Прикарпатський національний університет імені Василя Стефаника
Факультет математики та інформатики
Кафедра комп'ютерних наук та інформаційних систем

БАКАЛАВРСЬКА РОБОТА

на здобуття першого (бакалаврського) рівня вищої освіти

на тему "Гейміфікована система підтримки вивчення технологій HTML/CSS"

Виконав: студент 4-го курсу, групи
ІСТ-41
спеціальності
126 Інформаційні системи та технології
(шифр і назва спеціальності)

Невмержицький В.Р.
(прізвище та ініціали студента)
Керівник
ст. викл. Ізмайлов А.В.
(прізвище та ініціали)
Рецензент
ст. викл. Никорак Я.Я
(прізвище та ініціали)

Івано-Франківськ – 2024 р.

АНОТАЦІЯ

Дипломна робота присвячена розробці гейміфікованої системи підтримки вивчення технологій HTML/CSS. Обсяг роботи включає частини з технічного проектування, роботи з ігровим рушієм та програмування. Розроблено карткову гру, яка допомагає вивченню Веб-технологій. Ігролад спрямований на покращення сприйняття й розуміння тегів HTML та правил CSS.

Робота включає в себе вступ, 3 розділи, висновки і перелік джерел. Проаналізовано використання різноманітних платформ та ігрових рушіїв для підвищення ефективності навчання. У роботі наведено 17 рисунків, 15 джерел інформації, 53 сторінки та 2 додатки .

Ключові слова: Веб-технології, Гейміфікація, Ігрові рушії, Інтерактивні методи навчання, Карткова гра.

ABSTRACT

This thesis is dedicated to the development of a gamified system to support learning HTML/CSS technologies. The scope of the work includes sections on technical design, working with a game engine, and programming. A card game has been developed to aid in learning web technologies. The gameplay is aimed at improving the understanding and perception of HTML tags and CSS rules.

The thesis includes an introduction, three chapters, conclusions, and a list of references. Various platforms and game engines were analyzed to enhance learning efficiency. The work contains 12 figures, 15 references, 39 pages, and 1 appendix.

Keywords: Card game, Game engines, Gamification, Interactive learning methods, Web-technologies.

ЗМІСТ

ВСТУП

Актуальність та місце проблеми

В сучасному освітньому середовищі вивчення технологій HTML/CSS стає обов'язковою складовою для багатьох напрямків навчання. Проте, несвідоме засвоєння цих мов може призвести до поверхневого розуміння їх сутності та невпевненості у власних навичках. Тому, питання розробки ефективних методів навчання HTML/CSS виходить на передній план. Використання гейміфікованого підходу може стати не лише цікавим засобом вивчення, але й сприятиме глибшому засвоєнню концепцій та практичних навичок цих технологій.[1]

Сучасний стан проблеми

Сучасне вивчення HTML і CSS серед учнів стикається з кількома проблемами, включаючи низьку мотивацію та зацікавленість через абстрактність теоретичних концепцій і відсутність інтерактивного підходу. Багато існуючих навчальних ресурсів не забезпечують практичної реалізації, що ускладнює засвоєння матеріалу. Гра в даному контексті може стати ефективним інструментом для вирішення цих проблем [2, 3].

Мета роботи

Метою роботи є створення карткової гри як інтерактивного навчального інструмента для поглиблення знань та вмінь у галузі HTML та CSS та підвищення зацікавленості учнів навчанням веб-технологій.

Центральна проблема, яку поставлено перед собою у цій роботі, - це розробка ефективного засобу для підтримки вивчення технологій HTML/CSS через використання карткової гри. Для досягнення цієї мети, необхідно дослідити існуючі проблеми в навчанні, розробити гру, провести аналіз результатів з метою вдосконалення навчального процесу.

Об'єктом дослідження є гейміфікація навчання веб-технологіям. **Предметом** роботи є карткова гра для підтримки вивчення технологій HTML/CSS.

Основні цілі

Розробити ефективний навчальний інструмент та вивчити його вплив на процес вивчення. Завдання включають аналіз проблем в навчанні, розробку гри, проведення експерименту та аналіз результатів. Для досягнення поставлених завдань будуть використані методи аналізу літературних джерел, тестування, спостереження.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Гейміфікація в освітньому процесі

- Поняття гейміфікації

Гейміфікація як метод використання елементів ігрових механік у контекстах, не пов'язаних з іграми, стає все більш популярною в сучасному навчальному процесі. Особливо це стосується вивчення веб-технологій, таких як HTML і CSS. Розгляд цієї предметної області може виявити переваги та проблеми, пов'язані з інтеграцією гейміфікації в освітні програми [2].

- Мотивація та інтерес

Гейміфікація може суттєво підвищити мотивацію до вивчення веб-технологій, яка часто буває недостатньою через абстрактність матеріалу та його спеціалізацію. Ігрові елементи, такі як завдання, рівні складності та нагороди, можуть зробити процес навчання більш захоплюючим, що підвищує інтерес учнів. Важливою перевагою є можливість формування внутрішньої мотивації, коли студенти відчувають задоволення від самого процесу навчання і досягнення результатів [3].

- Підвищена активність і залученість

Графічні та анімаційні елементи гейміфікації можуть значно полегшити вивчення веб-технологій. Використання візуальних ефектів для вирішення проблем може допомогти учням краще зрозуміти поняття HTML/CSS та заохотити їх до активної участі у навчанні. Гейміфікація сприяє глибшій взаємодії з матеріалом та підвищує рівень залученості студентів. Це особливо важливо в дистанційному навчанні, де підтримка високого рівня активності є ключовим фактором успішності.

