МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність:

123 – «Комп'ютерна інженерія»

Освітня програма:

«Комп'ютерна графіка і Web-дизайн»

Група: 4КГ-06

Дипломний проект

студента денної форми навчання
КГ 06.26.000.00 ДП

ТЄРОВА
МИХАЙЛА
ОЛЕКСАНДРОВИЧА

м. Одеса
2023 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «Одеський технічний фаховий коледж ОНТУ»

Спеціальність *123 – «Комп'ютерна інженерія»*

Освітня програма *«Комп'ютерна графіка і Web-дизайн»*

Група *4КГ-06*

ПОЯСНЮВАЛЬНА ЗАПИСКА

До дипломного проекту (роботи) на тему:

Розробка браузерної гри типу Arcade на мові програмування Java Script

Проектний матеріал складається з пояснювальної записки на 76 сторінках та графічного матеріалу на 16 аркушах.

Дипломник _____ (*Тєров М.О.*)

Керівник проекту _____ (*Скорняков В.С.*)

Консультанти:

з економічної частини _____ (*Копайгородська Т.Г.*)

з охорони праці _____ (*Чорновол Н.І.*)

за дотриманням вимог ЄСКД _____ (*Петрашова В.І.*)

старший консультант _____ (*Кривченко Ю.В.*)

До захисту допущений

Голова циклової комісії _____ (*Кривченко Ю.В.*)

Завідувач відділенням _____ (*Скорнякова О.В.*)

Захист «24» червня 202 р. Протокол ДКК № 6

Оцінка ДКК 5/відмінно

Секретар ДКК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «Одеський технічний фаховий коледж ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ і ПІ*

Спеціальність *123 "Комп'ютерна інженерія"*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР _____

Беркань І.В.

"_____" _____ 20___ р.

ЗАВДАННЯ

на дипломний проект (роботу)

Здобувачу освіти ***Тєрову Михайлу Олександровичу***

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) ***Розробка браузерної гри типу Arcade на мові програмування Java Script***

затверджена наказом по коледжу від **17 жовтня 2023 р. № 235-А2-ОД**

2. Термін здачі здобувачем освіти закінченого проекту (роботи) **10 червня 2023 р.**

3. Вихідні дані до проекту (роботи) ***Мови програмування. Мова програмування JavaScript. Ігрове середовище Phaser. Програма для написання коду Visual Studio Code.***

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1. Технологічний розділ. 2. Економічний розділ. 3. Охорона праці Висновки. Список використаних джерел. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) *Слайд 1 - Титульний слайд (тема, виконавець, керівник). Слайд 2 - Вступ. Слайд 3 Історія мов програмування. Слайд 4 - Апаратне та програмне забезпечення розробника JavaScript. Слайд 5 - Ігрові движки (середовища). Мобільні платформи (Android, IOS) Слайд 6 - Розробка гри на JavaScript Слайд 7 – 10 – Індикатор прогресу завантаження ресурсу гри. Слайд 11 - Phaser.Game. Слайд 12 – Loader 13 – TileSprite. Слайд 14 - Tween. Слайд 15 - ВИСНОВКИ. Слайд 16 – Дякую за увагу.*

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

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний	*Скорняков В.С.*		
Економ.частина	*Копайгородська Т.Г.*		
Охорона праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		

7. Дата видачі завдання 01.05.23

Керівник *Скорняков В.С.* _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	*Вступ. Постановка задачі проектування*	*20.02.2023*	виконав
2	*Аналітичний розділ. Огляд існуючих рішень.*	*01.03.2023*	виконав
3	*Історія мов програмування. Ігрові платформи*	*20.03.2023*	виконав
4	*Технічний розділ. Етапи створення продукту*	*10.04.2023*	виконав
5	*Інтегровані середовища розробки. Створення сценарію*	*17.04.2023*	виконав
6	*Анімація гри. Фізика гри. Формування об'єктів.*	*01.05.2023*	виконав
7	*Фіксація подій. Звукові ефекти в грі.*	*15.05.2023*	виконав
8	*Фіксація результатів гри.*	*22.05.2023*	виконав
9	*Робота над кодом програми*	*01.06.2023*	виконав
10	*Економічний розділ*	*04.06.2023*	виконав
11	*Питання охорони праці*	*06.06.2023*	виконав
12	*Висновки*	*08.06.2023*	виконав
13	*Чистове оформлення роботи*	*10.06.2023*	виконав
14	*Підготовка презентації*	*до 12.06.2023*	виконав
15	*Підготовка доповіді до захисту*	*до 30.06.2023*	

Дипломник _____
(підпис)

Керівник проекту _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КГ 06. 26 000. 00 ДП	Дипломний проект		
А4			КГ 06. 26 000. 00 ДП ПЗ	Пояснювальна	1	
				записка		
				Презентація		

					КГ 06. 26 000. 00 ДП	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		Тєров М.О.			*Розробки браузерної гри типу Arcade на мові програмування Java Script*	Літ. / Аркуш / Аркушів: Н Д П / 4 /
Перевірив		Скорняков В.С.				
Н. Контр.		Петрашова В.І.				ОТФК ОНТУ 4КГ-06
Затверд.		Кривченко Ю.В.				

ЗМІСТ

ВСТУП

Розробка ігор передбачає ретельне поєднання креативності, вирішення проблем і технічної майстерності. Мови програмування надають розробникам інструменти та фреймворки, необхідні для створення інтерактивного ігрового процесу, приголомшливих візуальних ефектів і динамічних світів.

Кожна мова програмування має власний унікальний набір функцій, сильних сторін і спільнот, що задовольняють різні аспекти розробки ігор. Від потужності та універсальності C++ до доступності JavaScript і простоти Python, розробники ігор мають у своєму розпорядженні цілу низку мов. Вибір правильної мови програмування залежить від різних факторів, таких як платформа, на яку ви орієнтуєтесь, складність вашої ігрової механіки, вимоги до продуктивності та особисті уподобання. Щоб почати розробляти ігри за допомогою Phaser, потрібно базове розуміння JavaScript, HTML і CSS. Phaser можна легко інтегрувати в будь-який робочий процес веб-розробки та вимагає мінімального налаштування. Фреймворк надає комплексний API і потужний набір інструментів для розробки ігор. Отже, незалежно від того, чи ми досвідчений програміст, який прагне розширити свій кругозір, чи починаючий розробник ігор, який робить перші кроки, приготуйтеся вирушити у захоплюючу пригоду у царині розробки ігор за допомогою мов програмування.

Тема дипломної роботи – розробка гри типу Arcade на мові програмування JavaScript.

Мета дипломної роботи – розглянути ігрові середовища та розробити гру та проаналізувати економічну частину розробки та опрацювати матеріал по охороні праці.

У основному розділі дипломного проекту було розглянуто розробку на мові JS та було проаналізовано ігрові середовища які використовувалися при розробці даної роботи вивчено матеріали по вдосконаленню та розробки. У останніх розділах було розглянуто питання економічної доцільності розробки та питання охорони праці.

1 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

1.1 Історія мов програмування

Історія мов програмування бере свій початок від ранніх днів комп'ютерної техніки. Він розпочався з народженням перших програмованих комп'ютерів у середині 20-го століття та продовжує розвиватися разом із досягненнями технологій і потребами програмістів.

З історії мов програмування:

Машинна мова: у 1940-х і 1950-х роках програмістам доводилося спілкуватися з комп'ютерами за допомогою машинної мови, яка складалася з двійкового коду, що представляв конкретні інструкції. Написання програм машинною мовою було виснажливим і спричиняло помилки, оскільки вимагало розуміння складних деталей апаратного забезпечення комп'ютера.

Мова асемблера: для спрощення програмування була введена мова асемблера. Він використовував мнемонічні коди для представлення інструкцій машинної мови, що робило його більш читабельним і легшим для роботи. Мова асемблера безпосередньо відповідала апаратному забезпеченню комп'ютера, дозволяючи програмістам писати більш ефективний код.

Fortran. Розроблений наприкінці 1950-х років Fortran (скорочення від «Formula Translation») був першою мовою програмування високого рівня. Він представив концепцію написання коду в більш зрозумілій людині формі, схожій на математичні формули. Fortran швидко набув популярності серед науковців та інженерів, дозволяючи їм легше писати складні числові обчислення.

COBOL: на початку 1960-х COBOL (Common Business-Oriented Language) була створена для задоволення потреб бізнес-орієнтованого програмування. COBOL був розроблений таким чином, щоб його легко зрозуміли нетехнічні працівники, що дозволяє підприємствам автоматизувати свої операції [1]. Вона стала домінуючою мовою для бізнес-додатків і досі використовується в застарілих системах сьогодні.

LISP: розроблений наприкінці 1950-х років LISP (LIST Processing) був однією з найперших мов програмування, зосереджених на штучному інтелекті (AI) і символьній обробці. LISP представив такі поняття, як списки як фундаментальні структури даних і рекурсію як потужну техніку програмування. Він залишається впливовим у дослідженнях ШІ та функціональному програмуванні.

C: У 1970-х роках Денніс Річі створив мову програмування C у Bell Labs. C була мовою загального призначення з акцентом на ефективності та низькорівневому системному програмуванні. Він став широко використовуватися для операційних систем, компіляторів та інших засобів розробки програмного забезпечення. C послужив основою для багатьох наступних мов і все ще широко використовується сьогодні.

Pascal: У 1970-х роках Ніклаус Вірт розробив Pascal як мову навчання. Паскаль зробив акцент на структурованому програмуванні та представив такі поняття, як сильна типізація та блокові структури. Його простота й читабельність зробили його популярним для освітніх цілей і вплинули на дизайн пізніших мов.

C++: у 1980-х роках Б’ярн Страуструп розширив мову C, щоб створити C++, додавши функції об’єктно-орієнтованого програмування. C++ поєднала ефективність і низькорівневий контроль C із потужністю об’єктно-орієнтованого програмування, що зробило його універсальною мовою для розробки ігор, системного програмного забезпечення та великомасштабних програм.

Java: У середині 1990-х Java з’явилася як портативна, незалежна від платформи мова, розроблена Sun Microsystems (нині належить Oracle) [1]. Java представила концепцію «записати один раз, запустити будь-де» через свою модель виконання байт-коду, дозволяючи розробникам створювати програми, які можуть працювати на різних платформах. Java набула популярності для веб-аплетів, корпоративного програмного забезпечення та розробки програм для Android.

Python: створений Гвідо ван Россумом наприкінці 1980-х років, Python наголошував на читабельності та простоті коду. Він здобув популярність завдяки

простоті використання, чіткому синтаксису та великій стандартній бібліотеці. Python широко використовується в різних областях, включаючи веб-розробку, аналіз даних, наукові обчислення та розробку ігор.

Це лише деякі моменти з багатої історії мов програмування. З тих пір було розроблено багато мов, кожна зі своєю метою, сильними сторонами та спільнотами. Мови програмування продовжуватимуть розвиватися, дозволяючи розробникам створювати більш складні та інноваційні програми.

1.2 Апаратне та програмне забезпечення розробника JavaScript

Розробники ігорипрацюють з комбінацією апаратних і програмних інструментів, щоб створювати привабливі ігрові враження. Давайте розглянемо основні задіяні компоненти:

Комп'ютер: надійний комп'ютер є основною апаратною вимогою для розробки ігор на JavaScript. Він повинен мати достатню обчислювальну потужність, пам'ять і ємність для зберігання, щоб працювати з середовищем розробки та іграми, які ви створюватимете.

Дисплей: монітор із високою роздільною здатністю або кілька моніторів можуть підвищити вашу продуктивність, забезпечуючи достатній простір на екрані для кодування, проектування ігрових ресурсів і попереднього перегляду ваших ігор.[3]

Пристрої введення: Стандартні пристрої введення, такі як клавіатура та миша, необхідні для програмування та навігації середовищ розробки. Крім того, деякі розробники ігор надають перевагу використанню спеціалізованих пристроїв введення, таких як графічні планшети або ігрові контролери, для виконання певних завдань або тестування.

Програмне забезпечення. Текстові редактори/IDE: розробка ігор на JavaScript зазвичай передбачає роботу з текстовим кодом. Ви можете вибрати з різних текстових редакторів або інтегрованих середовищ розробки (IDE), щоб ефективно писати свій код і керувати ним. Серед популярних варіантів – Visual Studio Code, Sublime Text, Atom або WebStorm.

Ігрові движки JavaScript: Ігрові движки забезпечують структуру та інструменти для ефективного створення ігор. Для розробки ігор на JavaScript ви можете використовувати механізми, такі як Phaser, PixiJS або CreateJS. Ці механізми пропонують такі функції, як керування ресурсами, симуляції фізики, анімація та підтримка звуку, що спрощує процес розробки.

Інструменти графіки та анімації. Створення візуально привабливих ігрових ресурсів має вирішальне значення. Ви можете використовувати таке програмне забезпечення, як Adobe Photoshop або GIMP, для редагування зображень, інструменти векторної графіки, такі як Adobe Illustrator або Inkscape, для створення масштабованих ілюстрацій, а також інструменти анімації, такі як Adobe Animate або Spine, для анімації персонажів і об'єктів.

Системи контролю версій. Контроль версій життєво важливий для керування змінами коду, співпраці з членами команди та забезпечення стабільності проекту. Git, популярна розподілена система контролю версій, разом із хостинговими платформами, такими як GitHub або GitLab, можна використовувати для ефективного відстеження та керування кодовою базою.

Браузери та інструменти розробника: ігри на JavaScript зазвичай виконуються у веб-браузерах. Дуже важливо тестувати й оптимізувати свої ігри в різних браузерах. Сучасні веб-браузери, такі як Chrome, Firefox або Edge, надають потужні інструменти для розробників, зокрема консолі налагодження, аналізатори продуктивності та мережеві монітори, які допомагають у розробці ігор і вирішенні проблем.

Тестування та налагодження гри. Такі інструменти, як інструменти налагодження консолі браузера, автоматизовані системи тестування (наприклад, Jest або Mocha) і специфічні для гри системи тестування (наприклад, Selenium або Cypress) можуть допомогти виявити та виправити проблеми у функціональності вашої гри та її продуктивності. , і досвід користувача.

Це лише кілька прикладів апаратних і програмних компонентів, з якими ви зіткнетеся як розробник ігор на JavaScript. Залежно від конкретних вимог і вподобань вашого проекту ви можете досліджувати додаткові інструменти та

бібліотеки, щоб покращити свій робочий процес розробки ігор і забезпечити виняткові ігрові враження.[4]

1.3 Ігрові движки (середовища)

Ігрові механізми, також відомі як середовища розробки ігор, — це програмні інфраструктури, які надають розробникам інструменти, бібліотеки та функції для ефективного створення та створення ігор.

Ці механізми абстрагуються від низькорівневих деталей, таких як рендеринг графіки, керування ресурсами, обробка вхідних даних і реалізація фізики, дозволяючи розробникам зосередитися на логіці та дизайні гри. Ось кілька популярних ігрових движків:

Unity: Unity — це широко використовуваний і універсальний ігровий движок, який підтримує кілька платформ, включаючи ПК, консолі, мобільні пристрої та VR/AR. Він пропонує візуальний редактор, потужну мову сценаріїв під назвою C# і величезне сховище активів із готовими до використання ресурсами, сценаріями та плагінами.

Unreal Engine: Unreal Engine, розроблений компанією Epic Games, є надійним і багатофункціональним ігровим движком, який використовується для створення високоякісних візуально приголомшливих ігор. Він надає візуальну систему сценаріїв під назвою Blueprints, а також програмування на C++ для більш розширених налаштувань. Unreal Engine підтримує широкий спектр платформ, включаючи ПК, консолі, мобільні пристрої та VR/AR.

Godot: Godot — це ігровий движок із відкритим кодом, який пропонує зручний інтерфейс і підтримує різноманітні платформи. Він має унікальну гнучку систему сцен на основі вузлів, вбудовану мову сценаріїв під назвою GDScript і можливість писати логіку гри на C# або C++.

Phaser: Phaser — це легкий і популярний фреймворк 2D-ігор для JavaScript і TypeScript. Він зосереджений на іграх на основі веб-браузера та пропонує такі функції, як завантаження активів, фізика, обробка введення та керування звуком. Phaser відомий своєю простотою використання та великою документацією.

Cocos2d-x: Cocos2d-x — це потужна платформа з відкритим вихідним кодом, яка в основному використовується для розробки 2D-ігор. Він підтримує різні мови програмування, включаючи C++, JavaScript і Lua. Cocos2d-x пропонує багатий набір функцій, включаючи анімацію спрайтів, керування сценою та розгортання на кількох платформах.

Construct: Construct — це візуальний інструмент розробки ігор, який дозволяє користувачам створювати ігри без програмування. Він використовує інтерфейс перетягування та надає потужну систему подій для визначення поведінки гри. Construct підтримує розробку як 2D, так і 3D ігор.

