МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА
Відокремлений структурний підрозділ «Фаховий коледж Чернівецького національного університету імені Юрія Федьковича»

Природниче відділення
Циклова комісія комп'ютерної інженерії

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітньо-професійного ступеня «фаховий молодший бакалавр»

зі спеціальності *123 Комп'ютерна інженерія*

підготовки за освітньо-професійною програмою «*Комп'ютерна інженерія*»

на тему: **«Логічна гра на основі графів Ейлера»**

Виконав:
студент 4 курсу, 41 групи
Олійник Владислав Ярославович_____
(підпис)
Керівник:
Букурос Олеся Вікторівна_____
(підпис)
Рецензент:
Тащук Олександр Юрійович_____
(підпис)

До захисту допущено
на засіданні циклової комісії
протокол №____від _____ 2024 р.
Голова ЦК _____ Тащук О.Ю.

Чернівці – 2024

Завдання та календарний план роботи

Міністерство освіти і науки України
Чернівецький національний університету імені Юрія Федьковича
Відокремлений структурний підрозділ «Фаховий коледж
Чернівецького національного університету імені Юрія Федьковича»

Затверджую
Голова циклової комісії,
Олександр ТАЩУК.

_____
«____» _____2023 р.

Завдання

На кваліфікаційну роботу	Олійник Владислав Ярославович
Тема кваліфікаційної роботи	Логічна гра на основі графів Ейлера
Постановка завдання в короткій формі	Розробка логічної гри для побудови циклів Ейлера на відповідних графах
Вихідні дані (2-3 адреси сайтів, матеріали яких рекомендує керівник кваліфікаційної роботи)	

Календарний план роботи

Дата отримання версії звіту керівником	Етап виконання роботи	Виконання (зазначає керівник)
10.10.23	Отримання завдання до кваліфікаційної роботи.	
02.11.23	Огляд літератури, присвяченої розробці гри на основі графів Ейлера.	
21.12.23	Аналіз існуючих конструкторських рішень. Оформлення розділу 1.	
27.01.24	Вибір компонентної бази, узгодження модулів системи.	
29.02.24	Написання теоретичного матеріалу по використовуваних модулях.	
30.03.24	Проектування та розробка апаратної частини, підключення Оформлення розділу 2.	
28.04.24	Проектування та розробка програмної частини, підключення. Оформлення розділу 3.	
17.05.24	Оформлення та редагування кваліфікаційної роботи.	

31.05.24	Подання роботи на перевірку Інтернет-сервісом Unicheck.	
03.06.24	Подання роботи на рецензію	
06.06.24	Представлення роботи на засіданні Циклової комісії	
за графіком	Захист кваліфікаційної роботи	

Дата видачі завдання _____.

Студент Владислав ОЛІЙНИК

Керівник (П.І.Б) Олеся БУКУРОС

АНОТАЦІЯ

В даній роботі виконано розробку алгоритмів, проектування коду, програмну реалізацію і тестування застосунку – логічної гри для побудови циклів Ейлера на відповідних графах, що має 9 рівнів складності, перехід між якими виконується тільки після проходження поточного рівня. Застосунок розроблений з використанням веб-технологій HTML5, CSS3, JS, jQuery, що дає йому певну кросплатформність, оскільки може бути запущений в браузері на персональному комп'ютері з будь-якою операційною системою.

Робота складається з 5 розділів, вступу, висновків, додатків з js-кодом та схемами застосунку. В першому розділі описана класифікація ігор, платформи та основні моменти для розробки ігор. Другий присвячений опису технологій розробки, зокрема актуальністю та перевагам розробки ігор з використанням мови програмування JS. В третьому розділі описана структура та проектування коду застосунку. Четвертий присвячений тестуванню, а п'ятий – охороні праці.

Дипломна робота містить 58 сторінок, 17 рисунків та 14 посилань на літературні джерела.

Ключові слова: граф, ребро, вершина, веб-застосунок, логічна гра, JS-гра, головоломка, Ейлер.

ABSTRACT

In this work, the development of algorithms, code design, software implementation and testing of the application - a logic game for building Euler cycles on the corresponding graphs, which has 9 levels of complexity, the transition between which is performed only after passing the current level - is performed. The application is developed using web technologies HTML5, CSS3, JS, jQuery, which gives it a certain cross-platform, as it can be launched in a browser on a personal computer with any operating system.

The work consists of 5 sections, introduction, conclusions, applications with js code and application schemes. The first chapter describes the classification of games, platforms and the main points for game development. The second is devoted to the description of development technologies, in particular, the relevance and advantages of game development using the JS programming language. The third section describes the structure and design of the application code. The fourth is devoted to testing, and the fifth to labor protection.

The thesis contains 58 pages, 17 figures and 14 references to literary sources.

Keywords: graph, edge, vertex, web application, logic game, JS game, puzzle, Euler.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Граф	–	Математична структура, яка використовується для моделювання відносин між об'єктами. Вона складається з набору вершин (або вузлів) і набору ребер (або дуг), які з'єднують пари вершин.
Вершини графа		Об'єкти, які з'єднуються ребрами. У графі вони зазвичай позначаються як точки або кола.
Ребра графа	–	Лінії, які з'єднують пари вершин. Вони можуть бути спрямованими або не спрямованими.
JS	–	JavaScript: це мова програмування, що використовується для створення інтерактивних елементів на веб-сторінках.
HTML	–	HyperText Markup Language: це мова розмітки, яка використовується для створення структури веб-сторінок.
CSS	–	Cascading Style Sheets: це мова стилів, що використовується для опису зовнішнього вигляду HTML-елементів на веб-сторінці.
WWW	–	World Wide Web: Це система взаємопов'язаних документів та інших ресурсів, що доступні через Інтернет.
ЕОМ	–	Електронно-обчислювальна машина
ПК	–	Персональний комп’ютер
ПП	–	Програмний продукт
ІТ	–	Інформаційні технології
ІС	–	Інформаційна система
ТЗ	–	Технічне завдання
ПЗ	–	Програмне забезпечення

ВСТУП

Актуальність теми. Створення веб-застосунку для персональних комп’ютерів, що підтримує всі операційні системи і зосереджений на логічній грі з побудови ейлерових циклів на графах, що дозволяє перейти до наступного рівня лише пройшовши попередній, є актуальним та перспективним проектом з кількох причин:

1. Розвиток логічного мислення: гра, що вимагає побудови ейлерових циклів, сприяє розвитку логічного мислення, стратегічного планування та навичок вирішення проблем. Такі завдання допомагають гравцям розвивати алгоритмічне мислення та розуміння графів, що є важливими темами в області математики та інформатики.

2. Підготовка до академічних курсів: для студентів, що вивчають теорію графів, дискретну математику чи комп'ютерні науки, ця гра може служити ефективним інструментом для закріплення теоретичних знань через практичні завдання. Вона дозволяє краще зрозуміти принципи побудови ейлерових циклів і застосовувати ці знання на практиці.

3. Навчання через гру: використання гри як навчального інструменту є ефективним підходом для залучення уваги та утримання інтересу учнів. Гравці засвоюють складні концепції у легкій і невимушеній формі, що підвищує їх мотивацію до вивчення.

4. Цікавість та залучення: логічні ігри з графами часто викликають інтерес завдяки своїй інтелектуальній складності та можливості для гравців постійно вдосконалювати свої навички. Гра, яка пропонує кілька рівнів складності, забезпечує довготривалий інтерес, оскільки гравці прагнуть пройти всі рівні.

5. Челендж та досягнення: можливість переходу на наступний рівень лише після завершення поточного додає грі елемент виклику та досягнення. Це мотивує гравців докладати більше зусиль і відчувати задоволення від своїх успіхів.

6. Доступність: веб-застосунок, який може працювати на будь-якій

операційній системі, забезпечує широкий доступ до гри. Користувачі можуть грати на своїх улюблених пристроях, що робить гру зручною та легко доступною.

7. Розвиток когнітивних навичок: логічні ігри, як-от побудова ейлерових циклів, сприяють розвитку когнітивних навичок, включаючи пам'ять, увагу, концентрацію та просторове мислення.

8. Зміцнення знань у галузі комп'ютерних наук: ігри на основі графів сприяють глибшому розумінню структур даних та алгоритмів, що є фундаментальними елементами комп'ютерних наук. Вони також можуть надихати гравців на подальше вивчення цієї області.

9. Платформа для творчості та інновацій: розробка та використання такої гри може стати платформою для експериментів з новими алгоритмами, підходами до вирішення задач та навіть створення нових типів графових задач.

Підсумовуючи написане, зробимо висновки, що створення веб-застосунку для гри в побудову ейлерових циклів є не лише актуальним, але й надзвичайно корисним проектом, який поєднує в собі освітні, розважальні та розвиваючі аспекти. Такий застосунок сприятиме розвитку важливих навичок, забезпечуючи при цьому захоплююче та мотивуюче ігрове середовище.

Об’єктом дослідження є процес взаємодії користувача з веб-застосунком, призначеним для вирішення задач побудови ейлерових циклів.

Предметом дослідження є розробка веб-застосунку для персональних комп'ютерів, що підтримує всі операційні системи, з логічною грою, яка фокусується на побудові ейлерових циклів на графах.

Метою роботи є розробка функціонального та зручного веб-застосунку, який надає користувачам можливість вирішувати задачі з побудови ейлерових циклів на графах, сприяючи розвитку їх логічного мислення та навчальних навичок.

В рамках цієї мети потрібно виконати такі завдання:

- провести огляд наукових праць з теорії графів;
- провести огляд освітніх та логічних ігор;
- розробити концепцію гри: основні механіки, правила та рівні;
- створити дизайн користувацького інтерфейсу;
- спроектувати алгоритми для генерування графів;
- реалізувати алгоритми перевірки ейлерових циклів;
- реалізувати програмний код застосунку;
- провести функціональне тестування для перевірки роботи алгоритмів;
- провести користувацьке тестування.

Методи дослідження:

1. Аналіз літератури: вивчення наукових праць та існуючих рішень у галузі теорії графів, ейлерових циклів та освітніх ігор.
2. Розробка алгоритмів: проектування та впровадження алгоритмів для побудови та перевірки ейлерових циклів у графах.
3. Програмна інженерія: використання методів програмної інженерії для розробки веб-застосунку, включаючи дизайну інтерфейсу користувача, написання коду та тестування.

Практична цінність. Розробка веб-застосунку для побудови ейлерових циклів має значну практичну цінність, поєднуючи освітні, розважальні та дослідницькі аспекти, і сприяє всебічному розвитку користувачів.

1. КЛАСИФІКАЦІЯ КОМП'ЮТЕРНИХ ІГОР ТА ПЛАТФОРМ ДЛЯ РОЗРОБКИ

1.1. Класифікація комп'ютерних ігор

Класифікація комп'ютерних ігор може бути здійснена за різними критеріями, зокрема за жанром, платформою, аудиторією, економічною моделлю, режимом гри та іншими характеристиками. Розглянемо основні види класифікації [1]:

Ігри за жанром поділяються на

1. Екшн (Action): шутери від першої особи (FPS) ("Call of Duty", "Doom"), шутери від третьої особи (TPS) ("Gears of War", "Max Payne"), платформери ("Super Mario", "Sonic the Hedgehog"), файтинги ("Mortal Kombat", "Street Fighter").
2. Пригодницькі (Adventure): квестові ігри ("Monkey Island", "Broken Sword"), ігри на виживання ("Resident Evil", "Silent Hill").
3. Рольові ігри (RPG): японські RPG (JRPG) ("Final Fantasy", "Dragon Quest"), західні RPG (WRPG) ("The Elder Scrolls", "The Witcher").
4. Стратегічні (Strategy): стратегії в реальному часі (RTS) ("StarCraft", "Age of Empires"), покрокові стратегії (TBS) ("Civilization", "XCOM").
5. Симулятори (Simulation): симулятори життя ("The Sims"), симулятори будівництва та управління ("SimCity", "RollerCoaster Tycoon"), симулятори транспорту ("Microsoft Flight Simulator", "Euro Truck Simulator").
6. Спортивні (Sports): традиційні спортивні ігри ("FIFA", "NBA 2K"), гоночні ігри ("Need for Speed", "Gran Turismo").
7. Головоломки (Puzzle): логічні ігри ("Tetris", "Portal"), казуальні ігри ("Candy Crush Saga").
8. Масові багатокористувацькі онлайн-ігри (MMO): MMORPG ("World of Warcraft", "Guild Wars 2").

Ігри за платформою поділяються на

1. Персональні комп'ютери з Windows, MacOS, Linux.

2. Домашні консолі: PlayStation, Xbox, Nintendo Switch.

3. Портативні консолі: Nintendo 3DS, PlayStation Vita.

4. Смартфони та планшети: Android, iOS.

За аудиторією ігри поділяються

1. Для дорослих: Ігри з високим рівнем насильства або іншого контенту, призначеного для дорослої аудиторії. Наприклад, "Grand Theft Auto", "The Last of Us".

2. Для підлітків: Ігри з помірним рівнем насильства і складнішими темами, ніж дитячі ігри. Наприклад, "Fortnite", "Minecraft".

3. Для дітей: Ігри з простими механіками і дружнім контентом. Наприклад, "Lego" серії, "Animal Crossing".

За режимом гри ігри поділяються на

1. Однокористувацькі (Single-player): ігри, розраховані на одного гравця. Наприклад, "The Witcher 3".

2. Багатокористувацькі (Multiplayer): ігри, в які можна грати з іншими гравцями через інтернет або локальну мережу. Наприклад, "Overwatch", "Among Us".

За економічною моделлю ігри поділяються на:

1. Платні ігри (Premium): ігри, які потрібно купити для доступу, наприклад, "The Sims", "Cyberpunk 2077".

2. Безкоштовні ігри (Free-to-play): ігри, що поширюються безкоштовно, але можуть містити мікротранзакції, наприклад, "Fortnite", "League of Legends".

3. Ігри з підпискою: ігри, доступ до яких надається через щомісячну або щорічну підписку, наприклад, "World of Warcraft" [2].

1.2. Основні моменти розробки комп’ютерних ігор

Розробка комп'ютерних ігор є складним процесом, який включає кілька ключових етапів і аспектів. Ось основні моменти, які необхідно враховувати при розробці ігор: концептуалізація та планування, прототипування, розробка, тестування, випуск, підтримка після випуску.

Концептуалізація та планування є ключовими етапами в процесі розробки комп'ютерних ігор, що закладають основу для всього проекту. Першим кроком є розробка загальної концепції гри, де визначається основна ідея, жанр, сюжет і головні ігрові механіки. Наприклад, розробники можуть вирішити створити пригодницьку гру з елементами рольової гри в фентезійному світі, де гравець виконує різні квести.

На цьому етапі також важливо визначити цільову аудиторію. Знання про те, для кого створюється гра (діти, підлітки, дорослі, хардкорні геймери або казуальні гравці), допомагає приймати рішення щодо стилю гри, її складності та механік. Наприклад, якщо гра орієнтована на дітей, розробники можуть зробити її більш барвистою та простою в освоєнні.

Після визначення концепції та цільової аудиторії, необхідно розробити сеттінг і світ гри. Це включає створення світу гри, його історії, географії, культури та важливих подій. Сюди ж відноситься створення персонажів, включаючи їх біографії, мотивації та взаємодії. Наприклад, у фентезійній грі це можуть бути різні раси, магічні королівства та епічні битви між добром і злом [3].