- Адаптивне та індивідуалізоване навчання

Гейміфікація дозволяє створювати індивідуальні програми для учнів з різним рівнем підготовки та підходами до навчання. Адаптивні ігри можуть адекватно реагувати на різні потреби та темпи навчання, надаючи кожному студенту можливість адаптувати свій шлях до технологічного розвитку. Такий підхід дозволяє враховувати індивідуальні потреби кожного учня, що сприяє більш ефективному засвоєнню матеріалу та підвищенню загального рівня знань.

- Заохочення командної роботи

Гейміфікація може стимулювати колективне навчання та вирішення проблем. Створюючи групові завдання та конкурси, можна підвищити ефективність навчання, застосовуючи на практиці навички співпраці та комунікації. Командна робота сприяє розвитку соціальних навичок, що є важливим аспектом у професійній діяльності майбутніх фахівців. Крім того, колективне вирішення завдань дозволяє учням обмінюватися знаннями та досвідом, що сприяє глибшому розумінню матеріалу.

- Проблеми та обмеження

Незважаючи на численні переваги, гейміфікація має свої труднощі та обмеження. Перш за все, необхідно ретельно визначити правила гри та збалансувати рівень складності, щоб уникнути стресу та нудьги у учнів. Важливо також запобігти відволіканню уваги від основних цілей навчання. Занадто складні або занадто легкі завдання можуть негативно вплинути на мотивацію студентів. Крім того, інтеграція гейміфікації в навчальний процес вимагає додаткових ресурсів, таких як час та фінанси, для розробки та впровадження ефективних гейміфікованих рішень.

- Психологічні аспекти гейміфікації

Психологічний вплив гейміфікації на студентів також є важливим аспектом. Використання гейміфікації може сприяти розвитку позитивного ставлення до навчання та підвищення самооцінки учнів. Проте, необхідно враховувати можливість виникнення залежності від ігрових елементів, що

може призвести до зниження інтересу до традиційних методів навчання. Баланс між гейміфікованими та традиційними методами навчання є ключовим фактором для досягнення оптимальних результатів [3].

- Технічні аспекти впровадження гейміфікації

Впровадження гейміфікації в навчальний процес вимагає наявності певних технічних ресурсів та інфраструктури. Необхідно забезпечити доступ до сучасних комп'ютерів, програмного забезпечення та інтернету для всіх учасників навчання. Крім того, важливо, щоб викладачі мали необхідні навички для використання гейміфікованих інструментів та платформ. Це може вимагати додаткового навчання та підвищення кваліфікації педагогів.

- Оцінка ефективності гейміфікації

Для оцінки ефективності гейміфікації необхідно розробити відповідні методи та інструменти. Це можуть бути опитування, тести, аналіз результатів навчання та спостереження за активністю учнів. Важливо також враховувати зворотний зв'язок від студентів, щоб мати можливість коригувати підходи та вдосконалювати гейміфіковану систему навчання. Регулярний моніторинг і аналіз ефективності дозволить виявляти сильні та слабкі сторони гейміфікації та оптимізувати її використання в навчальному процесі.

Висновки

Гейміфікація є потужним інструментом для підвищення мотивації, залученості та ефективності навчання веб-технологій, таких як HTML і CSS. Вона дозволяє адаптувати навчальні програми до індивідуальних потреб учнів, сприяє розвитку командної роботи та соціальних навичок. Однак, впровадження гейміфікації в навчальний процес потребує ретельного планування та балансу для уникнення можливих проблем і обмежень. Важливими аспектами є також технічне забезпечення та

підготовка викладачів, а також регулярний моніторинг ефективності та коригування підходів на основі зворотного зв'язку від учнів.

1.2 Аналіз існуючих рішень

Існують численні гейміфіковані додатки для вивчення веб-технологій, які використовують ігрові елементи для підвищення мотивації та залучення студентів. Такі додатки часто включають інтерактивні уроки, де користувачі можуть навчатися, виконуючи завдання з HTML, CSS, JavaScript та інших технологій.

Також є платформи, які пропонують структуру навчання через серії проектів і завдань, що дозволяє користувачам створювати реальні веб-додатки. Інші додатки використовують короткі уроки та інтерактивні завдання, що нагадують міні-ігри, де студенти програмують, щоб вирішувати проблеми та проходити рівні. Існують і платформи з конкурсами і змаганнями, які заохочують користувачів брати участь у викликах з програмування, що допомагає вдосконалювати свої навички в умовах здорової конкуренції.

Coding Fantasy

Coding Fantasy — це гейміфікована освітня платформа для вивчення програмування, яка використовує елементи фантазії та гри, щоб зробити навчання більш захоплюючим. На цій платформі навчання подається у вигляді пригодницьких сюжетів, де користувачі грають роль персонажів, що виконують різноманітні квестові завдання [4].

Переваги:

- Геймплей тісно поєднаний з кодом:

Відмінності гри полягають у її тісному зв'язку з програмним кодом. Це створює стимулююче навчальне середовище, де студенти можуть навчатися шляхом практичного застосування концепцій веб-розробки.

- Просте та зрозуміле викладення теорії:

Один із сильних боків цього ресурсу - коротке та зрозуміле пояснення теоретичних аспектів. Це дозволяє швидко усвідомлювати ключові концепції, що є особливо корисним для новачків у веб-розробці.

- Практичні вправи та наочні приклади:

Наявність багатоцільових та наочних вправ допомагає студентам отримувати практичний досвід та закріплювати знання у веб-розробці (Рис.1.1).

