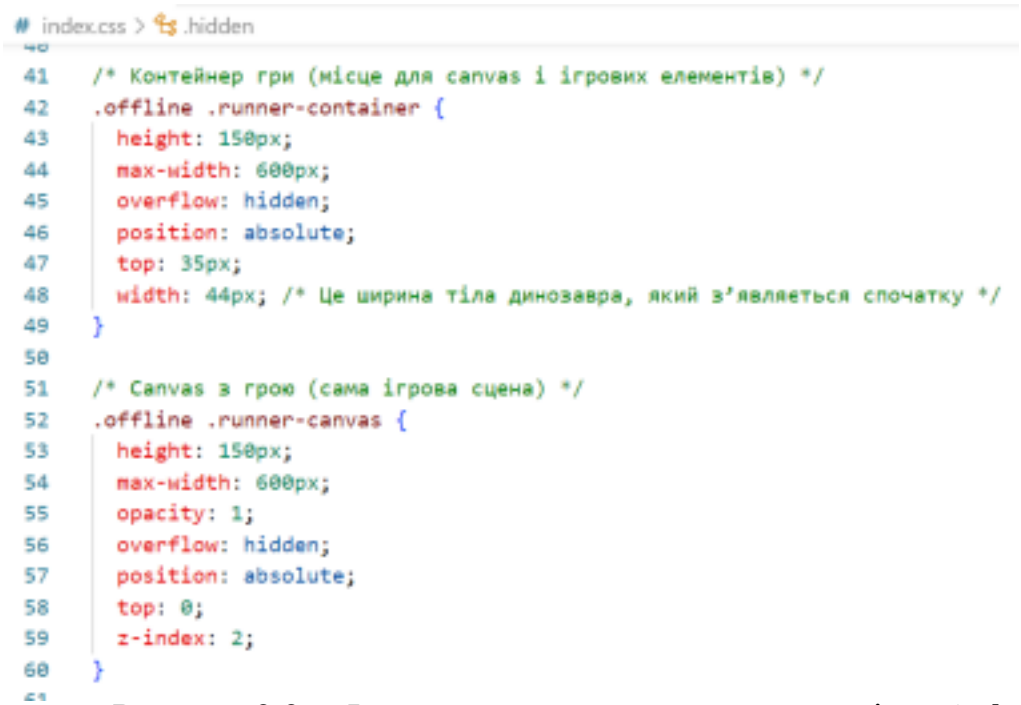

Рисунок 2.2 – Фрагмент програмного коду сторінки *index.css*

3. *JavaScript* – ключовий компонент, який реалізує ігрову логіку. Скрипти відповідають за:

- головний цикл гри (*game loop*) – постійне оновлення стану гри та рендеринг.
- обробку подій – реакцію на натискання клавіш (наприклад, пробіл або стрілка вгору для стрибка).
- фізику гри – стрибок, гравітація, зіткнення з перешкодами. - генерацію перешкод – створення кактусів і птеродактилів з випадковими інтервалами.
- систему очок – нарахування балів за подолані перешкоди та відображення результатів.
- адаптацію складності – поступове збільшення швидкості гри з часом.

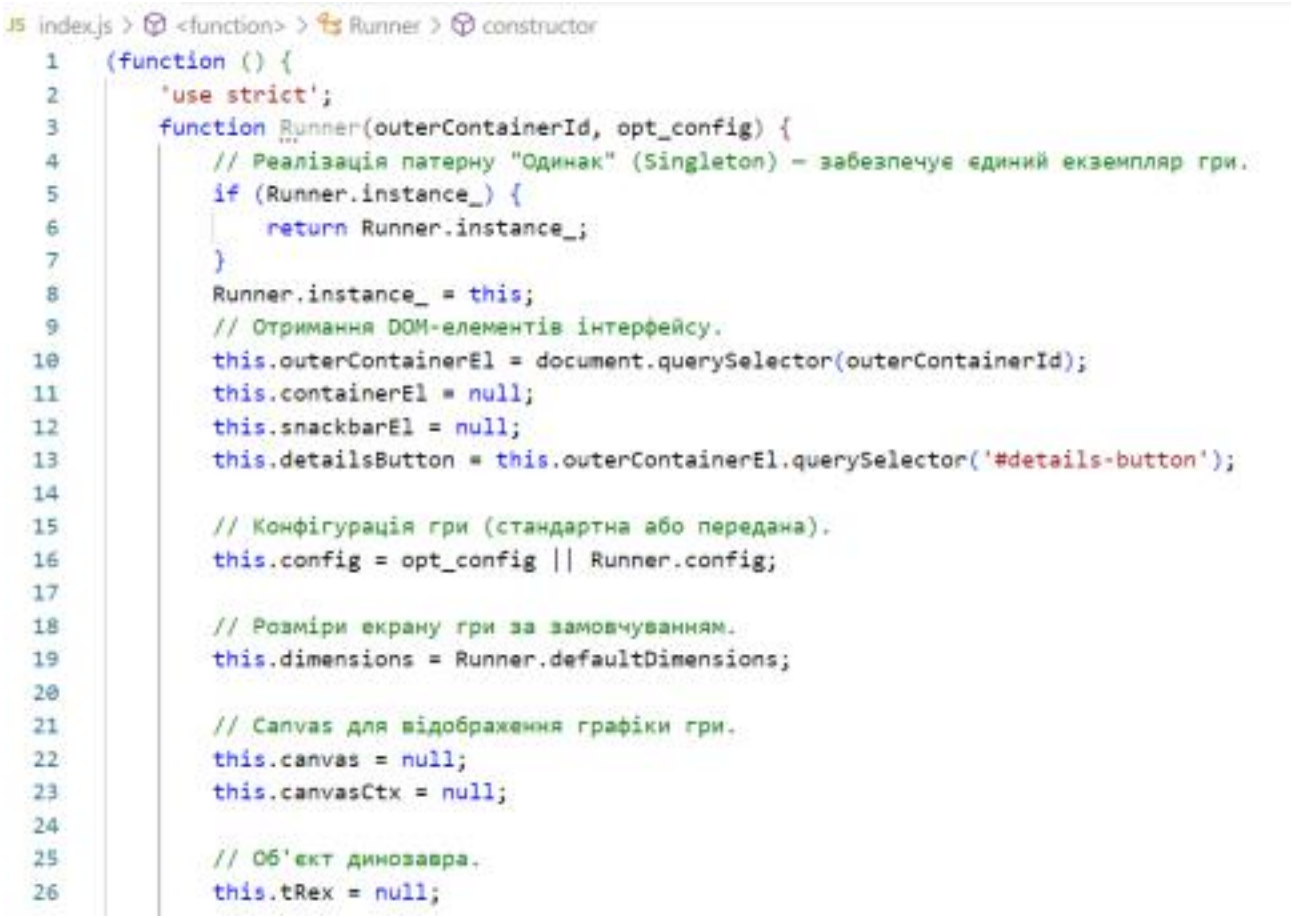

Рисунок 2.3 – Фрагмент програмного коду сторінки *index.js*

4. Аудіо та графіка – мінімалістичні піксельні зображення зберігаються у вигляді спрайтів, які завантажуються під час ініціалізації. Звукові ефекти додаються для підвищення динаміки, але музика у класичній версії відсутня.

5. Вивід результатів – після зіткнення гра зупиняється, і користувачу показується результат (набрана кількість очок) та кнопка для повтору гри. Архітектура гри є легковаговою та незалежною від серверної частини, що дозволяє запускати її навіть в офлайн-режимі. Це робить *Rex Chrome Dino* ефективною демонстрацією можливостей клієнтських веб-технологій для створення інтерактивних ігор у браузері.

2.2 Розробка функціональної схеми

У грі «*Rex Chrome Dino*» головний персонаж може перебувати в кількох станах залежно від дій гравця та ситуації в грі:

1. Очікування – початковий стан перед початком гри.

2. Біг – основний стан під час гри, коли динозавр рухається вперед.

3. Прижок – активується при натисканні клавіші стрибка.

4. Присідання – активується при натисканні клавіші вниз.

5. Зіткнення / Гра закінчена – стан після зіткнення з перешкодою.

6. Перезапуск – повернення до стану очікування.

На рисунку 2.4 представлено логіку переходів між різними станами динозавра в грі *Chrome Dino*. У початковому стані «Очікування» герой перебуває до натискання клавіші запуску гри. Після цього переходить у стан «Біг», який є основним під час проходження рівня. Зі стану бігу можливі переходи до «Стрибка» (при натисканні клавіші «Пробіл») або «Присідання» (при натисканні клавіші «Вниз»).

Кожна з цих дій повертає героя у стан «Біг» після завершення відповідної анімації. У разі зіткнення з перешкодою (кактусом або птахом), незалежно від поточного стану, настає стан «Зіткнення», що завершує гру. Після цього користувач може перезапустити гру, що призводить до переходу в початковий стан «Очікування». Така модель дозволяє чітко простежити послідовність взаємодій та забезпечує ефективну реалізацію ігрової логіки.

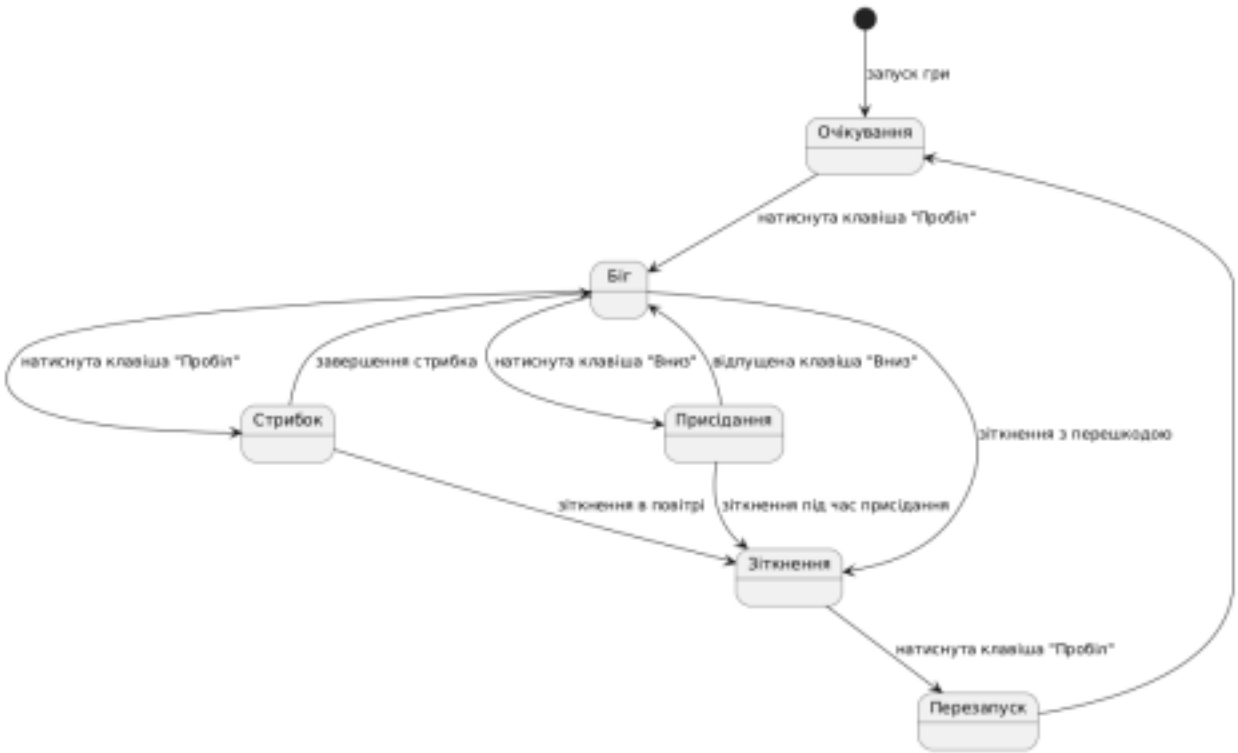

Рисунок 2.4 – Діаграми станів головного персонажа гри

2.3 Вибір та обґрунтування технологій реалізації гри

Під час розробки гри *Rex Chrome Dino* було обрано сучасний і легкий

стек вебтехнологій, до якого входять *HTML5*, *CSS3* та *JavaScript* (*ES6*). Такий вибір зумовлений бажанням забезпечити широку сумісність з браузерами, мінімальні вимоги до ресурсів, швидке завантаження гри, а також простоту розгортання та зручність подальшої підтримки й удосконалення.