RPG Maker: RPG Maker — спеціалізований інструмент розробки ігор для створення рольових ігор (RPG). Він пропонує візуальний редактор для розробки карт, персонажів і діалогів, а також мову сценаріїв для налаштування. RPG Maker спрощує створення традиційних RPG.[2]

Це лише кілька прикладів популярних ігрових движків і фреймворків, доступних для розробників. Кожен двигун має свої сильні сторони, функції та підтримку спільноти, тому вибір правильного залежить від конкретних вимог до гри, цільових платформ, уподобань мови програмування та рівня знань.

1.4 Мобільні платформи (Android, IOS)

Коли справа доходить до розробки ігор для мобільних платформ, на ринку домінують дві основні операційні системи: Android та iOS. Давайте детальніше розглянемо кожну платформу:

Android:

- ОС Android: Android — це операційна система з відкритим вихідним кодом, розроблена Google на основі ядра Linux. Він живить широкий спектр пристроїв, включаючи смартфони, планшети, смарт-телевізори тощо. Android надає розробникам гнучку та настроювану платформу для створення ігор.
- Java/Kotlin: розробка ігор для Android в основному спирається на мову програмування Java, хоча Kotlin набув популярності як альтернатива. Розробники

використовують Android SDK (Software Development Kit) і такі інструменти, як Android Studio, для створення ігор для пристроїв Android. Крім того, такі фреймворки, як LibGDX і Unity, часто використовуються для спрощення розробки ігор.

- Google Play Store: Google Play Store є основним ринком для розповсюдження ігор для Android. Розробники можуть публікувати свої ігри в Play Store, дозволяючи користувачам знаходити, завантажувати та насолоджуватися їхніми творами. Варіанти монетизації включають безкоштовну гру з покупками в програмі, платні ігри або моделі з підтримкою реклами.

iOS:

- iOS: iOS — це власна операційна система Apple, розроблена спеціально для пристроїв iPhone, iPad та iPod Touch. Він пропонує зручну взаємодію з користувачем і сувору апаратно-програмну інтеграцію.[3]
- Objective-C/Swift: Традиційно розробка ігор для iOS виконувалася за допомогою Objective-C, але Swift набув значної популярності як сучасна та ефективніша мова. Xcode, інтегроване середовище розробки (IDE) від Apple, використовується разом із такими фреймворками, як SpriteKit або Unity, для створення ігор для iOS.
- App Store: App Store є офіційною платформою Apple для розповсюдження ігор для iOS. Розробники можуть надсилати свої ігри в App Store, де користувачі можуть знаходити, завантажувати та купувати їх. App Store пропонує різні варіанти монетизації, включаючи платні ігри, покупки в програмі та підписки.

Кросплатформна розробка:

Щоб одночасно націлюватися на Android і iOS, розробники часто обирають кросплатформену ігорну базу розробки. Ці фреймворки дозволяють написати код один раз і розгорнути його на кількох платформах. Деякі популярні кросплатформні фреймворки для розробки мобільних ігор включають Unity, Cocos2d-x, Godot, Xamarin і React Native.

Міркування. Розробляючи ігри для Android та iOS, важливо враховувати відмінності в апаратних можливостях, розмірах екранів, методах введення та вказівках щодо платформи. Тестування ваших ігор на реальних пристроях і забезпечення оптимізації для кожної платформи мають вирішальне значення для забезпечення найкращої взаємодії з користувачем.

Зрештою, вибір між Android та iOS або вибір кросплатформної розробки залежить від таких факторів, як цільова аудиторія, ресурси розробки, вимоги до проекту та бізнес-цілі.[5]

1.5 Розробка гри на JavaScript

Phaser — це популярна та надійна платформа розробки ігор, створена спеціально для створення 2D-ігор у браузері за допомогою JavaScript. Він надає повний набір інструментів, функцій і ресурсів, які допоможуть вам втілити ваші ігрові ідеї в життя. JavaScript, з іншого боку, є широко поширеною мовою програмування, відомою своєю універсальністю та здатністю бездоганно працювати у веб-браузерах.[8]

Поєднання Phaser і JavaScript пропонує ідеальне середовище для розробки браузерних ігор, якими можна насолоджуватися на різних платформах і пристроях. Незалежно від того, чи є ви досвідченим розробником, чи новачком у розробці ігор, простота й гнучкість Phaser разом із знайомим синтаксисом JavaScript роблять його доступним вибором для втілення ваших концепцій гри.

З Phaser ми матимемо доступ до широкого набору функцій, включаючи обробку спрайтів, анімацію, керування введенням, симуляцію фізики, звукові ефекти тощо. Він також надає надійну та активну спільноту, розширену документацію, численні навчальні посібники та приклади, які допоможуть вам під час розробки гри.[6]

Поринаючи глибше у світ Phaser і JavaScript, ми досліджуватимемо такі ключові поняття, як ініціалізація гри, завантаження ресурсів, створення ігрових сцен, обробка введених користувачем даних, впровадження ігрової механіки та додавання візуальних і звукових ефектів для покращення досвіду гравця. Ми дамо

волю нашій творчості та уяві, щоб створити захоплюючі рівні, інтригуючих персонажів і захоплюючий ігровий процес.

Протягом цього процесу ми наголошуватимемо на важливості ітераційної розробки, тестування та вдосконалення вашої гри, щоб створити привабливу та приємну гру для гравців. Ми також розглянемо найкращі методи структурування вашого коду, оптимізації продуктивності та забезпечення сумісності з різними браузерами.[7]

Погляньмо на параметри функції *Phaser.Game*.

Phaser.Game*(width, height, renderer, parent, state, transparent, antialias, physicsConfig*)

Width - ширина гри, тобто ширина полотна, що використовується для рендерингу гри, у пікселях.

Height - висота гри, тобто висота полотна, що використовується для рендерингу гри, у пікселях.

Renderer - який метод рендерингу використовувати, *Phaser. CANVAS* повинен використовувати полотно *html5*, *Phaser.WEBGL* повинен використовувати *WebGL* з більш високою продуктивністю для рендерингу, *Phaser.AUTO* повинен автоматично визначати, якщо браузер підтримує WebGL, використовувати *WebGL*, інакше використовувати *Canvas*.

Parent - Батьківським елементом, який використовується для розміщення елемента canvas, може бути ідентифікатор елемента або сам елемент dom, Phaser автоматично створить полотно та вставить його в цей елемент.

State - Стан можна розуміти як сцену. Вказати тут стан означає дозволити грі спочатку завантажити цю сцену, але ви також можете не вказувати тут стан і вирішити, який стан завантажити першим у наступному коді.

Transparent - Чи використовується прозорий фон

Antialias - Чи вмикається згладжування

PhysicsConfig - Параметри конфігурації системи ігрової фізики

Усі наведені вище параметри є необов’язковими, їх значення за замовчуванням. Загалом нам потрібно вказати лише перші 4-5 параметрів.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>game</title>
<script src="js/phaser.js"></script>
</head>
<body>
<div id="game"></div>
<script>
var game = new Phaser.Game(288,505,Phaser.AUTO,'game');
</script>
</body>
</html>
var state1 = {
    preload : function(){ },
    create : function(){ },
    update : function(){ }
}
var state2 = function(){
    this.preload = function(){ };
    this.create = function(){ };
    this.update = function(){ };
}
var state3 = function(){
    this.update = function(){  };
```

Після створення екземпляра об’єкта Game наступне, що потрібно зробити, це створити різні сцени, які використовуватиме гра.

Стан може бути власним об’єктом js або функцією, якщо вони мають будь-який із трьох методів попереднього завантаження, створення та оновлення, це законний стан [1]

У цій грі ми будемо використовувати 4 стани. Ми можемо додати стан до гри за допомогою методу *game.state.add(),* а потім використовувати метод *game.state.start()* для виклику стану.

```
}
var state4 = function(){
    this.create = function(){ };
    this.aaa = function(){ };
    this.bbb = 'hello'; }
```

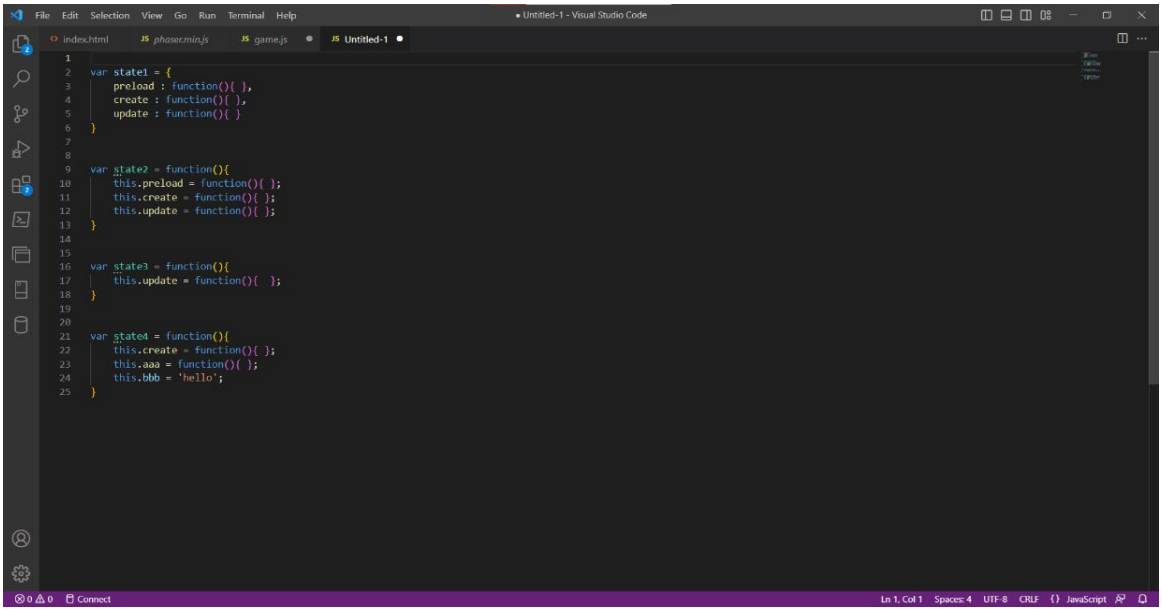


Рисунок 1.1 – функції *Phaser.Game* (1)

У цій грі ми будемо використовувати 4 стани. Ми можемо додати стан до гри за допомогою методу *game.state.add(),* а потім використовувати метод *game.state.start()* для виклику стану [1]

```
game.States.boot = function(){
    this.preload = function(){
      game.load.image('loading','assets/preloader.gif');
    };
    this.create = function(){
```

```
var game = new Phaser.Game(288,505,Phaser.AUTO,'game');
game.States = {};
game.State.boot = function(){ ... }
game.State.prelaod = function(){ ... }
game.State.menu = function(){ ... }
game.State.play = function(){ ... }
game.state.add('boot',game.States.boot);
game.state.add('preload',game.States.preload);
game.state.add('menu',game.States.menu);
game.state.add('play',game.States.play);
game.state.start('boot');

game.state.start('preload');
};
```

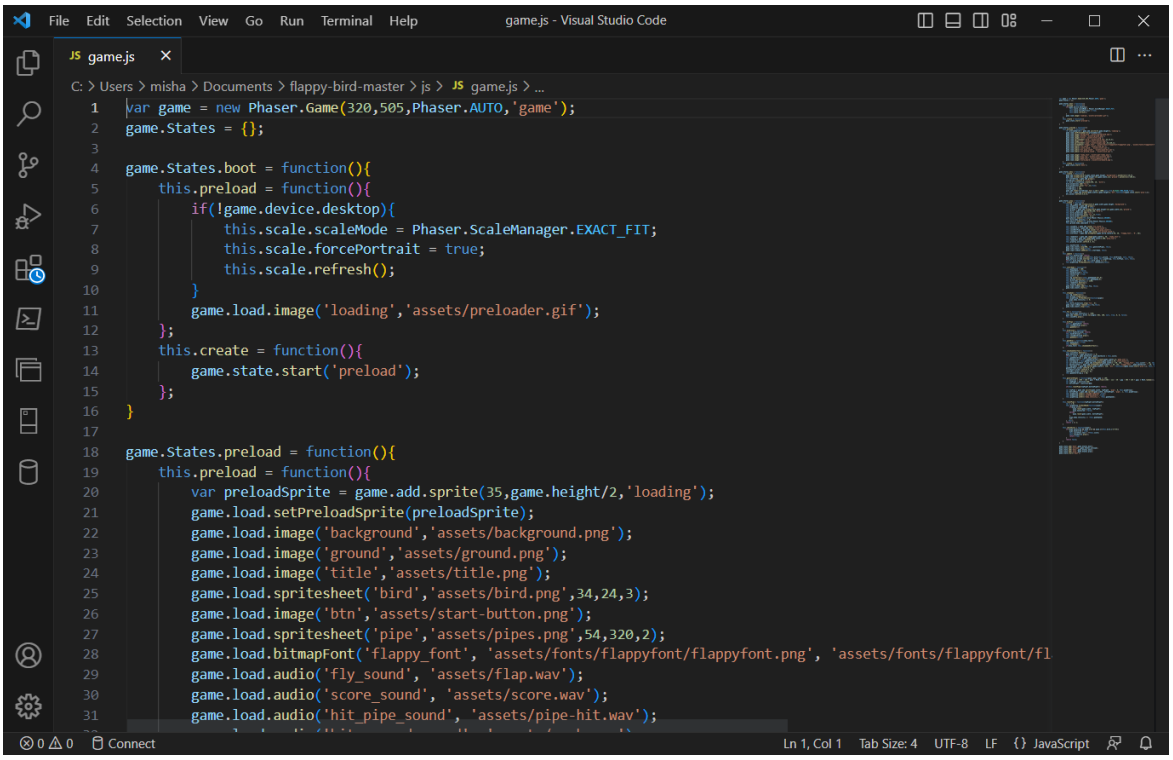


Рисунок 1.2 – Опис методу *var state 1.2.3 (2)*

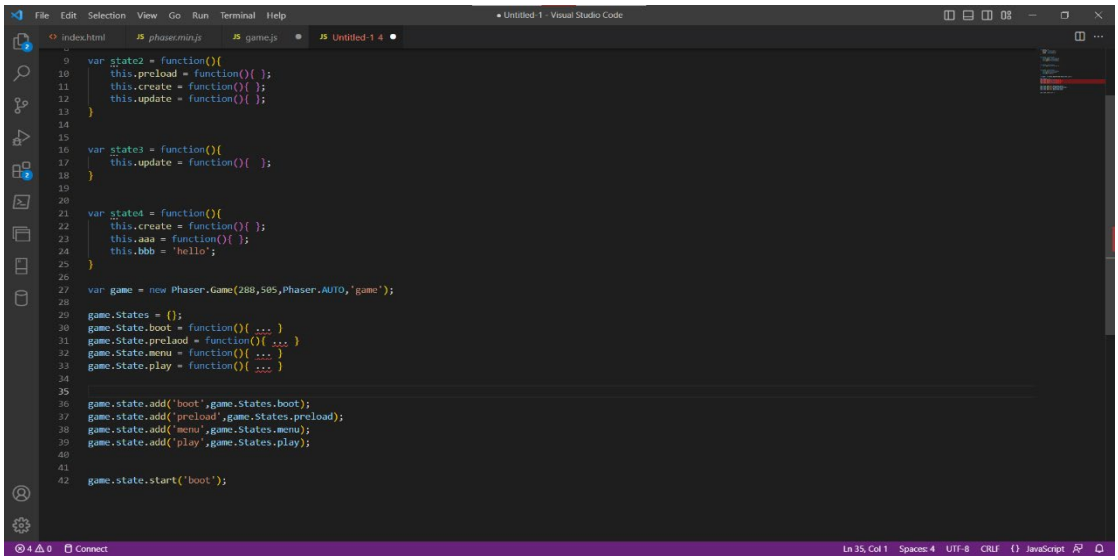


Рисунок 1.3 – Опис методу *game.state.add(3)*

Індикатор прогресу завантаження ресурсу гри

Деякі зображення, звуки та інші ресурси, які будуть використовуватися в грі, потрібно завантажити заздалегідь. Іноді, якщо ресурсів багато, необхідно зробити сторінку прогресу завантаження ресурсу, щоб покращити терпіння користувача в очікуванні.

Тут ми реалізуємо це за допомогою стану під назвою “*preload*”.

Оскільки індикатор прогресу завантаження ресурсу потребує фонового зображення індикатора прогресу, перш ніж зробити цей стан, нам потрібен ще один базовий стан для завантаження зображення індикатора прогресу, який ми назвали завантаженням.

Завантаження ресурсів у Phaser здійснюється за допомогою методу об’єкта Phaser.Loader [5]

Атрибут load екземпляра гри вказує на об’єкт Loader поточної гри, яким у нашому випадку є *game.load.* Об’єкт Loader має багато методів. Різні методи можуть завантажувати різні ресурси.

Ми використовуємо метод *game.load.image()* для завантаження зображень.

Об’єкт ***Loader*** надає метод *setPreloadSprite.*

У сценарії попереднього завантаження нам потрібно завантажити всі ресурси, які будуть використовуватися пізніше в грі, а потім відобразити індикатор прогресу завантаження, щоб користувач міг бачити його.

Поки об’єкт спрайту призначено цьому методу, ширина або висота об’єкта спрайту автоматично регулюватиметься відповідно до поточного відсотка завантаження, досягаючи ефекту динамічної індикатора прогресу.[5]

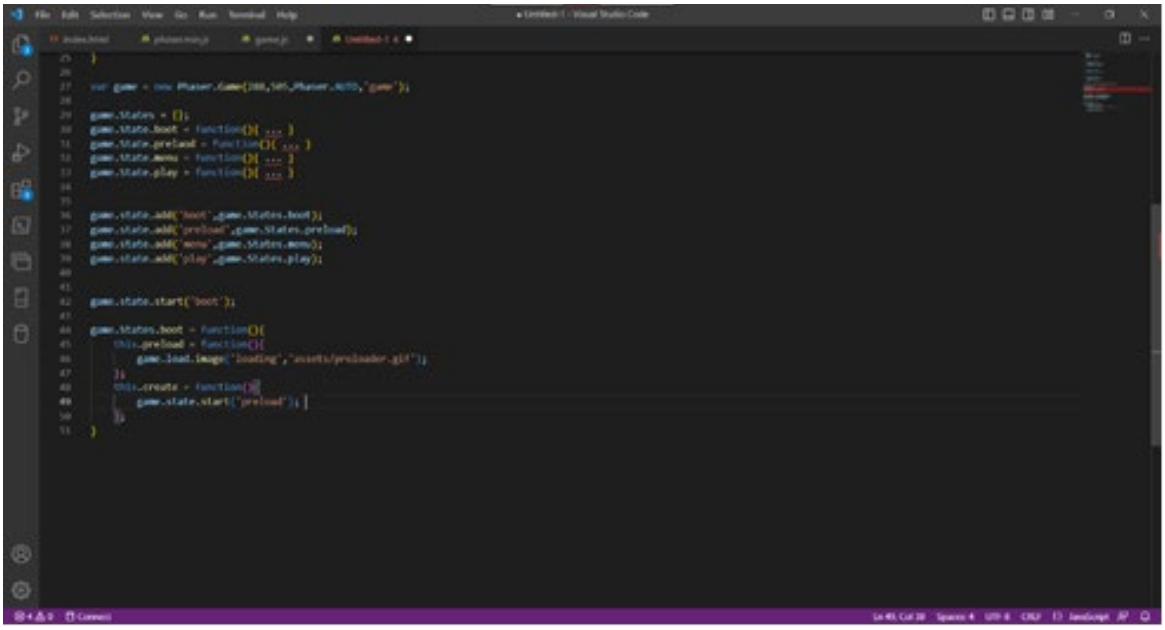


Рисунок 1.4 – Опис методу *preload(4)*

Об’єкт Sprite, який у розробці ігор зазвичай називають спрайтом.

Так само об’єкт спрайт у Phaser також є одним із найбільш використовуваних і найважливіших об’єктів у процесі створення ігор.

Ми використали зображення для створення спрайту, а потім використати багато властивостей і методів, які надає нам Phaser, щоб оперувати ним.

Вище ми використовуємо game.add.sprite() для створення спрайту, який буде автоматично додано до поточної гри після створення. різні компоненти гри.