Документ дизайну гри (Game Design Document, GDD) є наступним важливим етапом. Він містить докладний опис всіх аспектів гри, включаючи геймплей, рівні, персонажів та механіки. Це своєрідна "біблія" проекту, яка направляє всі наступні етапи розробки. У цьому документі описуються, наприклад, система бою, квести, система прокачування персонажів, а також плануються рівні та локації, які гравець буде досліджувати.

Технічне планування також є критично важливим. На цьому етапі вибираються технології та інструменти, які будуть використовуватися в процесі розробки. Це може включати вибір ігрового рушія (наприклад, Unity або Unreal Engine), мови програмування та інші інструменти розробки. Крім того, розробляється архітектура системи, яка визначає структуру коду, використання патернів проектування та інтеграцію сторонніх бібліотек або сервісів.

Проектний менеджмент займається встановленням ключових етапів розробки та відповідних дедлайнів. Це включає розподіл завдань між членами команди, контроль за виконанням плану та регулярні зустрічі для обговорення прогресу і вирішення проблем. Наприклад, розробка може бути поділена на кілька етапів, таких як прототипування, альфа-тестування, бета-тестування та фінальний реліз.

Отже, концептуалізація та планування є комплексним процесом, що охоплює розробку ідеї, визначення цільової аудиторії, створення світу та персонажів, документування всіх аспектів гри, вибір технологій та управління проектом. Ці етапи закладають основу для успішного розвитку гри та забезпечують чітку структуровану роботу над проектом.

У розробці ігор прототипування – це критичний етап, який допомагає визначити напрямок та основні елементи гри. Команди розробників створюють різноманітні прототипи для різних аспектів гри. Перше, це геймплейні прототипи, де швидкі прототипи геймплею створюються для тестування ідеї та механік гри. Це дозволяє команді зрозуміти, що працює, а що – ні, та швидко вносити зміни. Друге, прототипи рівнів – тут розробники створюють швидкі прототипи рівнів або окремих сегментів гри для оцінки їх дизайну та геймплею. Третє, художні прототипи, які зазвичай використовуються для виявлення атмосфери, стилю та естетики гри. Вони можуть бути швидкими макетами або навіть концептуальними малюнками. Четверте, прототипи інтерфейсів користувача, які допомагають визначити оптимальний дизайн інтерфейсу, його зрозумілість та зручність. П'яте, технічні прототипи, які створюються для випробування нових технологій, двигунів гри або інших технічних аспектів. І наостанок, тестування та ітерація, де отриманий від прототипів фідбек дозволяє команді удосконалювати гру в процесі розробки, вносячи потрібні зміни та додаткові функції.

Тестування ігор – це комплексний процес, який включає в себе різні аспекти та підходи. Ось детальніше про них [1]:

1. Функціональне тестування: цей тип тестування спрямований на перевірку роботи основних механік гри, включаючи управління персонажами, інтерфейс, взаємодію з оточенням та інші головні функції гри.

2. Тестування на стабільність: важливо перевірити, щоб гра працювала стабільно та без перебоїв. Це означає, що вона не повинна висаджуватися, крашитися або викликати інші технічні проблеми, які можуть вплинути на користувацький досвід.

3. Тестування на відповідність вимогам: гра повинна відповідати всім поставленим перед нею вимогам, включаючи функціональність, продуктивність, сумісність з різними пристроями та платформами.

4. Тестування на користувацький досвід: важливо перевірити, що гра забезпечує задоволення та комфортний ігровий досвід для користувачів. Це означає перевірку інтерфейсу користувача, контролів, балансу гри та інших аспектів, які впливають на сприйняття гри гравцями.

5. Тестування на різних пристроях та платформах: з огляду на різноманітність пристроїв та платформ, на яких може використовуватися гра, важливо перевірити її сумісність та оптимізацію для різних пристроїв, роздільних здатностей екрану, операційних систем тощо.

6. Тестування на відмовостійкість: гра повинна бути протестована на відмовостійкість, тобто на здатність витримувати великі навантаження, стресові ситуації та інші фактори, які можуть виникнути в процесі використання.

7. Тестування на безпеку: цей тип тестування важливий, особливо для онлайн-ігор або тих, що працюють у мережі. Гра повинна бути протестована на вразливості та можливість атак з боку користувачів.

Підтримка після випуску ігор є важливою частиною життєвого циклу гри і може значно вплинути на задоволення гравців та успіх проекту в цілому. Ось деякі ключові аспекти підтримки після випуску ігор:

1. Виправлення помилок (багфікси): першочерговим завданням є виявлення та виправлення будь-яких помилок або багів, які виявилися після

випуску гри. Це може включати патчі, оновлення або хотфікси, щоб виправити проблеми та покращити загальний досвід гравців.

2. Оновлення контенту: щоб зберегти інтерес гравців і привернути нових, розробники можуть випускати оновлення з новим контентом, таким як нові рівні, персонажі, предмети тощо. Це допомагає підтримувати активність гравців та розвивати гру після її випуску.

3. Взаємодія з гравцями: розробники повинні підтримувати активну взаємодію зі спільнотою гравців, приймаючи до уваги їхні запити, пропозиції та скарги. Це може включати фіксацію проблем, відповіді на запитання або навіть врахування ідей гравців для майбутніх оновлень.

4. Оптимізація та підтримка платформ: розробники повинні продовжувати підтримку гри на різних платформах, забезпечуючи її сумісність з новими версіями операційних систем, пристроями та іншими технічними змінами.

5. Безпека інформації та захист від шахрайства: розробники повинні вживати заходів для захисту гри та особистих даних користувачів від потенційних загроз безпеки, таких як шахрайство або вторгнення.

6. Підтримка онлайн-ігрових сервісів: якщо гра має онлайн-функціонал, важливо підтримувати роботу і підтримку серверів, щоб забезпечити стабільність мережевого геймплею та уникнути перерв у доступі до онлайн-функцій гри [2].

1.3. Платформи для розробки ігор

Існує багато платформ для розробки ігор, які відрізняються за функціональністю, спрямованістю та складністю використання. Ось кілька з них [3]:

1. Unity – одна з найпопулярніших платформ для розробки ігор у всьому світі. Вона підтримує створення ігор для різних платформ, включаючи ПК, консолі, мобільні пристрої та віртуальну реальність. Unity має потужний двигун гри, велику спільноту користувачів та багато

інструментів для розробки.

2. Unreal Engine – розроблений Epic Games, також є дуже популярною платформою для розробки ігор. Він відомий своїм потужним двигуном гри, відмінною графікою та фізикою. Unreal Engine також підтримує різні платформи, включаючи ПК, консолі та мобільні пристрої.

3. GameMaker Studio – це інструмент для розробки 2D-ігор, що надає зручний інтерфейс та простоту використання. Він підтримує різні платформи, включаючи ПК, консолі, мобільні пристрої та веб.

4. Godot Engine – це вільна та відкрита платформа для розробки ігор, яка надає повний доступ до свого вихідного коду. Вона має потужні можливості та підтримує різні платформи.

5. Construct – це інструмент для створення 2D-ігор без програмування. Він має інтуїтивний інтерфейс та велику бібліотеку готових до використання об'єктів та функцій.

Кожна з цих платформ має свої переваги та недоліки, тому вибір залежить від потреб та вимог конкретного проекту.

Висновки до розділу 1

В першому розділі проведений опис класифікації ігор, яка в свою чергу може бути здійснена за жанром, платформою, аудиторією, економічною моделлю, режимом гри та іншими характеристиками. Тут же детально описані моменти створення ігор: концептуалізація та планування, прототипування, розробка, тестування, випуск, підтримка після випуску. Окрім цього, наведені найпоширеніші платформи для розробки ігор.

2. ТЕХНОЛОГІЇ РОЗРОБКИ

2.1. Мова гіперрозмітки HTML5 та таблиці каскадних стилів CSS3 в створенні ігор

HTML5 не лише призначений для створення веб-сторінок, але також може бути використаний для розробки веб-ігор завдяки своїм мультимедійним та веб-графічним можливостям. Використання HTML5 для створення ігор називається HTML5-геймдевом. Нижче наведені деякі інструменти та технології, які використовуються для розробки ігор на базі HTML5.

Елемент <canvas> HTML5 дозволяє рендерити графіку на сторінці за допомогою JavaScript. Він часто використовується для створення 2D-графіки в іграх. Canvas працює за принципом малювання "піксель за пікселем", дозволяючи розробникам контролювати кожен піксель на вмісті, що відображається. За допомогою Canvas можна створювати різноманітні графічні ефекти, малюнки, анімацію, графіки чи діаграми. Він підтримує відображення 2D-графіки та анімації. Canvas надає JavaScript API для малювання різноманітних об'єктів, таких як лінії, криві, текст, картинки і т.д. Розробник може програмно маніпулювати цими об'єктами, створюючи динамічні інтерактивні веб-застосунки. Canvas має дуже ефективний механізм малювання, що дозволяє досягати високої продуктивності при роботі з графікою та анімацією. Canvas підтримується всіма сучасними браузерами, що дозволяє використовувати його в широкому спектрі веб-проектів. Canvas є потужним інструментом для реалізації різноманітних графічних ефектів та анімації в веб-розробці, зокрема для створення ігор, візуалізації даних, малюнків та інтерактивних додатків [4].

HTML5 має вбудовану підтримку для аудіо та відео, що робить його корисним інструментом для розробки ігор. Вбудована підтримка мультимедіа дозволяє розробникам інтегрувати звукові ефекти, музику та відео безпосередньо в свої ігри за допомогою HTML5-елементів <audio> та

<video>.

Елемент <audio> дозволяє вставляти аудіофайли безпосередньо в HTML5-документи. Це дозволяє розробникам використовувати звукові ефекти, фонову музику та інші аудіоелементи у своїх іграх.

Елемент <video> дозволяє вставляти відеофайли безпосередньо в HTML5-документи. Це корисно для створення відеороликів, анімацій та інших відеоелементів у графічних іграх.

Ці можливості дозволяють розробникам створювати ігри, які мають звукові та візуальні компоненти, що робить відтворення ігор більш цікавим та захоплюючим для гравців [4].

Використання локального сховища HTML5 є важливою можливістю для розробки ігор, оскільки воно дозволяє зберігати дані гри на боці клієнта. Це важливо для збереження прогресу гравців, налаштувань гри та іншої інформації, яка має зберігатися між сеансами гри. Ось деякі способи використання локального сховища в розробці ігор:

Локальне сховище може використовуватися для збереження прогресу гравця, такого як рівень, набрані бали, досягнуті досягнення тощо. Це дозволяє гравцям продовжувати гру з того ж місця, де вони зупинилися, навіть якщо вони закрили вкладку браузера чи перезавантажили сторінку.

Локальне сховище може також використовуватися для збереження налаштувань гри, таких як рівень складності, обрана мова, вибрані параметри графіки тощо. Це дозволяє гравцям налаштовувати гру під свої індивідуальні вподобання.

Локальне сховище може використовуватися для збереження історії дій гравця, такої як останні дії, виконані дії чи вибори, зроблені під час гри. Це може бути корисно для створення складних гілок сюжету або для аналізу поведінки гравців.

Локальне сховище може використовуватися для кешування графічних, звукових та інших ресурсів гри, що дозволяє зменшити час завантаження та покращити продуктивність гри.

Локальне сховище також може використовуватися для синхронізації даних між різними пристроями гравця, що дозволяє гравцям грати на різних пристроях та продовжувати гру там, де вони зупинилися.

Загалом, використання локального сховища HTML5 робить розробку ігор більш гнучкою та інтерактивною, дозволяючи зберігати та обмінюватися даними між різними частинами гри та між різними сеансами гри.

З використанням цих інструментів і технологій розробники можуть створювати різноманітні та захоплюючі ігри, які можна грати у веб-браузерах без необхідності встановлення додаткових програм або плагінів.

Таблиці каскадних стилів CSS3 (Cascading Style Sheets) грають важливу роль у створенні ігор, особливо веб-ігор. CSS дозволяє розробникам створювати стильовий дизайн інтерфейсу гри, анімації, а також різноманітні візуальні ефекти. Нижче наведено кілька способів використання CSS3 в розробці ігор.

CSS дозволяє визначати вигляд елементів інтерфейсу гри, таких як кнопки, меню, поля введення тощо. За допомогою CSS можна налаштовувати кольори, розміри, шрифти та інші властивості цих елементів, щоб створювати привабливий та користувацькозороїний дизайн.

CSS3 включає в себе розширені можливості для створення анімацій, таких як переходи (transitions), перетворення (transforms) та ключові кадри (keyframes). Ці можливості можна використовувати для створення різноманітних анімаційних ефектів у грі, таких як рух об'єктів, зміна їх розміру чи форми, зміна кольору та ін.

CSS дозволяє контролювати розташування елементів на сторінці, що дозволяє розміщувати їх в потрібних місцях з використанням плаваючих, абсолютних, відносних позицій тощо. Це особливо корисно для створення складних інтерфейсів гри з багатьма елементами.

CSS3 дозволяє створювати адаптивний дизайн, який змінюється залежно від розміру екрану пристрою. Це дозволяє створювати ігри, які добре виглядають на будь-яких пристроях, від комп'ютерів до мобільних

телефонів.

CSS дозволяє визначати стилі тексту, такі як колір, розмір, жирність, курсивність, тінь та інші ефекти, що дозволяє створювати ефектний вигляд текстових елементів у грі [5].

CSS може використовуватися для створення спрайтів – зображень, які містять набір графічних елементів, що використовуються у грі. Використання CSS-анімації дозволяє анімувати ці спрайти, що додає живості та динамічності до гри.

В цілому, CSS3 є потужним інструментом для створення візуально привабливих та інтерактивних ігор у веб-розробці. Використовуючи його разом з HTML та JavaScript, розробники можуть створювати складні та захоплюючі ігри для різних платформ.

2.2. Мова JavaScript та її бібліотека jQuery

JavaScript є однією з найпоширеніших мов програмування для розробки ігор, особливо у веб-розробці. Вона надає широкі можливості для створення інтерактивних, захоплюючих та складних ігор. Ось деякі ключові аспекти використання JavaScript в розробці ігор:

1. Взаємодія з DOM: JavaScript використовується для взаємодії з об'єктами веб-сторінки через DOM (Document Object Model). Це означає, що розробники можуть динамічно змінювати елементи сторінки, створювати анімацію, обробляти події натискання кнопок та інші взаємодії користувача.

2. Анімація та ефекти: JavaScript дозволяє створювати анімацію та різноманітні візуальні ефекти, такі як плавний рух об'єктів, зміна їх розміру, кольору, прозорості тощо. Це додає живості та динамізму до ігор.

3. Логіка гри: JavaScript використовується для реалізації логіки гри, такої як управління персонажем, обробка колізій, розрахунок балів, управління штучним інтелектом NPC (non-player characters) тощо.

4. Мультимедіа та звук: JavaScript дозволяє відтворювати аудіо та відео, а також створювати звукові ефекти у грі. Це важливо для створення

атмосфери гри та підвищення іммерсії гравців.

5. Модульність та фреймворки: JavaScript має багато фреймворків та бібліотек, спеціалізованих на розробці ігор, які дозволяють розробникам прискорити процес створення гри, надати доступ до готових рішень для фізики, анімації, штучного інтелекту тощо.

6. Мобільна розробка: JavaScript є важливим інструментом для розробки мобільних ігор. За допомогою фреймворків, таких як Phaser, Pixi.js, або платформ, таких як React Native чи Ionic, розробники можуть створювати ігри, які працюють на різних платформах, включаючи iOS та Android.