HTML5 слугує основою для структурування графічного інтерфейсу гри. За його допомогою створюється ігрове полотно з використанням елемента <*canvas*>, на якому відображаються основні об'єкти: динозавр, перешкоди, лінія землі, тощо. Використання семантичних можливостей *HTML5* сприяє кращій читабельності й підтримуваності коду. Крім того, вбудовані мультимедійні можливості HTML5, такі як <*audio*>, дозволяють реалізовувати звуковий супровід без сторонніх бібліотек.

CSS3 застосовується для оформлення інтерфейсу та елементів керування. Зокрема, стилізація фону, текстових повідомлень (наприклад, про поразку чи перемогу), а також анімаційні ефекти при наведенні чи натисканні кнопок реалізовані за допомогою властивостей *transition*, *animation*, псевдокласів і змінних стилів. Це забезпечує естетичний вигляд гри, при цьому не навантажуючи систему.

JavaScript (ES6) є основним інструментом для реалізації логіки гри. Саме за його допомогою реалізовано:

- обробку клавіш (стрибок динозавра),
- ігровий цикл (*game loop*) з постійним оновленням стану,
- механізм генерації перешкод та перевірку зіткнень,
- динамічне збільшення складності гри залежно від очок,
- систему підрахунку балів та виведення повідомлень про результат.

Застосування сучасних конструкцій *JavaScript* – таких як класи, стрілочні функції, модулі та шаблонні рядки – сприяє чистоті й масштабованості коду, дозволяючи без труднощів доповнювати функціональність чи адаптувати гру під інші потреби.

Обрана технологічна база дозволяє запускати гру на будь-якому пристрої з підтримкою сучасних браузерів – ПК, ноутбуках, планшетах або смартфонах – без потреби в установці додаткового програмного забезпечення. Завдяки цьому гра є портативною, кросплатформною, а також зручно інтегрується у навчальні або демонстраційні вебпортали, слугуючи прикладом використання *JavaScript* у веброзробці.

На рисунку 2.5 представлено діаграму компонентів, що відображає обрані технології для розробки гри *Chrome Dino* та взаємозв'язок між розробником і користувачем.

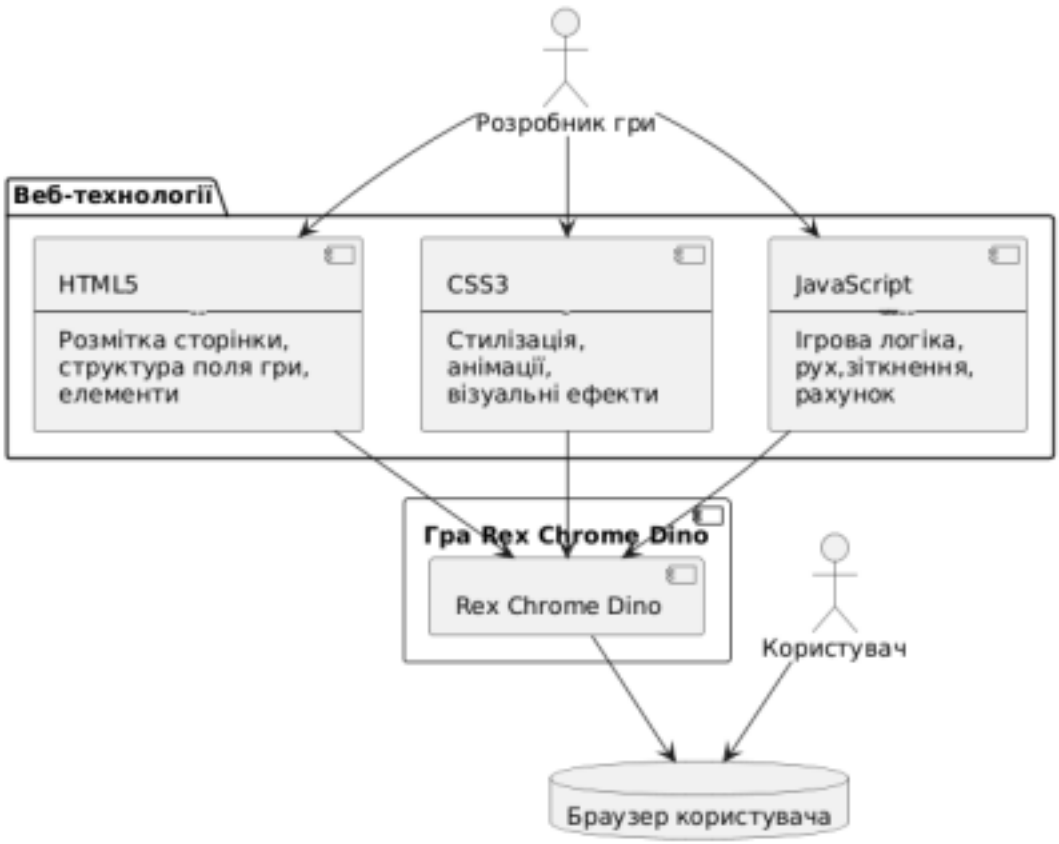

Рисунок 2.5 – Діаграма компонентів вибору технологій гри *Rex Chrome Dino*

Ця діаграма наочно демонструє, як розробник використовує *HTML5* для створення структури сторінки та виведення графічних об'єктів через <*canvas*>, *CSS3* – для стилізації елементів інтерфейсу та додавання базових анімацій, а *JavaScript* (*ES6*) – для реалізації основної ігрової логіки, керування рухом динозавра, генерації перешкод і підрахунку очок. Усі ці технології інтегруються в єдиний вебпроєкт – гру *Chrome Dino*, яка запускається

безпосередньо в браузері. Користувач взаємодіє з грою через інтерфейс

браузера, що забезпечує повну доступність без потреби встановлювати додаткове програмне забезпечення.

Для розробки гри *Chrome Dino* було обрано *Visual Studio Code* (*VS Code*) – сучасне, зручне та безкоштовне середовище розробки від компанії *Microsoft*. Воно підтримує широкий спектр мов програмування, включаючи *HTML5*, *CSS3* та *JavaScript*, що робить його ідеальним інструментом для створення веб застосунків і браузерних ігор.

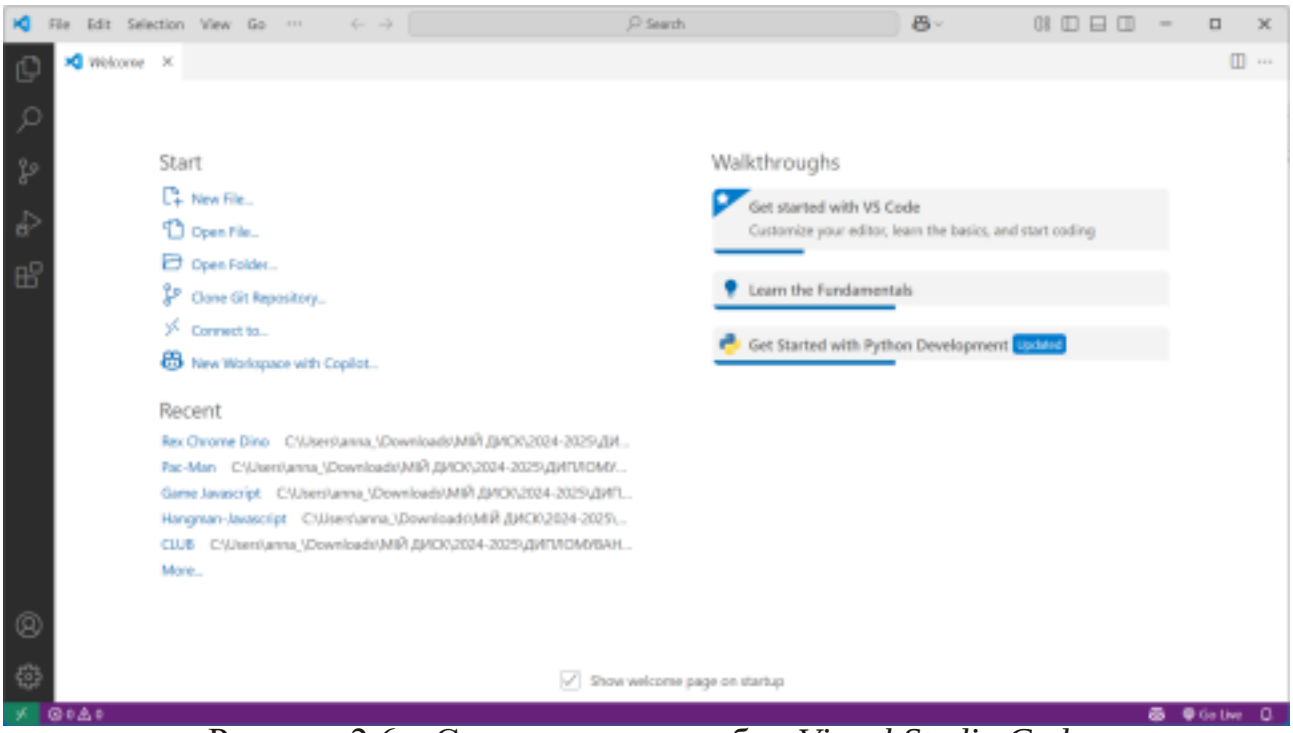

Рисунок 2.6 – Середовище розробки *Visual Studio Code*

Серед основних переваг *Visual Studio Code* варто виділити:

1. Підтримку розширень – користувач може легко встановити додаткові плагіни, які розширюють функціональність редактора (наприклад, *Live Server* для миттєвого перегляду результату у браузері, *Prettier* для форматування коду, *ESLint* для перевірки синтаксису тощо).
2. Підсвічування синтаксису та автодоповнення – інтелектуальні підказки значно пришвидшують написання коду та зменшують кількість помилок.