```
game.States.preload = function(){
    this.preload = function(){
        var preloadSprite = game.add.sprite(50,game.height/2,'loading');
        game.load.setPreloadSprite(preloadSprite);
```

Необхідно завантажити наступні ресурси

```
        game.load.image('background','assets/background.png');
        game.load.image('ground','assets/ground.png');
        game.load.image('title','assets/title.png');
        game.load.spritesheet('bird','assets/bird.png',34,24,3);
        game.load.image('btn','assets/start-button.png');
        game.load.spritesheet('pipe','assets/pipes.png',54,320,2);
        game.load.bitmapFont('flappy_font',
'assets/fonts/flappyfont/flappyfont.png', 'assets/fonts/flappyfont/flappyfont.fnt');
        game.load.audio('fly_sound', 'assets/flap.wav');
        game.load.audio('score_sound', 'assets/score.wav');
        game.load.audio('hit_pipe_sound', 'assets/pipe-hit.wav');
        game.load.audio('hit_ground_sound', 'assets/ouch.wav');
        game.load.image('ready_text','assets/get-ready.png');
        game.load.image('play_tip','assets/instructions.png');
        game.load.image('game_over','assets/gameover.png');
        game.load.image('score_board','assets/scoreboard.png');
    }
    this.create = function(){
        game.state.start('menu');
    }
}
```

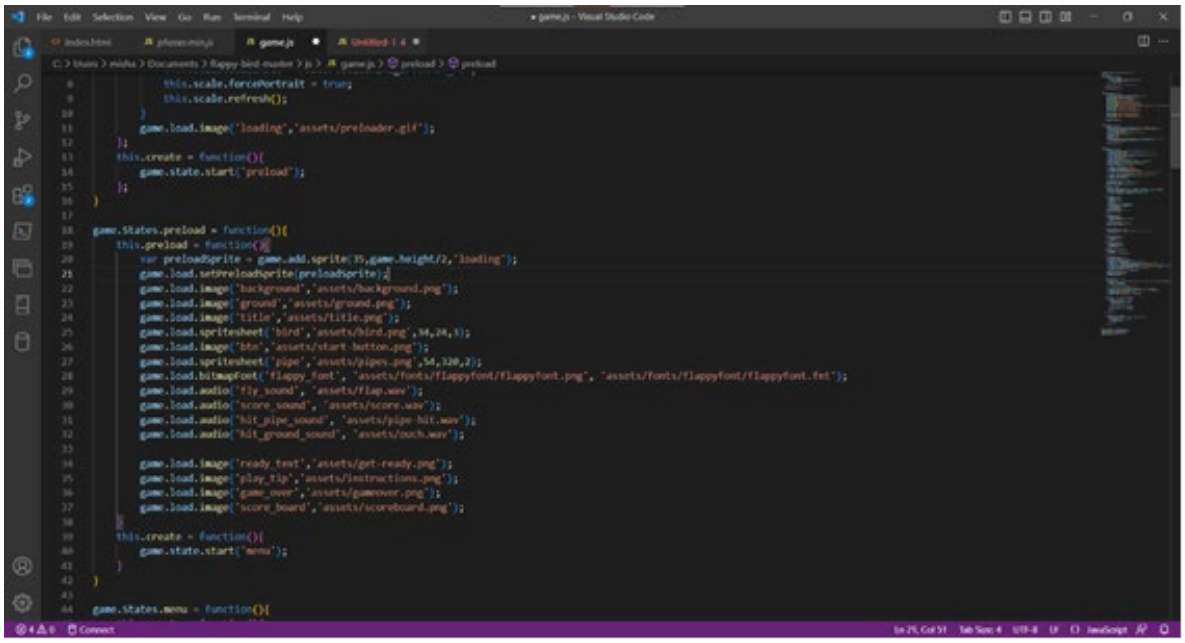


Рисунок 1.5 – Завантаження ресурсу гри (1.2.3)

Рисунок 1.6 – Завантаження ресурсу гри (2.3)

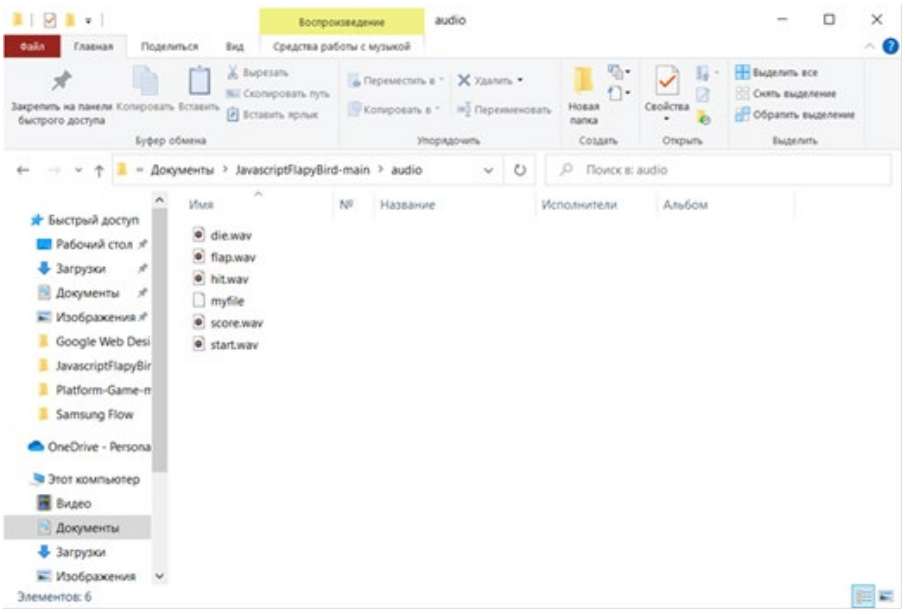


Рисунок 1.7 – Завантаження ресурсу гри (3)

Після завантаження ресурсів настав час увійти на сторінку меню гри, яка називається сторінкою меню, але тут ми пропонуємо лише кнопку для запуску гри, тому не робимо це настільки складним. Ефект після завершення наступний [8]

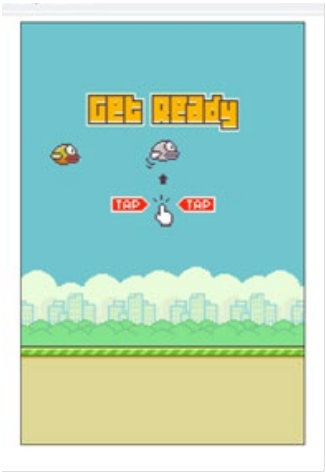


Рисунок 1.8 – Стартова анімаця (Початок гри)

По-перше, це фонове зображення та земля. Ми бачимо, що ці дві речі можуть рухатися. Земля рухається швидше, а фонове зображення повільніше.

У ***Phaser*** є спеціальна функція для обробки цього ефекту, яка називається *TileSprite*.

- ***TileSprite***, по суті, все ще є об'єктом спрайту, але текстуру цього спрайту можна переміщати, і він буде автоматично розміщуватися, щоб компенсувати проміжок після переміщення, тому, якщо наша матеріальна картинка не показує жодних прогалин після переміщення, це може бути тепер використовується як мобільна текстура *TileSprite*.

Текстуру *TileSprite* можна переміщати горизонтально чи вертикально, або обидва одночасно.

Нам потрібно лише викликати метод autoScroll(x,y) об'єкта *TileSprite*, щоб перемістити його текстуру, де x — швидкість у горизонтальному напрямку , y – швидкість у вертикальному напрямку.

Створюємо назву гри. Слово «Flappy bird» у назві гри — це зображення, а потім птах — це спрайт, і ми виконали анімацію на спрайті, щоб здавалося, що його крила рухаються. [10]

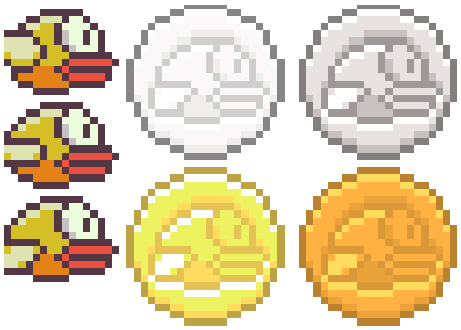


Рисунок 1.9 – Модель Птаха (Медалі за проходження)

Ми бачимо, що на цій картинці є три птахи, точніше три стани птаха, або три кадри в анімації.

В об'єкті *Loader* є метод spritesheet, який спеціально використовується для завантаження такого роду багатокадрових зображень. Давайте подивимося на цей метод: ***spritesheet****(key, url, frameWidth, frameHeight, frameMax, margin, spacing)*

- *Key - Ім'я, призначене цьому зображенню, буде використано пізніше під час створення таких об'єктів.*
- *Url - URL зображення*
- *FrameWidth : ширина кожного кадру в зображенні*
- *FrameHeight : висота кожного кадру на зображенні*
- *FrameMax : максимальна кількість кадрів*
- *Margin : поля на кадр*
- *Spacing : інтервал між кожним кадром*

На зображенні птаха вище ширина та висота кожного птаха дорівнюють 34 пікселям і 24 пікселям відповідно, тому *frameWidth* має бути 34, *frameHeight* — 24, а наша анімація має три кадри, *frameMax* — 3, між кадрами немає проміжку, margin Both інтервал і інтервал дорівнюють 0.

Фактично, метод спрайт-таблиці дозволяє нам завантажити зображення та розділити кадр на зображення. Пізніше спрайт, який використовує це зображення, може використовувати ці кадри для відтворення анімації. Щоб реалізувати анімацію на спрайті, ми повинні спочатку визначити анімацію, тобто визначити, з яких кадрів анімація складається.

Об’єкт спрайту має властивість *animations*, яка представляє об’єкт у **Phaser**, який спеціалізується на управлінні анімаціями: *AnimationManager*, який має метод add для додавання анімацій і метод play для відтворення анімацій.

Інший важливий об'єкт: *Phaser.Group*, який є групою. Група еквівалентна батьківському контейнеру. Ми можемо помістити багато об’єктів у групу, а потім використовувати методи, надані групою, для виконання пакетних або загальних операцій над цими об’єктами.

Наприклад, якщо ви хочете змусити об’єкти в групі погодитися на зміщення, вам потрібно виконати зміщення лише для групи. Наприклад, якщо ви хочете виконати виявлення зіткнень для всіх об’єктів у групі, вам потрібно лише щоб виконати виявлення зіткнень на цій групі об'єктів.

Назва гри, яку ми збираємося створити нижче, складається з текстового зображення та пташки. Ми поміщаємо ці дві речі в групу, а потім виконуємо загальну операцію.[11]

```
game.States.menu = function(){
    this.create = function(){
        var bg = game.add.tileSprite(0,0,game.width,game.height,'background');
        var ground = game.add.tileSprite(0,game.height-112,game.width,112,'ground').autoScroll(-100,0);
        bg.autoScroll(-10,0);
        ground.autoScroll(-100,0);
    }
}
```

Об’єкт Tween у наведеному вище коді спеціально використовується для реалізації анімації анімації. Отримав об’єкт *Tween* за допомогою методу анімації *game.add.* Параметром цього методу є об’єкт, який потрібно анімувати.

Тоді ми можемо використати метод to об’єкта *Tween* для реалізації анімації анімації.

***to**(properties, duration, ease, autoStart, delay, repeat, yoyo)*

- *Properties*: об’єкт js, який містить властивості, які потрібно анімувати, наприклад {y:120} у наведеному вище коді
- *Duration*: тривалість анімації в мілісекундах
- *Ease* : функція полегшення, за замовчуванням рівномірна анімація
- *AutoStart* : чи запускати автоматично
- *Delay*: затримка перед початком анімації в мілісекундах
- *Repeat* : кількість повторів анімації. Якщо вам потрібно, щоб анімація повторювалася вічно, встановіть це значення як Number.MAX_VALUE
- *Yoyo* : якщо значення істинне, анімація буде автоматично перевернута.

```
game.States.menu = function(){
        this.create = function(){
        .....
        var titleGroup = game.add.group();
        titleGroup.create(0,0,'title');
        var bird = titleGroup.create(190, 10, 'bird');
        bird.animations.add('fly');
        bird.animations.play('fly',12,true);
        titleGroup.x = 35;
        titleGroup.y = 100;
        game.add.tween(titleGroup).
    to ({ y:120 },1000,null,true,0,Number.MAX_VALUE,true);
      }
    }
```

Останнє, що потрібно додати кнопку для запуску гри. Phaser надає об’єкт *Button*, щоб ми могли легко реалізувати кнопку.[11]

Багато об’єктів у Phaser мають атрибут прив’язки, який вказує на центральну точку об’єкта, а вісь переміщення та обертання об’єкта базується на цій центральній точці.

Отже, у наведеному вище коді ми хочемо відцентрувати кнопку по горизонталі та вертикалі. Окрім встановлення властивостей кнопки x та y на половину ширини та висоти гри, нам також потрібно встановити центральну точку кнопки на центр кнопки.

Нарешті, ми збираємо всі коди разом, щоб отримати остаточний код стану меню, для якого потрібен лише один метод *create*.