JavaScript дозволяє розробникам створювати різноманітні ігри, починаючи від простих HTML5-ігор до складних ігрових двигунів та великих ігор для різних платформ. Вона є важливим інструментом для будь-якого розробника, який хоче створити ігри з використанням веб-технологій.

jQuery є потужною бібліотекою JavaScript, яка спрощує взаємодію з DOM, анімацію, обробку подій та інші завдання у веб-розробці. Хоча вона не спеціалізується на розробці ігор, вона може бути корисною у ряді сценаріїв для створення веб-ігор. Ось деякі способи використання jQuery в розробці ігор:

1. Взаємодія з DOM: jQuery робить роботу з DOM простішою та зручнішою, що дозволяє звертатися до елементів сторінки, змінювати їх властивості, додавати або видаляти елементи, обробляти події тощо. Це корисно для створення інтерактивних ігор з відгуком на дії користувача.

2. Анімація: jQuery має ряд функцій для створення анімацій, таких як fadeIn, fadeOut, slideUp, slideDown, animate та інші. Вони дозволяють створювати різноманітні анімаційні ефекти для елементів гри, що додають динамізм та привабливість [6].

3. Обробка подій: jQuery дозволяє легко прикріплювати обробники подій до елементів сторінки, таких як натискання кнопок, клікання мишею, рух миші тощо. Це корисно для взаємодії з користувачем та реалізації різноманітних геймплейних механік.

4. Ajax запити: jQuery має вбудовану підтримку Ajax, що дозволяє здійснювати асинхронні запити до сервера. Це може бути корисно для обміну даними з сервером у реальному часі, наприклад, для збереження прогресу гравця чи отримання додаткових ресурсів для гри.

5. Плагіни та розширення: у веб-розробці ігор часто використовуються різноманітні плагіни та розширення jQuery, які додають додаткові функціональність, наприклад, роботу з анімаціями, фізикою, обробкою дотиків тощо.

Хоча jQuery може бути корисною у розробці веб-ігор, варто також звернути увагу на те, що за останні роки стандартні функції JavaScript значно розширилися, а модульні бібліотеки, такі як React, Vue.js, або Angular, набули популярності у веб-розробці [].

2.3. Переваги розробки логічних головоломок на JavaScript, jQuery, HTML, CSS

Розробка логічних головоломок на JavaScript, jQuery, HTML та CSS має кілька переваг:

1. Широке охоплення: JavaScript є однією з найпоширеніших мов програмування у веб-розробці, тож використання його дозволяє розробляти головоломки, які можуть працювати на більшості сучасних веб-платформ.

2. Динамічність та взаємодія: JavaScript дозволяє створювати динамічні головоломки зі складними механіками. Він також дозволяє реалізувати взаємодію з користувачем, таку як перевірка правильності відповідей, відображення результатів та інше [7]

3. Анімація та візуальні ефекти: За допомогою CSS та jQuery можна додати різноманітні анімаційні та візуальні ефекти до головоломок, що робить їх більш привабливими та захоплюючими для користувачів.

4. Легкість розробки та розширення: HTML, CSS та JavaScript дуже добре поєднуються між собою, що спрощує розробку головоломок та дозволяє швидко додавати новий функціонал чи виправляти помилки.

5. Широкий вибір інструментів та бібліотек: Існує безліч бібліотек та фреймворків JavaScript, таких як jQuery, React, Vue.js тощо, які можна використовувати для розробки головоломок та забезпечення їхньої більшої функціональності.

6. Кросплатформенність: Головоломки, розроблені з використанням HTML, CSS та JavaScript, працюють у браузерах на різних пристроях та платформах, що робить їх доступними для широкого кола користувачів [8].

Висновки до розділу 2

В розділі описано про особливості використання JavaScript, jQuery, HTML та CSS для створення ігор.

Загалом, використання JavaScript, jQuery, HTML та CSS для розробки логічних головоломок дозволяє створювати цікаві та захоплюючі ігри, які працюють у веб-середовищі та забезпечують великий інтерактив з користувачами.

3. ОПИС СТВОРЕНОЇ ГРИ

3.1. Загальні відомості про гру

На вхід розробленої гри подається граф, який є ейлеровим, тобто це граф, у якому існує цикл, що проходить через кожне ребро рівно один раз. Цей цикл називається ейлеровим циклом або ейлеровим туром [9-11]. При цьому користувачу потрібно прокласти цей маршрут самостійно. Процес гри супроводжується інформаційними повідомленнями, що допомагають користувачу визначатись з правилами гри, окрім цього в застосунку є кнопка яке веде на інформаційну сторінку з правилами гри.

Суть гри полягає в побудові графа Ейлера послідовно вибираючи ребра для цього та починаючи з вершини (рисунок 3.1). Гра містить дев'ять рівнів, складність яких зростає зі збільшенням рівня.

Гра відноситься до жанру логічних головоломок, але окрім цього її можна віднести до освітніх ігор, оскільки дозволяє освоїти теорію графів, зокрема і ейлерових графів у цікавий та веселий спосіб.

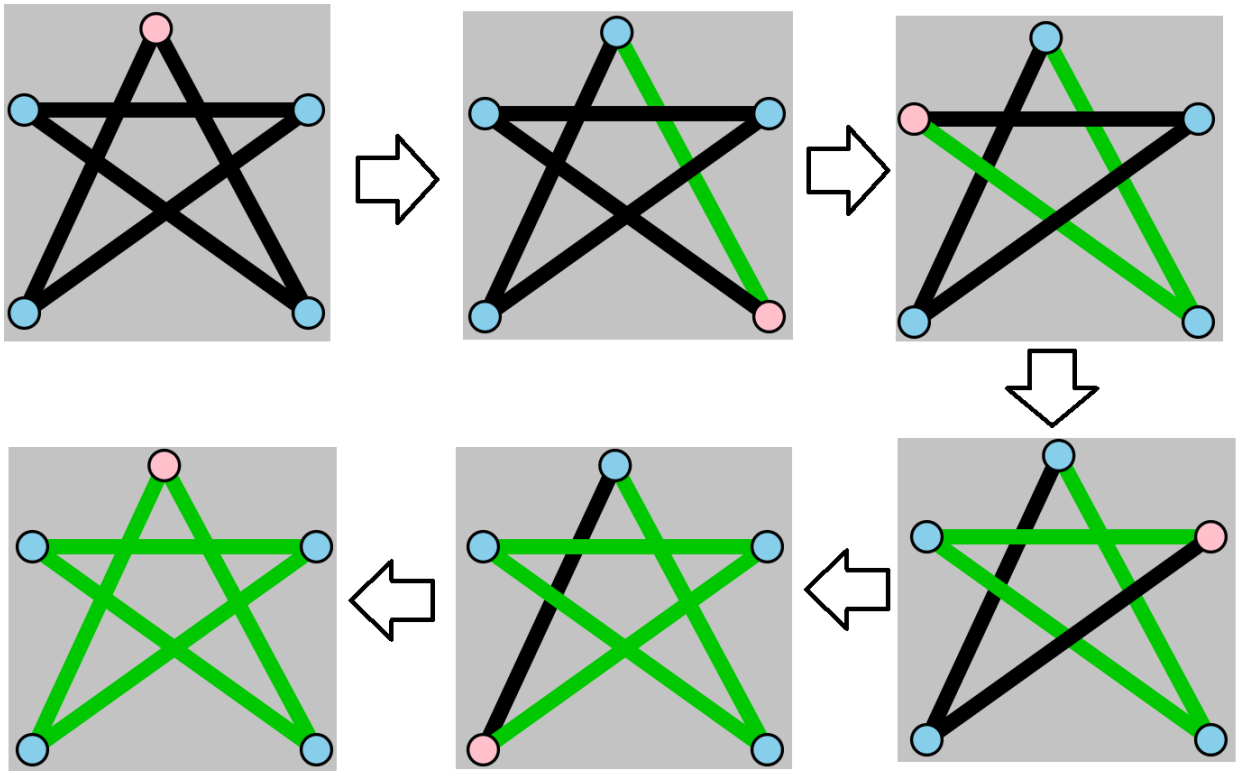


Рисунок 3.1. Процес успішного проходження рівня

Зрозуміло, що рисунком 3.1 представлений граф Ейлера, в якому з якої б вершини не почав користувач побудову свого циклу, то він би його завершив. Такий граф представляє перший рівень гри та, зрозуміло, що є найпростішим. Наступні рівні є складнішими, і в деяких випадках, навіть знаючи теорію графів, потрібно задуматись над вірним проходженням. Але взагалі гра розрахована на гравців, які незнайомі з теорію графів, а гра допоможе їм в її освоєнні.

3.2. Структура застосунку та опис коду

Застосунок розроблений з використанням мови гіпертекстової розмітки HTML5, таблиць каскадних стилів CSS3, мови програмування JavaScript та бібліотеки jQuery. Через це такий застосунок може бути запущений на персональному комп’ютеру з будь-якою операційною системою.

Складається застосунок гри з html-файлів index.html, game.html, info.html, js-файлів game.js, script.js, shapes.js, бібліотеки jQuery.min.js, файлу стилів game.css та зображень, що представляють задній фон сторінок застосунку. Структура файлів показана рисунком 3.2.

Файл index.html є головною веб-сторінкою з деякою статичною інформацією про Ейлера, ейлерові графи та перехід сторінки гри (game.html), на якій є кнопки для рестарту рівня, для переходу до правил гри (info.html), поле із самими графами та шкала рівнів. До файлу game.html, при цьому, підключені js-файли для функціонування самої гри.

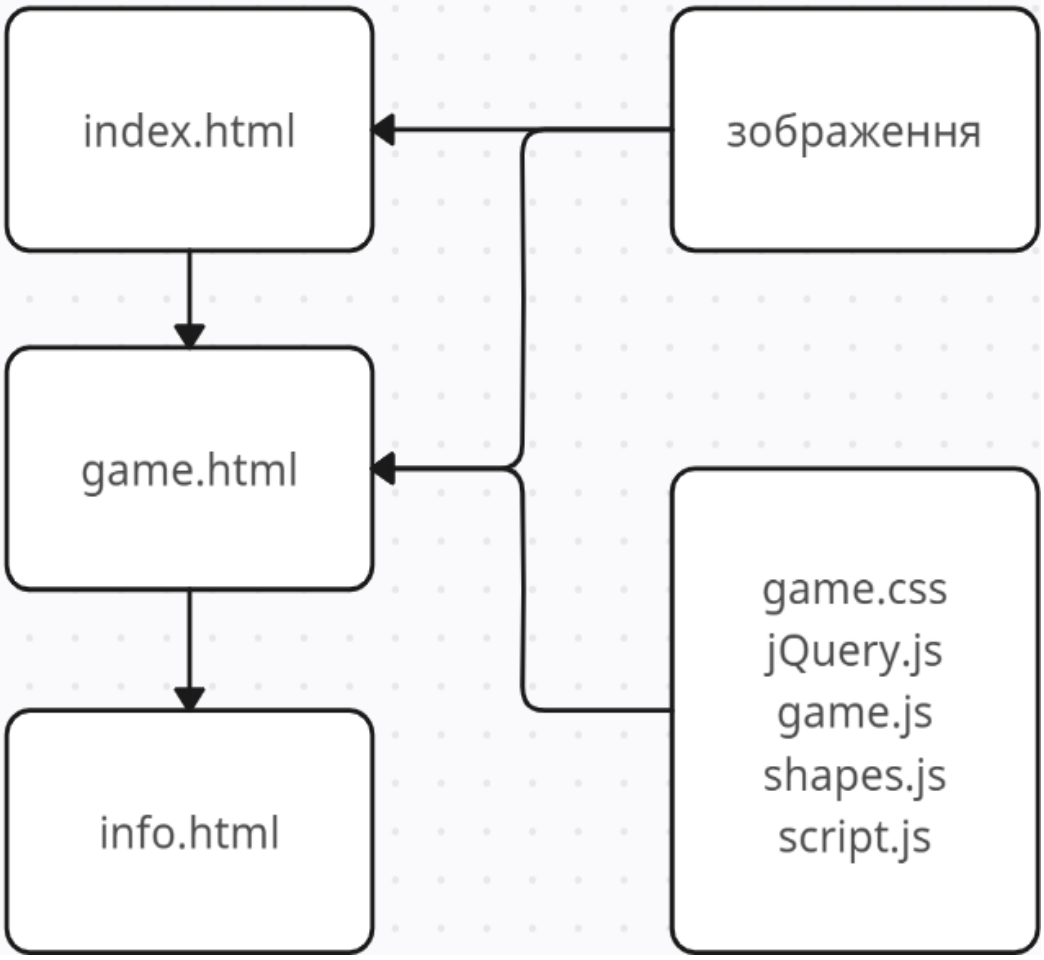


Рисунок 3.2. Структура файлів застосунку

Файл game.js містить JavaScript-код для гри, яка моделює побудову графів і вирішення ейлерових циклів. Гравець вибирає послідовно ребра, щоб побудувати ейлерів цикл. Файл містить (рисунок 3.3):

1. Клас Node, що містить методи та властивості:

- x і y: координати вузла.
- id: унікальний ідентифікатор вузла.
- edges: масив ребер, пов'язаних з вузлом.
- isEqualTo(node): перевіряє, чи рівні два вузли.
- addEdge(edge): додає ребро до вузла.
- isEdgeAdjacent(edge): перевіряє, чи є ребро суміжним до цього вузла.
- areAllEdgesVisited(): перевіряє, чи всі ребра вузла були відвідані.

2. Клас Edge, що містить методи та властивості:

- startNode і endNode: початковий і кінцевий вузли ребра;
- id: унікальний ідентифікатор ребра;
- visited: позначає, чи було відвідано ребро;
- isEqualTo(edge): перевіряє, чи рівні два ребра;
- getOtherNode(node): повертає інший вузол ребра;
- createShape(game): ініціалізує вузли та ребра для поточного рівня гри на основі даних про форму.

3. Клас Game, що містить властивості та методи:

- level: поточний рівень гри;
- shapesData: дані про форми для кожного рівня;
- noOfLevels: кількість рівнів у грі;
- nodes і edges: масиви вузлів і ребер для поточного рівня;
- noOfEdgeVisited: кількість відвіданих ребер;
- currentNode: поточний вузол, з якого відбувається вибір ребра;
- getEdgeById(edgeID): повертає ребро за його ID;
- isLevelComplete(): перевіряє, чи всі ребра були відвідані;
- getNodeById(nodeIdToFind): повертає вузол за його ID;
- visit(visitor): візуалізує вузли та ребра, показує рівень;
- selectNode(nodeId): вибирає поточний вузол;
- visitEdge(edgeID): відвідує ребро, перевіряє різні умови та повертає статус;
- restartLevel(): перезапускає поточний рівень;
- swicthToNextLevel(): переходить до наступного рівня.

Окрім цього, файл містить основну логіку гри для ініціалізації гри, взаємодії з користувачем, перевірку стану гри, візуалізацію, а саме:

1. Створюється новий об'єкт Game з переданими даними про форми.
2. Викликається функція createShape, яка створює вузли та ребра для поточного рівня.
3. Гравець може вибирати вузли та ребра, що моделює проходження ейлерового циклу.

4. Методи selectNode та visitEdge обробляють дії користувача, перевіряють умови і повертають відповідні статус-коди.