3. Інтегрований термінал – дає змогу виконувати команди безпосередньо з редактора, що зручно при роботі з системами

керування версіями (наприклад, *Git*).

4. Контроль версій – *VS Code* має вбудовану підтримку *Git*, що дозволяє відстежувати зміни у проєкті та працювати з репозиторіями без сторонніх інструментів.

5. Кросплатформеність – редактор доступний для *Windows, macOS* і *Linux*, що забезпечує гнучкість при виборі операційної системи. Завдяки цим функціям *Visual Studio Code* забезпечує комфортне середовище для розробника, сприяє швидкій реалізації функціоналу гри та полегшує її налагодження й супровід.

Однією з головних переваг *Visual Studio Code* є його багатофункціональність і розширюваність. Розробник може підлаштувати середовище під власні потреби за допомогою великої кількості розширень – від перевірки синтаксису до автоматичного форматування коду. Завдяки вбудованій підтримці *JavaScript, HTML* і *CSS, VS Code* є ідеальним інструментом для створення веб-застосунків та браузерних ігор, таких як *Chrome Dino*. Інтелектуальні підказки, автодоповнення та підсвічування синтаксису значно полегшують написання коду, зменшують кількість помилок і підвищують продуктивність. Ще однією вагомою перевагою є інтегрований термінал, що дозволяє виконувати команди безпосередньо з редактора, а також вбудована підтримка *Git* для керування версіями коду.

Незважаючи на всі переваги, *Visual Studio Code* має і певні недоліки. Зокрема, при великій кількості встановлених розширень програма може споживати значні ресурси системи, що впливає на продуктивність, особливо на слабших комп’ютерах. Крім того, попри зручність, *Visual Studio Code* не є повноцінною *IDE* (інтегрованим середовищем розробки) і може потребувати додаткових налаштувань для проєктів зі складнішою архітектурою. Для новачків деякі функції та параметри розширень можуть бути незрозумілими, що вимагає часу на освоєння. Також варто враховувати, що якість окремих

розширень залежить від їх розробників і не завжди є стабільною або добре підтримуваною.

Таблиця 2.1 – Переваги та недоліки середовища розробки *Visual Studio Code*

Переваги	Недоліки
Велика кількість розширень для підтримки різних мов та фреймворків	Може споживати значні ресурси системи при одночасному використанні багатьох розширень
Вбудована підтримка *HTML, CSS, JavaScript, TypeScript* та інших технологій	Не є повноцінною *IDE*, іноді потребує додаткових налаштувань
Інтелектуальні підказки, автодоповнення, підсвічування синтаксису	Деякі розширення можуть бути нестабільними або погано підтримуватися
Інтегрований термінал, можливість роботи з командним рядком безпосередньо в редакторі	Новачкам може бути складно зорієнтуватися у великій кількості налаштувань та функцій
Вбудована система керування версіями *Git*	Потребує певного часу на налаштування середовища для складніших проєктів
Швидке встановлення та підтримка кросплатформенності (*Windows, macOS, Linux*)	

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Розробка дизайну гри та структура коду гри *Rex Chrome Dino*

Процес створення гри починається з розробки її візуальної складової, яка відіграє важливу роль у сприйнятті та загальному досвіді користувача. У випадку гри у стилі *Rex Chrome Dino* особливу увагу було приділено мінімалістичному дизайну, що поєднує простоту та функціональність. Графічне оформлення є чорно-білим, що не лише наслідує оригінальний стиль гри від *Google*, а й дозволяє зосередити увагу гравця на динаміці та керуванні.

Інтерфейс гри складається з основних елементів: головного персонажа (динозавра), перешкод (кактусів та птахів), лічильника очок, а також

інформаційного екрану після завершення гри. Всі елементи інтерфейсу реалізовані з використанням технологій *HTML5* та *CSS3*, що дозволяє адаптувати зовнішній вигляд гри під різні розміри екранів, включаючи мобільні пристрої.

Дизайн дотримується принципу «*flat design*», що забезпечує чистоту і лаконічність візуального сприйняття. Основна увага приділена зрозумілості ігрового процесу: користувач одразу бачить, як керувати персонажем (наприклад, за допомогою клавіші пробілу або стрілок), а елементи управління зведено до мінімуму.

Для анімації рухів динозавра та перешкод використано *CSS*-анімації та *JavaScript*, що створює ілюзію постійного руху. При цьому плавність анімацій дозволяє досягти приємного візуального ефекту навіть за обмежених обчислювальних ресурсів, що є особливо важливим для гри, яка має працювати в браузері без встановлення додаткових програм.

Таким чином, розробка дизайну гри поєднує естетичну простоту, інтуїтивну взаємодію з користувачем та технічну ефективність, що разом створює позитивний користувацький досвід. Дотримання цього балансу є особливо важливим у реалізації браузерних аркадних ігор.

Таблиця 3.1 – Основні елементи інтерфейсу гри *Rex Chrome Dino*

№	Назва елемента	Опис	Технологія реалізації
1	Головний персонаж	Динозавр, яким керує гравець. Реалізовано анімацію бігу та стрибка.	*HTML5* (*Canvas*), *CSS, JS*
2	Перешкоди	Кактуси та птахи, що з’являються на шляху.	*HTML5* (*Canvas*), *JS*
3	Лічильник очок	Відображає поточний рахунок гравця.	*HTML, CSS, JS*
4	Екран завершення гри	Виводиться при програші, містить рахунок і кнопку перезапуску.	*HTML, CSS, JS*

5	Фоновий ландшафт	Елементи заднього плану: земля, небо.	*CSS*-анімація, *HTML*
6	Кнопки управління	Альтернативне керування для сенсорних пристроїв.	*HTML, JS*

На рисунку 3.1 представлено діаграму послідовностей взаємодії між основними об'єктами гри *Rex Chrome Dino*, яка ілюструє процес гри з моменту її запуску до завершення. У діаграмі задіяні такі учасники: Гравець, Ігровий рушій, Дино (головний персонаж), Перешкода та Система підрахунку очок.

На початку гри гравець ініціює її запуск, натискаючи кнопку «Почати гру». Ігровий рушій у відповідь ініціалізує стартовий стан головного персонажа, скидає очки та запускає генерацію перешкод.

У основному циклі гри ігровий рушій постійно оновлює положення персонажа та перешкод. Коли гравець натискає клавішу стрибка, Дино змінює свій стан на «Стрибає». Рушій перевіряє можливість зіткнення з перешкодами. У разі зіткнення гра завершується, і гравцю відображається екран програшу. Якщо зіткнення не відбулося, система підрахунку очок додає відповідні бали.

Після завершення гри гравець може повторно її запустити, натиснувши кнопку «Спробувати ще», що ініціює перезапуск гри через ігровий рушій. Ця діаграма чітко демонструє логіку взаємодії об'єктів гри та забезпечує розуміння послідовності подій, що відбуваються під час ігрового процесу.

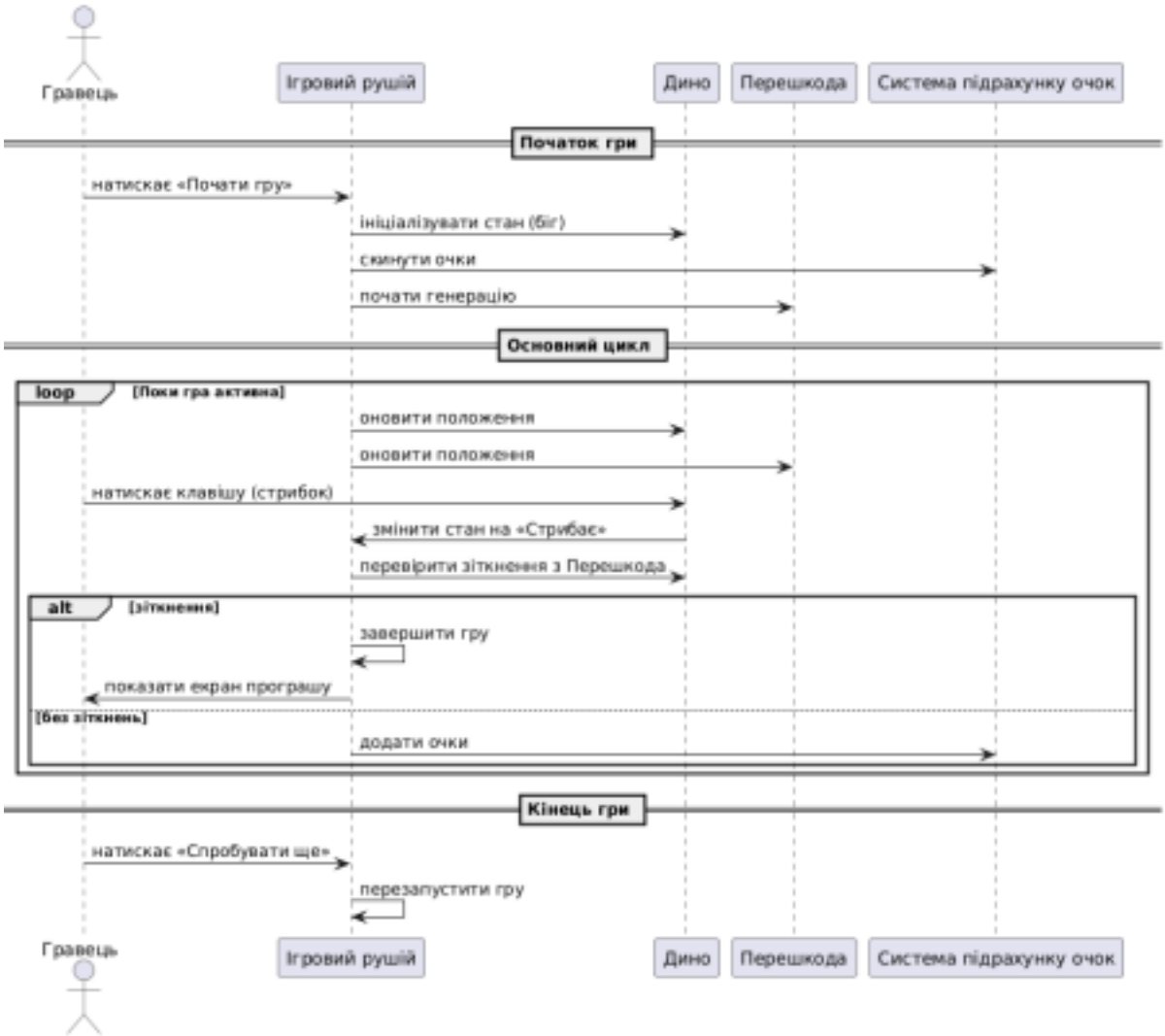

Рисунок 3.1 – Діаграма послідовностей взаємодії між основними об'єктами гри *Rex Chrome Dino*

Варто звернути увагу на структуровану послідовність викликів, що дозволяє розмежувати відповідальність між компонентами:

- Гравець виступає ініціатором подій – він не лише запускає гру, але й взаємодіє з персонажем під час гри через клавіатурні дії (зокрема, стрибки).