```
game.States.menu = function(){
    this.create = function(){
    game.add.tileSprite(0,0,game.width,game.height,'background').autoScroll(-10,0);
        game.add.tileSprite(0,game.height-112,game.width,112,'ground').autoScroll(-100,0);
        var titleGroup = game.add.group();
        titleGroup.create(0,0,'title');
        var bird = titleGroup.create(190, 10, 'bird');
        bird.animations.add('fly');
        bird.animations.play('fly',12,true);
        titleGroup.x = 35;
        titleGroup.y = 100;
        game.add.tween(titleGroup)
    .to({ y:120 },1000,null,true,0,Number.MAX_VALUE,true);
        var btn = game.add.button(game.width/2,game.height/2,'btn',function(){
            game.state.start('play');
        });
        btn.anchor.setTo(0.5,0.5);
    }
}
```

Далі ми розпочнемо створення основного стану гри. Розглянемо деякі ключові фрагменти коду. Після натискання кнопки «Пуск» у меню гри спочатку з’являється цей екран [14].

Рисунок 1.10 – Стартова анімація (Початок гри)

```
game.States.play = function(){
  this.create = function(){
    this.bg = game.add.tileSprite(0,0,game.width,game.height,'background');
    this.pipeGroup = game.add.group();
    this.pipeGroup.enableBody = true;
    this.ground = game.add.tileSprite(0,game.height-112,game.width,112,'ground');
     game.States.menu = function(){
       this.create = function(){
          .....

          var btn = game.add.button(game.width/2,game.height/2,'btn',function(){
            game.state.start('play');
          });
          btn.anchor.setTo(0.5,0.5);
       }
     }
          this.bird = game.add.sprite(50,150,'bird');
          this.bird.animations.add('fly');
          this.bird.animations.play('fly',12,true);
```

```
        this.bird.anchor.setTo(0.5, 0.5);
        game.physics.enable(this.bird,Phaser.Physics.ARCADE);
        this.bird.body.gravity.y = 0;
        game.physics.enable(this.ground,Phaser.Physics.ARCADE);
        this.ground.body.immovable = true;

        this.readyText = game.add.image(game.width/2, 40, 'ready_text');
        this.playTip = game.add.image(game.width/2,300,'play_tip');
        this.readyText.anchor.setTo(0.5, 0);
        this.playTip.anchor.setTo(0.5, 0);
        this.hasStarted = false;
        game.time.events.loop(900, this.generatePipes, this);
        game.time.events.stop(false);
        game.input.onDown.addOnce(this.statrGame, this);
    };
```

У першій частині роботи ми створили сцену ігрового меню, ця сторінка схожа на ту, за винятком того, що на цій сторінці видалено назву гри та кнопку «Пуск», а також додано «Готуйся» та підказку клацнути по екрану, щоб керувати грою. Дві картинки.

У цьому стані нам потрібно ввімкнути фізичний механізм і мати можливість реагувати на події клацання миші [15].

1.6 Фізика гри

За замовчуванням система фізики кожного об’єкта в грі вимкнена. Щоб увімкнути систему фізики об’єкта, ви можете скористатися методом *game.physics.enable()*.

- enable(object, system, debug)
- Object : об’єкт для ввімкнення фізичної системи, один об’єкт або масив із кількох об’єктів

- System : фізична система, яку потрібно ввімкнути, за замовчуванням Phaser.Physics.ARCADE, наразі Phaser підтримує три фізичні механізми, а саме Arcade, P2 і Ninja.
- Debug : чи вмикати налагодження

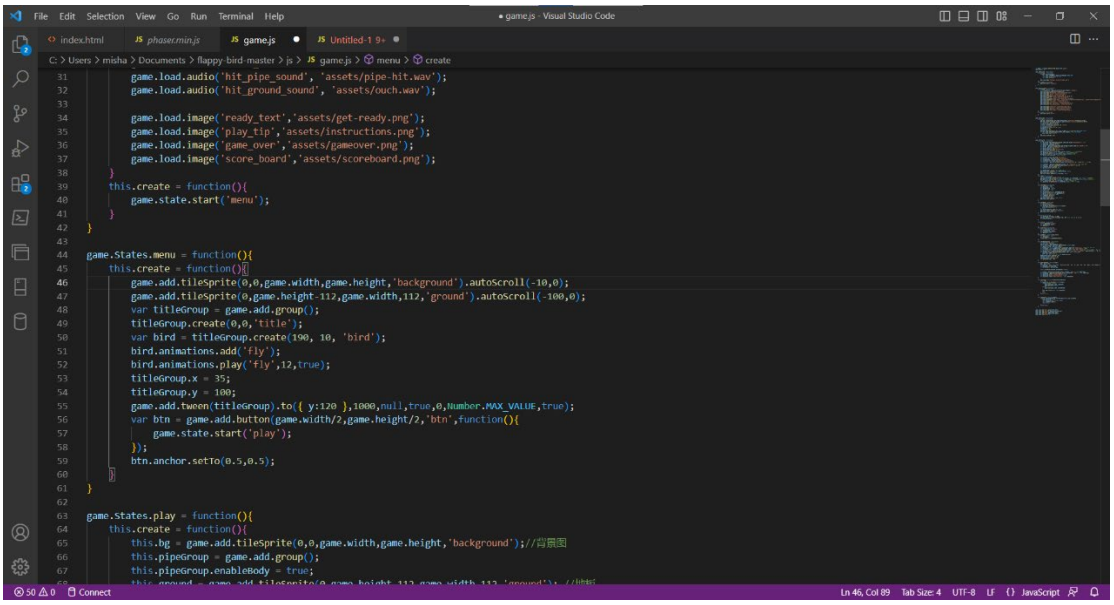


Рисунок 1.11 – Реалізація події *Create*

Лише коли фізичну систему об’єкта ввімкнено, об’єкт має фізичні характеристики. Після ввімкнення фізичної системи властивість тіла об’єкта вказує на фізичну систему, якою володіє об’єкт.

Необхідно керувати всіма властивостями чи методами, пов’язаними з фізикою на тілі.

Миша, клавіатура, дотик та інші інтерактивні події у Phaser однаково обробляються об’єктом Input.

Нам потрібна відповідь миші після натискання екрана. Ми можемо використовувати властивість onDown об’єкта Input, яка вказує на об’єкт Phaser.Signal. Ми можемо прив’язувати події до цього об’єкта.

Щоразу, коли натискається кнопка миші, сигнал onDown з’являється Якщо події прив’язані до об’єкта сигналу onDown, тоді ці події будуть виконані.

Наприклад: var input = game.input; - Вхідний об’єкт поточної гри

- var signal = input.onDown; - Сигналізує об'єкт при натисканні миші
- signal.add(function(){}); - Прив’язати обробники подій до Signal

- signal.add(function(){}); - Зв'язати інше натискння
- signal.addOnce(function(){}); - Прив’язав функцію події, яка буде виконана лише один раз

Об'єкт годинник. Іноді нам потрібно виконувати частину коду регулярно або через рівні проміжки часу. У нативному js ми можемо реалізувати це за допомогою setTimeout і setInterval. Phaser надає нам більш потужний об’єкт Timer для досягнення цих функцій.

Об’єкт Timer в основному має такі методи:

loop(delay, callback, callbackContext, arguments); - Безкінечне виконання функції в заданий проміжок часу, доки не буде викликано метод stop() об’єкта Timer

repeat(delay, repeatCount, callback, callbackContext, arguments); **-** Нехай функція виконується багаторазово, ви можете вказати кількість повторень.

Ми можемо отримати поточний об’єкт Timer через *game.time.events*.

Після виклику циклу або методу повторення об’єкта Timer ми також повинні викликати метод start, щоб запустити його.

game.time.events.loop(900, this.generatePipes, this); - Використовуйте об’єкт годинника для повторного створення каналів

game.time.events.stop(false); - Нехай він зупиниться спочатку, тому що він запуститься автоматично, навіть якщо метод start не викликається, це має бути помилка[16]

Політ птаха що виконується за допомогою функції this.fly

```
this.fly = function(){
    this.bird.body.velocity.y = -350;
    game.add.tween(this.bird).to({angle:-30}, 100, null, true, 0, 0, false);
    this.soundFly.play();
}
```

Сила тяжіння і швидкість.

Об’єкт *Phaser.Physics.Arcade.Body*, який є об’єктом, на який вказує sprite.body, коли ви використовуєте механізм аркадної фізики, має багато

властивостей і методів, пов'язаних з фізикою. Гравітаційний об'єкт представляє силу тяжіння, яка має дві властивості x і y, які представляють силу тяжіння в горизонтальному та вертикальному напрямку відповідно.

Ми можемо використовувати його метод *setTo*(x,y), щоб установити силу тяжіння в обох напрямках одночасно. На рух об'єкта зі встановленою силою тяжіння впливатиме сила тяжіння, що узгоджується з фізичними явищами в реальному житті. Тоді тіло також має об'єкт швидкості, який представляє швидкість об'єкта. [12]

Подібно до сили тяжіння, вона розділена на два напрямки: горизонтальну та вертикальну. Її також можна встановити за допомогою методу *setTo*(x,y).

Як тільки для об'єкта встановлено правильну швидкість, він може рухатися.

Генерація Колон.

Давайте подивимося на функцію генерації конвеєра *this.generatePipes*

```
this.statrGame = function(){
    this.gameSpeed = 200;
    this.gameIsOver = false;
    this.hasHitGround = false;
    this.hasStarted = true;
    this.score = 0;
    this.bg.autoScroll(-(this.gameSpeed/10),0);
    this.ground.autoScroll(-this.gameSpeed,0);
    this.bird.body.gravity.y = 1150;
    this.readyText.destroy();
    this.playTip.destroy();
    game.input.onDown.add(this.fly, this);
    game.time.events.start();
}
this.generatePipes = function(gap){gap = gap || 100;
    var position = (505 - 320 - gap) + Math.floor((505 - 112 - 30 - gap - 505 + 320 + gap) * Math.random());
```

```
        var topPipeY = position-360;
        var bottomPipeY = position+gap;
        if(this.resetPipe topPipeY,bottomPipeY    return;
        var topPipe = game.add.sprite game.width, topPipeY, 'pipe', 0,
this.pipeGroup);
        var bottomPipe = game.add.sprite game.width, bottomPipeY, 'pipe', 1,
this.pipeGroup);
        this.pipeGroup.setAll('checkWorldBounds',true);
        this.pipeGroup.setAll 'outOfBoundsKill',true);
        this.pipeGroup.setAll 'body.velocity.x', -this.gameSpeed);
    }
```

Ідея створення конвеєра: використовуйте випадкові числа для обчислення положення верхнього та нижнього конвеєрів, а потім перевірив, чи не вийшли якісь конвеєри за межі. Якщо так, скиньте положення групи конвеєрів, які вийшли за межі Якщо ні, створіть новий набір конвеєрів Конвеєр, щоб ви могли уникнути зайвої втрати пам'яті. Всі конвеєри об'єднуємо в групу для централізованого управління.

Тут потрібно освоїти метод скидання об'єкта спрайту: *reset(x, y, здоров'я)*

Цей метод може скинути положення спрайт-об'єкта.

Що більш важливо, якщо цей метод виконується на спрайт-об'єкті, який було вбито (kill), такі властивості спрайту, як «живий», «існує», «видимий» і «придатний для відтворення», будуть змінені на правда.

Цей метод часто використовується, коли існуючий об'єкт спрайту потрібно використовувати повторно. Давайте подивимося, як ми використовуємо цей метод у нашій грі [11].

```
    this.resetPipe = function(topPipeY,bottomPipeY){
        var i = 0;
        this.pipeGroup.forEachDead(function(pipe){
            if(pipe.y<=0){
                pipe.reset(game.width, topPipeY);
```

```
            pipe.hasScored = false; }else{
            pipe.reset(game.width, bottomPipeY);
        }
        pipe.body.velocity.x = -this.gameSpeed;
        i++;
    }, this);
    return i == 2; i==2
```

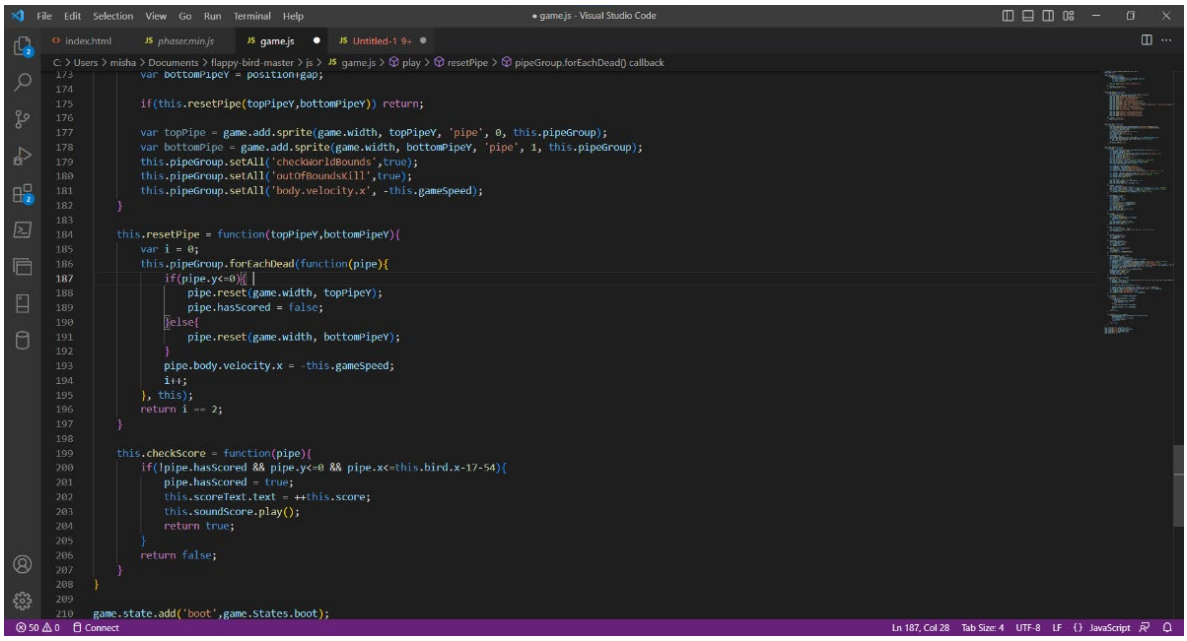


Рисунок 1.12 – Реалізація події this.generatePipes та reset(x, y, здоров'я)

1.7 Анімації гри

Перевірка впливу на гру. Труба і пташка можуть рухатися. Наступним кроком є реалізація функції, що гра закінчується, коли пташка вдаряється об трубу або землю.

У фізичному движку Arcade виявлення зіткнень в основному використовує дві функції: одна — зіткнення, а інша — перекриття.

Різниця між методом зіткнення та перекриття полягає в тому, що зіткнення вплине на фізичний стан між двома об’єктами, які потрібно виявити. Наприклад, використовуйте функцію зіткнення, щоб виявити два об’єкти. [14]

Якщо об’єкти зіткнуться, між ними буде сильна взаємодія два об’єкти, можливо, один із них відштовхнеться від іншого, або обидва відштовхнуться один від одного. Але якщо ви використовуєте метод перекриття, він лише визначить, чи два об’єкти зіткнулися або перекрилися, і не спричинить фізичних ефектів. Очевидно, що якщо вам потрібно лише знати, чи перекриваються два об’єкти, ефективність перекриття буде краще. Виявлення зіткнень може виявити один об’єкт проти одного об’єкта, один об’єкт із групою та групу з групою.

Метод зіткнення має викликатися в кожному кадрі, щоб створити фізику після зіткнення.

Управління оцінками (Бали).

1 бал нараховується, коли птах пролітає через набір труб. Проліт крізь групу труб означає, що ця група труб уже знаходиться ліворуч від птаха, тому ви можете судити, чи отримали ви бали за координатою x труб.

- *this.checkScore = function(pipe){* - Відповідає за виявлення та оновлення балів, канал представляє конвеєр, який потрібно виявити
- *pipe.hasScored* - Атрибут використовується для ідентифікації того, чи оцінено конвеєр
- *pipe.y<0* - Це стосується зазначеного вище трубопроводу в групі трубопроводів. Нам потрібно лише виявити один конвеєр у групі трубопроводів.

Коли координата x труби плюс ширина труби менші за координату x птаха, це означає, що птах пролетів через трубу і може забити.Нам потрібно лише один раз викликати цю функцію для кожного конвеєра в кожному кадрі, і все готово [14]

Звукові ефекти

Відтворити звук у Phaser дуже просто, потрібно лише заздалегідь завантажити звуковий ресурс. Потім викличте метод відтворення для відтворення.

Спочатку використовуйте *game.load.audio(),* щоб завантажити звуковий ресурс. Давайте візьмемо для прикладу звук, який відтворюється під час

забивання в цій грі, і попередньо завантажимо звуковий ресурс у метод попереднього завантаження стану:[18] [1]

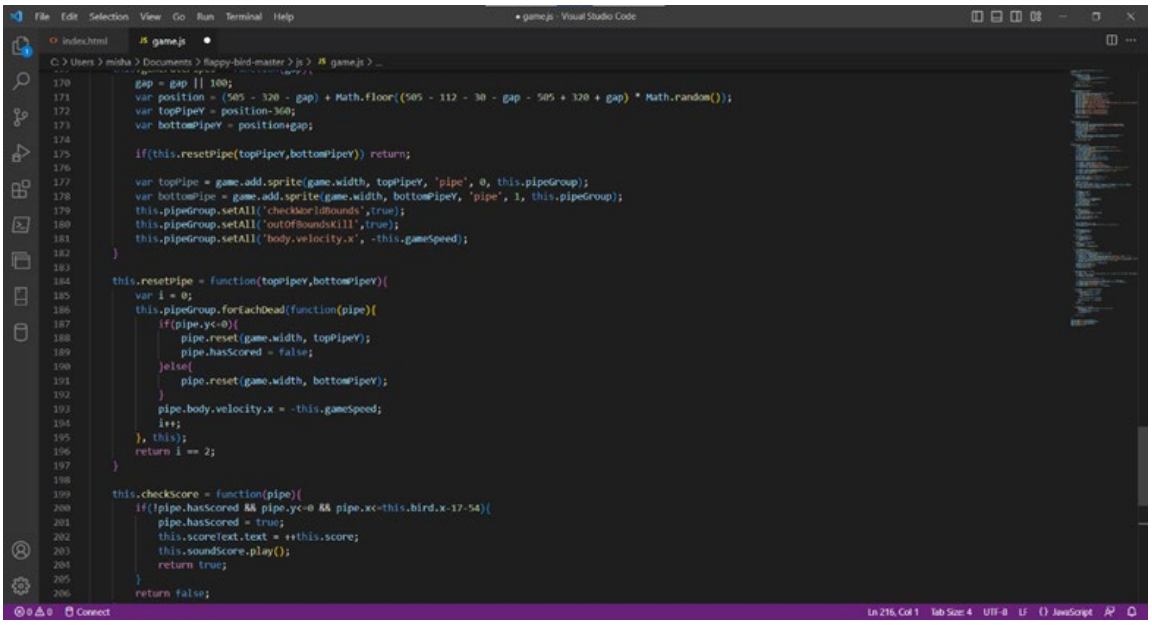
```
this.update = function(){
    if(!this.hasStarted) return;
    game.physics.arcade.collide(this.bird,this.ground, this.hitGround, null, this);
    game.physics.arcade.overlap(this.bird, this.pipeGroup, this.hitPipe, null, this);
    if(this.bird.angle < 90) this.bird.angle += 2.5;
    this.pipeGroup.forEachExists(this.checkScore,this);
}
```

- ***game.load.audio('score_sound', 'assets/score.wav');*** - Звуковий ефект оцінки