5. Методи isLevelComplete та areAllEdgesVisited перевіряють, чи виконані всі умови для завершення рівня чи гри.

6. Метод visit використовується для візуалізації поточного стану гри через переданий об'єкт visitor, який рендерить вузли та ребра.

Цей файл є основною частиною логіки гри, яка дозволяє користувачам проходити ейлерові цикли, вибираючи послідовно ребра графа.

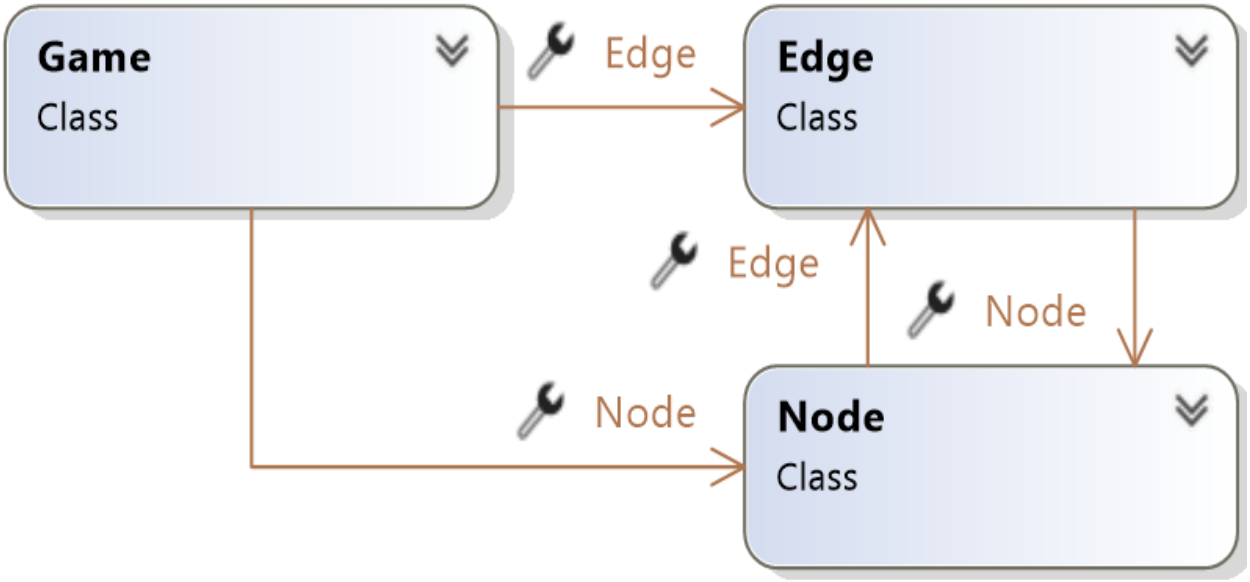


Рисунок 3.3. Діаграма класів з файлу game.js

Файл script.js містить JavaScript-код для управління взаємодією користувача з грою, яка моделює побудову ейлерових графів, де гравець вибирає послідовно ребра, щоб побудувати ейлерів цикл. Основні функції цього коду зосереджені на обробці подій, візуалізації гри та відображенні повідомлень про помилки.

Файл містить:

1. Об'єкт shower: цей об'єкт містить методи, які обробляють різні дії та помилки під час гри.

2. Функція performAction: виконує відповідні дії на основі статус-коду, що повертається методом гри. Викликає методи об'єкта shower для обробки

конкретних подій або помилок.

3. Функція onShapeReady: ініціалізує та візуалізує форму графа для поточного рівня гри. Створює SVG-елементи для вузлів та ребер графа і додає їх у контейнер.

Об'єкт shower містить такі складові:

- onEdgeVisited: позначає ребро як відвідане та вибирає новий вузол;
- onNodeSelected: змінює колір обраного вузла;
- onLevelComplete: обробляє завершення рівня та приховує певні елементи;
- showErrorMessage: відображає повідомлення про помилки з анімацією;
- обробники помилок (onStartNodeAlreadySelected, onNonAdjacentVisit, onStartNodeNotSelected, onEdgeRevisit, onNoPossibleMoves): викликають функцію showErrorMessage з відповідними повідомленнями;
- onGameFinished: обробляє завершення гри та відображає фінальні елементи.

Функція init файлу script.js ініціалізує гру та встановлює обробники подій для взаємодії з елементами гри (ребрами та вузлами). Окрім цього ця функція викликає onShapeReady для першого рендерингу графа та встановлює обробники подій для перезапуску рівня та переходу на наступний рівень.

Файл script.js містить обробку подій:

1. Обробку кліків на ребрах (line) та вузлах (circle) для вибору поточного вузла та відвідування ребра.
2. Обробку кліків на кнопках для перезапуску рівня та переходу на наступний рівень.

Взаємодія з користувачем згідно файлу script.js виконується наступним чином. Коли гравець клікає на ребро або вузол, викликаються відповідні методи гри (visitEdge або selectNode), і результати передаються у функцію performAction.

Залежно від статус-коду, performAction викликає відповідний метод об'єкта shower для оновлення візуалізації або відображення повідомлення про

помилку. Повідомлення про помилки відображаються з анімацією, яка змінює видимість і прозорість елементів, щоб привернути увагу користувача.

Файл script.js забезпечує логіку взаємодії між користувачем та грою, обробляючи події та відображаючи відповідні зміни в графі, повідомлення про помилки та стан гри. Він відповідає за візуалізацію поточного стану гри, реагуючи на дії користувача, і підтримує плавний перехід між рівнями гри.

Файл shapes.js містить JSON-об'єкт x, що містить дані про графи для різних рівнів гри, яка моделює побудову ейлерових графів. Кожен об'єкт у масиві x представляє один рівень і містить інформацію про вузли та ребра графа. Нижче наведено детальний опис структури даних:

Ключ nodesData JSON-об'єкту x – це масив, який містить координати вузлів графа. Кожен елемент масиву nodesData – це масив з двох чисел, які представляють координати (x, y) вузла.

Ключ edgesData JSON-об'єкту x – Це масив, який містить інформацію про ребра графа. Кожен елемент масиву edgesData – це масив з чотирьох чисел, які представляють координати двох кінцевих вузлів ребра (x1, y1, x2, y2).

Розглянемо перший об'єкт у масиві x, візуалізація якого показана рисунком 3.1 вище:

```
"nodesData": [
        [150, 20],
        [20, 100],
        [300, 100],
        [20, 300],
        [300, 300]
],
"edgesData": [
        [150, 20, 20, 300],
        [150, 20, 300, 300],
        [20, 100, 300, 100],
```

```
            [20, 100, 300, 300],
            [300, 100, 20, 300]
        ]
```

При цьому, nodesData містить координати п'яти вузлів:

- вузол 1: (150, 20);
- вузол 2: (20, 100);
- вузол 3: (300, 100);
- вузол 4: (20, 300);
- вузол 5: (300, 300).

А edgesData містить інформацію про п'ять ребер, кожне з яких з'єднує дві вершині:

- ребро 1: з (150, 20) до (20, 300);
- ребро 2: з (150, 20) до (300, 300);
- ребро 3: з (20, 100) до (300, 100);
- ребро 4: з (20, 100) до (300, 300);
- ребро 5: з (300, 100) до (20, 300);

Дані edgesData та nodesData використовуються для побудови та візуалізації графів у грі. Гравець може взаємодіяти з цими графами, вибираючи ребра, щоб створити ейлеровий цикл, де кожне ребро має бути пройдено рівно один раз. Дані про вузли та ребра з цього JSON-об'єкта будуть використовуватися для створення об'єктів Node та Edge. Метод createShape використовує ці дані для побудови графа на основі поточного рівня гри. Користувач взаємодіє з графом, вибираючи вузли та ребра, що оновлює стан гри відповідно до правил.

Ця структура забезпечує гнучкість у додаванні нових рівнів, просто додаючи нові об'єкти до масиву x.

Висновки до розділу 3

В розділі проведено опис структури розробленого програмного застосунку для логічної гри, яка дозволяє будувати цикли Ейлера у відповідних графах. Тут же описані основні класи, методи та властивості файлів застосунку та самі файли застосунку.

4. ТЕСТУВАННЯ, ОПИС ІНТЕРФЕЙСУ ТА МОЖЛИВОСТЕЙ ГРИ

4.1. Опис інтерфейсу та можливостей гри

На рисунку 4.1 показана головна сторінка застосунку, що створюється файлом index.html. З неї можна перейти вже безпосередньо на саму гру, яка зразу ж буде представлена першим рівнем, шкалою проходження рівнів та кнопками рестарту і переходу на інформаційну сторінку із правилами гри (рисунок 4.2).

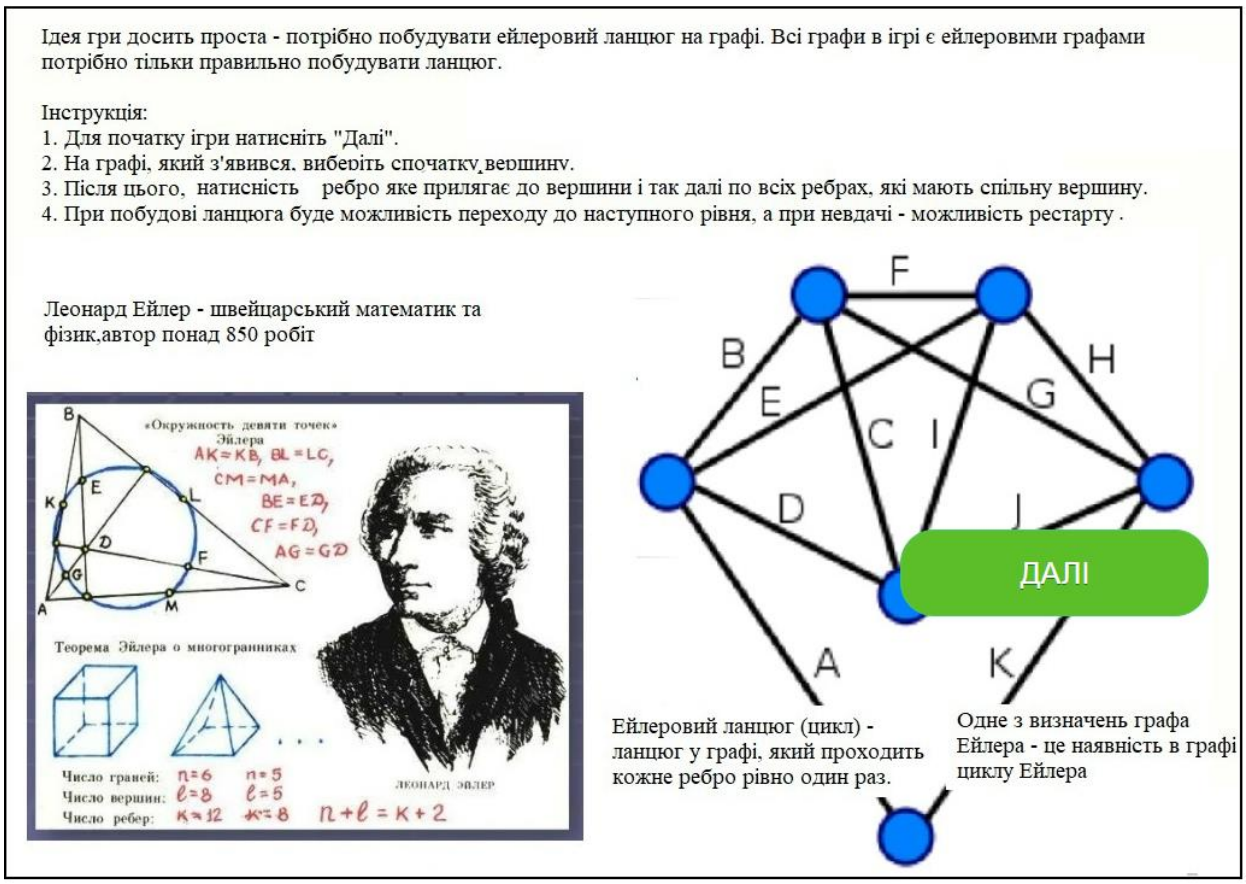


Рисунок 4.1. Головна сторінка застосунку

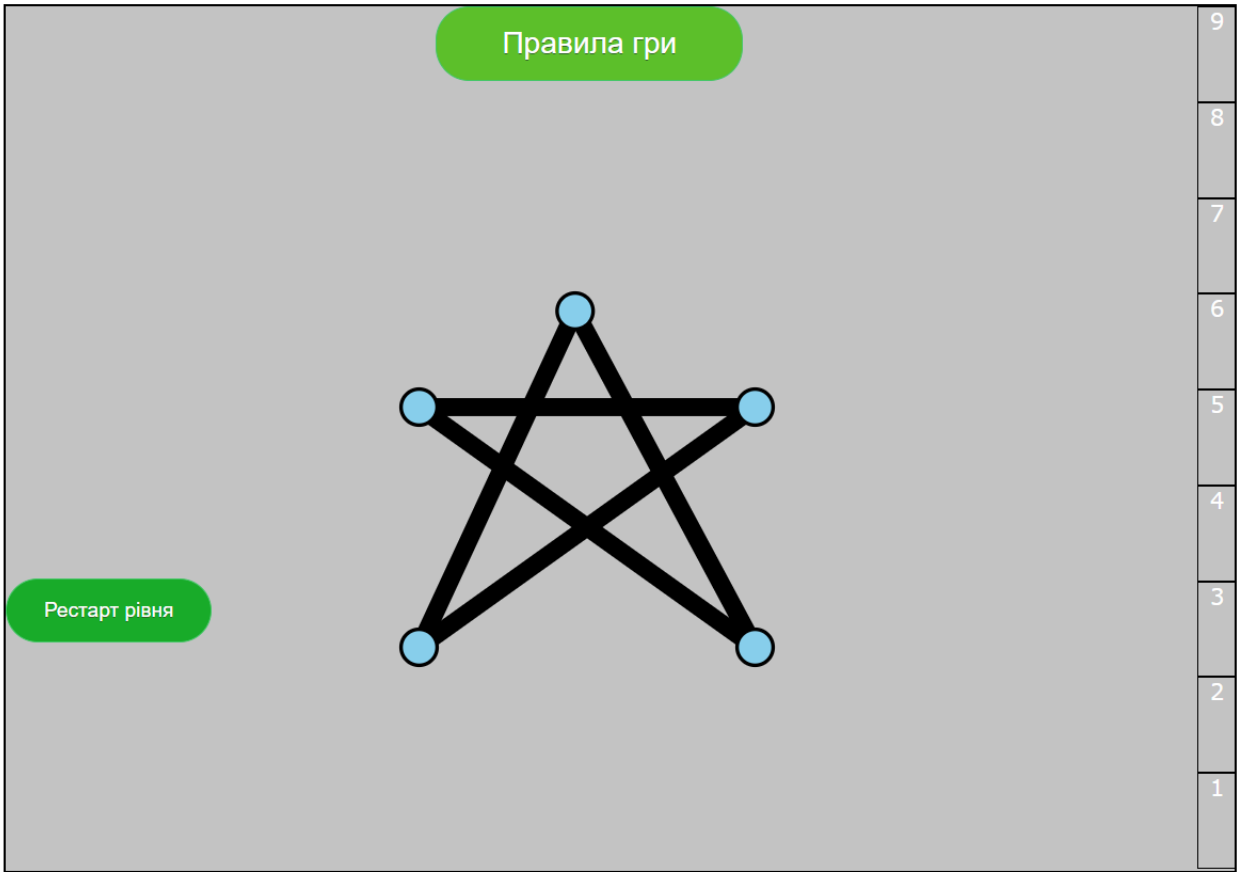


Рисунок 4.2. Перший рівень гри

Процес проходження супроводжується побудовою циклів Ейлера на всіх графах, які створюються згідно JavaScript-коду, що знаходиться у файлі shapes.js. Рисунком 4.3 показаний п’ятий рівень гри, а рисунком 4.4 – восьмий рівень гри. Як видно з цих рисунків збоку на шкалі пройдені рівні показані зафарбованим зеленим кольором.