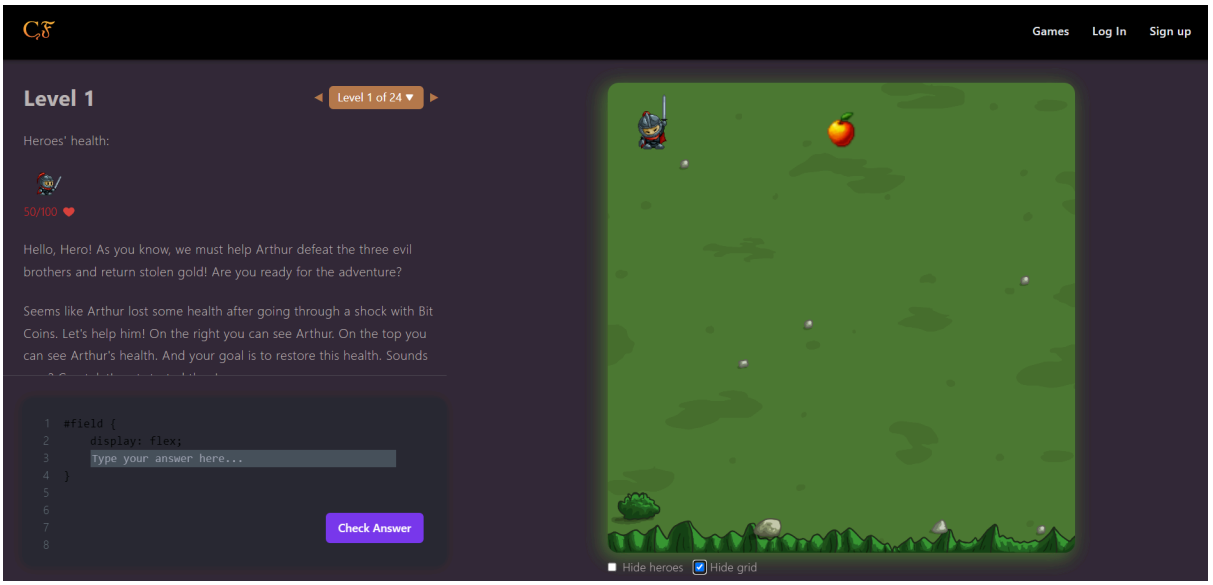

Рисунок 1.1 – Інтерфейс ігрового процесу Coding Fantasy [4]

Недоліки:

- Вузька направленість на flex та grid:

Одним з недоліків є вузька спеціалізація на конкретних технологіях, таких як flex та grid. Це може обмежити можливості студентів у повній

вивченні широкого спектру веб-розробки, оскільки існують інші важливі теми, які не охоплені цим ресурсом (Рис. 1.2).

- Обмеженість у вивченні різних аспектів веб-розробки:

Навчання на Coding Fantasy обмежується лише деякими аспектами веб-розробки, що може ускладнити отримання повного уявлення про цю галузь програмування.

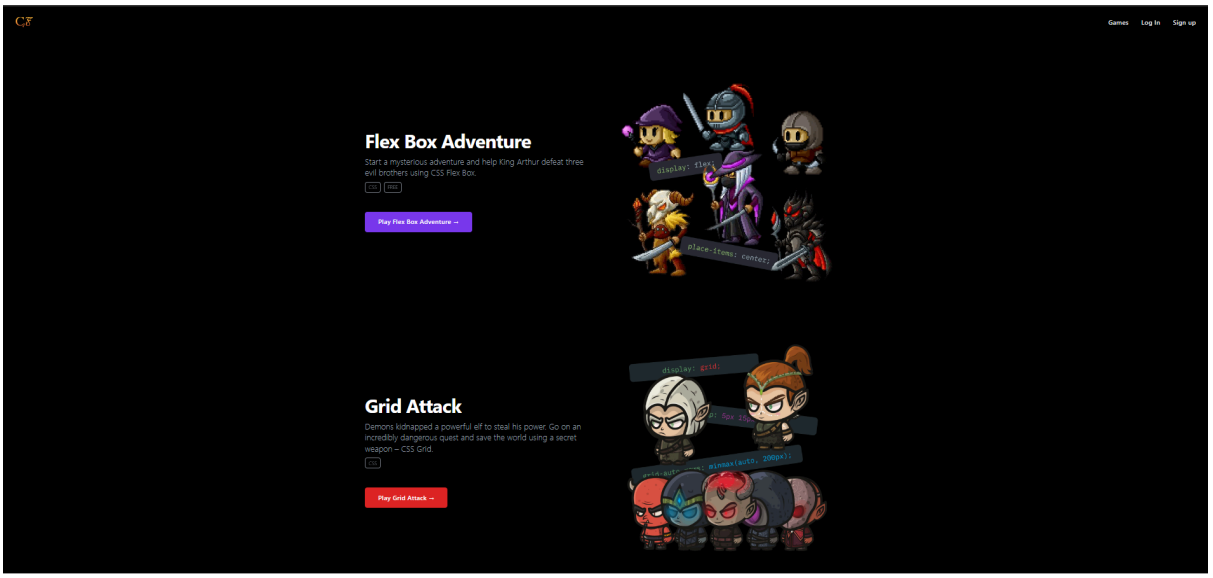

Рисунок 1.2 – Обмеженість тем Coding fantasy [4]

Загальна мета будь-якого навчального ресурсу – забезпечити студентам якісне та всебічне навчання, що дозволяє їм отримати не лише базові знання, а й глибокі розуміння предметної області. У випадку з Coding Fantasy, хоча він має свої очевидні переваги, такі як інтеграція геймплею з кодом та доступність лаконічних пояснень, важливо розглянути, які саме аспекти веб-розробки він охоплює та як це відповідає загальному контексту програмування та розробки.