```
this.checkScore = function(pipe)
    if(!pipe.hasScored && pipe.y<=0 && pipe.x<=this.bird.x-17-54){
        pipe.hasScored = true;
        this.scoreText.text = ++this.score;
        this.soundScore.play();
        return true;
    }
    return false;
}
```

Потім отримати звуковий об’єкт через game.add.sound()

- ***this.soundScore = game.add.sound('score_sound'); -*** Звуковий об’єкт має багато методів для керування відтворенням і паузою звуку тощо. Щоб відтворити звук, вам потрібно лише викликати його метод відтворення.
- ***this.soundScore.play();*** - відтворення звуку



```
gap = gap || 100;
var position = (505 - 320 - gap) + Math.floor((505 - 112 - 30 - gap - 505 + 320 + gap) * Math.random());
var topPipeY = position-360;
var bottomPipeY = position+gap;

if(this.resetPipe(topPipeY,bottomPipeY)) return;

var topPipe = game.add.sprite(game.width, topPipeY, 'pipe', 0, this.pipeGroup);
var bottomPipe = game.add.sprite(game.width, bottomPipeY, 'pipe', 1, this.pipeGroup);
this.pipeGroup.setAll('checkWorldBounds',true);
this.pipeGroup.setAll('outOfBoundsKill',true);
this.pipeGroup.setAll('body.velocity.x', -this.gameSpeed);
}

this.resetPipe = function(topPipeY,bottomPipeY){
var i = 0;
this.pipeGroup.forEachDead(function(pipe){
if(pipe.y<0){
pipe.reset(game.width, topPipeY);
pipe.hasScored = false;
}else{
pipe.reset(game.width, bottomPipeY);
}
pipe.body.velocity.x = -this.gameSpeed;
i++;
}, this);
return i == 2;
}