- Ігровий рушій виконує роль координатора: він відповідає за зміну станів об'єктів, обробку взаємодії між ними, перевірку зіткнень та керування завершенням гри.

- Персонаж Дино реагує на дії гравця (наприклад, змінює стан на «стрибає») та взаємодіє з перешкодами.

- Об’єкти перешкод рухаються незалежно, але їхнє положення постійно перевіряється рушієм для виявлення зіткнень.

- Система підрахунку очок оновлюється лише у разі, якщо зіткнення не сталося, таким чином реалізуючи мотиваційний механізм гри. - Логіка перезапуску реалізована як окремий сценарій, що повторює ініціалізаційні дії – це дозволяє гравцеві миттєво повернутися до гри після завершення без повного перезавантаження сторінки.

Загалом, діаграма послідовностей забезпечує чітке уявлення про потік подій, дозволяє легко ідентифікувати ролі кожного об'єкта та спростити процес налагодження або масштабування гри, зокрема при розширенні функціональності (додаванні нових станів, елементів або анімацій). Вона є важливим елементом у документації до гри, що дозволяє як розробникам, так і рецензентам краще розуміти внутрішню логіку ігрового процесу.

Структура коду гри *Chrome Dino*, реалізованої з використанням сучасних вебтехнологій (*HTML5*, *CSS3* та *JavaScript*), є логічно впорядкованою та поділеною на окремі складові, що відповідають за різні аспекти функціонування гри.

1. *HTML* (*index.html*). Файл *index.html* визначає структуру вебсторінки, в якій відображається гра. Він містить:

- базову розмітку з оголошенням <!*doctype html*>,

- метатеги для адаптивного відображення на різних пристроях, - посилання на зовнішній файл стилів (*index.css*) та скрипт (*index.js*), - блоки для графічних ресурсів (зображення спрайтів) і шаблон аудіоелементів,

- контейнер гри (*#main-frame-error*), де динамічно відображається інтерфейс гри або повідомлення про помилку.

2. *CSS* (*index.css*). Файл стилів *index.css* відповідає за візуальне оформлення гри. У ньому визначено:

- стилі для фону, динозавра, перешкод, текстів і кнопок,

- анімації руху персонажа та об’єктів за допомогою ключових кадрів (*@keyframes*),
- адаптивність і стильову узгодженість з темою офлайн-режиму браузера.

3. *JavaScript* (*index.js*). Файл *index.js* містить основну ігрову логіку: - визначення об’єктів гри (динозавр, перешкоди, рахунок),

- обробка подій клавіатури (стрибок при натисканні клавіші), - реалізація циклу оновлення кадрів (рух, зіткнення, генерація об’єктів),
- механізм обробки зіткнень і завершення гри,
- логіка підрахунку очок і відображення результатів.

Код організовано у вигляді функцій та об’єктів, що забезпечує модульність і легкість підтримки. Загалом структура коду побудована за принципами розділення відповідальності (структура – *HTML*, стиль – *CSS*, логіка – *JS*), що дозволяє ефективно підтримувати та розширювати гру.

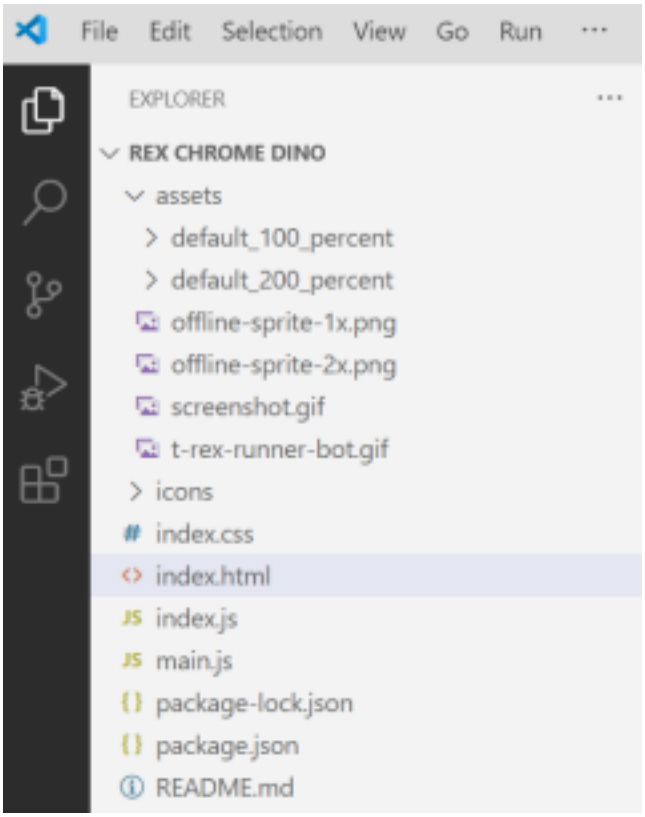

Рисунок 3.2 – Структура коду гри *Rex Chrome Dino*

3.2 Алгоритм розробки програми

На рисунку 3.3 представлено алгоритм гри *Rex Chrome Dino* у вигляді діаграми діяльності. Ця діаграма демонструє послідовність дій, які виконуються під час гри, починаючи з моменту запуску й до завершення гри внаслідок зіткнення з перешкодою.

Рисунок 3.3 – Алгоритм гри *Rex Chrome Dino* у вигляді діаграми діяльності

У верхній частині діаграми зображено початок гри, що включає ініціалізацію основних компонентів: встановлення початкового стану динозавра як «біжить», обнулення лічильника очок та запуск генерації перешкод.

Основний цикл гри представлено у вигляді блоку повторення (*loop*), в якому постійно відбувається:

- оновлення положення динозавра та перешкод,

- перевірка на натискання клавіші користувачем (для виконання стрибка),

- перевірка на зіткнення між динозавром і перешкодою.

У разі зіткнення гра завершується, і відображається екран поразки. Якщо зіткнення не відбулося, система нараховує очки за подолання перешкод. Завершується діаграма дією «Показати екран завершення», яка позначає кінець гри. Такий підхід до візуалізації алгоритму дозволяє наочно продемонструвати логіку роботи гри, що є корисним як для розробників, так і для аналізу в рамках навчального процесу.

3.3 Інструкція користувача

У цьому розділі наведено інструкцію користувача до гри *Rex Chrome Dino*, яка дозволяє швидко ознайомитися з правилами, елементами керування, принципами роботи гри та особливостями інтерфейсу. Гра реалізована у вигляді браузерної сторінки, що не потребує встановлення додаткового програмного забезпечення та є доступною для запуску на будь-якому сучасному пристрої з веббраузером.

Користувач взаємодіє з грою через клавіатуру. Основна мета – допомогти головному персонажу, динозавру, уникати перешкод, таких як кактуси та птеродактилі, за допомогою стрибків або присідань. У процесі гри нараховуються очки за подолані відстані та уникнення зіткнень. При досягненні певних порогів швидкість руху перешкод поступово збільшується, що робить гру дедалі складнішою.

Успішне користування грою передбачає ознайомлення з основними елементами управління, розуміння механіки гри та можливостей повторного запуску після програшу. У наступному підрозділі наведено детальну покрокову інструкцію щодо використання гри.

На рисунку 3.4 представлено інтерфейс гри *Rex Chrome Dino* на

початковому етапі, одразу після завантаження сторінки. У центрі екрана розміщено головного персонажа – динозавра, який стоїть на стартовій позиції. Фон гри виконано у мінімалістичному стилі, що імітує пустелю, з горизонтальною лінією землі та можливими хмарами.

У верхній частині ігрового поля згодом з’являється лічильник очок, який починає працювати після старту. На цьому етапі гра очікує на дію користувача – перше натискання клавіші (пробіл або стрілка вгору), після чого динозавр починає біг, запускається генерація перешкод, і розпочинається основний ігровий цикл.

Інтерфейс вирізняється своєю простотою та зручністю: відсутність зайвих елементів дозволяє користувачу зосередитися на процесі гри. Такий підхід особливо ефективний для демонстрації принципів *HTML5*-ігор та взаємодії з клавіатурою в середовищі веббраузера.

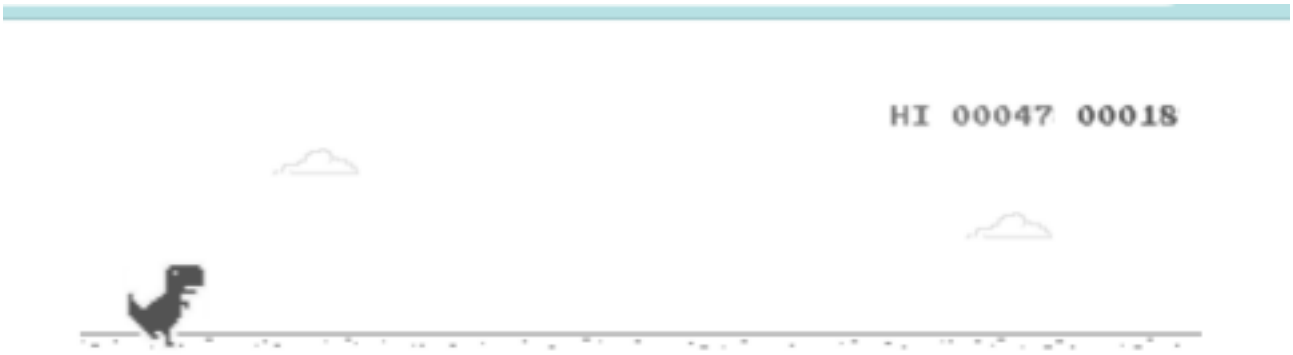

Рисунок 3.4 – Інтерфейс гри *Rex Chrome Dino* на початковому етапі

На рисунку 3.5 представлено можливість управління персонажем гри *Rex Chrome Dino*, зокрема – виконання дії присідання за допомогою натискання

клавіші стрілка вниз. У цьому стані головний герой динозавр змінює своє положення: з положення стоячи переходить у нахилену позу, що дозволяє уникати перешкод, які з'являються на певній висоті (наприклад, птахів).

Анімація присідання реалізується за допомогою зміни спрайту, а також модифікації висоти й положення персонажа на ігровому полі. Це надає грі динамічності та додає новий рівень складності, оскільки користувач має

реагувати не лише на кактуси, а й на інші типи загроз.