Ключовим аспектом розгляду є спрямованість Coding Fantasy на певні технології, зокрема flex та grid. Це може бути як перевагою, так і недоліком, залежно від потреб та очікувань студентів. З одного боку, фокус

на конкретних технологіях дозволяє глибше вивчити їхні можливості та особливості. З іншого боку, це може обмежити студентів у розумінні ширшого спектру інструментів і методів, які також є важливими для успішного розвитку веб-розробника.

Важливо також враховувати, як вміщається навчання на Coding Fantasy в загальний контекст програмування та розробки. Хоча цей ресурс може бути корисним для отримання базових навичок та ознайомлення з веб-розробкою, він може бути недостатнім для тих, хто прагне стати професійним веб-розробником і має амбіції працювати з різноманіттям технологій та платформ.

Отже, при виборі навчального ресурсу важливо зважити на його переваги та обмеження, а також на власні потреби та мету вивчення. Хоча Coding Fantasy може бути корисним для тих, хто шукає захоплюючий та інтерактивний спосіб вивчення веб-розробки, варто також розглянути додаткові ресурси та інструменти, які допоможуть розширити свої знання та навички у цій області.

Codepip

Codepip — це освітня платформа для вивчення веб-технологій через інтерактивні ігри. Вона надає користувачам можливість вчитися програмуванню за допомогою гейміфікованих вправ, що робить процес навчання захоплюючим і ефективним [5].

Переваги:

- Гнучкість навчання:

Codepip пропонує гнучкий підхід до навчання веб-розробки, що дозволяє студентам вибирати теми та завдання відповідно до їхніх індивідуальних потреб і рівня навичок.

- Інтерактивність та практика:

Ресурс створений з акцентом на інтерактивність та практичну складову навчання, що дозволяє студентам навчатися шляхом практичного застосування знань у веб-розробці.

- Різноманітність тем:

Codepip пропонує широкий вибір тем, що охоплюють різні аспекти веб-розробки, що дає студентам можливість поглибленого вивчення конкретних областей за індивідуальними потребами (Рис. 1.3).

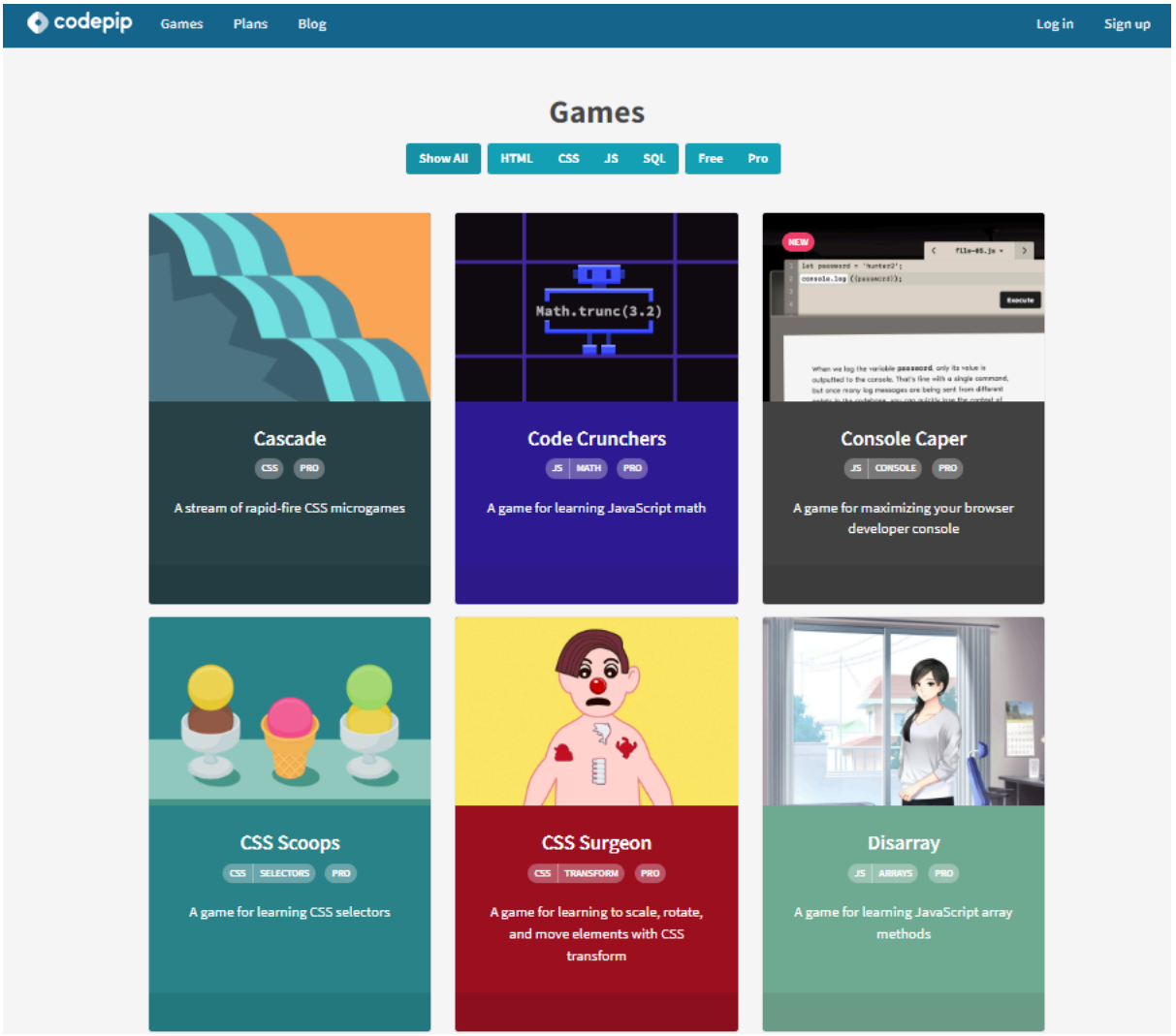

Рисунок 1.3 – Широкий вибір codepip [5]

Недоліки:

- Платний контент:

Більшість доступного контенту на ресурсі є платним, що може створити обмеження для студентів з обмеженим бюджетом або тих, хто не бажає інвестувати кошти у навчання.