this.checkScore = function(pipe){
if(!pipe.hasScored && pipe.y<=0 && pipe.x<=this.bird.x-17-54){
pipe.hasScored = true;
this.scoreText.text = ++this.score;
this.soundScore.play();
return true;
}
return false;
```

Рисунок 1.13 – Реалізація події *this.checkScore* та *game.load.audio*

1.8 Переваги та недоліки створення ігор на JS

Створення ігор на JavaScript (JS) має ряд переваг і недоліків.

Давайте вивчимо їх: Переваги створення ігор на JavaScript:

- Доступність: JavaScript — це широко підтримувана мова програмування, яка працює безпосередньо у веб-переглядачах, що робить її доступною для широкої аудиторії. Гравці можуть легко отримати доступ і грати в ігри на JavaScript, не вимагаючи додаткових установок або плагінів.
- Сумісність із різними платформами: ігри на JavaScript можна запускати на кількох платформах, включаючи настільні комп'ютери, смартфони, планшети та навіть смарт-телевізори, за умови, що пристрої мають сумісний веб-браузер. Ця крос-платформна сумісність забезпечує ширше розповсюдження та охоплення ваших ігор.
- Веб-інтеграція: JavaScript легко інтегрується з веб-технологіями, такими як HTML і CSS. Ця інтеграція дозволяє вам використовувати наявні знання та інструменти веб-розробки, полегшуючи створення інтерактивних і візуально привабливих ігор, які можна вставляти на веб-сайти або ділитися за допомогою веб-посилань.

- Швидке створення прототипів: динамічна природа і простий синтаксис JavaScript сприяють швидкому створенню прототипів та ітераційній розробці. Його гнучкість дозволяє розробникам швидко тестувати та впроваджувати ідеї, повторювати механізми ігрового процесу та вдосконалювати ігрові функції без значних витрат часу.
- Велике співтовариство та ресурси: у JavaScript є велика та активна спільнота розробників, що означає, що розробникам ігор доступні численні ресурси, навчальні посібники, фреймворки та бібліотеки. Ця мережа підтримки полегшує пошук рішень для поширених проблем, вивчення нових методів і бути в курсі останніх тенденцій у розробці ігор.[19]

Недоліки створення ігор на JavaScript:

- Обмеження продуктивності: JavaScript є інтерпретованою мовою, і порівняно з мовами нижчого рівня, такими як C++ або C#, він може мати обмеження продуктивності, особливо для складних і ігор з інтенсивним обчисленням. Однак оптимізація та ефективні методи кодування можуть певною мірою пом’якшити ці обмеження.
- Залежності від апаратного забезпечення: ігри на JavaScript залежать від продуктивності та обмежень пристроїв, на яких запущено веб-браузер. У результаті ви можете зіткнутися з відмінностями в продуктивності на різних пристроях, роздільній здатності екрану та конфігурації апаратного забезпечення.
- Обмежений доступ до системних ресурсів: ігри на JavaScript запускаються в межах ізольованого програмного середовища веб-браузера, яке обмежує прямий доступ до певних системних ресурсів. Це обмеження може вплинути на можливість використовувати функції апаратного забезпечення або отримати доступ до певних функцій низького рівня, необхідних для певних типів ігор.
- Сумісність з веб-переглядачами: хоча JavaScript широко підтримується, різні веб-браузери можуть мати незначні відмінності у реалізації JavaScript. Це може призвести до проблем сумісності, вимагаючи додаткового тестування та

потенційних обхідних шляхів, щоб переконатися, що ваша гра працює стабільно в різних браузерах.

- Обмежені можливості автономного режиму: ігри на JavaScript зазвичай потребують підключення до Інтернету, оскільки вони покладаються на веб-браузери. Хоча існують способи ввімкнення функціональності в автономному режимі за допомогою таких технологій, як Service Workers і локальне кешування, можливості в автономному режимі все одно можуть бути обмеженими порівняно з рідними програмами.[18]

Отже, Flappy Bird — проста, але захоплююча мобільна гра, яка набула величезної популярності. Хоча оригінальну гру було розроблено з використанням різних інструментів і технологій, давайте коротко розглянемо, як я розробив подібну гру за допомогою JavaScript і ігрового механізму Phaser:

Налаштував середовища:

- Встановив необхідні інструменти, такі як редактор коду, Visual Studio Code, і ігровий движок Phaser.
- Створив нову папку проекту та налаштував базову файлову структуру.

Створив ігрової сцени:

- Визначив ігрове полотно та налаштував ширину та висоту гри.
- Налаштував фізичну систему для обробки зіткнень і гравітації.
- Створив фонове зображення або колір для ігрової сцени.

Додав персонажа гравця:

- Створив спрайт для персонажа птаха та розмістив його на екрані.
- Застосував фізику до птаха, щоб увімкнути рух і ефект гравітації.
- Реалізував обробку введення для керування рухом птаха, як правило, за допомогою клавіатури або сенсорних подій.

Реалізував ігрову механіки:

- Створив труби або перешкоди як спрайти та розміщуйте їх на екрані.
- Визначив їхні моделі руху та швидкість, щоб імітувати прокручування.

- Налаштував виявлення зіткнень, щоб виявити, коли птах зіткнеться з трубами або межами гри, що призведе до завершення гри.

Оцінка та інтерфейс користувача:

- Впровадив систему нарахування очок, яка збільшується щоразу, коли гравець успішно проходить через набір каналів.
- Виведення рахунку на екран за допомогою текстових елементів.
- Показувати поточні та найкращі результати гравця, а також повідомлення про закінчення гри, коли гравець програє.

Логіка гри та обробка подій:

- Оброблував такі події, як зіткнення та умови гри, щоб керувати процесом гри.
- Реалізуйвав логіку, щоб перезапустити гру, коли гравець захоче грати знову.

Полірування та естетика:

- Додав візуальні ефекти, анімацію та звукові ефекти, щоб покращити враження від гри.
- Експериментував з візуальними стилями, кольорами та графічними ресурсами, щоб надати грі унікального вигляду та відчуття.

Тестування та оптимізація:

- Протестував гру на різних пристроях і розмірах екрана, щоб переконатися в сумісності та швидкості реагування.
- Оптимізував продуктивність за рахунок зменшення непотрібних обчислень, оптимізації завантаження ресурсів і мінімізації використання пам’яті.

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

В даному дипломному проекті було розроблене програмне забезпечення зі створенню 2D гри на ігровому середовища JavaScript. З його допомогою можна значною підвищити свій ігровий та розумовий досвід ще можна дуже весело проводити свій час. Використання створеного програмного забезпечення дозволить нам зрозуміти як створюють ігри та як на них заробляють.

Ефективність кожного програмного продукту визначається його якістю та ефективністю процесу розробки. Якість ПП визначається наступними складовими: з точки зору користувача; з позиції використання ресурсів; виконання вимог до програмного забезпечення.

Оцінка якості програмного продукту з точки зору користувача визначається необхідним на стадії функціонування розміром оперативної пам'яті ЕОТ, витратами машинного часу, пропускною спроможністю каналів передачі даних. Оцінка якості програмного продукту включає визначення трудомісткості і вартості його створення. Проведемо розрахунки визначення трудомісткості розробки даного програмного продукту.

2.2 Розрахунок ціни програмного продукту нормативним методом

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку.

Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначається обсяг програмних засобів, у тисячах умовних машинних команд програми аналога.[28]

2.3 Визначення трудомісткості розробки програмного забезпечення

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт.

Таблиця 2.1 - Каталог аналогів

Найменування ПП	Обсяг функції ПП – V_o, усл. машинних командах.
1. ПП СУБД	2500 – 9800
2. Комплексні системи ведення БД	950 – 7430
3. ПП введення інформації	1060 – 5750
4. ПП оптимізації розрахунків	1300 – 4200
5. ПП автоматизації засобів по каталогу	680 – 7000
6. ПП автоматизованих розрахунків	1300 – 8600
7. ПП загальної математики і ПП імітаційного моделювання	7800 – 8800
8. ПП організації обчислювального процесу	13000 – 10200
9. ПП оптимізаційних розрахунків	1300 – 4200

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл. 2.2.

Таблиця 2.2 - Машині Команди

Обсяг ПП, тис. умов. машинних команд	Норма часу, люд/год
1.00	229
2.00	244
3.00	262
4.00	283
5.00	306
6.00	330
7.00	357

8.00	385
9.00	414
10.00	445
12.00	510
14.00	580
16.00	654
18.00	731
20.00	812

На підставі отриманого значення, по таблиці 2.2, визначається укрупнена норма часу на розробку аналога програмного забезпечення, яка коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8:

$T^a p = 445$ (люд/годин).

Трудомісткість програмного продукту визначається по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

- Розробка технічного завдання

$$T_{ТЗ} = T^a p \times L_1 \times K_H \qquad (2.1)$$

- Розробка технічного проекту

$$T_{ТП} = T^a p \times L_2 \times K_H \qquad (2.2)$$

- Розробка робочого проекту

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \qquad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.3.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.4.);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.5.).

Таблиця 2.3 - Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Таблиця 2.4 - Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	Ступінь новизни	Значення $К_н$
А	Принципово нові ПП	1,75 – 1,2
Б	ПП – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПП маючий аналог	0,7

Тому що розробка системи є ПО, що має аналоги програмних продуктів, то код ступеня новизни для мого ПО – В, а значення коефіцієнта $К_н$= 0,7. По таблиці 2.4, знаючи код ступеня новизни, тепер можна визначити значення питомих коефіцієнтів трудомісткості (по таблиці 2.3):

L_1 = 0,12

L_2 = 0,11

L_3 = 0,61

Таблиця 2.5 - Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПП типовими програмами, %	Значення $К_т$
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

У розробленому програмному продукті використовується від 20 до 40 відсотків існуючих функцій, це значить, що $К_т$= 0,8 Тепер потрібно розрахувати трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$Т_{тз}=^{T^a}p *L_1*К_н= 445*0.12*0.7= 37,38 \quad \text{(люд/годин)} \quad (2.1)$$

Трудомісткість розробки технічного проекту

$$Т_{тп}=^{T^a}p *L_2*К_н =445*0.11*0.7= 34,265 \quad \text{(люд/годин)} \quad (2.2)$$

Трудомісткість розробки робочого проекту

$$Т_{рп}=^{T^a}p *L_3*К_н*К_т= 445*0.61*0.7*0.8=152,012 \quad \text{(люд/годин)} \quad (2.3)$$

Для подальших розрахунків необхідно визначити кількість папера, витраченого на кожен етап. $N_{тз}$= 13 (стр), $N_{тп}$= 20 (стр), $N_{рп}$= 25 (стр), $N_{пз}$= 5 (стр) – технічне завдання, розробка технічного проекту, розробка робочого проекту, пояснювальна записка відповідно. Розрахунок зведений у таблицю 2.6

Таблиця 2.6 - Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, години		
1	Розробка ПП	Контроль керівника	Нормоконтроль
1.ТЗ	$Т_{Ртз}$= 37,38	$Т_{кк}$=0,7*13= 9,1	$Т_{нк}$=0,15*13= 1,95
2.Розробка ТП	$Т_{Ртп}$= 34,265	$Т_{кк}$=0,7*20= 14	$Т_{нк}$=0,15*20= 3

3.Розробка РП	$T_{Ррп}$= 152,012	$T_{кк}$=0,7*25= 17,5	$T_{нк}$=0,15*25= 3,75
4.Розробка пояснювальної записки	$Tр_{пз}$=1,5*$N_{пз}$= 7,5	$T_{кк}$=0,7*13= 9,1	$T_{нк}$=0,15*5 0,75
Усього, в т.ч.:	T = 231,15	49,7	9,45
- на розробку	$\Sigma T_р$= 231,157		
- контроль керівника		$\Sigma T_{кк}$= 49,7	
- нормоконтроль			$\Sigma T_{нк}$= 9,45

На основі таблиці 2.6 розрахуємо тривалість розробки в роках:

Тпп=Т/(8.0*0.73*360) = 231,15/(8.0*0.73*360) = 1 (р), (2.4)

Де 8,0 – тривалість робочого дня;

0,73 – коефіцієнт перекладу в календарні дні;

2.4 Розрахунок ціни програмного продукту

У цьому розділі для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, вартість машино – години і витрати на розробку ПО. Розрахунок основної заробітної плати виконавців приведений у таблиці 6.7. Відповідно до статті 8 «Закону про Державний бюджет України на 2023» встановлено мінімальну заробітну плату у місячному розмірі з 1 січня 2023 року – 6700 гривень; мінімальну погодинну тарифну ставку – 40.43 грн.[30}

Таблиця 2.7 - Розрахунок основної заробітної плати виконавців

Найменування робіт	Трудомісткість робіт, роб. години	Годинна тарифна ставка, грн.	Розрахунок, грн.
1.Розробка ПП	231,157	40,43	9 357,23
2.Контроль керівника	49,7	40,43	2 009.37

3.Нормоконт-роль	9,45	40,43	382,06
Усього ($З_о$)	290,7		Σ Зо= 11 748,66

Розрахунок основної заробітної плати виконуємо по формулі:

$$Зо = Тмj \times Згод, \quad (2.5)$$

Де $Т_{мj}$ – трудомісткість j – того виду робіт, робоч. год;

Згод – погодинна тарифна ставка, грн.

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в таблицю 2.8

Таблиця 2.8 - Розрахунок матеріальних витрат на розробку ПО

Найменування матеріальних витрат	Тип, модель	Кількість, шт	Ціна одиниці, грн.	Вартість, грн.
Папір	Лист	85	2	170
Фотокартки	Фото	10	15	150
Разом	Лист, Фото	95	30	$В_{м\ i}$ = 320
Транспортно – заготівельні Витрати 10%				$Втр\_з = 0{,}1 \times Вм1$ = 32
Усього				$В_м=В_{м1}+В_{тр\_з=}$ 352

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.9

Таблиця 2.9 - Розрахунок статей витрат планової собівартості

Стаття витрат	Значення, грн.	Формула розрахунку
1. Матеріали	352	$В_м$ (див. табл. 6.8)
2. Основна заробітна плата	11 748,66	$З_о$ (див. табл. 6.7)

3.Додаткова заробітна плата	1 174,86	$Зд = 0{,}1 \times Зо$ = 0,1 * 11 748,66
4.Відрахування до єдиного фонду соціального внеску	2 843,17	$Вє.с.в. = 0{,}22 \times (Зо + Зд)$ = 0,22*(11 748,66 + 1 174,86)
5.Накладні Витрати	7 049,19	$Внак. = 0{,}6 \times Зо$ = 0,6 * 11 748,66
6. Повна собівартість	23 167,88	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак = 352 + 11 748,66 + 1 174,86 + 2 843,17+ 7 049,19

Розмір прибутку, що включається в ціну, визначається по наступній формулі:

$$П=(С_{пов}*Р)/100 = 23\ 167{,}88*10/100 = 2\ 316{,}78 \text{ (грн)}, \quad (2.6)$$

Де р – плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$$Ц_о=С_{пов}+П=23\ 167{,}88+2\ 316{,}78 = 25\ 484{,}68 \text{ (грн)}; \quad (2.7)$$

Податок на додану вартість визначається по наступній формулі:

$$ПДВ = 0.2 * Ц_о = 0{,}2*25\ 484{,}68 = 5\ 096{,}93 \text{ (грн)}; \quad (2.8)$$

Виходячи з отриманих даних, ціна реалізації розробленого програмного продукту на основі наступної формули, становитиме:

$$Ц_р=Ц_о+ПДВ = 25\ 484{,}68+ 5\ 096{,}93 = 30\ 581{,}61 \text{ (грн)}; \quad (2.9)$$

3 ОХОРОНА ПРАЦІ

3.1 Вступ

Охорона праці стосується заходів і правил, які вживаються для захисту прав, добробуту та умов праці працівників. Творці ігор, зокрема програмісти, художники, дизайнери та письменники, вносять свої навички та досвід у розробку складних та інноваційних віртуальних світів. Однак інтенсивний характер розробки ігор часто призводить до подовження робочого дня, стислих дедлайнів і надмірного стресу, що може призвести до виснаження, проблем із психічним здоров'ям і зниження задоволеності роботою.

Отже, задоволення потреб розробників ігор щодо охорони праці стає вирішальним для розвитку стійкої та процвітаючої галузі. Визнаючи важливість охорони праці для творців ігор, різні зацікавлені сторони виступають за реформи та ініціативи для покращення умов праці та забезпечення справедливого ставлення в ігровій індустрії [20]. Розробники ігор виконують свою роботу за допомогою персональних комп'ютерів, тому можна прирівняти їх працю до праці користувачів (програмістів) ПК.

3.2 Аналіз небезпечних і шкідливих факторів, що впливають на програміста при розробці ігор даного програмного комплексу

Розробка ігор передбачає роботу працівників з передовими технологіями, включаючи потужне обладнання, складне програмне забезпечення та потенційно небезпечні матеріали., що може призвести до проблем з опорно-руховим апаратом, напруги очей та інших проблем зі здоров'ям.

Запровадження ергономічних принципів на робочому місці є життєво важливим для мінімізації цих ризиків. Надання регульованих і зручних робочих станцій, ергономічних клавіатур і мишок, а також сприяння правильній поставі можуть значно зменшити ймовірність травм і дискомфорту, пов'язаних з роботою. Регулярні перерви та заохочувальні вправи на розтяжку також можуть допомогти зменшити фізичний стрес.[23]

3.3 Гігієнічні вимоги до виробничого середовища

Регулярне оцінювання та покращення умов праці має бути постійним процесом. Необхідно вживати проактивних заходів для впровадження необхідних змін і створення робочого середовища, яке відповідає мінливим потребам і очікуванням розробників ігор.

Забезпечення належної температури, вологості, повітряного потоку, контролю шуму, освітлення та ергономічних умов сприяє добробуту та продуктивності розробників ігор, сприяючи їхньому загальному здоров'ю та безпеці.[23]

3.4 Вимоги до організації робочого місця працівника

Організація робочого місця відіграє вирішальну роль у сприянні ефективності, співпраці та творчості серед працівників, які створюють ігри в студіях розробників. Добре спроектоване та продумано структуроване робоче місце може значно сприяти загальному успіху та продуктивності розробників ігор.

Робочі станції розташовують біля вікон , щоб отримати максимальний доступ до природного денного світла. Робочі станції мають бути організовані таким чином, щоб полегшити спілкування та сприяти командній роботі. Ергономічні потреби передбачають регульовані столи, ергономічні стільці. Меблі з ергономічним дизайном, у тому числі стільці, столи та монітори, повинні бути забезпечені для сприяння правильній поставі, зменшення напруги опорно-рухового апарату та запобігання повторюваним стресовим травмам.

Забезпечення розробників ігор необхідними ресурсами має вирішальне значення для їх продуктивності та творчості. Це включає доступ до потужного обладнання, програмного забезпечення, бібліотек і наборів для розробки.

Передбачають створення спеціальних просторів для спільної роботи на робочому місці. Добре організоване робоче місце не тільки підвищує ефективність особистості та команди, але й сприяє загальному успіху проектів розробки ігор.[20]

3.5 Мікроклімат

Мікроклімат відноситься до конкретних умов навколишнього середовища на робочому місці, включаючи температуру, вологість, якість повітря та потік повітря. Підтримка здорового та комфортного мікроклімату має вирішальне значення для здоров'я та безпеки праці розробників ігор.

Температуру в робочому приміщенні слід регулювати, щоб забезпечити комфортне робоче середовище. Належний рівень вологості є важливим для добробуту розробників ігор. Надмірна вологість може викликати дискомфорт, а низька може викликати сухість і подразнення. Підтримка збалансованого рівня вологості, як правило, від 40% до 60%, допомагає запобігти таким проблемам, як проблеми з диханням, сухість очей і захворювання шкіри.[23]

Правильний потік повітря допомагає регулювати температуру, зменшує накопичення запахів і мінімізує концентрацію забруднень у повітрі. Важливо підтримувати функціональні системи вентиляції, регулярний повітрообмін і належну фільтрацію повітря для сприяння здоровому робочому середовищу.

3.6 Безпека праці

Забезпечення безпеки розробників ігор вимагає впровадження належних заходів безпеки. Це включає в себе надання засобів захисту, якщо це необхідно, наприклад захисних окулярів, рукавичок і ергономічних стільців, а також регулярне технічне обслуговування та перевірку обладнання для запобігання нещасним випадкам [23].

3.7 Освітлення приміщень

Правильне освітлення має вирішальне значення для здоров'я та безпеки праці розробників ігор. Недостатнє або неправильне освітлення може призвести до напруги очей, головного болю, зниження продуктивності та інших проблем зі здоров'ям. [25]

Робочий простір повинен мати достатнє природне або штучне освітлення, рівномірно розподілене, уникаючи відблисків або тіней на екранах комп'ютерів.

Регульовані параметри освітлення можуть дозволити розробникам ігор персоналізувати свої переваги освітлення на основі завдань і візуального комфорту. Освітлення має бути рівномірним і достатнім, щоб уникнути відблисків або тіней на екранах комп'ютерів. Рекомендовані рівні освітлення -500-750 люкс для роботи в офісі та 750-1000 люкс для завдань, які вимагають більшої точності зору.

Природне освітлення створює більш приємне та візуально комфортне середовище. Відблиски та надмірна яскравість контролюються за допомогою жалюзі, штор або плівки для вікон.

Штучне освітлення використовується щоб підтримувати постійний рівень освітлення в усьому робочому просторі. Освітлювальні прилади забезпечують рівномірне і розсіяне освітлення. Регульоване робоче освітлення може бути корисним для доповнення освітлення для конкретних робочих зон або завдань, дозволяючи розробникам ігор персоналізувати своє освітлення відповідно до своїх потреб. Для більшості офісних середовищ підходить колірна температура близько 4000-5000 K (нейтральний білий), оскільки вона забезпечує збалансоване природне освітлення. Джерела мерехтіння світла можуть спричинити втому очей, головний біль і труднощі з фокусуванням. Світлодіодне освітлення зазвичай є кращим через його низьку швидкість мерехтіння порівняно з люмінесцентним освітленням.

Освітлювальні прилади розташовані так, щоб уникнути прямих відблисків на екранах комп'ютерів або відбиваючих поверхнях. Регульовані освітлювальні прилади можна розташувати під відповідним кутом, щоб забезпечити сфокусоване освітлення, не створюючи візуального дискомфорту. Достатній рівень освітлення, врахування природного освітлення, правильна колірна температура, освітлення без мерехтіння, ергономічне розташування та регулярне технічне обслуговування сприяють здоров'ю та безпеці розробників ігор, зменшуючи ризик напруги очей і пов'язаного з цим дискомфорту.

Студії розробки ігор можуть бути шумними через наявність кількох робочих станцій, обладнання та спільних заходів. Надмірний рівень шуму може

призвести до стресу, зниження концентрації та пошкодження слуху. Застосування заходів для зменшення шуму, як-от звукоізоляція стін, використання навушників із шумозаглушенням або створення тихих зон, може допомогти захистити слух і благополуччя розробників ігор.[20]

3.8 Пожежна безпека

Пожежна безпека є критично важливим аспектом охорони праці . Протипожежні заходи:

- визначати потенційну небезпеку пожежі на робочому місці та оцініть їхню ймовірність і потенційний вплив. Ця оцінка допомагає визначити відповідні профілактичні заходи.[23]
- підтримувати чисте та організоване робоче місце, щоб звести до мінімуму накопичення легкозаймистих матеріалів, таких як папір, упаковка чи безлад.
- зберігати легкозаймисті речовини, такі як засоби для чищення, розчинники або фарби, у спеціально відведених місцях подалі від джерел займання та переконайтеся, що вони зберігаються в дозволених контейнерах.
- регулярно перевіряти електричні системи, електропроводку та обладнання, щоб запобігти електричним несправностям, які можуть призвести до пожежі. Заохочувати безпечні практики, як-от уникнення перевантажених розеток і своєчасне повідомлення про будь-які проблеми з електрикою.
- забороняти паління на робочому місці або виділити спеціальні місця для куріння подалі від легкозаймистих матеріалів і з належними засобами для утилізації.

Протипожежне обладнання та системи:

- встановити відповідні вогнегасники в доступних місцях на робочому місці (як правило це вуглекислотні та порошкові вогнегасники);
- встановлення та обслуговування систем пожежної сигналізації, які включають детектори диму, теплові сповіщувачі та ручні сповіщувачі.
- встановити аварійне освітлення, щоб забезпечити видимість і безпечну евакуацію під час відключення електроенергії або в умовах задимленості.

- розглянути можливість встановлення автоматичних спринклерних систем для забезпечення швидкого реагування на локалізацію або гасіння пожежі.

Розроблений комплексний план екстреної евакуації, який включає маршрути евакуації, пункти збору та процедури для різних сценаріїв. Всі співробітники мають бути ознайомлені з планом. Регулярні тренування та навчання ознайомлюють працівників із процедурами евакуації та виходами. Проводити навчання щодо реагування на надзвичайні ситуації, пов’язані з пожежею, включно з правильним використанням вогнегасників і методів евакуації. Чіткі канали зв'язку для сповіщення співробітників у разі пожежі. Це може включати системи внутрішнього зв’язку, системи масового сповіщення або призначений персонал, відповідальний за подачу тривоги. Обов’язковий інструктаж з пожежної безпеки для всіх працівників, у тому числі для нових працівників.

Впроваджуючи заходи протипожежної безпеки, забезпечуючи належне навчання та розвиваючи культуру пожежної безпеки, студії розробки ігор можуть зменшити ризик виникнення пожеж і забезпечити добробут своїх співробітників. Для підтримки безпечного робочого середовища вкрай важливо дотримуватися місцевих правил пожежної безпеки, проводити регулярні оцінки та адаптувати протоколи пожежної безпеки, якщо це необхідно [22].

ВИСНОВКИ

Темою даної дипломної роботи є розробка гри типу Arcade на мові програмування JavaScript. Для досягнення мети було розглянуто ігрові середовища та представлено усі етапи створення програмного продукту, розроблено гру на мові програмування JavaScript, а також проаналізовано питання економічної доцільності розробки та опрацьовано матеріал з охороні праці.

Проведений детальний аналіз можливостей мови JavaScript доводить, що це універсальна мова - гнучкість JavaScript і широка підтримка браузера роблять його чудовим вибором для створення ігор. Його здатність обробляти динамічний вміст, взаємодіяти з елементами HTML і маніпулювати моделлю об’єктів документа (DOM) робить його добре придатним для розробки ігор. Фреймворки та бібліотеки підвищують продуктивність. Ці інструменти забезпечують готові функції, такі як обробка графіки, фізики та введення, зменшуючи час і зусилля, необхідні для розробки гри з нуля.

Реалізація добре структурованої архітектури гри має вирішальне значення для управління складністю та забезпечення зручності коду. Розділення завдань на модулі, використання принципів об’єктно-орієнтованого програмування та дотримання шаблонів проектування, таких як система сутності-компонента (ECS), може призвести до більш чистих і керованих кодових баз.

Оптимізація продуктивності має життєво важливе значення: оскільки ігри вимагають рендерингу в реальному часі та інтерактивності, оптимізація продуктивності є надзвичайно важливою. Такі методи, як об’єднання об’єктів, групування спрайтів і мінімізація операцій DOM, можуть значно покращити швидкість реакції та плавність гри. Ретельне тестування та налагодження є життєво важливими для забезпечення приємних ігор без помилок. Такі інструменти, як Jest, Mocha або Jasmine, можна використовувати для модульного тестування, а інструменти розробника браузера допомагають перевіряти та налагоджувати код JavaScript.

Розробка гри з упором на досвід користувача є ключем до успіху. Інтуїтивно зрозумілі елементи керування, привабливі візуальні ефекти, захоплюючий геймплей і добре продумані рівні сприяють захоплюючому та приємному ігровому досвіду.

JavaScript дозволяє крос-платформну розробку, дозволяючи запускати ігри на різних пристроях і операційних системах. Використання принципів адаптивного дизайну та забезпечення сумісності з різними браузерами та розмірами екрану покращує охоплення та доступність гри.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мова програмування JavaScript [Електроний ресурс] - https://uk.javascript.info/js
2. Беляев С. А. Розробка ігор на мові JavaScript: Навчальний посібник. - СПб.: Видавництво «Лань», 2016. - 128 с.
3. Bruce Johnson. Visual Studio Code: End-to-End Editing and Debugging Tools for Web Developers. Published by John Wiley & Sons, Inc.10475 Crosspoint Boulevard Indianapolis, IN 46256.193 c.
4. Douglas Crockford. How JavaScript Works. 279 c.
5. Мінник, Кріс, Холланд, Єва. М62 JavaScript для чайників: Пер. з англ. – М.: ТОВ “І.Д. Вільяме”, 2017. – 320 с. ил. - Парал. тит. англ.
6. Ackermann, Philipp, author. JavaScript : the comprehensive guide. 2018.
7. Roberto Brunetti Vanni Boncinelli. Advanced Windows Store App Development Using HTML5 and JavaScript. 435 c. O’Reilly Media, Inc.1005 Gravenstein Highway NorthSebastopol, California 95472
8. Alessandro Del Sole. Visual Studio Code Distilled: Evolved Code Editing for Windows, macOS, and Linux. 268 c. Cremona, Italy.
9. Marc Wandschneider. Learning Node.js, Second Edition.769 c. Printed in the United States of America. First printing: December 2016.
10. Мови програмування [Електроний ресурс] - https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F
11. Яку мову програмування краще вибрати для старту в IT? [Електроний ресурс] - https://happymonday.ua/yaku-movu-programuvannya-vybraty-dlya-startu-v-it#:~:text=%D0%9E%D1%82%D0%B6%D0%B5%2C%20%D1%8F%D0%BA%D1%83%20%D0%BC%D0%BE%D0%B2%D1%83%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20%D0%BE%D0%B1%D1%80%D0%B0%D1%82%D0%B8,%D0%

B2%20%D1%8F%D0%BA%D1%96%D0%B9%20%D0%B2%D0%B8%20%D1%85%D0%BE%D1%87%D0%B5%D1%82%D0%B5%20%D1%80%D0%BE%D0%B7%D0%B2%D0%B8%D0%B2%D0%B0%D1%82%D0%B8%D1%81%D1%8F.

12. Мови програмування – список найпопулярніших [Електроний ресурс] – https://futurenow.com.ua/movy-programuvannya-spysok-najpopulyarnishyh/

13. 2D гра на Phaser [Електроний ресурс] – https://developer.mozilla.org/ru/docs/Games/Tutorials/2D_breakout_game_Phaser

14. Створення першої гри на Phaser. Частина 1 – Вступ [Електроний ресурс] – https://habr.com/ru/articles/324896/

15. Як створити аркадну гру за допомогою Phaser 3 [Електроний ресурс] – https://yocton.ru/phaser3/kak-sozdat-arkadnuyu-igru-s-pomoshyu-phaser-3

16. Phaser (game framework) [Електроний ресурс] – https://en.wikipedia.org/wiki/Phaser_(game_framework)

17. Travis Faas (Author). An Introduction to HTML5 Game Development with Phaser.js.284 с. A K Peters/CRC Press; 1st edition (August 12, 2016).

18. Stephen Gose. Phaser.js Game Design Workbook: Game development guide using Phaser JavaScript Game Framework.280 с. December 2, 2017.

19. Stephen Gose. Making Multiplayer Online Games: A Game Development Workbook for any Phaser JavaScript Gaming Framework.377 с. Scribl (November 7, 2016).

20. Винокурова Л. Е., Васильчук М. В., Гаман М. В. Основи охорони праці: Підручн. для проф. -техн. навч. закладів. - 2-ге вид., допов., перероб. - К. : Вікторія, 2001. - 192 с.

21. Голінько В.І. Г 60 Основи охорони праці: підручник / В.І. Голінько; М-во освіти і науки України; Нац. гірн. ун-т. – 2-ге вид. – Д.: НГУ, 2014. – 271 с.

22. Основи охорони праці:Підручник. 2ге видання, доповнене та перероблене. / К.Н.Ткачук, М.О.Халімовський, В. В. Зацарний, Д. В. Зеркалов, Р. В. Сабарно, О. І. Полукаров, В. С. Коз'яков, Л. О. Мітюк. За ред. К. Н. Ткачука і М. О. Халімовського. — К.: Основа, 2006 — 448 с.

23. Грищук М. В. Осюв охорони праці: Підручник. -К.: Кондор. 2007 -.. 240 с.

24. Охорона праці на підприємствах та в організаціях: Бібліографічний покажчик / Укладач О.Ю. Бут. – Запоріжжя: ЗНТУ, 2009. – 23 с.

25. Одарченко М.С., Одарченко А.М., Степанов В.І., Черненко Я.М.. «Основи охорони праці». - 2017.

26. Економіка підприємства : підручник / під заг. ред. д.е.н., проф. Ковальської Л.Л. та проф. Кривов'язюка І.В. Київ : Видавничий дім «Кондор», 2020. 700 с.

27. Бойчик І.М Економіка підприємства: підручник. / І.М.Бойчик. - К.: Кондор -Видавництво, 2016. – 378 с.

28. Економіка підприємства: Підручник/ За ред. С. Ф. Покропивного. - К.: КНЕУ, 1999.

29. Економіка підприємства: навчальний посібник / О.І. Лисак, Л.О. Андрєєва, Л.О.Болтянська .- Мелітополь: Люкс, 2020. – 272 с.

30. Економіка підприємства: навч. посіб. / за заг. ред. Л. С. Шевченко. – Х.: Нац. ун-т «Юрид. акад. України ім. Ярослава Мудрого», 2011. – 208 с.

31. Економіка підприємства: навч. посіб. / [І. М. Посохов, В. Г. Дюжев, С. В. Сусліков, К. О. Тимофєєва] ; М-во освіти і науки України, Нац. техн. ун-т. «Харків. Політехн. ін-т». – Харків: НТУ «ХПІ», 2016. – 380 с.

Додаток 1. Код програми