Функціональність клавіші «вниз» є інтуїтивно зрозумілою для користувача та відповідає загальноприйнятим схемам управління в аркадних іграх. Така інтеракція розширює можливості гравця, роблячи ігровий процес більш захопливим та інтерактивним.

Рисунок 3.5 – Анімація присідання головного героя

На рисунку 3.6 представлено процес анімації стрибка головного героя гри. Ця дія є ключовою для уникнення перешкод, зокрема кактусів, які розташовані на землі. Стрибок активується натисканням клавіші стрілка вгору або пробіл, що є стандартним методом взаємодії в більшості аркадних платформерів.

Під час виконання стрибка персонаж змінює своє положення по вертикалі, відтворюється відповідна анімація, яка забезпечує плавний рух вгору та вниз. Завдяки зміні координат у просторі *canvas* та оновленню кадрів гри, створюється ілюзія реального фізичного руху. Крім того, стрибок

супроводжується звуковим ефектом, що додає динамічності і підвищує рівень занурення в гру.

Анімація реалізується через *JavaScript* шляхом зміни властивостей об’єкта героя в ігровому циклі. Це дає змогу контролювати висоту стрибка, швидкість підйому й падіння, а також обробляти можливі зіткнення з перешкодами навіть під час перебування в повітрі.

Таким чином, анімація стрибка – це не лише візуальний ефект, а й важливий функціональний компонент ігрової механіки, який безпосередньо впливає на ігрову стратегію та реакцію користувача.

Рисунок 3.6 – Анімація стрибка головного героя

Гра *Rex Chrome Dino* має поступове ускладнення, яке реалізується через зміну ключових параметрів під час проходження. Основні фактори, що впливають на складність гри, – це зростання швидкості, поява нових типів перешкод, а також непередбачуваність їх розміщення.

Із самого початку швидкість руху ігрового середовища відносно динозавра є помірною, що дозволяє гравцеві звикнути до управління. Проте в міру набору очок швидкість поступово зростає. Це відбувається за рахунок зменшення інтервалів між оновленнями положення об’єктів на *canvas*. Внаслідок цього:

- динаміка гри стає стрімкішою;

- вимагається швидша реакція гравця;

- зменшується час на прийняття рішень (стрибок або присідання).

Це підвищує рівень напруги та зацікавленості, характерний для аркадного типу ігор. На рисунку 3.7 представлений підвищений рівень з появою птахів.

Рисунок 3.7 – Інтерфейс гри з появою птахів

На певному етапі гри, коли гравець набирає достатньо очок, до стандартних перешкод (кактусів) додається новий тип об’єкта – птахи. Вони з’являються у повітрі на різній висоті та рухаються горизонтально з тією ж швидкістю, що й інші перешкоди. Їх поява має такі особливості:

- гравець має реагувати не лише на землю, а й слідкувати за повітряним простором;
- для уникнення зіткнення з птахом іноді потрібно присісти, якщо він летить низько;
- складність значно зростає, коли кактуси та птахи йдуть у комбінації.

Рисунок 3.8 – Інтерфейс гри

Таким чином, з кожною хвилиною гри *Rex Chrome Dino* перетворюється з

простої розваги в випробування на уважність, реакцію та витривалість. Збільшення швидкості та поява нових перешкод – це основні механізми ескалації складності, які утримують інтерес гравця і створюють мотивацію до встановлення нових рекордів.

На рисунку 3.9 представлений скріншот завершення гри, який з’являється у разі зіткнення головного героя з перешкодою – кактусом або птахом. Після фатального зіткнення ігровий процес зупиняється, і на екрані з’являється повідомлення про поразку, а також зображення ігрової сцени в момент помилки.

Користувач бачить іконку «*Game Over*», яка сигналізує про закінчення поточної спроби, а також поточний рахунок, що дозволяє оцінити власний результат. Під цією інформацією зазвичай розміщується кнопка перезапуску гри, що дозволяє гравцеві розпочати нову спробу без перезавантаження сторінки. Такий підхід забезпечує зручність повторного проходження та мотивацію до покращення власних результатів.

Рисунок 3.9 – Скріншот завершення гри

ВИСНОВОК

Кваліфікаційна робота присвячена розробці програмного забезпечення, яке реалізує браузерну гру «*Rex Chrome Dino*» з використанням сучасних вебтехнологій. Додаток дозволяє організувати динамічне дозвілля користувача, забезпечуючи при цьому швидкий доступ до гри без встановлення додаткового програмного забезпечення.

В аналітичній частині було розглянуто особливості існуючих ігор цього

типу, зокрема популярну гру *Chrome Dino*, вбудовану в браузер *Google Chrome*. У межах роботи було розроблено функціональну структуру гри, описано архітектуру, створено діаграми діяльності, станів та послідовностей. Було обґрунтовано вибір інструментів розробки, серед яких – середовище *Visual Studio Code* і мова програмування *JavaScript*, у поєднанні з *HTML5* і *CSS3*.

У результаті виконання кваліфікаційної роботи було розроблено браузерну гру у стилі *Chrome Dino*, що функціонує безпосередньо у веббраузері без потреби встановлення додаткового програмного забезпечення. Для реалізації гри було використано сучасні вебтехнології – *HTML5*, *CSS3* та *JavaScript* (*ES6*). Кожна з технологій виконувала важливу роль: *HTML5* забезпечував структуру сторінки та графіку через елемент <*canvas*>, *CSS3* відповідав за візуальне оформлення й анімації, а *JavaScript* – за реалізацію ігрової логіки та взаємодії з користувачем.

У процесі проєктування було розглянуто архітектуру гри, побудовано діаграми станів, послідовностей і діяльності, що дозволило формалізувати логіку взаємодії між об'єктами гри: головним персонажем, перешкодами, підрахунком очок тощо. Було створено простий і зручний користувацький інтерфейс, що адаптується до різних пристроїв.

Гра передбачає поступове зростання складності: збільшення швидкості, появу нових типів перешкод (зокрема птахів), що стимулює гравця розвивати реакцію та стратегічне мислення. Особливу увагу приділено оптимізації коду та зручності подальшого розширення функціоналу.

Отже, розроблений застосунок є наочним прикладом реалізації інтерактивного вебпроєкту, який може використовуватись як в освітніх цілях (для навчання основам розробки ігор та роботи з *JavaScript*), так і для демонстрації можливостей *HTML5* у створенні динамічних ігор.

У перспективі розробка гри може бути продовжена в кількох напрямках:

1. Розширення функціоналу гри – додавання нових типів перешкод, бонусів, рівнів складності, системи досягнень або кількох режимів гри (наприклад, нічний режим або мультиплеєр).
2. Адаптація під мобільні пристрої – оптимізація гри для сенсорного керування, зокрема реалізація свайпів для стрибка чи присідання,

додавання повноекранного режиму та адаптивної графіки.

3. Збереження статистики гравця – впровадження механізму збереження рекордів, ігрових сесій або авторизації користувача (наприклад, через локальне сховище браузера або базу даних).
4. Підключення звукових ефектів та музики – розширення звукового супроводу гри для підвищення занурення користувача в ігровий процес.
5. Публікація гри у вигляді *PWA* (*Progressive Web App*) – створення прогресивного вебдодатку з можливістю встановлення гри на робочий стіл або мобільний пристрій безпосередньо з браузера.

Таким чином, проєкт має потенціал до масштабування та модернізації, що робить його не лише навчальним прикладом, а й платформою для подальшого вдосконалення знань у сфері веброзробки та геймдизайну.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дакетт, Дж. *HTML* і *CSS*. Розробка та дизайн веб-сайтів / Джон Дакетт ; пер. з англ. – К. : Видавництво «Пітер», 2017. – 512 с.
2. Флэнаган, Д. *JavaScript*. Повне керівництво / Девід Флэнаган ; пер. з англ. – 6-е вид. – К. : Діалектика, 2020. – 1104 с.
3. Лабудін, О. Ю. Розробка веб-додатків з використанням HTML5, CSS3, *JavaScript* / О. Ю. Лабудін, А. В. Іванов. – Харків : ХНУРЕ, 2021. – 240 с. 4. Фріман, Е. HTML5 і *JavaScript* для створення ігор / Е. Фріман, Е. Робсон. – Львів : Видавництво «БХВ», 2018. – 480 с.
5. Матвєєв, О. О. Веб-програмування. Основи *HTML, CSS, JavaScript* / О. О. Матвєєв. – К. : НАУ-друк, 2020. – 272 с.
6. *Mozilla Developer Network* (*MDN*). HTML5 *Reference*. [Електронний ресурс] Режим доступу: *https://developer.mozilla.org/en US/docs/Web/HTML* (дата звернення: 05.05.2025)
7. *Mozilla Developer Network* (*MDN*). CSS3 *Reference*. [Електронний ресурс] Режим доступу: *https://developer.mozilla.org/en-US/docs/Web/CSS* (дата звернення: 01.05.2025)

8. *Mozilla Developer Network* (*MDN*). *JavaScript Guide*. [Електронний ресурс] Режим доступу: *https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide* (дата звернення: 01.05.2025)
9. W3Schools. HTML5 *Canvas*. [Електронний ресурс] Режим доступу: *https://www.w3schools.com/html/html5_canvas.asp* (дата звернення: 05.05.2025)
10. W3Schools. *JavaScript Game Tutorial*. [Електронний ресурс] Режим доступу: *https://www.w3schools.com/graphics/game_intro.asp* (дата звернення: 05.05.2025)
11. *Visual Studio Code Documentation*. [Електронний ресурс] Режим доступу: *https://code.visualstudio.com/docs* (дата звернення: 05.05.2025)
12. *FreeCodeCamp*. *How to Build a Chrome Dino Game Clone with JavaScript*. [Електронний ресурс] Режим доступу: *https://www.freecodecamp.org/news/javascript-game-tutorial/* (дата звернення: 05.05.2025)
13. *Stack Overflow*. *JavaScript Collision Detection*. [Електронний ресурс] Режим доступу: *https://stackoverflow.com/questions/2440377/javascript-collision-detection* (дата звернення: 05.05.2025)
14. *YouTube*. *Build a Dino Game using JavaScript | Tutorial*. [Електронний ресурс] Режим доступу: *https://www.youtube.com/watch?v=4YQ4svkETS0* (дата звернення: 01.05.2025)
15. *PlantUML Documentation*. [Електронний ресурс] Режим доступу: *https://plantuml.com/* (дата звернення: 01.05.2025)
16. HTML5 *Game Devs Forum*. [Електронний ресурс] Режим доступу: *https://www.html5gamedevs.com/* (дата звернення: 01.05.2025)