- Можливий бар'єр для доступу:

Висока концентрація платного контенту може стати бар'єром для доступу до важливих навчальних ресурсів для широкого кола користувачів, що може ускладнити процес навчання (Рис. 1.4).

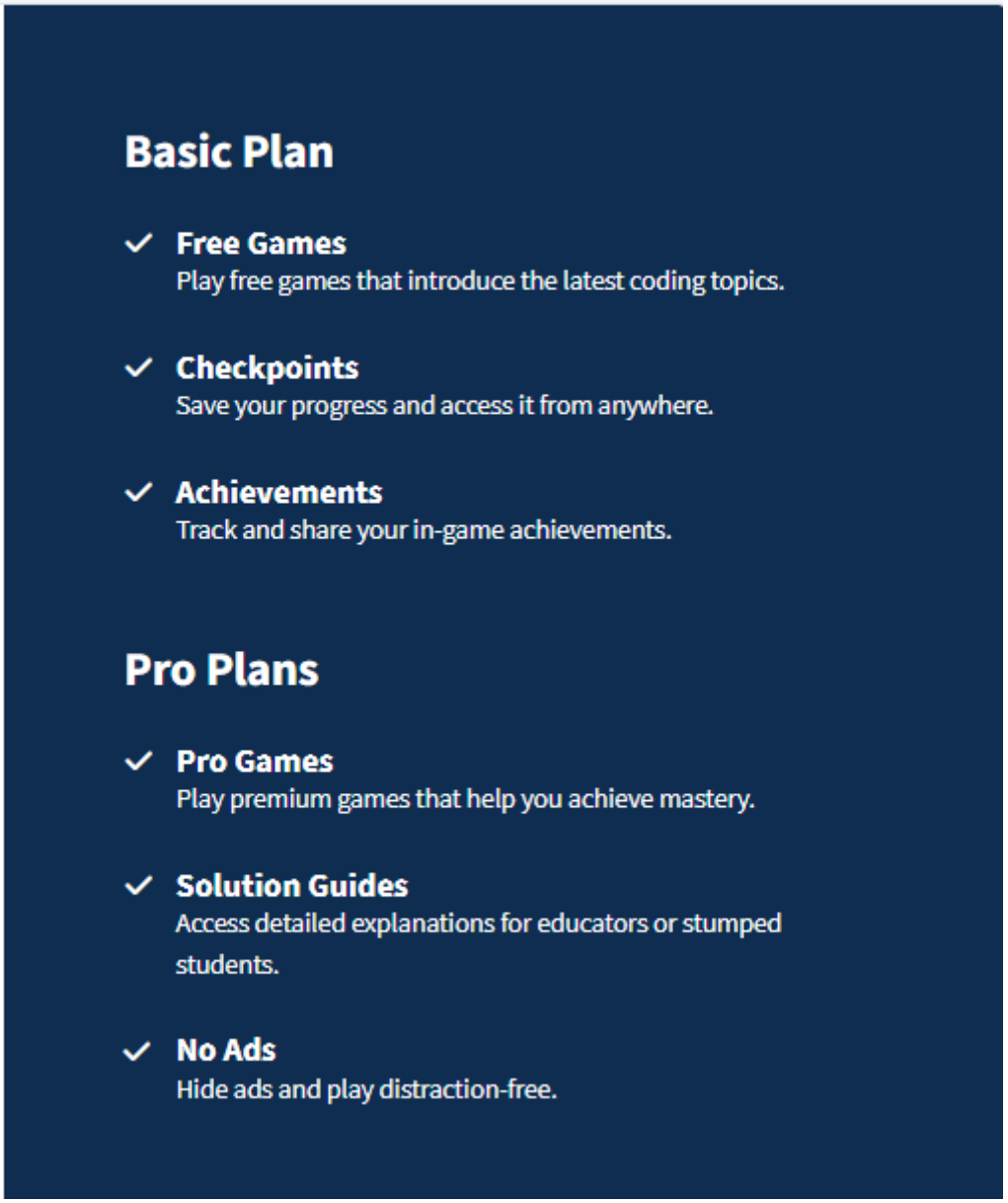

Рисунок 1.4 – Суттєва перевага Pro плану над Basic [5]

Ціллю Codepip є забезпечення студентам гнучкого та ефективного інструменту для навчання веб-розробки. Це означає створення платформи, що дозволяє кожному учневі розвивати свої навички у веб-розробці на власних умовах, враховуючи їхні індивідуальні потреби та рівень підготовки. Незважаючи на те, що деякий контент на ресурсі є платним, важливо розуміти, що це можуть бути виправдані витрати, якщо ресурс надає якісний та глибокий контент для розвитку навичок у веб-розробці.

Крім того, Codepip розуміє, що ефективне навчання веб-розробки потребує не лише доступу до інформації, але й можливості її практичного застосування. Тому платформа акцентує на інтерактивності та практичних завданнях, які допомагають студентам закріпити знання та вміння шляхом активного застосування їх у веб-розробці.