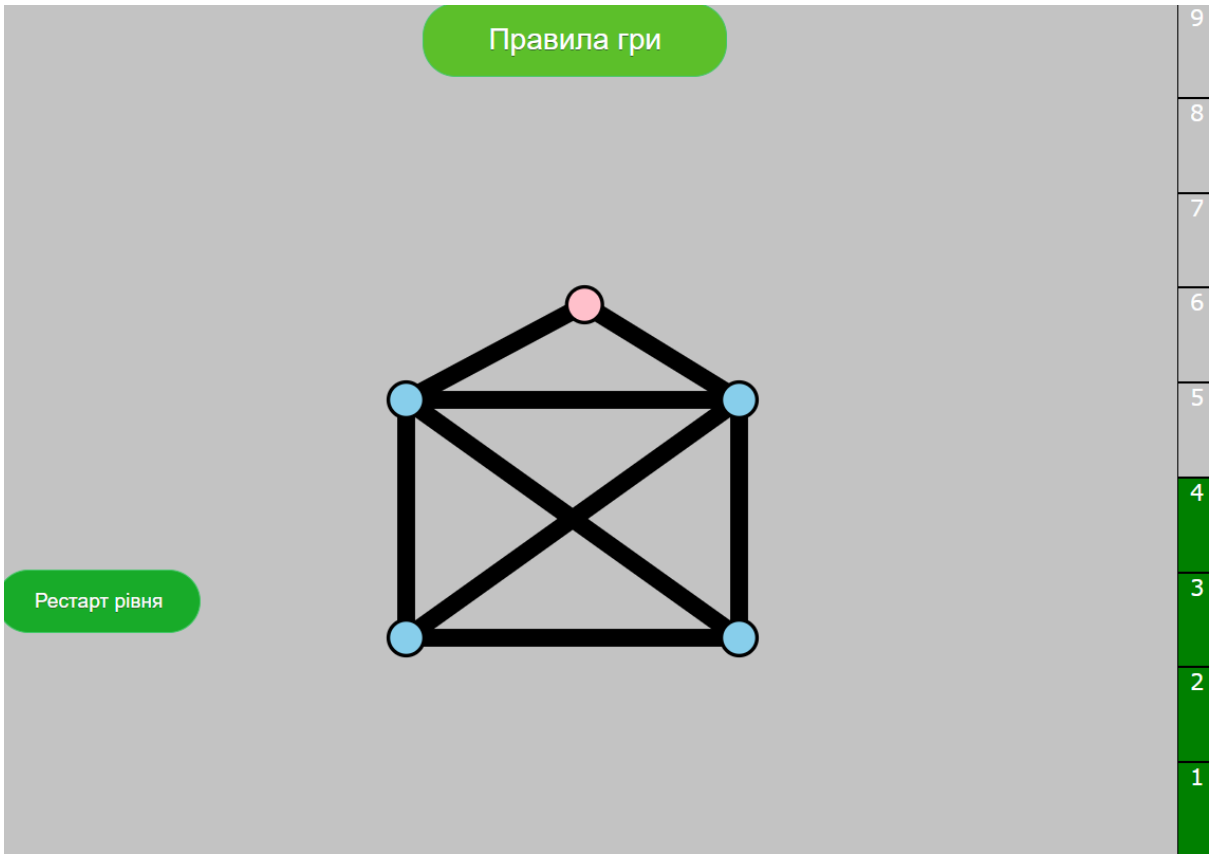


Рисунок 4.3. П’ятий рівень гри

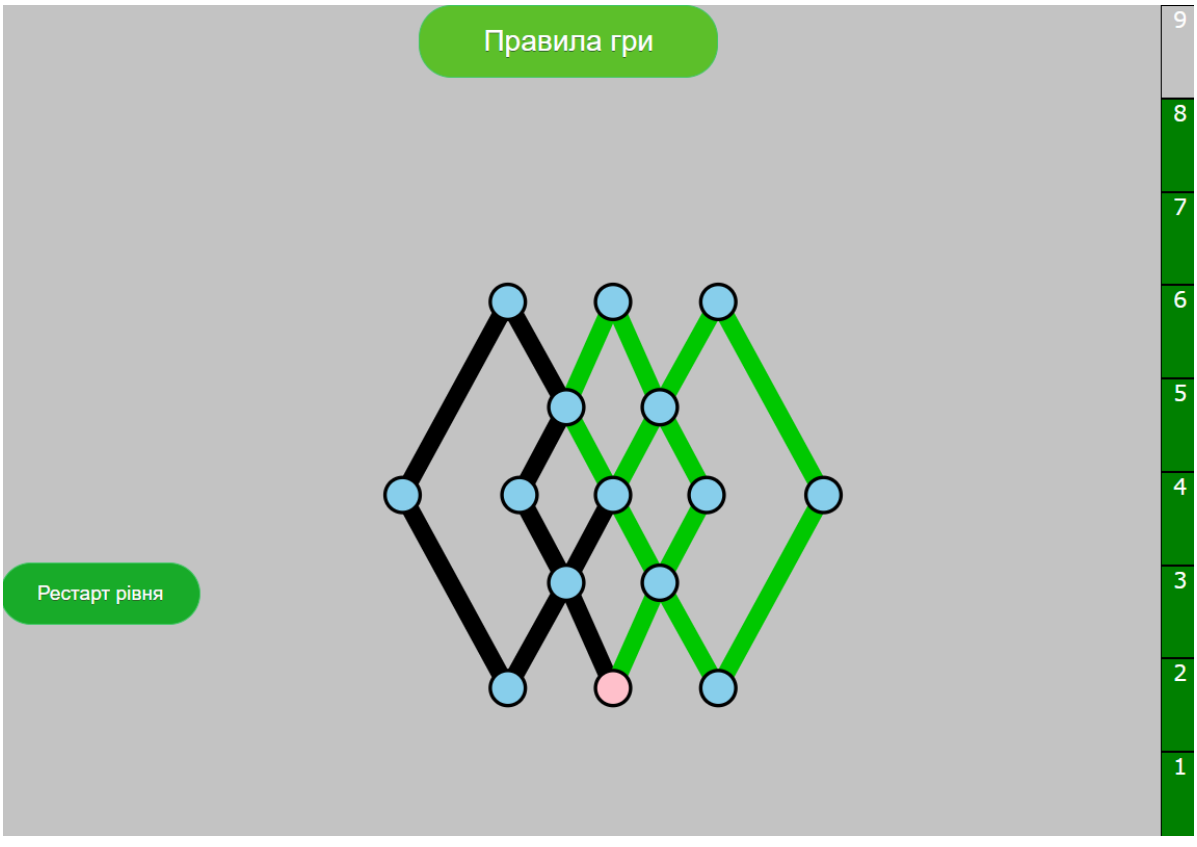


Рисунок 4.4. Восьмий рівень гри

Гра дає можливість користувачу проходити рівень за рівнем, освоюючи теорію графів, весело проводячи час. При цьому, можливості для користувача на кожному рівні можна продемонструвати діаграмою прецедентів на рисунку 4.5.

Рисунок 4.5. Діаграма прецедентів користувача для рівня гри

4.2. Тестування функціоналу гри

Тестування застосунку базувалось на тестуванні функціоналу вибору некоректного вузла чи ребра для того щоб продовжувати гру. Кожен рівень гри повинен бути початий вибором певної вершини, в інакшому випадку буде виведене інформаційне повідомлення, яке показане рисунком 4.6.

У випадку, якщо була вибрана вершина, а після того ребро, або є десь в процесі гри випадково можна вибрати уже відвідане ребро, то програмою буде виведене інформаційне повідомлення, яке показане рисунком 4.7.

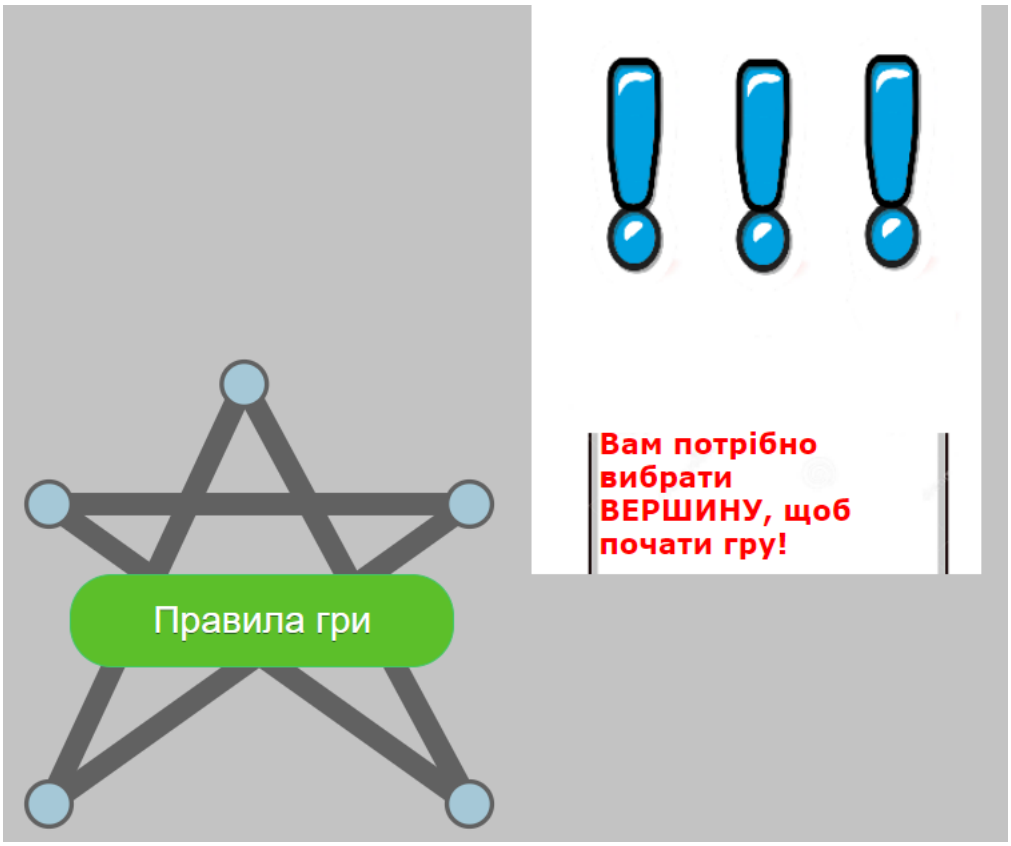


Рисунок 4.6. Повідомлення про некоректний старт проходження рівня

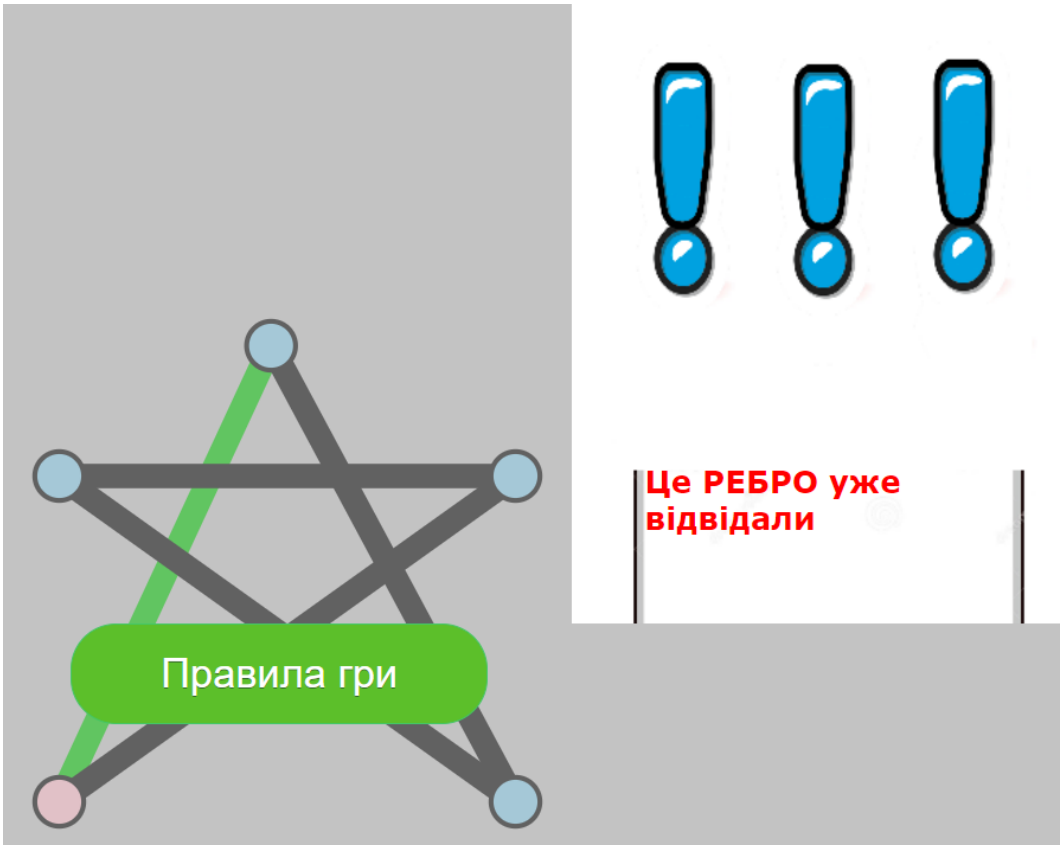


Рисунок 4.7. Повідомлення про те, що ребро вже відвідане

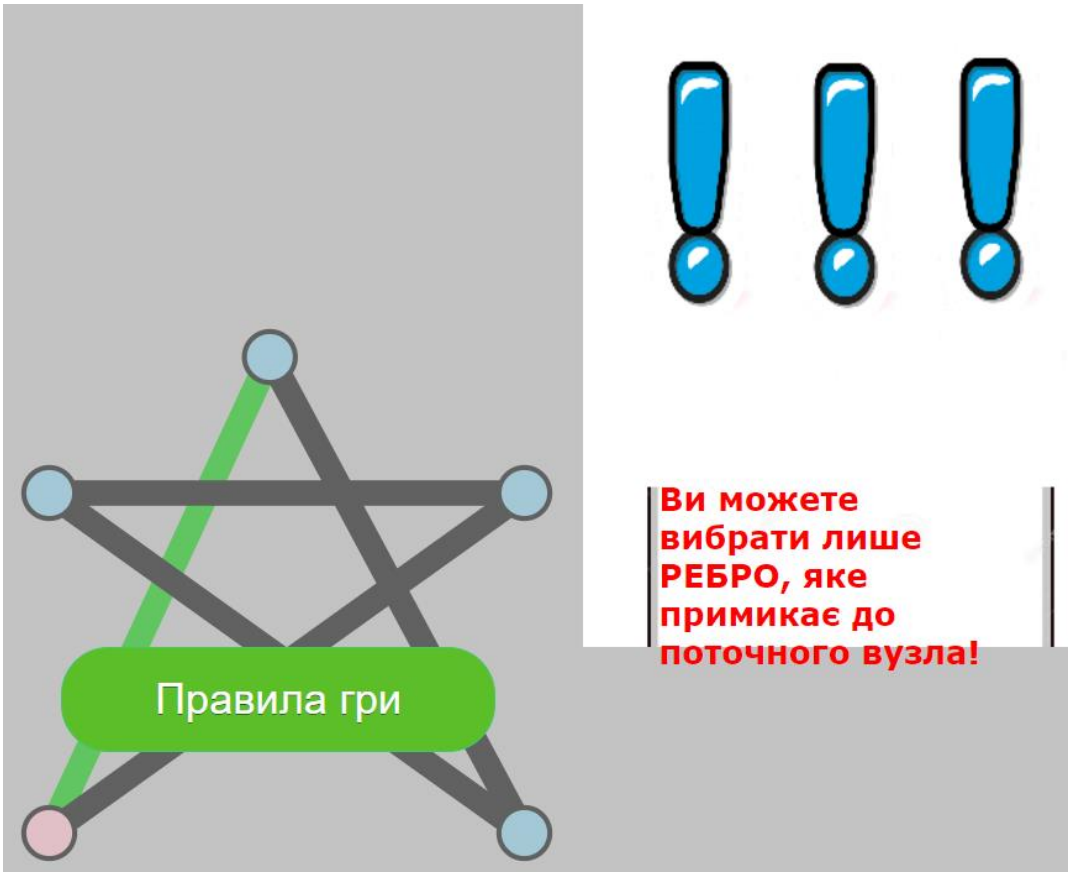


Рисунок 4.8. Повідомлення про вибір не прилягаючого ребра

Після вибору якогось певного ребра в процесі гри, користувач може вибрати тільки прилягаюче ребро, бо в цьому і полягає суть гри – побудувати цикл на графі, який би включав кожне ребро тільки по одному разу. Якщо ж користувач вибере ребро, що не прилягає до поточного, то йому буде показане повідомлення, що показане рисунком 4.8.