```
var game = new Phaser.Game(320,505,Phaser.AUTO,'game');
game.States = {};

game.States.boot = function(){
	this.preload = function(){
		if(!game.device.desktop){
			this.scale.scaleMode = Phaser.ScaleManager.EXACT_FIT;
			this.scale.forcePortrait = true;
			this.scale.refresh();
		}
		game.load.image('loading','assets/preloader.gif');
	};
	this.create = function(){
		game.state.start('preload');
	};
}

game.States.preload = function(){
	this.preload = function(){
		var preloadSprite = game.add.sprite(35,game.height/2,'loading');
		game.load.setPreloadSprite(preloadSprite);
		game.load.image('background','assets/background.png');
	game.load.image('ground','assets/ground.png');
	game.load.image('title','assets/title.png');
	game.load.spritesheet('bird','assets/bird.png',34,24,3);
	game.load.image('btn','assets/start-button.png');
	game.load.spritesheet('pipe','assets/pipes.png',54,320,2);
	game.load.bitmapFont('flappy_font', 'assets/fonts/flappyfont/flappyfont.png',
'assets/fonts/flappyfont/flappyfont.fnt');
```

```
        game.load.audio('fly_sound', 'assets/flap.wav');
        game.load.audio('score_sound', 'assets/score.wav');
        game.load.audio('hit_pipe_sound', 'assets/pipe-hit.wav');
        game.load.audio('hit_ground_sound', 'assets/ouch.wav');

        game.load.image('ready_text','assets/get-ready.png');
        game.load.image('play_tip','assets/instructions.png');
        game.load.image('game_over','assets/gameover.png');
        game.load.image('score_board','assets/scoreboard.png');
        }
        this.create = function(){
                game.state.start('menu');
        }
}

game.States.menu = function(){
        this.create = function(){

        game.add.tileSprite(0,0,game.width,game.height,'background').autoScroll(-
10,0);
                game.add.tileSprite(0,game.height-
112,game.width,112,'ground').autoScroll(-100,0);
                var titleGroup = game.add.group();
                titleGroup.create(0,0,'title');
                var bird = titleGroup.create(190, 10, 'bird');
                bird.animations.add('fly');
                bird.animations.play('fly',12,true);
                titleGroup.x = 35;
                titleGroup.y = 100;
```

```
            game.add.tween(titleGroup).to({ y:120
},1000,null,true,0,Number.MAX_VALUE,true);
            var btn =
game.add.button(game.width/2,game.height/2,'btn',function(){game.state.start('play');}
);
            btn.anchor.setTo(0.5,0.5);
        }
}

game.States.play = function(){
        this.create = function(){
            this.bg = game.add.tileSprite(0,0,game.width,game.height,'background');
            this.pipeGroup = game.add.group();
            this.pipeGroup.enableBody = true;
            this.ground = game.add.tileSprite(0,game.height-
112,game.width,112,'ground');
            this.bird = game.add.sprite(50,150,'bird');
            this.bird.animations.add('fly');
            this.bird.animations.play('fly',12,true);
            this.bird.anchor.setTo(0.5, 0.5);
            game.physics.enable(this.bird,Phaser.Physics.ARCADE);
            this.bird.body.gravity.y = 0;
            game.physics.enable(this.ground,Phaser.Physics.ARCADE);
            this.ground.body.immovable = true;

            this.soundFly = game.add.sound('fly_sound');
            this.soundScore = game.add.sound('score_sound');
            this.soundHitPipe = game.add.sound('hit_pipe_sound');
            this.soundHitGround = game.add.sound('hit_ground_sound');
```

```
        this.scoreText = game.add.bitmapText(game.world.centerX-20, 30, 'flappy_font', '0', 36);

        this.readyText = game.add.image(game.width/2, 40, 'ready_text');
        this.playTip = game.add.image(game.width/2,300,'play_tip');
        this.readyText.anchor.setTo(0.5, 0);
        this.playTip.anchor.setTo(0.5, 0);

        this.hasStarted = false;
        game.time.events.loop(900, this.generatePipes, this);
        game.time.events.stop(false);
        game.input.onDown.addOnce(this.statrGame, this);
    };
    this.update = function(){
        if(!this.hasStarted) return;
        game.physics.arcade.collide(this.bird,this.ground, this.hitGround, null, this);
        game.physics.arcade.overlap(this.bird, this.pipeGroup, this.hitPipe, null, this);
        if(this.bird.angle < 90) this.bird.angle += 2.5;
        this.pipeGroup.forEachExists(this.checkScore,this);
    }

    this.statrGame = function(){
        this.gameSpeed = 200;
        this.gameIsOver = false;
        this.hasHitGround = false;
        this.hasStarted = true;
        this.score = 0;
        this.bg.autoScroll(-(this.gameSpeed/10),0);
```

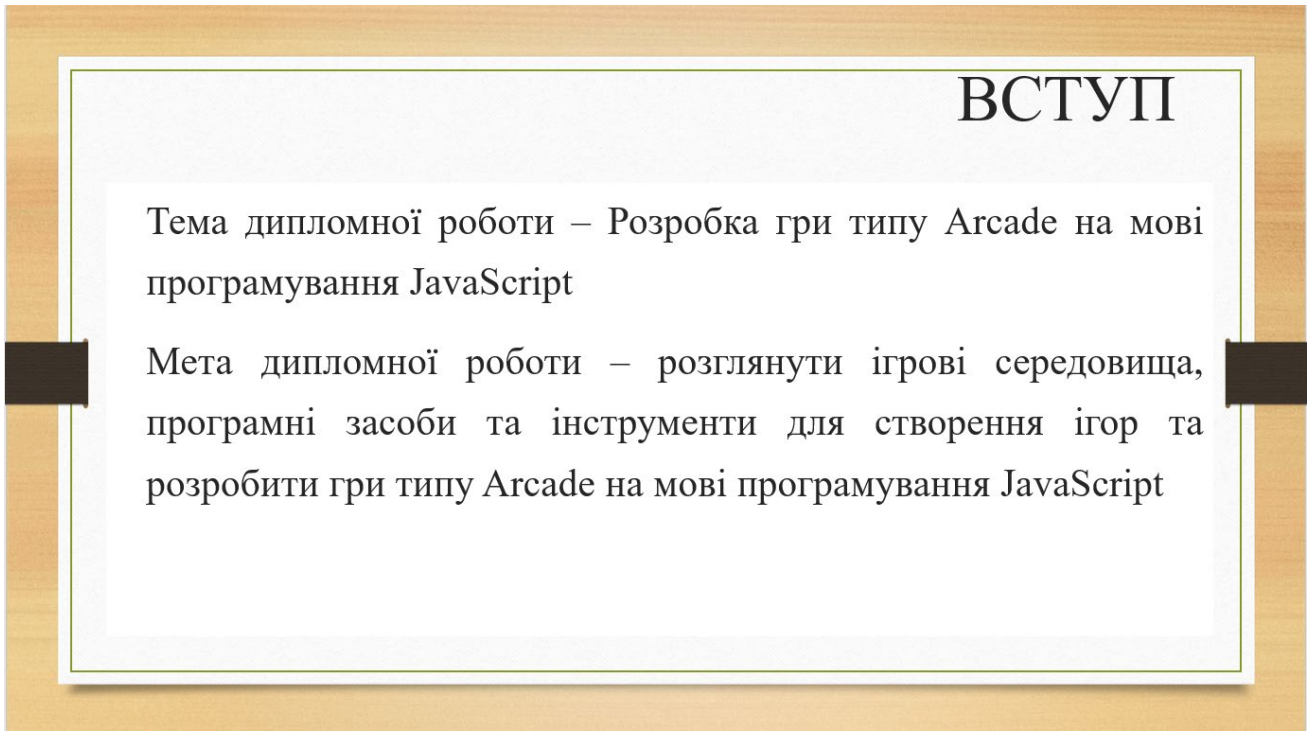
```
        this.ground.autoScroll(-this.gameSpeed,0);
        this.bird.body.gravity.y = 1150;
        this.readyText.destroy();
        this.playTip.destroy();
        game.input.onDown.add(this.fly, this);
        game.time.events.start();
}

this.stopGame = function(){
        this.bg.stopScroll();
        this.ground.stopScroll();
        this.pipeGroup.forEachExists(function(pipe){
                pipe.body.velocity.x = 0;
        }, this);
        this.bird.animations.stop('fly', 0);
        game.input.onDown.remove(this.fly,this);
        game.time.events.stop(true);
}

this.fly = function(){
        this.bird.body.velocity.y = -350;
        game.add.tween(this.bird).to({angle:-30}, 100, null, true, 0, 0, false);
        this.soundFly.play();
}

this.hitPipe = function(){
        if(this.gameIsOver) return;
        this.soundHitPipe.play();
        this.gameOver();
}
```

```
	this.hitGround = function(){
		if(this.hasHitGround) return;
		this.hasHitGround = true;
		this.soundHitGround.play();
		this.gameOver(true);
	}
	this.gameOver = function(show_text){
		this.gameIsOver = true;
		this.stopGame();
		if(show_text) this.showGameOverText();
	};

	this.showGameOverText = function(){
		this.scoreText.destroy();
		game.bestScore = game.bestScore || 0;
		if(this.score > game.bestScore) game.bestScore = this.score;
		this.gameOverGroup = game.add.group();
		var gameOverText = this.gameOverGroup.create(game.width/2,0,'game_over');
		var scoreboard = this.gameOverGroup.create(game.width/2,70,'score_board');
		var currentScoreText = game.add.bitmapText(game.width/2 + 60, 105, 'flappy_font', this.score+'', 20, this.gameOverGroup);
		var bestScoreText = game.add.bitmapText(game.width/2 + 60, 153, 'flappy_font', game.bestScore+'', 20, this.gameOverGroup);
		var replayBtn = game.add.button(game.width/2, 210, 'btn', function(){game.state.start('play');}, this, null, null, null, null, this.gameOverGroup);
		gameOverText.anchor.setTo(0.5, 0);
		scoreboard.anchor.setTo(0.5, 0);
		replayBtn.anchor.setTo(0.5, 0);
```

```
        this.gameOverGroup.y = 30;
    }

    this.generatePipes = function(gap){ gap = gap || 100;
        var position = (505 - 320 - gap) + Math.floor((505 - 112 - 30 - gap - 505 + 320 + gap) * Math.random());
        var topPipeY = position-360;
        var bottomPipeY = position+gap;

        if(this.resetPipe(topPipeY,bottomPipeY)) return;

        var topPipe = game.add.sprite(game.width, topPipeY, 'pipe', 0, this.pipeGroup);
        var bottomPipe = game.add.sprite(game.width, bottomPipeY, 'pipe', 1, this.pipeGroup);
        this.pipeGroup.setAll('checkWorldBounds',true);
        this.pipeGroup.setAll('outOfBoundsKill',true);
        this.pipeGroup.setAll('body.velocity.x', -this.gameSpeed);
    }

    this.resetPipe = function(topPipeY,bottomPipeY){
        var i = 0;
        this.pipeGroup.forEachDead(function(pipe){
            if(pipe.y<=0){
                pipe.reset(game.width, topPipeY);
                pipe.hasScored = false;
            }else{
                pipe.reset(game.width, bottomPipeY);
            }
            pipe.body.velocity.x = -this.gameSpeed;
```