Забезпечуючи різноманітність тем та завдань, Codepip створює можливість для студентів вивчати не лише основи веб-розробки, але й розширювати свої знання у більш специфічних областях, таких як адаптивний дизайн, робота з серверами чи розробка веб-додатків. Це сприяє розвитку комплексного розуміння та навичок у всіх аспектах веб-розробки.

Таким чином, враховуючи цільову орієнтацію на гнучкість, ефективність та якість навчання, платформа Codepip відображає сучасні потреби учнів у вивченні веб-розробки, забезпечуючи їм засоби для успішного розвитку у цій галузі.

CodeCombat

CodeCombat — це освітня платформа, яка навчає програмуванню через гру, використовуючи реальний код для керування персонажами в ігровому світі. Вона орієнтована на різні рівні підготовки, від початківців до просунутих користувачів, і пропонує вивчення кількох мов програмування, таких як JavaScript, Python, та інші.

Переваги:

- Ігровий підхід до навчання:

CodeCombat інтегрує процес навчання програмування в захоплюючий ігровий формат. Учні керують персонажами, використовуючи код для виконання завдань і проходження рівнів. Це не лише підвищує мотивацію та залученість, але й допомагає краще засвоїти матеріал через активне застосування знань [6].

- Підтримка кількох мов програмування:

Платформа пропонує навчання на таких мовах програмування, як Python, JavaScript, і C++. Це дозволяє учням обирати мову, яка найкраще відповідає їхнім потребам і цілям, забезпечуючи гнучкість у навчанні.

- Індивідуалізоване навчання:

CodeCombat дозволяє учням навчатися у власному темпі, пропонуючи адаптивний підхід до навчання. Учні можуть повторювати завдання або переходити до наступних рівнів, що допомагає враховувати їхні індивідуальні потреби та рівень підготовки (Рис. 1.5).

- Ресурси для вчителів:

Платформа надає інструменти для вчителів, які допомагають організувати уроки, відстежувати прогрес учнів та надавати підтримку під час навчання. Це включає детальні звіти про успішність та рекомендації щодо подальшого навчання, що полегшує процес викладання.

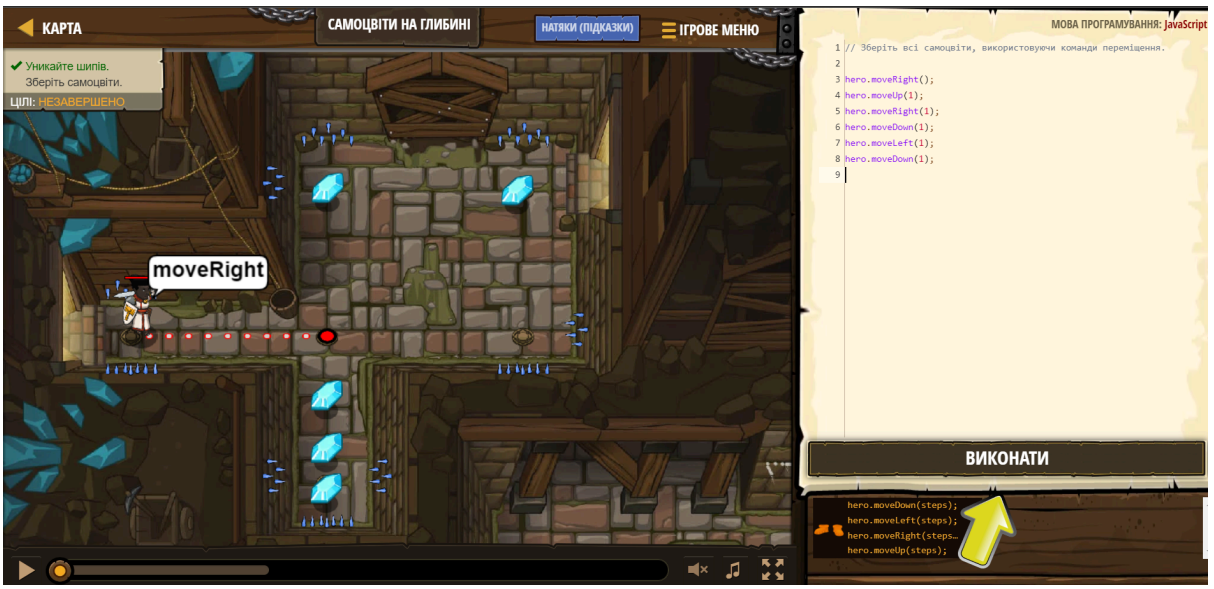

Рисунок 1.5 – Приклад ігроладу CodeCombat [6]

Недоліки:

- Вартість:

Деякі курси та функції доступні лише за передплатою, що може бути обмеженням для деяких користувачів. Це може створювати бар'єр для тих, хто не може собі дозволити платні опції.

- Обмежений контент:

Платформа зосереджена на базових та середніх рівнях програмування, що може бути недостатньо для просунутих користувачів. Відсутність розширених тем може обмежити можливості для подальшого розвитку та поглибленого вивчення (Рис. 1.6).

Рисунок 1.6 – Обмеженість контенту CodeCombat [6]

- Потреба в доступі до інтернету:

Для використання платформи необхідний постійний доступ до інтернету. Це може бути проблемою для користувачів з обмеженим доступом до мережі або нестабільним з'єднанням.

CodeCombat є потужним інструментом для навчання програмування, який поєднує освіту та розваги, забезпечуючи ефективне та захоплююче навчання. Проте, як і будь-який інший ресурс, він має свої переваги та обмеження, які варто враховувати при виборі платформи для навчання.