Після вибору на першому етапі вершини, як і повинно бути згідно інструкцій гри, далі потрібно вибирати тільки ребра. У випадку вибору якоїсь вершини виведеться повідомлення, яке показане рисунком 4.9.

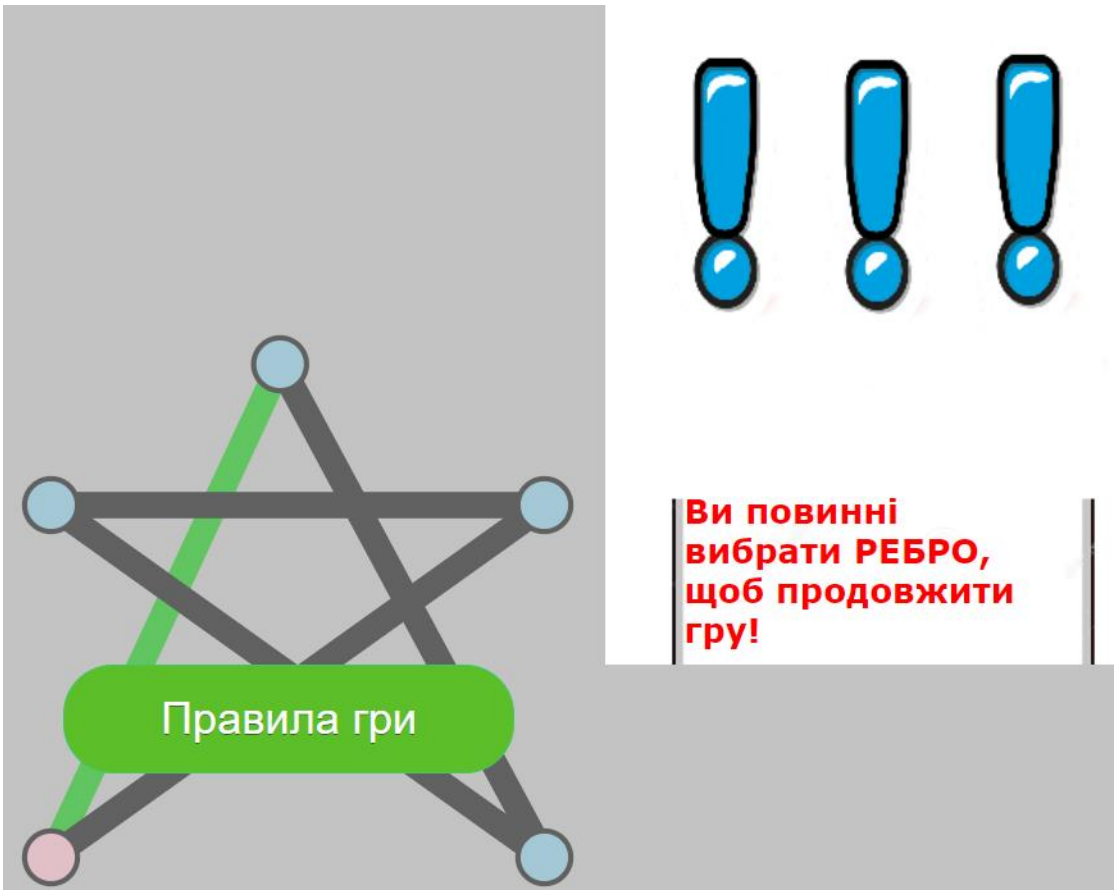


Рисунок 4.9. Повідомлення про те, що треба вибрати ребро,а не вершину

Після успішного проходження рівня користувачу буде відображений інтерфейс, що показаний рисунком 4.10. Після натиснення на відповідну кнопку буде виконаний перехід до наступного рівня.

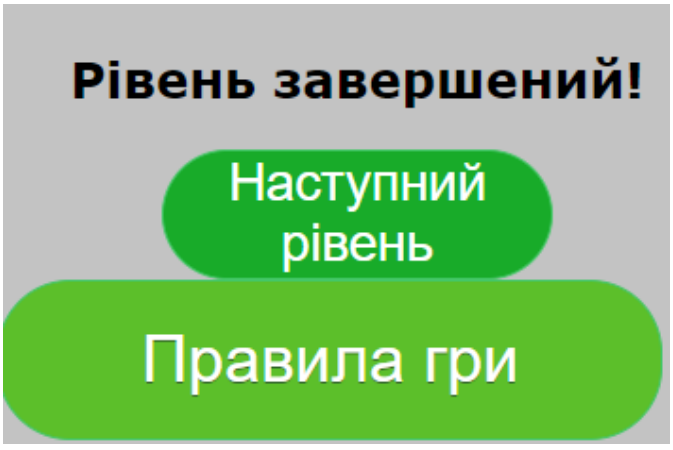


Рисунок 4.9. Завершення рівня

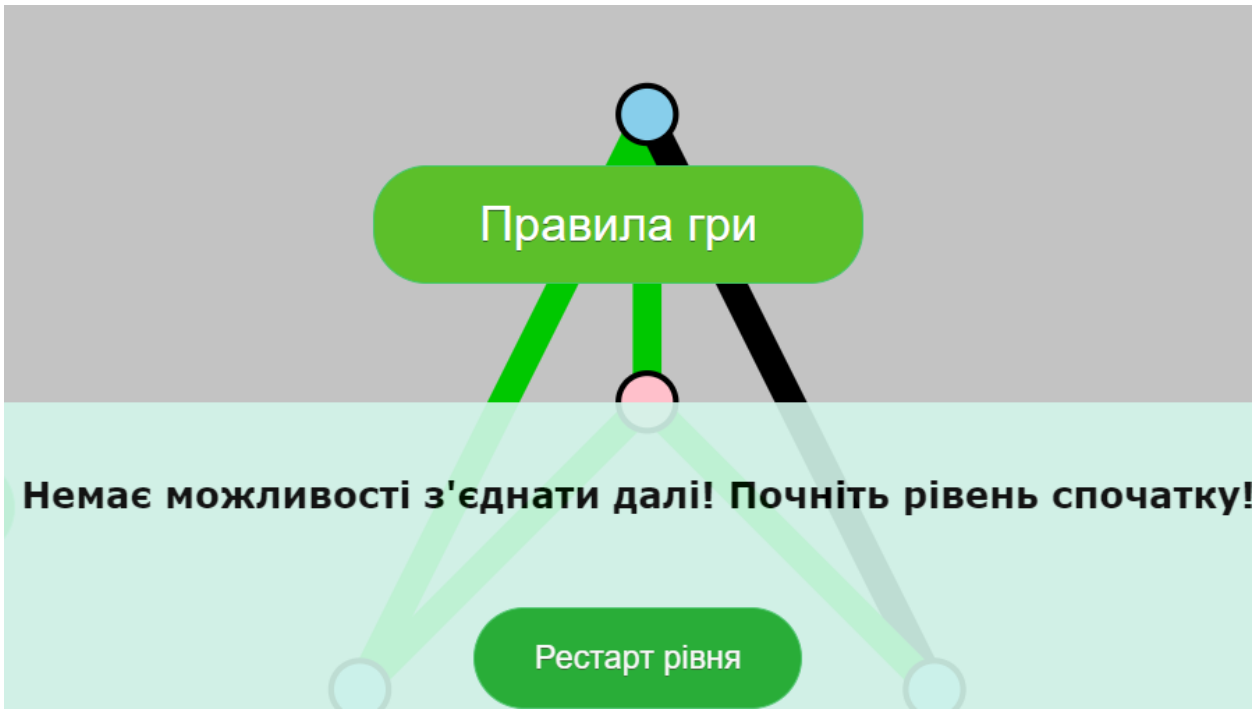


Рисунок 4.11.

У випадку, якщо користувач почав складати ейлеровий цикл почавши не з тієї вершини, або ж в процесі гри виконав вибір не того ребра, і йому не вдається завершити цикл, оскільки просто на якомусь певному етапі немає можливості продовжувати, то йому буде показане відповідне повідомлення про це і можливість рестарту рівня (рисунок 4.11).

Висновки до розділу 4

В розділі проведено опис тестування, що базувалось на проходженні рівнів, на всеможливому некоректному виборі ребер та вершин для отримання інформаційних повідомлень. В ході тестування помилок не виявлено, а застосунок, тобто гра готова для використання, наприклад для ілюстрації студентам з відповідних курсів.

5. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

У даному дипломному проекті проводиться розробка апаратно-програмного модуля управління освітленням для побутових та виробничих приміщень, у цьому розділі доцільно розглянути питання з охорони праці при виготовленні друкованих плат та при виконанні пайки. В ході проведення цих робіт утворюється низка шкідливих та небезпечних факторів, розгляд яких і буде проведено в цьому розділі. Також доцільним буде й розгляд електробезпеки та пожежної безпеки на робочому місці при виконанні описаних вище робіт [12-14].

5.1. Аналіз умов праці

На працівників, зайнятих паянням виробів, можливий вплив наступних небезпечних і шкідливих виробничих факторів:

– підвищена загазованість повітря робочої зони парами шкідливих хімічних речовин;

– підвищена температура поверхні виробу, обладнання, інструменту, розплавів припоїв і солей;

– підвищена температура повітря робочої зони;

– небезпечний рівень напруги в електричному ланцюзі, замикання

– якого може відбутися через тіло людини;

– підвищений рівень шуму від двигунів вакуумних насосів, ультразвукового устаткування;

– підвищений рівень вібрації на робочому місці від двигунів вакуумних

– насосів;

– підвищений рівень електромагнітного випромінювання від неекранованих індукторів, трансформаторів, конденсаторів;

– підвищений рівень ультразвуку;

– підвищений рівень ультрафіолетової радіації під час виконання пайки;

– рентгенівське випромінювання через незахищені частини вакуумних камер електронно-променевих установок;

– пожежонебезпека;

– рухомі механізми і машини;

– бризки припоїв і розчинів.

При виконанні технологічних процесів виробництва необхідно виконувати вимоги зазначені правилами з охорони праці. Далі розглянуто основні вимоги [16].

5.1.1. Організація робочого місця

Організація і проведення робіт з пайки виробів повинні відповідати вимогам міжгалузевим правилам з охорони праці при проведенні робіт з пайки і лудіння виробів і чинних нормативних актів, що містять державні нормативні вимоги охорони праці.

Будівлі, в яких розташовуються ділянки пайки, а також інші цехи і ділянки слід будувати з вогнестійкого матеріалу, розміщувати по відношенню до житлових забудов з підвітряного боку і на відстані, що визначається відповідно до розрахунку розсіювання шкідливих речовин, але не менше 50 м від житлових забудов.

На території організації повинно бути ізольоване приміщення для збору, короткочасного зберігання та утилізації відходів, отриманих при виконанні робіт по пайці.

У виробничих приміщеннях повинні бути передбачені безпечні проходи і проїзди для руху людей і транспортних засобів. Ширина проїздів установлюється залежно від габаритів транспортних засобів і виробів. Межі проходів та проїздів повинні бути відзначені білими смугами шириною не менше 50 мм, металевими кнопками або іншими способами. Відстань від межі проїжджої частини до елементів конструкції будівель та обладнання повинна бути не менше 0,5 м, при русі людей –не менше 0,8 м.

У виробничих приміщеннях має бути не менше двох евакуаційних

виходів. Двері повинні мати ширину не менше 0,8 м і висоту – не менше 2 м.

Ворота, двері та інші прорізи в капітальних стінах, зроблені для технологічних цілей, повинні бути утеплені й обладнані тамбурами або тепловими повітряними завісами. Двері повинні мати пристосування для примусового закривання.

На ділянках пайки повинні знаходитися первинні засоби пожежогасіння відповідно до вимог Правил пожежної безпеки. Не допускається застосовувати для гасіння магнієвих сплавів воду та пінні вогнегасники.

5.1.2. Мікроклімат робочих приміщень

Мікроклімат виробничих приміщень повинен відповідати гігієнічним вимогам норм мікроклімату виробничих приміщень, згідно ГОСТ 12.1.005-88 і ДСН 3.3.6.042-99

Температура в приміщенні ділянки, де виконується пайка, через джерела тепла може перевищувати границю припустимої норми. Для видалення надлишку тепла використовується місцева механічна вентиляція. У теплий період року використовується природна вентиляція через віконні прорізи. Це дозволяє знизити температуру до нормального значення. У холодний період року може застосовуватись система центрального опалення у відповідності з вимогами СНиП 2.04.05-84 та ГОСТ 12.4.021-75.

5.1.3. Шкідливі речовини в повітрі робочої зони

Всі різновиди процесів пайки і залуження супроводжуються забрудненням повітряного середовища: аерозолями припою і флюсу, парами різних рідин, які використовуються для змивання флюсу та розчинення лаків.

Операції пайки і залужування супроводжуються забрудненням повітряного середовища в приміщеннях парами окису свинцю, олова, сурми та інших елементів, що входять до складу припою, а також парами каніфолі. Пари, потрапляючи в атмосферу цеху, конденсуються і перетворюються в

аерозоль конденсації.