ДОДАТОК А

Код програми головного модуля

```
<!doctype html>
<!-- Вказує, що це документ HTML5 -->
```

```
<html>
<head>
   <!-- Встановлення кодування сторінки -->
   <meta charset="utf-8">
<!-- Адаптивність: ширина екрана, масштаб 1.0, без можливості масштабування
-->
   <meta name="viewport" content="width=device-width, initial
scale=1.0,maximum-scale=1.0, user-scalable=no">
   <!-- Назва сторінки у вкладці браузера -->
   <title>Chrome Dino — Run Rex!</title>
      <!-- Підключення зовнішнього CSS-файлу для стилів -->
   <link rel="stylesheet" href="index.css">
      <!-- Підключення зовнішнього JS-файлу для логіки гри -->
   <script src="index.js"></script>
</head>
<body id="t" class="offline">
   <!-- Основний контейнер для повідомлення про помилку або гру -->
   <div id="main-frame-error" class="interstitial-wrapper">
      <!-- Контейнер з іконкою (наприклад, динозавром) -->
      <div id="main-content">
         <div class="icon icon-offline" alt=""></div>
      </div>
      <!-- Блок з ресурсами для офлайн-режиму (зображення і звуки) -->
      <div id="offline-resources">
         <!-- Зображення спрайтів для нормального та ретіна-екранів -->
```



```
         <img id="offline-resources-1x" src="assets/default_100_percent/100-offline
sprite.png">
         <img id="offline-resources-2x" src="assets/default_200_percent/200-offline
sprite.png">
         <!-- Шаблон елементів аудіо (не вставляється напряму у DOM до
активації) -->
```

```
        <template id="audio-resources">
          <!-- Звуки для гри: натискання, зіткнення, досягнення мети --> <audio
        id="offline-sound-press" src="data:audio/mpeg;basAA"></audio> <audio
        id="offline-sound-hit" src="data:audio/mpeg;base64QA"></audio> <audio
        id="offline-sound-reached" src="data:audio mpeasAA"> </audio>
        </template>
      </div>
    </div>
</body>
</html>
```

Програмний код файлу стилів *index.css*

```
/* Скидання зовнішніх відступів і установка розмірів на весь екран */
html, body {
  padding: 0;
  margin: 0;
  width: 100%;
  height: 100%;
}
/* Іконка — елемент, який не можна виділити */
.icon {
  -webkit-user-select: none;
  user-select: none;
  display: inline-block;
}

/* Іконка в режимі офлайн — задає графіку через image-set */
.icon-offline {
  content: -webkit-image-set(
    url(assets/default_100_percent/100-error-offline.png) 1x,
```

```
    url(assets/default_200_percent/200-error-offline.png) 2x
  );
  position: relative;
 }
/* Схований елемент (наприклад, при завантаженні або помилках) */
.hidden {
  display: none;
 }
/* Обгортка офлайн-сторінки */
.offline .interstitial-wrapper {
  color: #2b2b2b;
  font-size: 1em;
  line-height: 1.55;
  margin: 0 auto;
  max-width: 600px;
  padding-top: 100px;
  width: 100%;
 }
/* Контейнер гри (місце для canvas і ігрових елементів) */
.offline .runner-container {
  height: 150px;
  max-width: 600px;
  overflow: hidden;
  position: absolute;
  top: 35px;
```



```
  width: 44px; /* Це ширина тіла динозавра, який з’являється спочатку */
 }
/* Canvas з грою (сама ігрова сцена) */
.offline .runner-canvas {
  height: 150px;
  max-width: 600px;
```

```
  opacity: 1;
  overflow: hidden;
  position: absolute;
  top: 0;
  z-index: 2;
 }
/* Контролер для мобільного сенсорного управління (поверх гри) */
.offline .controller {
  background: rgba(247, 247, 247, .1);
  height: 100vh;
  left: 0;
  position: absolute;
  top: 0;
  width: 100vw;
  z-index: 1;
 }
/* Схований блок з ресурсами (зображення чи аудіо у вигляді тегів) */
#offline-resources {
  display: none;
 }
/* Для екранів шириною до 420px (зазвичай мобільні телефони)
 */ @media (max-width: 420px) {
  /* Центрує кнопки управління */
  .suggested-left > #control-buttons, .suggested-right > #control-buttons
   { float: none;
```



```
  }
  /* Повідомлення (snackbar) розтягується на всю ширину внизу */
  .snackbar {
    left: 0;
    bottom: 0;
    width: 100%;
```

```
    border-radius: 0;
  }
 }
/* Для екранів з дуже маленькою висотою (менше 350px) — наприклад, в ландшафтній орієнтації на старих телефонах */
 @media (max-height: 350px) {
  h1 {
    margin: 0 0 15px;
  }
  .icon-offline {
    margin: 0 0 10px;
  }
  .interstitial-wrapper {
    margin-top: 5%;
  }
  .nav-wrapper {
    margin-top: 30px;
  }
 }
/* Для планшетів або широких телефонів у горизонтальній орієнтації */
@media (min-width: 600px) and (max-width: 736px) and (orientation: landscape) {
.offline .interstitial-wrapper {
    margin-left: 0;
    margin-right: 0; }
 } /* Для невеликих екранів у горизонтальній орієнтації */
```



```
 @media (min-width: 420px) and (max-width: 736px) and (min-height: 240px) and (max-height: 420px) and (orientation:landscape) {
  .interstitial-wrapper {
    margin-bottom: 100px;
  } }
/* Для екранів з висотою від 240px у горизонтальному режимі
```

```
*/ @media (min-height: 240px) and (orientation: landscape) {
  .offline .interstitial-wrapper {
    margin-bottom: 90px;
  }
  .icon-offline {
    margin-bottom: 20px;
  } }
/* Для дуже маленької висоти (до 320px) у горизонтальному режимі */
@media (max-height: 320px) and (orientation: landscape) {
  .icon-offline {
    margin-bottom: 0;
  } .offline .runner-container { top: 10px;
  }
} /* Для надзвичайно вузьких екранів (до 240px шириною)
*/ @media (max-width: 240px) {
  .interstitial-wrapper {
    overflow: inherit;
    padding: 0 8px;
  }
}
```


ДОДАТОК Б

Програмний код файлу сторінки *index.js*

```
(function () {
    'use strict';
     /**
     * Основний клас гри T-Rex Runner.
     * @param {string} outerContainerId — ID зовнішнього контейнера для
     гри. * @param {Object} opt_config — необов’язкова конфігурація гри. *
     @constructor
```

```
   * @export
   */
  function Runner(outerContainerId, opt_config) {
// Реалізація патерну "Одинак" (Singleton) — забезпечує єдиний екземпляр гри.
    if (Runner.instance_) {
      return Runner.instance_;
    }
    Runner.instance_ = this;
    // Отримання DOM-елементів інтерфейсу.
    this.outerContainerEl =
    document.querySelector(outerContainerId); this.containerEl = null;
    this.snackbarEl = null;
    this.detailsButton = this.outerContainerEl.querySelector('#details-
    button'); // Конфігурація гри (стандартна або передана).
    this.config = opt_config || Runner.config;
    // Розміри екрану гри за замовчуванням.
    this.dimensions = Runner.defaultDimensions;
    // Canvas для відображення графіки гри.
    this.canvas = null;
    this.canvasCtx = null;
```



```
    // Об'єкт динозавра.
    this.tRex = null;
    // Відстань, яку пробіг динозавр.
    this.distanceMeter = null;
    this.distanceRan = 0;
    // Найвищий результат гри.
    this.highestScore = 0;
    // Таймери та швидкість.
    this.time = 0;
    this.runningTime = 0;
```

```
        this.msPerFrame = 1000 / FPS; // Тривалість одного кадру.
        this.currentSpeed = this.config.SPEED; // Початкова швидкість гри.
        // Масив перешкод на шляху динозавра.
        this.obstacles = [];
        // Стани гри.
        this.activated = false; // Чи активована гра (пасхалка).
        this.playing = false; // Чи запущена гра.
        this.crashed = false; // Чи динозавр зіткнувся з
        перешкодою. this.paused = false; // Чи гра на паузі.
        this.inverted = false; // Чи змінено кольори екрану.
        this.invertTimer = 0;
        this.resizeTimerId_ = null;
        this.playCount = 0; // Кількість запусків гри.
        // Звукові ефекти.
        this.audioBuffer = null;
        this.soundFx = {};
        // Аудіоконтекст для відтворення звуків.
        this.audioContext = null;
        // Зображення для гри.
        this.images = {};
```



```
        this.imagesLoaded = 0;
        // Якщо гра відключена (наприклад, на мобільних) — ініціалізувати
заглушку.
        if (this.isDisabled()) {
            this.setupDisabledRunner();
        } else {
            // Інакше — завантажити зображення.
            this.loadImages();
        }
    }
    // Робимо конструктор Runner глобально доступним
```

window['Runner'] = *Runner*;

/*Стандартна ширина ігрового поля */

var DEFAULT_WIDTH = 600;

/*Кількість кадрів на секунду (*FPS*) */

var FPS = 60;

/*Визначення, чи екран високої роздільності (*HiDPI* / *Retina*) */

var IS_HIDPI = *window*.*devicePixelRatio* > 1;

/*Перевірка, чи запущено на пристрої iOS */

var IS_IOS = /*iPad*|*iPhone*|*iPod*/.*test*(*window*.*navigator*.*platform*);

/*Перевірка, чи гра запущена на мобільному пристрої (*Android* або *iOS*) */

var IS_MOBILE = /*Android*/.*test*(*window*.*navigator*.*userAgent*) || *IS_IOS*; /*

Перевірка, чи пристрій підтримує сенсорний екран */

var IS_TOUCH_ENABLED = 'ontouchstart' *in window*;

/*Стандартна конфігурація гри */

Runner.*config* = {

ACCELERATION: 0.001, // Прискорення з часом.

BG_CLOUD_SPEED: 0.2, // Швидкість руху хмар на фоні. *BOTTOM_PAD*: 10, // Відступ знизу для відображення динозавра. *CLEAR_TIME*: 3000, // Час очищення повідомлень (мс). *CLOUD_FREQUENCY*: 0.5, // Частота появи хмар.

GAMEOVER_CLEAR_TIME: 750, // Час очищення після *Game Over* (мс).

GAP_COEFFICIENT: 0.6, // Коефіцієнт відстані між перешкодами.

GRAVITY: 0.6, // Сила гравітації.

INITIAL_JUMP_VELOCITY: 12, // Початкова швидкість стрибка.

INVERT_FADE_DURATION: 12000, // Тривалість анімації зміни кольору (інверсії).

INVERT_DISTANCE: 700, // Відстань до інверсії кольору.

MAX_BLINK_COUNT: 3, // Максимальна кількість моргань динозавра перед стартом.

MAX_CLOUDS: 6, // Максимальна кількість хмар на екрані.

MAX_OBSTACLE_LENGTH: 3, // Максимальна довжина перешкоди.