Висновки по недоліках Coding Fantasy, Codepip та CodeCombat:

- Вузька направленість та обмежена тематика:

У всіх трьох платформах виявлено вузьку спеціалізацію або обмежену тематику, яка може обмежувати розвиток навичок учасників. Виправлення цього недоліку полягає у розширенні спектру тем та технологій, щоб охопити більше аспектів веб-розробки та програмування.

- Вартість контенту:

Більшість контенту на цих платформах платна, що може створювати бар'єри для користувачів, особливо для тих, хто не може собі дозволити платні опції. Виправлення цього може полягати у наданні безкоштовного базового контенту, щоб дозволити користувачам ознайомитися з платформою та здобути основні навички, а також у пропозиції платного контенту або підписки для глибокого та розширеного вивчення.

В цілому, з урахуванням недоліків Coding Fantasy, Codepip та CodeCombat, важливою є необхідність розширення тематичного охоплення, розширення доступності для користувачів та розробка більш гнучких моделей оплати, щоб зробити ці платформи більш привабливими та доступними для широкого кола користувачів.

Запропоноване рішення

- Інтеграція широкого спектру тем:

Рішення: Розробити гру, яка охоплює різні аспекти веб-розробки, включаючи HTML, CSS, JavaScript, адаптивний дизайн, взаємодію з сервером тощо. Забезпечити глибокий інструкційний контент для кожної теми.

- Безкоштовний базовий контент:

Рішення: Надати безкоштовні рівні гри та базовий набір завдань для введення гравців у вивчення веб-розробки. Платний контент може бути доступний для розширення можливостей та отримання додаткового матеріалу.

- Ігровий процес та педагогічний аспект:

Рішення: Забезпечити ігровий процес, який тісно взаємодіє з веб-розробкою та подає завдання у формі гри. Додати педагогічні елементи, які сприяють ефективному вивченню та розвитку навичок.

- Інтерактивний теоретичний матеріал:

Рішення: Включити інтерактивний теоретичний матеріал, що допомагає користувачам засвоювати теорію шляхом активної участі та практики.

Головною метою є розробка гри, яка не лише розважатиме, але й сприятиме якісному вивченню веб-розробки через інтерактивний та цікавий підхід.

Висновки:

Обидва ресурси, Coding Fantasy та Codepip, виявилися цікавими інструментами для вивчення програмування через гейміфікацію. У Coding Fantasy ігровий процес тісно взаємодіє з кодом, створюючи активне середовище для вивчення, а також надається коротка і зрозуміла теорія, що разом з наочною практикою робить його ефективним для розуміння концепцій [4, 5].

Codepip також пропонує взаємодію з кодом через геймплей та відзначається різноманітністю тем та наочною практикою. Проте, його обмеженням є те, що більшість контенту є платним, що може вплинути на доступність для користувачів.

У той час як CodingCombat також використовує гейміфікацію для навчання програмування, забезпечуючи інтеграцію з кодом через ігровий процес. Він також має широкий спектр тем та надає різні рівні складності для користувачів на різних етапах навчання. Проте, недоліком є те, що деякі курси та функції доступні лише за передплатою, що може створювати бар'єри для користувачів [6].

Отже, всі три ресурси мають свої переваги та недоліки. Вибір між ними може залежати від індивідуальних потреб та уподобань користувача, а також доступності ресурсу та його відповідності конкретним цілям навчання.

1.3 Постановка задачі:

Розробити карткову гру для вивчення HTML/CSS.

Цілі:

- Розробка інтерактивного інтерфейсу:

Створення інтерфейсу, що надає користувачам можливість взаємодії з грою та вивчення основ HTML/CSS.

- Підвищення мотивації користувачів:

Впровадження механік або елементів, які підвищать зацікавленість та мотивацію користувачів у вивченні технологій.

- Адаптивність та сумісність:

Забезпечення адаптивності гри до різних розмірів екранів та сумісності з різними веб-браузерами.

Підзадачі:

- Генерація графіки за допомогою штучного інтелекту (ШІ):

Використання ШІ для створення графічних елементів гри, наприклад, ілюстрацій та фонів.

- Реалізація інтерфейсу:

Створення інтерактивного інтерфейсу з елементами взаємодії, що допоможе вивчати HTML/CSS.

- Створення всіх ігрових об'єктів:

Розробка 2D моделей та текстур для карток, та інших об'єктів гри.

- Написання логіки для гри:

Реалізація головних взаємодій користувача та гри, включаючи логіку перемоги та поразки.

Цей проект спрямований на створення ефективної та захоплюючої карткової гри, яка допоможе користувачам вивчати основи HTML/CSS, забезпечуючи при цьому стимулюючий інтерфейс та навчальний досвід.

Висновки:

Гейміфікація в навчанні дозволяє зробити процес вивчення більш привабливим та заохочує студентів до активної участі та досягнення навчальних цілей [2, 3].

Coding Fantasy, Codepip та Codecombat - всі ці ресурси пропонують інноваційний підхід до навчання програмування через гейміфікацію. Вони допомагають студентам отримати практичні навички, використовуючи ігровий формат, що створює захопливе навчальне середовище [4, 5, 6].

Позитивні аспекти: Заохочення самодисципліни, покращення вивчення матеріалу та підвищення мотивації – це основні позитивні аспекти використання гейміфікації в освіті та навчанні програмування.

Ресурси, які пропонують різноманітні та захоплюючі завдання, можуть бути більш привабливими для широкої аудиторії та сприяти глибшому розумінню предмету.