Знаходячись у запиленій атмосфері, робітники піддаються впливу пилу та парів, шкідливі речовини осідають на шкірі, потрапляють на слизову оболонку порожнини рота, очей, верхніх дихальних шляхів, зі слиною потрапляють у травний тракт, вдихаються в легені. Поряд із забрудненням повітряного середовища забруднюються робочі поверхні й одяг працюючих. Ступінь впливу аерозолів залежить від хімічного складу, який визначається хімічним складом припою. Вміст шкідливих речовин у повітрі робочої зони не повинен перевищувати гранично допустимих концентрацій [18].

5.2 Розробка заходів з охорони праці

У даному підрозділі розглядаються заходи, які забезпечують здоров'я працівника і безпеку умов праці на робочому місці. До небезпечних та шкідливих виробничих факторів, які мають місце при технологічному процесі пайки відноситься:

– запиленість та загазованість повітря робочої зони;

– наявність інфрачервоних випромінювань від розплавленого припою у ванні чи від паяльника;

– наявність електромагнітного випромінювання високої частоти;

– дія ультразвуку на організм людей при використанні технологічного процесу пайки хвилею;

– вплив електростатичного розряду;

– незадовільна освітленість робочого місця чи підвищена яскравість;

– незадовільні метеорологічні умови в робочий зоні;

– вплив бризок та капель розплавленого припою;

– можливість ураження електричним струмом

– група психофізіологічних шкідливих виробничих факторів: фізичні перевантаження (статичні та динамічні), нервово-психічні перевантаження (монотонність роботи, емоційні перевантаження).

5.2.1. Електробезпека

Згідно ГОСТу 12.2.007.0-75 все електрообладнання, що знаходиться в робочому приміщені, можна віднести до 0I та I класу по електрозахисту (0I клас – обладнання, що має робочу ізоляцію, елемент заземлення і провід без заземлюючої жили; I клас – обладнання, що приєднується до ланцюга живлення трьох контактними вилками, один з цих контактів з'єднується з заземленим контактом розетки) [18].

Робоче приміщення за ступенем небезпеки ураження людей електричним струмом можна віднести, згідно ПУЕ та ГОСТ 12.1.013-78, до приміщень без підвищеної небезпеки, так як:

– відносна вологість повітря не перевищує 75%;

– матеріал підлоги (паркет) є діелектриком;

– температура повітря не досягає значень, більших ніж 35єС;

– відсутня можливість одночасного торкання людини до елементів металоконструкцій будівель, технологічних апаратів, механізмів, що мають з'єднання з землею, з одного боку і до металевих корпусів електрообладнання з іншого.

При нормальному (неаварійному) режимі роботи електрообладнання, гранично припустимі рівні напруг дотику і струмів через людину не повинні перевищувати значень (ГОСТ 12.1.038-88 ССБТ).

В робочому приміщені передбачене захисне відключення напруги живлення мережі при аварійному режимі роботи обладнання. Для зменшення значення напруги дотику і відповідних їй величин струмів при нормальному і аварійному режимах роботи електрообладнання необхідно виконати повторне захисне заземлення нульового дроту.

ВИСНОВКИ

В результаті виконання дипломної роботи створений веб-застосунок, що представляє певну гру – логічну головоломку, яка дозволить не тільки весело проводити час, а й освоювати теорію графів, зокрема теорію графів Ейлера, в зручний та цікавий спосіб.

Для цього, було виконано такі завдання:

1. Досліджені класифікація ігор, платформи для розробки ігор та основні моменти розробки комп'ютерних ігор.

2. Поглиблені знання з мови програмування JavaScript та її бібліотеки jQuery, мови гіперрозмітки HTML5 та таблиць каскадних стилів CSS3, вибір яких є кросплатформним рішенням, оскільки гра запускається в браузері на ПК з будь-якою операційною системою, тим самим створюючи апаратну незалежність для гри.

3. Аналіз наведених технологій на предмет розробки не тільки веб-застосунків, а й комп'ютерних ігор.

4. Опрацьовані матеріали з теорії графів, зокрема з побудови ейлерових графів та побудови циклів Ейлера на графах.

5. Створені правила гри, згідно яких користувач буде будувати цикли Ейлера, зокрема починати потрібно з вершини, а потім вибирати тільки ребра.

6. Розроблена логічна головоломка, що складається з 9 рівнів з можливістю перейти до наступного, пройшовши поточний рівень.

7. Проведене тестування комп'ютерної гри на предмет задоволення всіх функціональних вимог та перевірки виведення різного роду повідомлень при не коректній побудові циклу Ейлера згідно правил гри.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Класифікація і жанри комп’ютерних і відеоігор. Портал шкільної преси : веб-сайт. URL: https://zps19.at.ua/publ/mediateksti_vid_ekspertiv /klasifikacija_i_zhanri_komp_juternikh_i_videoigor/15-1-0-226 (дата звернення: 01.12.2023).
2. Усі жанри комп’ютерних ігор. Все про ігри : веб-сайт. URL: https://uaplay.com.ua/usi-zhanry-pc-ihor/ (дата звернення: 05.01.2024).
3. Joseph Hocking. Unity in action. Manning Publications, 2015, 352 p.
4. Мельник Р.А. Програмування веб-застосувань (фронт-енд та бек-енд) : навч. посіб. Львів : Львівська політехніка, 2018. 248 с.
5. Посібник по HTML5 и CSS3. Сайт про програмування : веб-сайт. URL: https://metanit.com/web/html5/ (дата звернення: 01.03.2024).
6. Документація по HTML5 и CSS3. Сайт про програмування : веб-сайт. URL: https://abitap.com/category/html-5-ta-css-3/ (дата звернення: 15.03.2024).
7. Посібник по JavaScript. Сайт про програмування : веб-сайт. URL: https://metanit.com/web/javascript/ (дата звернення: 20.03.2024).
8. Онлайн-книга "Вивчаєм jQuery". Сайт про програмування : веб-сайт. URL: https://metanit.com/web/jquery/ (дата звернення: 01.04.2024).
9. Бондаренко М.Ф., Білоус Н.В., Руткас А.Г. Комп’ютерна дискретна математика: Підручник. Харків, 2004. 480 с.
10. Kenneth H. Rosen Discrete Mathematics and Its Applications. McGrawHill Science, 928 p.
11. Кузьменко І.М. Теорія графів. Київ: КПІ ім. Ігоря Сікорського, 2020, 71 с.
12. Руснак І.С., Бабюк А.В., Федорова О.Є. Охорона праці при роботі з комп’ютером. Чернівці: Рута, 2005.128 с.
13. Березуцький В. В. Васьковець Л.А., Вершиніна Н.П. Безпека життєдіяльності : навчальний посібник. Х. : Факт, 2005. 384 с.
14. Вимоги безпеки до електричного устаткування для вимірювання, керування та лабораторного застосування. Частина 1.Загальні вимоги IEC

Додаток А. Лістинг файлів script.js, game.js

script.js

```
// Об'єкт, що зберігає функції для відображення різних подій гри
var shower = {};

// Функція для виконання дій на основі статусу info та id
елементу
var performAction = function(info, id) {
    // Словник, що містить відповідності статус-кодів та функцій
для їх обробки
    var dictionary = {
        "action301": shower.onNodeSelected,  // Вибір вершини
        "action302": shower.onEdgeVisited,   // Відвідування
ребра
        "action303": shower.onLevelComplete, // Завершення рівня
        "action304": shower.onGameFinished,  // Завершення гри

        "error401": shower.onStartNodeAlreadySelected,  //
Вибрано стартову вершину повторно
        "error402": shower.onStartNodeNotSelected,      // Не
вибрано стартову вершину
        "error403": shower.onEdgeRevisit,               // Ребро
відвідано повторно
        "error404": shower.onNonAdjacentVisit,          //
Відвідування не сусіднього ребра
        "error405": shower.onNoPossibleMoves            // Немає
можливих ходів
    };

    // Виконати відповідну дію чи обробку помилки на основі
статусу
    if (info.statusCode < 400)
        dictionary["action" + info.statusCode](id, info); //
Виклик функції для дії
    else
        dictionary["error" + info.statusCode](id, info); //
Виклик функції для помилки
};
```

```
// Функція для підготовки та відображення форми гри
var onShapeReady = function(shape) {
    var container = $('#container'); // Отримання контейнера для
SVG
    var svgHTML = '<svg width="400" height="400">'; // Початок
SVG

    // Об'єкт відвідувача для рендерингу компонентів гри
    var visitor = {};

    // Функція для рендерингу ребра
    visitor.renderEdge = function(edge) {
        var lineHTML = '<line id="ID" x1="X1" y1="Y1" x2="X2"
y2="Y2"/>';
        // Заповнення шаблону даними ребра
        lineHTML = lineHTML.replace('ID', edge.id)
                           .replace('X1', edge.startNode.x)
                           .replace('Y1', edge.startNode.y)
                           .replace('X2', edge.endNode.x)
                           .replace('Y2', edge.endNode.y);
        svgHTML += lineHTML; // Додавання лінії до SVG
    };

    // Функція для рендерингу вершини
    visitor.renderNode = function(node) {
        var circleHTML = '<circle id="ID" cx="CX" cy="CY" r="15"
stroke="black" stroke-width="3" fill="skyblue"/>';
        // Заповнення шаблону даними вершини
        circleHTML = circleHTML.replace('ID', node.id)
                               .replace('CX', node.x)
                               .replace('CY', node.y);
        svgHTML += circleHTML; // Додавання кола до SVG
    };

    // Функція для відображення прогресу рівня
    visitor.showLevel = function(noOfLevels, currentLevel) {
        var progressBar = $('#progress'); // Отримання прогрес-
бару
        var spanHeight = Math.round(progressBar.height() /
noOfLevels); // Визначення висоти кожного рівня
        var progressHTML = '';
```

Змн.	Арк.	№ докум.	Підпис	Дата	Логічна гра на основі графів Ейлера	Літ.	Маса	Масштаб
Розроб.		Олійник В.Я.						
Перевір.		Букурос О.В.				Арк.	Аркушів 1	
Т. Контр.						Циклова комісія Комп'ютерної інженерії		
Реценз.								
Н. Контр.								
Затверд.								

```
        // Заповнення прогрес-бару відповідними рівнями
        for (var i = noOfLevels; i >= 1; i--) {
            if (i < currentLevel)
                progressHTML += '<span class="complete">' + i +
'</span>'; // Відображення завершених рівнів
            else
                progressHTML += '<span>' + i + '</span>'; //
Відображення поточних та незавершених рівнів
        }

        progressBar.html(progressHTML); // Оновлення HTML
прогрес-бару
        $('#progress span').css({'height': spanHeight - 2 +
'px'}); // Встановлення висоти рівнів
    };

    // Виклик рендерингу форми гри через відвідувача
    shower.game.visit(visitor);
    svgHTML += '</svg>'; // Закриття SVG
    container.html(svgHTML); // Вставка SVG в контейнер
};

// Функція, що викликається при відвідуванні ребра
shower.onEdgeVisited = function(edgeID, info) {
    $('#' + edgeID).attr('class', 'visited'); // Додавання класу
"відвіданий" до ребра
    shower.onNodeSelected(info.nodeId); // Виклик функції для
вибору вершини
};

// Функція, що викликається при виборі вершини
shower.onNodeSelected = function(nodeID) {
    var circles = $('circle'); // Отримання всіх вершин
    circles.attr('fill', 'skyblue'); // Скидання кольору всіх
вершин
    $('#' + nodeID).attr('fill', 'pink'); // Встановлення
кольору вибраної вершини
};
```

Змн.	Арк.	№ докум.	Підпис	Дата	Логічна гра на основі графів Ейлера	Літ.	Маса	Масштаб
Розроб.		Олійник В.Я.						
Перевір.		Букурос О.В.				Арк.	Аркушів 1	
Т. Контр.						Циклова комісія Комп'ютерної інженерії		
Реценз.								
Н. Контр.								
Затверд.								

```
// Функція, що викликається при завершенні рівня
shower.onLevelComplete = function(edgeID, info) {
    shower.onEdgeVisited(edgeID, info); // Відмітка ребра як
відвіданого
    setTimeout(function() {
        // Приховування та показ різних елементів інтерфейсу
через 500 мс
        $('#error').hide();
        $('#container').hide();
        $('#finish').hide();
        $('#level').show();
        $('#instruction').hide();
        $('.resetLevel').hide();
    }, 500);
};

// Функція для показу повідомлень про помилки
var showErrorMessage = function(message) {
    var error = $('#error'); // Отримання блоку з помилкою
    var container = $('#container'); // Отримання контейнера гри
    error.find('h2').text(message); // Встановлення тексту
повідомлення
    $('circle').prop('disabled', true); // Вимкнення всіх вершин
    $('line').prop('disabled', true); // Вимкнення всіх ребер
    $('.resetLevel').prop('disabled', true); // Вимкнення кнопки
перезапуску рівня
    error.animate({
        height: "toggle",
        display: 'inline-block'
    }, 1000, function() {});
    error.css({
        display: 'inline-block'
    });

    container.css({
        opacity: 0.5
    });
```

```
        setTimeout(function() {
            error.animate({
                height: "toggle"
            }, 1000, function() {
                container.css({
                    opacity: 1.0
                });
            });
            $('circle').prop('disabled', false); // Увімкнення всіх
вершин
            $('line').prop('disabled', false); // Увімкнення всіх
ребер
            $('.resetLevel').prop('disabled', false); // Увімкнення
кнопки перезапуску рівня
        }, 3000);
        // Завершення анімації.
    };

    // Обробники помилок для різних ситуацій
    shower.onStartNodeAlreadySelected = function(nodeID) {
        showErrorMessage('Ви повинні вибрати РЕБРО, щоб продовжити
гру!');
    };

    shower.onNonAdjacentVisit = function() {
        showErrorMessage('Ви можете вибрати лише РЕБРО, яке примикає
до поточного вузла!');
    };

    shower.onStartNodeNotSelected = function() {
        showErrorMessage('Вам потрібно вибрати ВЕРШИНУ, щоб почати
гру!');
    };

    shower.onEdgeRevisit = function() {
        showErrorMessage('Це РЕБРО уже відвідали');
    };
```

					Логічна гра на основі графів Ейлера	Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Олійник В.Я.						
Перевір.		Букурос О.В.						
Т. Контр.						Арк.	Аркушів	1
Реценз.								
Н. Контр.						Циклова комісія Комп'ютерної інженерії		
Затверд.								

```
shower.onNoPossibleMoves = function(edgeID, info) {
    shower.onEdgeVisited(edgeID, info); // Відмітка ребра як
відвіданого
    setTimeout(function() {
        $('#movesFinished').show(); // Показ повідомлення про
завершення ходів
        $('circle').prop('disabled', true); // Вимкнення всіх
вершин
        $('line').prop('disabled', true); // Вимкнення всіх
ребер
    }, 500);
};

// Функція, що викликається при завершенні гри
shower.onGameFinished = function() {
    $('#container').hide(); // Приховування контейнера гри
    $('#error').hide(); // Приховування блоку з помилками
    $('#level').hide(); // Приховування рівня
    $('.resetLevel').hide(); // Приховування кнопки перезапуску
рівня
    $('#finish').show(); // Показ повідомлення про завершення
гри
    $('#progress').hide(); // Приховування прогрес-бару
};

// Функція ініціалізації гри
var init = function() {
    shower.game = new Game(); // Створення нового об'єкта гри
    onShapeReady(); // Підготовка та відображення форми гри
    $('#container').on('click', 'line', function() {
        var edgeID = $(this).attr('id'); // Отримання ID ребра
        var info = shower.game.visitEdge(edgeID); //
Відвідування ребра у грі
        performAction(info, edgeID); // Виконання відповідної
дії
```