MAX_OBSTACLE_DUPLICATION: 2, // Максимальна кількість повторів одного типу перешкод поспіль.
MAX_SPEED: 13, // Максимальна швидкість гри.
MIN_JUMP_HEIGHT: 35, // Мінімальна висота стрибка.
MOBILE_SPEED_COEFFICIENT: 1.2, // Коефіцієнт для збільшення швидкості на мобільних.
RESOURCE_TEMPLATE_ID: 'audio-resources', // *ID* шаблону ресурсів для звуків.
SPEED: 6, // Початкова швидкість гри.
SPEED_DROP_COEFFICIENT: 3 // Коефіцієнт зниження швидкості при падінні.
};
/* Стандартні розміри ігрового поля */
Runner.defaultDimensions = {
WIDTH: *DEFAULT_WIDTH*, // Ширина гри (за замовчуванням 600) *HEIGHT*: 150 // Висота гри
};
/* *CSS*-класи, що використовуються в грі */
Runner.classes = {
CANVAS: 'runner-canvas', // Клас для елемента *canvas*
CONTAINER: 'runner-container', // Клас для головного контейнера гри

CRASHED: 'crashed', // Клас, що сигналізує про зіткнення *ICON*: 'icon-offline', // Клас для іконки режиму офлайн *INVERTED*: 'inverted', // Клас інверсії кольорів (нічний режим) *SNACKBAR*: 'snackbar', // Клас для повідомлень (наприклад, підказки) *SNACKBAR_SHOW*: 'snackbar-show', // Клас для показу *snackbar TOUCH_CONTROLLER*: 'controller' // Клас контролера для сенсорних пристроїв
};
/* Розміщення спрайтів на спрайт-листі */
Runner.spriteDefinition = {
LDPI: { // Для пристроїв з низькою роздільністю (*Low DPI*)
CACTUS_LARGE: { *x*: 332, *y*: 2 }, // Великий кактус
CACTUS_SMALL: { *x*: 228, *y*: 2 }, // Малий кактус

```
        CLOUD: { x: 86, y: 2 }, // Хмара
        HORIZON: { x: 2, y: 54 }, // Лінія горизонту
        MOON: { x: 484, y: 2 }, // Місяць
        PTERODACTYL: { x: 134, y: 2 }, // Птеродактиль
        RESTART: { x: 2, y: 2 }, // Кнопка перезапуску
        TEXT_SPRITE: { x: 655, y: 2 }, // Текстові елементи
        TREX: { x: 848, y: 2 }, // Динозавр
        STAR: { x: 645, y: 2 } // Зірка (нічний режим)
    },
    HDPI: { // Для пристроїв з високою роздільністю (High DPI)
        CACTUS_LARGE: { x: 652, y: 2 },
        CACTUS_SMALL: { x: 446, y: 2 },
        CLOUD: { x: 166, y: 2 },
        HORIZON: { x: 2, y: 104 },
        MOON: { x: 954, y: 2 },
        PTERODACTYL: { x: 260, y: 2 },
        RESTART: { x: 2, y: 2 },
        TEXT_SPRITE: { x: 1294, y: 2 },
        TREX: { x: 1678, y: 2 },
        STAR: { x: 1276, y: 2 }
    }
};
/* Звукові ефекти. Посилання на ID аудіо-елементів на сторінці */
Runner.sounds = {
    BUTTON_PRESS: 'offline-sound-press', // Звук натискання кнопки
    HIT: 'offline-sound-hit', // Звук зіткнення
    SCORE: 'offline-sound-reached' // Звук досягнення нової відстані };
 /* Відповідність клавіш діям у грі */
Runner.keycodes = {
    JUMP: { '38': 1, '32': 1 }, // Стрибок: клавіша вгору (↑) або
    пробіл DUCK: { '40': 1 }, // Присісти: клавіша вниз (↓)
```

```
    RESTART: { '13': 1 } // Перезапуск гри: клавіша Enter
};
/* Назви подій, які використовуються у грі */
Runner.events = {
 ANIM_END: 'webkitAnimationEnd', // Завершення анімації (для підтримки
WebKit)
    CLICK: 'click', // Клік миші
    KEYDOWN: 'keydown', // Натискання клавіші
    KEYUP: 'keyup', // Відпускання клавіші
    MOUSEDOWN: 'mousedown', // Натискання кнопки миші
    MOUSEUP: 'mouseup', // Відпускання кнопки миші
    RESIZE: 'resize', // Зміна розміру вікна
    TOUCHEND: 'touchend', // Завершення торкання на сенсорних
пристроях
    TOUCHSTART: 'touchstart', // Початок торкання на сенсорних пристроях
    VISIBILITY: 'visibilitychange', // Зміна видимості вкладки
    BLUR: 'blur', // Втрата фокуса вікном
```



```
    FOCUS: 'focus', // Отримання фокуса вікном
    LOAD: 'load' // Завантаження сторінки
};
Runner.prototype = {
    /*Чи відключено гру. Використовується на пристроях з корпоративною
політикою Chrome OS */
    isDisabled: function () {
        // Можна розкоментувати для підтримки ChromeOS:
        // return loadTimeData && loadTimeData.valueExists('disabledEasterEgg');
        return false; // За замовчуванням гра не вимкнена
    },
    /*Якщо гра вимкнена, відображає повідомлення через snackbar */
    setupDisabledRunner: function () {
        // Створюємо контейнер з повідомленням
```

```
    this.containerEl = document.createElement('div');
    this.containerEl.className = Runner.classes.SNACKBAR;
    this.containerEl.textContent =
    loadTimeData.getValue('disabledEasterEgg');
    this.outerContainerEl.appendChild(this.containerEl);
    // При спробі активації гри показуємо повідомлення
    document.addEventListener(Runner.events.KEYDOWN, function (e) {
      if (Runner.keycodes.JUMP[e.keyCode]) {
        this.containerEl.classList.add(Runner.classes.SNACKBAR_SHOW)
        ; document.querySelector('.icon').classList.add('icon-disabled');
      }
    }.bind(this));
  },
  /**
  * Налаштування окремих параметрів для налагодження (debugging).
  * @param {string} setting — назва параметра (наприклад,
  'GRAVITY') * @param {*} value — нове значення параметра
  */
                                                                    56
  updateConfigSetting: function (setting, value) {
    // Якщо параметр існує в конфігурації і передано значення
    if (setting in this.config && value != undefined) {
      this.config[setting] = value;
      // Залежно від параметра оновлюються також значення у tRex
      switch (setting) {
        case 'GRAVITY':
        case 'MIN_JUMP_HEIGHT':
        case 'SPEED_DROP_COEFFICIENT':
          this.tRex.config[setting] = value;
          break;
        case 'INITIAL_JUMP_VELOCITY':
      this.tRex.setJumpVelocity(value); // встановлюємо нову швидкість стрибка
```

```
                break;
            case 'SPEED':
                this.setSpeed(value); // змінюємо швидкість гри
                break;
        }
    }
},
/*Завантаження зображень зі сторінки та вибір відповідного спрайту. Визначає, яку версію (HDPI або LDPI) використовувати залежно від щільності пікселів */
loadImages: function () {
    if (IS_HIDPI) {
        // Висока щільність пікселів — беремо HD спрайт
        Runner.imageSprite = document.getElementById('offline-resources-2x');
        this.spriteDef = Runner.spriteDefinition.HDPI;
    } else {
        // Стандартна щільність — LD спрайт
        Runner.imageSprite = document.getElementById('offline-resources-1x');
        this.spriteDef = Runner.spriteDefinition.LDPI;
```



```
    }
    if (Runner.imageSprite.complete) {
        // Якщо зображення вже завантажене — ініціалізуємо гру
        this.init();
    } else {
        // Інакше чекаємо на завантаження зображення
        Runner.imageSprite.addEventListener(Runner.events.LOAD,
            this.init.bind(this));
    }
},
/* Завантаження та декодування звуків, закодованих у форматі base64 */
loadSounds: function () {
    // На iOS відтворення звуку обмежене, тому пропускаємо
```

```
    if (!IS_IOS) {
        // Створюємо аудіо-контекст для відтворення звуків
        this.audioContext = new AudioContext();
        // Отримуємо шаблон з ресурсами зі сторінки
        var resourceTemplate =
      document.getElementById(this.config.RESOURCE_TEMPLATE_ID).content;
        // Проходимо по кожному звуку з переліку Runner.sounds
        for (var sound in Runner.sounds) {
            // Отримуємо джерело звуку з елемента audio
            var soundSrc =
                resourceTemplate.getElementById(Runner.sounds[sound]).src
            ; // Видаляємо префікс "data:audio/wav;base64,"
            soundSrc = soundSrc.substr(soundSrc.indexOf(',') + 1);
            // Декодуємо base64 у масив байтів
            var buffer = decodeBase64ToArrayBuffer(soundSrc);
            // Декодуємо аудіо-дані асинхронно і зберігаємо в this.soundFx
            this.audioContext.decodeAudioData(buffer, function (index, audioData) {
            this.soundFx[index] = audioData;
```



```
            }.bind(this, sound)); // Прив'язуємо sound як ключ
        }
    }
},
/**
 * Встановлює швидкість гри.
 * Якщо екран менший за стандартну ширину, зменшує
швидкість. * @param {number} opt_speed — нова швидкість
(необов’язково) */
setSpeed: function (opt_speed) {
    var speed = opt_speed || this.currentSpeed;
    // Якщо ширина екрана менша за стандартну — зменшуємо швидкість
    if (this.dimensions.WIDTH < DEFAULT_WIDTH) {
```

```
        var mobileSpeed = speed * this.dimensions.WIDTH / DEFAULT_WIDTH *
            this.config.MOBILE_SPEED_COEFFICIENT;
        // Встановлюємо або розраховану швидкість, або вихідну (меншу з них)
      this.currentSpeed = mobileSpeed > speed ? speed : mobileSpeed; } else if
      (opt_speed) {
        // Якщо передано конкретну швидкість — встановлюємо її
        this.currentSpeed = opt_speed;
      }
    },
    /*Ініціалізація гри. */
    init: function () {
      // Приховуємо статичну іконку динозавра (використовується до старту гри)
      document.querySelector('.' + Runner.classes.ICON).style.visibility = 'hidden';
      // Налаштовуємо розміри екрана гри
      this.adjustDimensions();
      // Встановлюємо стартову швидкість гри
      this.setSpeed();
```



```
      // Створюємо головний контейнер для гри
      this.containerEl = document.createElement('div');
      this.containerEl.className = Runner.classes.CONTAINER;
      // Створюємо елемент canvas для малювання гравця і всього ігрового світу
      this.canvas = createCanvas(this.containerEl, this.dimensions.WIDTH,
      this.dimensions.HEIGHT, Runner.classes.PLAYER);
      // Отримуємо контекст для малювання та встановлюємо фон
      this.canvasCtx = this.canvas.getContext('2d');
      this.canvasCtx.fillStyle = '#f7f7f7';
      this.canvasCtx.fill();
      // Масштабуємо canvas, якщо потрібно (наприклад, для HiDPI-екранів)
      Runner.updateCanvasScaling(this.canvas);
      // Створюємо об'єкт горизонту — включає хмари, землю та перешкоди
```

```
        this.horizon = new Horizon(this.canvas, this.spriteDef, this.dimensions,
        this.config.GAP_COEFFICIENT);
        // Створюємо лічильник пройденої відстані
        this.distanceMeter = new DistanceMeter(this.canvas,
            this.spriteDef.TEXT_SPRITE, this.dimensions.WIDTH);
        // Створюємо динозавра (T-Rex)
        this.tRex = new Trex(this.canvas, this.spriteDef.TREX);
        // Додаємо контейнер до зовнішнього елемента
        this.outerContainerEl.appendChild(this.containerEl);
// Якщо гра запускається на мобільному пристрої — створюємо сенсорне
керування
        if (IS_MOBILE) {
            this.createTouchController();
        }
        // Запускаємо обробники подій (клавіатура, кліки тощо)
        this.startListening();
        // Запускаємо основний цикл оновлення гри
```



```
        this.update();
        // Додаємо обробник зміни розміру вікна з затримкою
        window.addEventListener(Runner.events.RESIZE,
            this.debounceResize.bind(this));
    },
        createTouchController: function () {
            this.touchController = document.createElement('div');
            this.touchController.className =
            Runner.classes.TOUCH_CONTROLLER;
            this.outerContainerEl.appendChild(this.touchController);
        },
        debounceResize: function () {
            if (!this.resizeTimerId_) {
                this.resizeTimerId_ =
```

```
            setInterval(this.adjustDimensions.bind(this), 250);
        }
    },
    adjustDimensions: function () {
        clearInterval(this.resizeTimerId_);
        this.resizeTimerId_ = null;
        var boxStyles = window.getComputedStyle(this.outerContainerEl);
        var padding = Number(boxStyles.paddingLeft.substr(0,
            boxStyles.paddingLeft.length - 2));
    this.dimensions.WIDTH = this.outerContainerEl.offsetWidth - padding *
        2; if (this.canvas) {
            this.canvas.width = this.dimensions.WIDTH;
            this.canvas.height = this.dimensions.HEIGHT;
            Runner.updateCanvasScaling(this.canvas);
            this.distanceMeter.calcXPos(this.dimensions.WIDTH);
            this.clearCanvas();
            this.horizon.update(0, 0, true);
            this.tRex.update(0);
                                                                    61
            if (this.playing || this.crashed || this.paused) {
                this.containerEl.style.width = this.dimensions.WIDTH + 'px';
                this.containerEl.style.height = this.dimensions.HEIGHT + 'px';
                this.distanceMeter.update(0, Math.ceil(this.distanceRan));
                this.stop();
            } else {
                this.tRex.draw(0, 0);
            }
            if (this.crashed && this.gameOverPanel) {
                this.gameOverPanel.updateDimensions(this.dimensions.WIDTH)
                ; this.gameOverPanel.draw();
            }
        }
```

```
  },
playIntro: function () {
    if (!this.activated && !this.crashed) {
      this.playingIntro = true;
      this.tRex.playingIntro = true;
      var keyframes = '@-webkit-keyframes intro { ' +
        'from { width:' + Trex.config.WIDTH + 'px }' +
        'to { width: ' + this.dimensions.WIDTH + 'px }' +
        '}';
      var sheet = document.createElement('style');
      sheet.innerHTML = keyframes;
      document.head.appendChild(sheet);
      this.containerEl.addEventListener(Runner.events.ANIM_END
        , this.startGame.bind(this));
     this.containerEl.style.webkitAnimation = 'intro .4s ease-out 1
  both'; this.containerEl.style.width = this.dimensions.WIDTH + 'px';
      this.playing = true;
      this.activated = true;
    } else if (this.crashed) {
      this.restart();
    }
  },
 startGame: function () {
    this.runningTime = 0;
    this.playingIntro = false;
    this.tRex.playingIntro = false;
    this.containerEl.style.webkitAnimation = '';
    this.playCount++;
    document.addEventListener(Runner.events.VISIBILITY,
      this.onVisibilityChange.bind(this));
    window.addEventListener(Runner.events.BLUR,
```

```
        this.onVisibilityChange.bind(this));
    window.addEventListener(Runner.events.FOCUS,
        this.onVisibilityChange.bind(this));
},
clearCanvas: function () {
    this.canvasCtx.clearRect(0, 0, this.dimensions.WIDTH,
        this.dimensions.HEIGHT);
},
update: function () {
    this.updatePending = false;
    var now = getTimeStamp();
    var deltaTime = now - (this.time || now);
    this.time = now;
    if (this.playing) {
        this.clearCanvas();
        if (this.tRex.jumping) {
            this.tRex.updateJump(deltaTime);
        }
                                                                    63
        this.runningTime += deltaTime;
        var hasObstacles = this.runningTime > this.config.CLEAR_TIME;
        if (this.tRex.jumpCount == 1 && !this.playingIntro) {
            this.playIntro();
        }
        if (this.playingIntro) {
            this.horizon.update(0, this.currentSpeed, hasObstacles);
        } else {
            deltaTime = !this.activated ? 0 : deltaTime;
            this.horizon.update(deltaTime, this.currentSpeed,
                hasObstacles, this.inverted);
        }
        var collision = hasObstacles &&
```

```
    checkForCollision(this.horizon.obstacles[0], this.tRex);
if (!collision) {
    this.distanceRan += this.currentSpeed * deltaTime /
    this.msPerFrame; if (this.currentSpeed < this.config.MAX_SPEED) {
        this.currentSpeed += this.config.ACCELERATION;
    }
} else {
    this.gameOver();
}
var playAchievementSound = this.distanceMeter.update(deltaTime,
    Math.ceil(this.distanceRan));
if (playAchievementSound) {
    this.playSound(this.soundFx.SCORE);
}
if (this.invertTimer > this.config.INVERT_FADE_DURATION)
    { this.invertTimer = 0;
    this.invertTrigger = false;
    this.invert();
```



```
} else if (this.invertTimer) {
    this.invertTimer += deltaTime;
} else {
    var actualDistance =
        this.distanceMeter.getActualDistance(Math.ceil(this.distanceRan))
    ; if (actualDistance > 0) {
        this.invertTrigger = !(actualDistance %
            this.config.INVERT_DISTANCE);
        if (this.invertTrigger && this.invertTimer === 0) {
            this.invertTimer += deltaTime;
            this.invert();
        }
    }
```

```
            }
          }
          if (this.playing || (!this.activated &&
            this.tRex.blinkCount < Runner.config.MAX_BLINK_COUNT)) {
            this.tRex.update(deltaTime);
            this.scheduleNextUpdate();
          }
        },
function onDocumentLoad() {
    new Runner('.interstitial-wrapper');
}
document.addEventListener('DOMContentLoaded', onDocumentLoad);
```