```
                i++;
            }, this);
            return i == 2;
        }

        this.checkScore = function(pipe){
            if(!pipe.hasScored && pipe.y<=0 && pipe.x<=this.bird.x-17-54){
                pipe.hasScored = true;
                this.scoreText.text = ++this.score;
                this.soundScore.play();
                return true;
            }
            return false;
        }
}

game.state.add('boot',game.States.boot);
game.state.add('preload',game.States.preload);
game.state.add('menu',game.States.menu);
game.state.add('play',game.States.play);
game.state.start('boot');
```

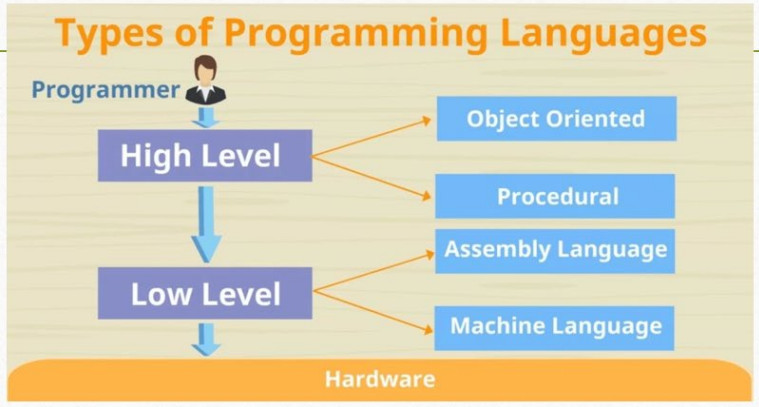
Додаток 2. Презентаційний матеріал до проекту



Історія мов програмування

May 2022	May 2021	Change	Programming Language	Ratings	Change
1	2	↑	Python	12.74%	+0.86%
2	1	↓	C	11.59%	-1.80%
3	3		Java	10.99%	-0.74%
4	4		C++	8.83%	+1.01%
5	5		C#	6.39%	+1.98%
6	6		Visual Basic	5.86%	+1.85%
7	7		JavaScript	2.12%	-0.33%
8	8		Assembly language	1.92%	-0.51%
9	10	↑	SQL	1.87%	+0.16%
10	9	↓	PHP	1.52%	-0.34%
11	17	⇈	Delphi/Object Pascal	1.42%	+0.22%
12	18	⇈	Swift	1.23%	+0.08%

Апаратне та програмне забезпечення розробника JavaScript



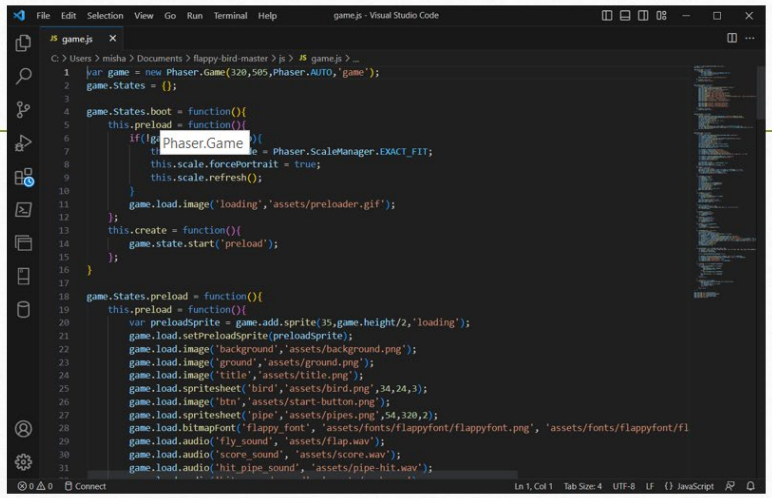
Ігрові движки (середовища)

Мобільні платформи (Android, IOS)

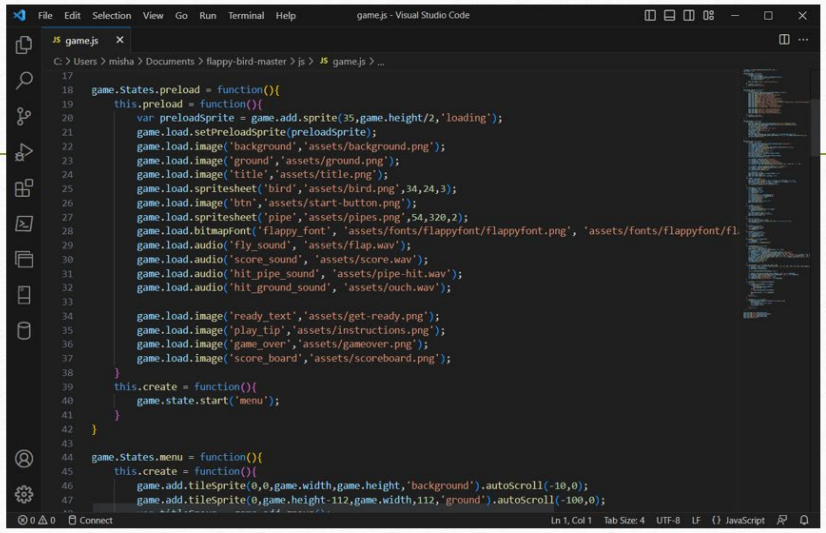
Розробка гри на JavaScript

Індикатор прогресу завантаження ресурсу гри

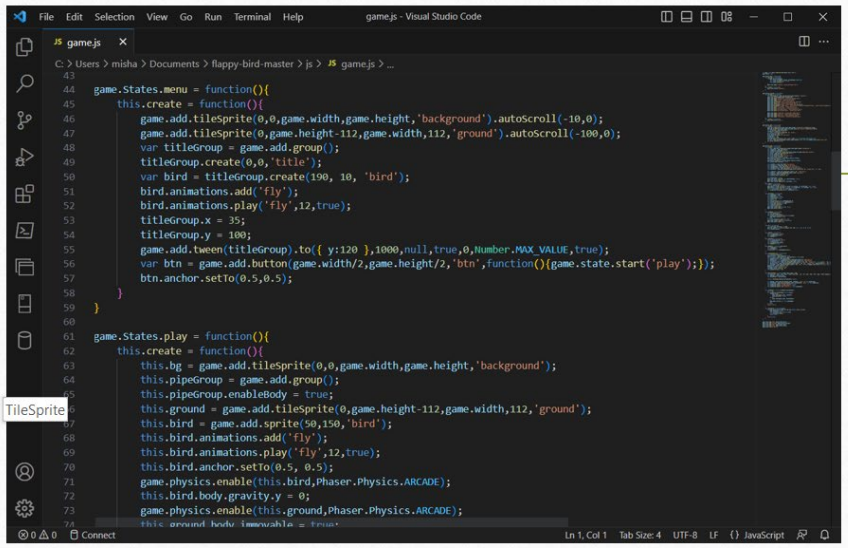
Phaser.Game



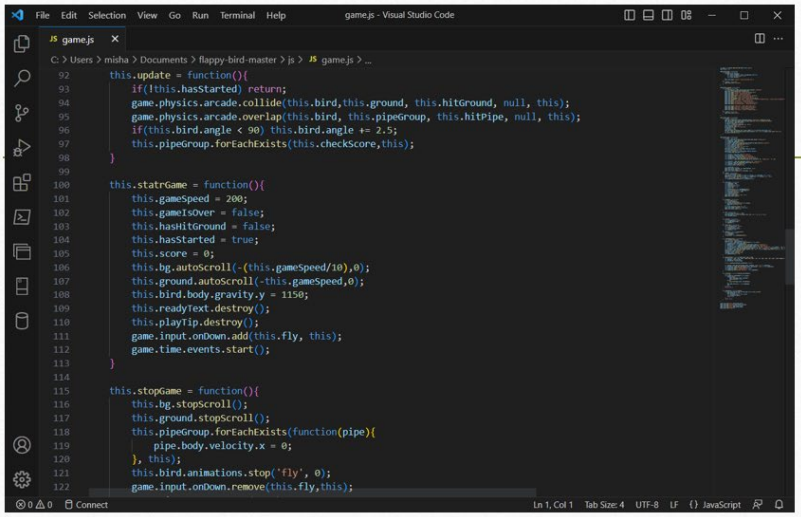
Loader



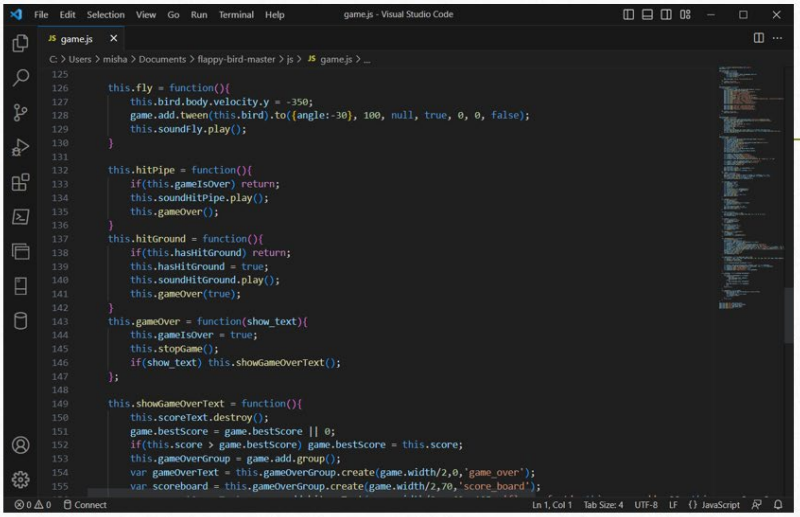
TileSprite



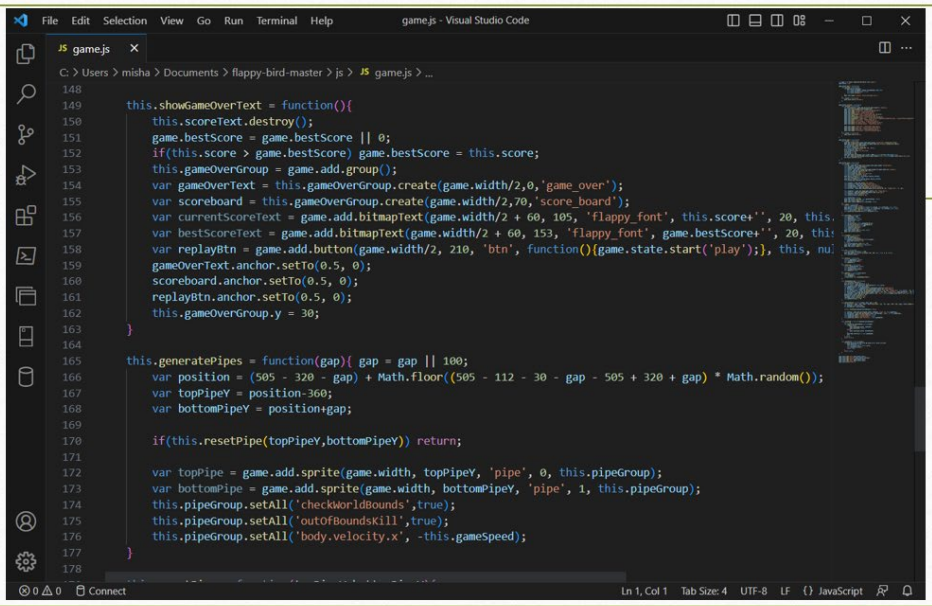
Tween



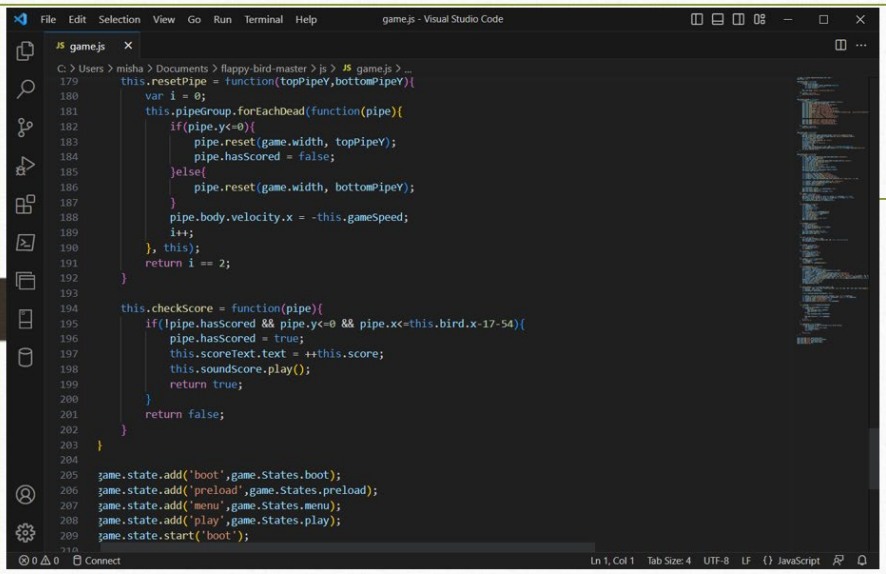
Фізика гри



Timer



Анімації гри



ВИСНОВКИ

Темою даної дипломної роботи є розробка гри типу Arcade на мові програмування JavaScript. Для досягнення мети було розглянуто ігрові середовища та представлено усі етапи створення програмного продукту, розроблено гру на мові програмування JavaScript, а також проаналізовано питання економічної доцільності розробки та опрацьовано матеріал з охороні праці.

Проведений детальний аналіз можливостей мови JavaScript доводить, що це універсальна мова - гнучкість JavaScript і широка підтримка браузера роблять його чудовим вибором для створення ігор. Його здатність обробляти динамічний вміст, взаємодіяти з елементами HTML і маніпулювати моделлю об'єктів документа (DOM) робить його добре придатним для розробки ігор. Фреймворки та бібліотеки підвищують продуктивність. Ці інструменти забезпечують готові функції, такі як обробка графіки, фізики та введення, зменшуючи час і зусилля, необхідні для розробки гри з нуля.

ДЯКУЮ ЗА УВАГУ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект (роботу) здобувача (здобувачки) освіти

відділення комп'ютерних систем

Тєрова Михайла Олександровича

(прізвище, ім`я та по батькові)

Спеціальність **123 "Комп'ютерна інженерія"**

Освітня програма **Комп'ютерна графіка та Web-дизайн**

Керівник дипломного проекту (роботи) **Скорняков Вячеслав Сергійович**

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи):

Розробки браузерної гри типу Arcade на мові програмування Java Script

Обсяг розрахунково-пояснювальної записки _____ сторінок

Обсяг графічної (презентаційної) частини _____ аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту (роботи) завданню

Дипломний проект повністю відповідає завданню до дипломного проектування

б) характеристика виконання кожного розділу дипломного проекту (роботи) _____

Пояснювальна записка дипломного проекту виконана у повному обсязі та відповідає поставленим завданням. Конкретизовано на основі проведеного теоретичного аналізу вимоги до дипломного проекту, визначено завдання та визначено технічні рішення, що дозволяють реалізувати завдання дипломного проекту – здійснено огляд популярних мов програмування, ігрових платформ. Описано поетапну розробку гри та створення коду

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту (роботи) _____

Презентаційні матеріали виконані якісно, демонстративно та відповідають вмісту теоретичного матеріалу

г) перелік позитивних якостей дипломного проекту (роботи)

Серед позитивних якостей проекту - детальний аналітичний огляд існуючих мов програмування, виважений підхід до реалізації завдань до дипломного проекту, вибору програмних засобів та інструментів, демонстрація гри

д) основні недоліки дипломного проекту (роботи)

В роботі зустрічаються відхилення від вимог щодо оформлення пояснювальної записки дипломного проекту.

Етапи розробки слід було описати докладніше

Оцінка розрахункової частини **4 (добре)**

Оцінка графічної частини **5 (відмінно)**

Загальна оцінка **4 (добре)**

Прізвище, ім'я, по батькові рецензента **Василіу Євген Вікторович**

Місце роботи і посада рецензента **Державний університет інтелектуальних технологій і зв'язку, д.т.н., проф. кафедри КБ та ТЗІ, декан факультету інформаційних технологій та кібербезпеки**

Підпис:

«16» червня 2023 р.

ПІДПИС ПОСВІДЧУЮ
НАЧАЛЬНИК ВІДДІЛУ КАДРІВ ДУІТЗ

ВІДДІЛ КАДРІВ

ВСП «Одеський технічний фаховий коледж ОНТУ»

ВІДГУК

Керівника на дипломний проект здобувача освіти

Тєрова Михайла Олександровича

(прізвище, ім`я та по батькові)

Спеціальність: *123 «Комп'ютерна інженерія»*

Тема дипломного проекту: *Розробка браузерної гри типу Arcade на мові програмування Java Script*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) Обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки): *Пояснювальна записка дипломного проекту виконана якісно, у повному обсязі. В дипломному проекті здобувачем проведено огляд існуючих технологій та мов програмування, розроблено гру, продемонстровано її роботу, наведено код программного продукту. В дипломному проекті в останніх розділах проаналізовано питання економічної доцільності та охорони праці. Створено презентацію до захисту.*

б) Самостійність роботи над проектом: *Здобувач самостійно визначався з напрямом роботи, дослухався до рекомендацій керівника дипломного проекту, своєчасно надавав результати роботи, якісно виконував основні етапи роботи за вимогою керівника.*

в) Теоретична підготовка випускника: *Теоретична підготовка випускника в цілому відповідає державним вимогам до фахівців відповідного рівня кваліфікації*

г) Вміння розв`язувати виробничі і конструкторські питання на базі останніх досліджень науки і техніки, передових методів виробництва

В процесі роботи над дипломним проектом здобувач продемонстрував уміння використовувати останні досягнення науки та техніки в предметній галузі на підставі відповідної навчальної та науково-технічної літератури, достатньо впевнено користувався програмним забезпеченням при роботі над дипломним проектом та створенням презентації.

Оцінка розрахункової частини *відмінно*

Оцінка графічної частини *добре*

Загальна оцінка *відмінно*

Прізвище, ім`я, по батькові *Скорняков В'ячеслав Сергійович*

Місце роботи і посада керівника дипломного проекту: викладач комісії КТ та ПІ_ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету»

Підпис

«12» 06 2023 р.

UNICHECK

Ім'я користувача:
Наталія Вікторівна Копусь

ID перевірки:
1015484030

Дата перевірки:
07.06.2023 14:02:23 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
07.06.2023 14:04:34 EEST

ID користувача:
100011688

Назва документа: **4КГ-06 Тєров Михайло**

Кількість сторінок: **50** Кількість слів: **10019** Кількість символів: **73165** Розмір файлу: **1.33 MB** ID файлу: **1015141247**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

7.76%
Схожість

Найбільша схожість: 4.12% з Інтернет-джерелом (http://www.meeetang.com/9750/h5game/Phaser/20190415_43842.html.

7.76% Джерела з Інтернету	233	..... Сторінка 52

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	26

Підозріле форматування	9 сторінок

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Тєров Михайло Олександрович,
здобувач освіти гр. 4КГ-06, та

Скорняков В'ячеслав Сергійович,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускної кваліфікаційної роботи молодшого спеціаліста на тему:

«Розробки браузерної гри типу Arcade на мові програмування Java Script» (автор роботи – Тєров М.О., керівник роботи – Скорняков В.С.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2023 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи, і даємо згоду на обробку персональних даних.

Виконавець _____ (Тєров М.О.)

Керівник _____ / Скорняков В.С./

«12» 06 2023 р.