					Логічна гра на основі графів Ейлера	*Літ.*	*Маса*	*Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Олійник В.Я.*						
Перевір.		*Букурос О.В.*						
Т. Контр.						*Арк.*	*Аркушів*	1
Реценз.								
Н. Контр.						*Циклова комісія Комп'ютерної інженерії*		
Затверд.								

```
    });
    $('#container').on('click', 'circle', function() {
        var nodeID = $(this).attr('id'); // Отримання ID вершини
        var info = shower.game.selectNode(nodeID); // Вибір
вершини у грі
        performAction(info, nodeID); // Виконання відповідної
дії
    });
    $('.resetLevel').click(function() {
        shower.game.restartLevel(); // Перезапуск рівня гри
        onShapeReady(); // Підготовка та відображення форми гри
        $('#movesFinished').hide(); // Приховування повідомлення
про завершення ходів
    });
    $('#changeLevel').click(function() {
        shower.game.swicthToNextLevel(); // Перехід на наступний
рівень
        onShapeReady(); // Підготовка та відображення форми гри
        $('#error').hide(); // Приховування блоку з помилками
        $('#level').hide(); // Приховування рівня
        $('#finish').hide(); // Приховування повідомлення про
завершення гри
        $('#container').show(); // Показ контейнера гри
        $('.resetLevel').show(); // Показ кнопки перезапуску
рівня
    });
};

// Запуск ініціалізації гри при завантаженні сторінки
$(window).load(init);
```

					Логічна гра на основі графів Ейлера	Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Олійник В.Я.						
Перевір.		Букурос О.В.				Арк.	Аркушів	1
Т. Контр.								
Реценз.						Циклова комісія Комп'ютерної інженерії		
Н. Контр.								
Затверд.								

game.js

```
// Конструктор для створення об'єктів Node (вершин)
var Node = function(x, y, id) {
     this.x = x; // Координата x вершини
     this.y = y; // Координата y вершини
     this.id = 'node' + id; // Унікальний ідентифікатор вершини
     this.edges = []; // Масив для збереження ребер, пов'язаних з
цією вершиною

     // Метод для перевірки, чи дві вершини є рівними (за
координатами)
     this.isEqualTo = function(node) {
          return this.x === node.x && this.y === node.y;
     };

     // Метод для додавання ребра до вершини
     this.addEdge = function(edge) {
          this.edges.push(edge);
     };

     // Метод для перевірки, чи ребро є суміжним з вершиною
     this.isEdgeAdjacent = function(edge) {
          return this.edges.some(function(edge1) {
               return edge.id === edge1.id;
          });
     };

     // Метод для перевірки, чи всі ребра вершини відвідані
     this.areAllEdgesVisited = function() {
          return this.edges.every(function(edge) {
               return edge.visited;
          });
     };
};
```

					Логічна гра на основі графів Ейлера	Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Олійник В.Я.						
Перевір.		Букурос О.В.						
Т. Контр.						Арк.	Аркушів	1
Реценз.								
Н. Контр.						Циклова комісія Комп'ютерної інженерії		
Затверд.								

```
// Конструктор для створення об'єктів Edge (ребер)
var Edge = function(startNode, endNode, id) {
     this.startNode = startNode; // Початкова вершина ребра
     this.endNode = endNode; // Кінцева вершина ребра
     this.id = 'edge' + id; // Унікальний ідентифікатор ребра
     this.visited = false; // Стан відвіданості ребра

     // Метод для перевірки, чи два ребра є рівними (за
вершинами)
     this.isEqualTo = function(edge) {
          return this.startNode.isEqualTo(edge.startNode) &&
this.endNode.isEqualTo(edge.endNode);
     };

     // Метод для отримання іншої вершини ребра
     this.getOtherNode = function(node) {
          if (node.isEqualTo(this.startNode))
               return this.endNode;
          return this.startNode;
     };
};

// Функція для створення форми (графа) гри
var createShape = function(game) {
     var shapeData = game.shapesData[game.level - 1]; // Дані
форми для поточного рівня
     game.noOfEdgeVisited = 0; // Лічильник відвіданих ребер
     game.nodes = shapeData.nodesData.map(function(nodeData,
index) {
          return new Node(nodeData[0], nodeData[1], index); //
Створення вершин
     });
     game.edges = shapeData.edgesData.map(function(edgeData,
index) {
          var start = game.nodes.filter(function(node) {
               return node.isEqualTo({ x: edgeData[0], y:
edgeData[1] });
          })[0]; // Пошук початкової вершини для ребра
          var end = game.nodes.filter(function(node) {
               return node.isEqualTo({ x: edgeData[2], y:
edgeData[3] });
```

					Логічна гра на основі графів Ейлера	*Літ.*	*Маса*	*Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Олійник В.Я.*						
Перевір.		*Букурос О.В.*						
Т. Контр.						*Арк.*	*Аркушів* 1	
Реценз.								
Н. Контр.						*Циклова комісія Комп'ютерної інженерії*		
Затверд.								

```
            })[0]; // Пошук кінцевої вершини для ребра
            var edge = new Edge(start, end, index); // Створення
ребра
            start.addEdge(edge); // Додавання ребра до початкової
вершини
            end.addEdge(edge); // Додавання ребра до кінцевої
вершини
            return edge; // Повернення ребра
      });
};

// Конструктор для створення об'єктів Game (гри)
var Game = function(shapesData) {
      this.level = 1; // Поточний рівень гри
      this.shapesData = shapesData || x; // Дані форм для гри
      this.noOfLevels = this.shapesData.length; // Кількість
рівнів у грі
      createShape(this); // Створення форми для поточного рівня
};

// Метод для отримання ребра за його ідентифікатором
Game.prototype.getEdgeById = function(edgeID) {
      return this.edges.filter(function(edge) {
            return edge.id === edgeID;
      })[0];
};

// Метод для перевірки, чи рівень завершено
Game.prototype.isLevelComplete = function() {
      return this.noOfEdgeVisited === this.edges.length;
};

// Метод для отримання вершини за її ідентифікатором
Game.prototype.getNodeById = function(nodeIdToFind) {
      return this.nodes.filter(function(node) {
            return node.id === nodeIdToFind;
      })[0];
};
```

					Логічна гра на основі графів Ейлера	*Літ.*	*Маса*	*Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Олійник В.Я.*						
Перевір.		*Букурос О.В.*						
Т. Контр.						*Арк.*	*Аркушів* 1	
Реценз.						*Циклова комісія Комп'ютерної інженерії*		
Н. Контр.								
Затверд.								

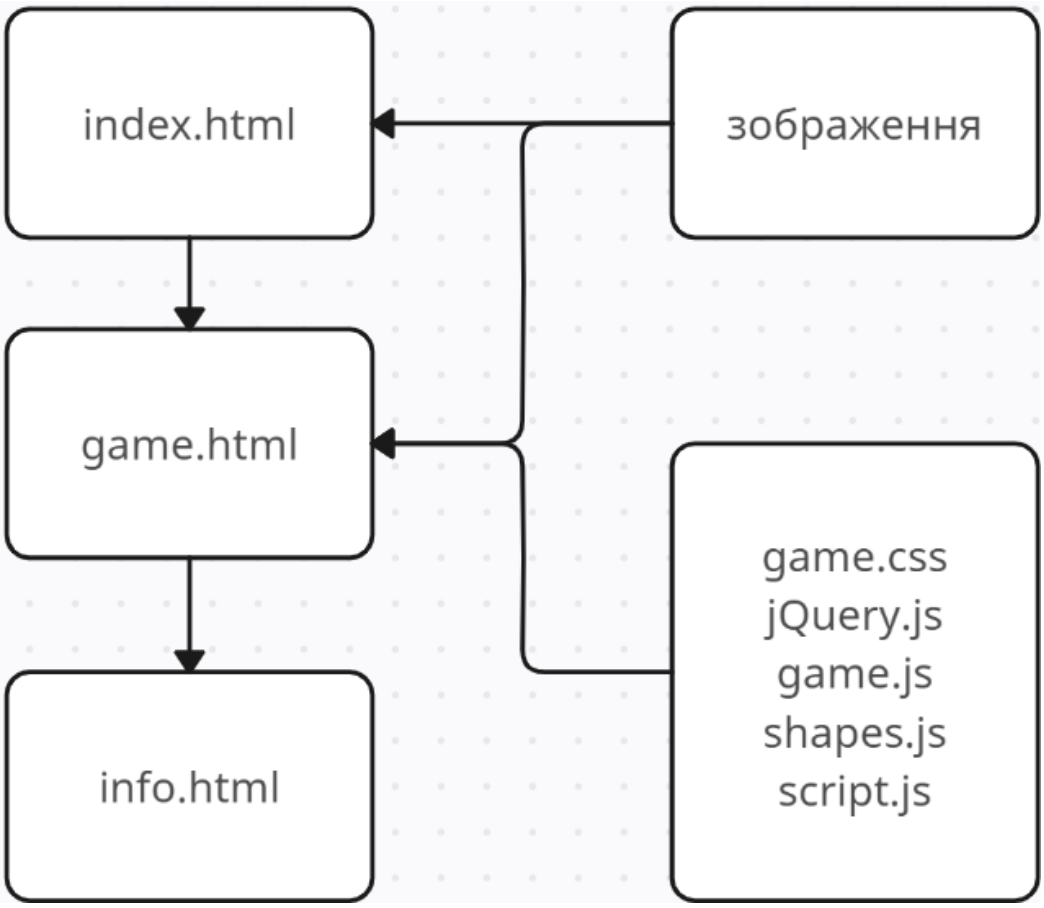
```
// Метод для відвідування відвідувача (рендеринга)
Game.prototype.visit = function(visitor) {
     this.edges.forEach(visitor.renderEdge); // Рендеринг ребер
     this.nodes.forEach(visitor.renderNode); // Рендеринг вершин
     visitor.showLevel(this.noOfLevels, this.level); // Показ
рівня
};

// Метод для вибору вершини
Game.prototype.selectNode = function(nodeId) {
     if (!this.currentNode) {
          this.currentNode = this.getNodeById(nodeId); //
Встановлення поточної вершини
          return { statusCode: 301 }; // Повернення статусу
вибору вершини
     }
     return { statusCode: 401 }; // Вершина вже вибрана
};

// Метод для відвідування ребра
Game.prototype.visitEdge = function(edgeID) {
     var edge = this.getEdgeById(edgeID);
     if (!this.currentNode) {
          return { statusCode: 402 }; // Поточна вершина не
вибрана
     }
     if (edge.visited) {
          return { statusCode: 403 }; // Ребро вже відвідано
     }
     if (!this.currentNode.isEdgeAdjacent(edge)) {
          return { statusCode: 404 }; // Відвідування несуміжного
ребра
     }
     edge.visited = true; // Позначення ребра як відвіданого
     this.noOfEdgeVisited++; // Збільшення лічильника відвіданих
ребер
     this.currentNode = edge.getOtherNode(this.currentNode); //
Встановлення нової поточної вершини
```

```
        if (this.isLevelComplete()) {
            if (this.level === this.noOfLevels) {
                return { statusCode: 304 }; // Гра завершена
            }
            return {
                statusCode: 303,
                nodeId: this.currentNode.id
            }; // Рівень завершений
        }

        if (this.currentNode.areAllEdgesVisited()) {
            return {
                statusCode: 405,
                nodeId: this.currentNode.id
            }; // Немає можливих ходів
        }

        return {
            statusCode: 302,
            nodeId: this.currentNode.id
        };
    };

    // Метод для перезапуску рівня
    Game.prototype.restartLevel = function() {
        createShape(this); // Створення форми для поточного рівня
        this.currentNode = undefined; // Скидання поточної вершини
    };

    // Метод для переходу на наступний рівень
    Game.prototype.swicthToNextLevel = function() {
        this.level++; // Збільшення поточного рівня
        this.currentNode = undefined; // Скидання поточної вершини
        createShape(this); // Створення форми для нового рівня
    };
```

					Логічна гра на основі графів Ейлера	*Літ.*	*Маса*	*Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Олійник В.Я.*						
Перевір.		*Букурос О.В.*						
Т. Контр.						*Арк.*	*Аркушів* 1	
Реценз.						*Циклова комісія Комп'ютерної інженерії*		
Н. Контр.								
Затверд.								

Додаток Б.

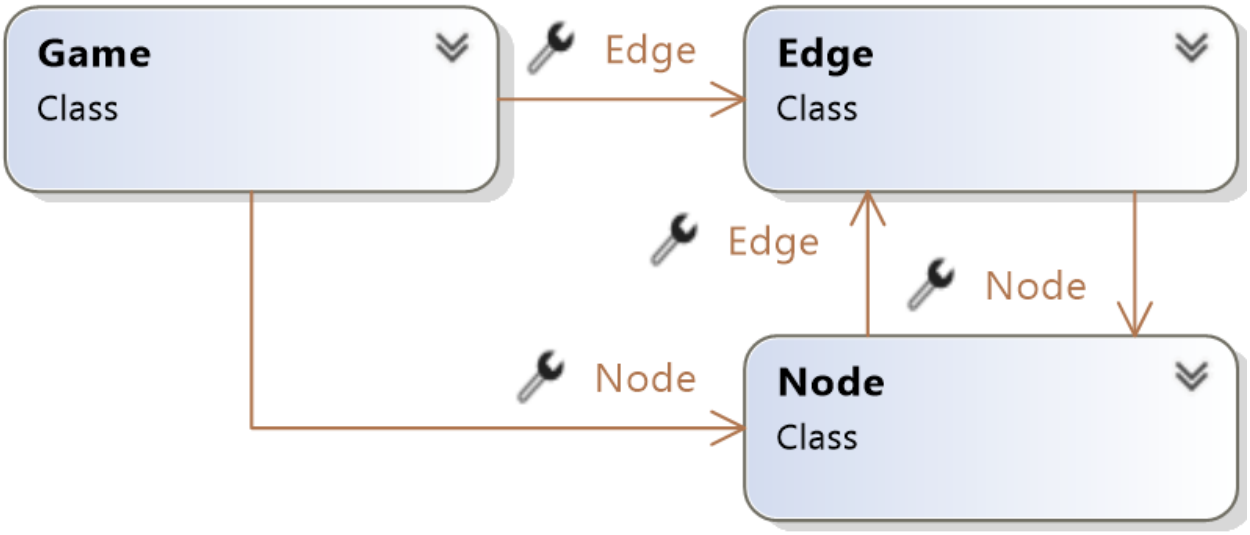
Структура файлів застосунку



					Логічна гра на основі графів Ейлера	Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Олійник В.Я.					
Перевір.		Букурос О.В.				Арк.	Аркушів 1
Т. Контр.							
Реценз.						Циклова комісія Комп'ютерної інженерії	
Н. Контр.							
Затверд.							

Додаток В.

Діаграма класів файлу game.js



					Логічна гра на основі графів Ейлера	Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Олійник В.Я.						
Перевір.		Букурос О.В.				Арк.	Аркушів 1	
Т. Контр.								
Реценз.						Циклова комісія Комп'ютерної інженерії		
Н. Контр.								
Затверд.								

Додаток Г.

Діаграма прецедентів користувача

					Логічна гра на основі графів Ейлера	Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Олійник В.Я.						
Перевір.		Букурос О.В.				Арк.	Аркушів	1
Т. Контр.								
Реценз.						Циклова комісія Комп'ютерної інженерії		
Н. Контр.								
Затверд.								