Програмний код файлу сторінки *main.js*

```
// Модулі для керування життєвим циклом застосунку та створення вікна браузера
const { app, BrowserWindow } = require('electron')
```



```
// Зберігаємо глобальне посилання на об'єкт вікна.
// Якщо цього не зробити, вікно буде автоматично закрите, коли об'єкт буде зібраний збирачем сміття.
let mainWindow
function createWindow () {
  // Створення вікна браузера.
  mainWindow = new BrowserWindow({ width: 800, height: 600
  }) // Завантаження файлу index.html у вікно програми.
  mainWindow.loadFile('index.html')
  // Відкриття інструментів розробника.
  // mainWindow.webContents.openDevTools()
  // Подія, яка виникає при закритті вікна.
```

```
  mainWindow.on('closed', function () {
    // Видалення посилання на об'єкт вікна.
    // Якщо програма підтримує декілька вікон, це місце, де слід видалити відповідний елемент.
    mainWindow = null
  })
}
// Цей метод буде викликаний, коли Electron завершить ініціалізацію
// і буде готовий створити вікна браузера.
// Деякі API можуть використовуватись лише після цієї події. app.on('ready', createWindow)
// Завершення роботи програми при закритті всіх вікон.
app.on('window-all-closed', function () {
  // На macOS прийнято, щоб програми залишалися активними, //
  доки користувач явно не завершить їх за допомогою Cmd + Q. if
  (process.platform !== 'darwin') {
    app.quit()
  }
})
app.on('activate', function () {
  // На macOS прийнято повторно створювати вікно програми,
  // коли натискається іконка в Dock і немає відкритих вікон.
  if (mainWindow === null) {
    createWindow()
  }
```

КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

РЕЦЕНЗІЯ
на кваліфікаційну роботу
випускника спеціальності: 123 «Комп'ютерна інженерія»
відділення: комп'ютерної та програмної інженерії
циклова комісія: комп'ютерних систем та мереж
Юрій РЯБОШАПКА
(ім'я, прізвище)

Обрана тема кваліфікаційної роботи «Гра у стилі Rex Chrome Dino з використанням сучасних веб-технологій» є актуальною з огляду на зростання популярності браузерних ігор, розширення можливостей веб-платформ та постійний попит на інтерактивний та доступний контент для різних аудиторій. Реалізація гри, яка не потребує встановлення, із використанням сучасних технологій фронтенд-розробки, відповідає сучасним тенденціям розвитку веб-індустрії, зокрема підвищенню продуктивності, кросплатформеності та доступності.

Кваліфікаційна робота повністю відповідає заявленій темі, затвердженій відповідним наказом. Всі поставлені завдання були виконані у повному обсязі, зокрема: аналіз аналогічних ігор, розробка ігрової логіки, створення адаптивного графічного інтерфейсу, реалізація механіки взаємодії користувача з грою, а також забезпечення стабільної роботи додатку в різних браузерах і на різних пристроях.

В результаті виконання кваліфікаційної роботи створено повноцінну браузерну гру, яка відтворює класичну ідею Chrome Dino, але з власними модифікаціями та графічними рішеннями. Робота демонструє вміння здобувача освіти працювати з сучасними веб-технологіями, зокрема HTML5, CSS3, JavaScript та фреймворками, а також застосовувати принципи адаптивного дизайну та оптимізації для продуктивної роботи додатку. Якість оформлення пояснювальної записки та графічного матеріалу відповідає вимогам ДСТУ. Матеріал викладений логічно, з послідовним описом етапів проєктування та реалізації гри. У роботі наведено фрагменти програмного коду, що підтверджує практичну реалізацію розробленого рішення.

Результати кваліфікаційної роботи можуть бути використані як основа для створення подібних браузерних ігор, навчальних проєктів або розширення у вигляді мобільних веб-додатків. Проєкт має прикладне значення та демонструє високий рівень володіння сучасними веб-інструментами.

Кваліфікаційна робота виконана на високому рівні та засвідчує належну підготовку здобувача освіти.Робота заслуговує на оцінку "добре".

Рецензент _____ *викладач вищої категорії* _____
(науковий ступінь, посада)

«____»_____ 2025 р. _____ Тетяна НОВІК
(підпис) (ім'я, прізвище)

З рецензією ознайомлений _____ Юрій РЯБОШАПКА
(підпис) (ім'я, прізвище)

})