РОЗДІЛ 2. АНАЛІЗ ЗАСОБІВ РОЗРОБКИ

2.1 Аналіз платформ для розробки

Розробка гри за допомогою веб-технологій (HTML, CSS та JS)

Розробка ігор на HTML/CSS та JavaScript (JS) включає створення веб-ігор, які працюють у браузері без необхідності встановлювати додаткові плагіни. HTML і CSS використовуються для структурування та стилізації контенту, тоді як JavaScript забезпечує інтерактивність та логіку гри [7].

Переваги:

- Висока доступність:

Ігри, створені з використанням веб-технологій, легко доступні через будь-який веб-браузер, що робить їх надзвичайно зручними для користувачів. Це означає, що гравці можуть запускати такі ігри на різних пристроях, включаючи комп’ютери, планшети та смартфони, без необхідності завантажувати або встановлювати додаткове програмне забезпечення [8].

- Легкість розповсюдження:

Веб-ігри можна легко поширювати через інтернет, надаючи гравцям миттєвий доступ до них за допомогою простого посилання. Це значно спрощує процес обміну та залучення нових гравців, оскільки вони можуть почати грати безпосередньо у браузері [7].

- Простий доступ для розробників:

Розробка ігор за допомогою HTML, CSS та JavaScript є досить простою і не вимагає спеціалізованих середовищ розробки. Веб-розробники можуть використовувати свої знання та навички для створення інтерактивних ігор, використовуючи звичайні текстові редактори та стандартні веб-інструменти [9].

- Можливість використання звичайних інструментів:

Для розробки веб-ігор можна використовувати звичайні текстові редактори (наприклад, Visual Studio Code, Sublime Text) та веб-інструменти, такі як браузерні консолі для налагодження та тестування коду. Це дозволяє розробникам швидко розпочати роботу, використовуючи знайомі інструменти.

- Кросплатформність:

Веб-технології забезпечують кросплатформну сумісність, дозволяючи одному й тому ж коду працювати на різних операційних системах та пристроях. Це забезпечує ширше охоплення аудиторії, оскільки гравці можуть використовувати будь-який пристрій з веб-браузером для доступу до гри.

- Широкий вибір бібліотек та фреймворків:

Існує безліч бібліотек та фреймворків, таких як Phaser, Three.js та Babylon.js, які значно полегшують процес розробки ігор. Вони надають готові рішення для роботи з графікою, анімаціями, фізикою та іншими важливими аспектами ігрового процесу, що допомагає швидше створювати якісні ігри.

Недоліки:

- Обмежені графічні можливості:

Попри те, що сучасні веб-технології дозволяють створювати досить складні графічні ефекти, вони все ж обмежені у порівнянні зі спеціалізованими ігровими рушіями, такими як Unity або Unreal Engine. Це може вплинути на якість візуальних ефектів та деталізацію ігрових світів.

- Відсутність доступу до певних ресурсів:

Веб-середовище обмежує доступ до певних системних ресурсів, таких як файлові системи та деякі апаратні можливості. Це може ускладнити реалізацію деяких функцій, таких як збереження прогресу гри на локальних пристроях або взаємодія з апаратними пристроями.

- Обмеження продуктивності:

Веб-ігри можуть стикатися з обмеженнями продуктивності через обмеження веб-браузерів та апаратного забезпечення. Це може призвести до проблем з продуктивністю на старіших або менш потужних пристроях, що може вплинути на якість ігрового досвіду.

- Безпека та конфіденційність:

Оскільки веб-ігри працюють через інтернет, вони можуть бути вразливими до різних видів атак та проблем з безпекою. Це вимагає додаткових заходів безпеки для захисту даних користувачів та забезпечення безпечної роботи гри.

Розробка гри на ігровому рушії (Unity або Unreal Engine)

Ігровий рушій – це програмне забезпечення, яке надає розробникам інструменти та середовище для створення відеоігор. Основна мета ігрового рушія – спрощення та прискорення процесу розробки ігор, дозволяючи розробникам зосередитися на креативних аспектах, а не на технічних деталях. Ігровий рушій зазвичай включає в себе набір бібліотек, фреймворків, інструментів та середовище розробки, що допомагає створювати, тестувати та налагоджувати ігрові проекти.

Популярні ігрові рушії:

Unity: Популярний ігровий рушій, який підтримує розробку як 2D, так і 3D ігор, відомий своєю зручністю та широкою підтримкою платформ [10].

Unreal Engine: Високоякісний рушій, що надає потужні інструменти для створення реалістичних графічних ефектів та складних ігрових світів, часто використовуваний для AAA-ігор [11].

Переваги:

- Графічні можливості:

Один із головних аргументів на користь використання Unity та Unreal Engine - це їхні вражаючі графічні можливості. Обидва рушії надають розробникам можливість створювати візуально захопливі ігрові світи з деталізованими та реалістичними об'єктами та ефектами.

- Зручний інтерфейс розробника:

Інтуїтивний інтерфейс є ще однією перевагою Unity та Unreal Engine. Він спрощує розробку та налаштування ігрових об'єктів та логіки, дозволяючи розробникам швидше і ефективніше втілювати свої ідеї.

- Мови програмування:

Використання різних мов програмування, таких як C# в Unity або C++ в Unreal Engine, відкриває безліч можливостей для розробників. Це дозволяє їм створювати складну логіку гри та реалізовувати різноманітні інтеракції між об'єктами [12].

- Кросплатформність:

Ще одна вагома перевага - це кросплатформеність рушіїв. Розробка гри на Unity або Unreal Engine дозволяє охоплювати різні платформи, включаючи персональні комп'ютери, ігрові консолі та мобільні пристрої, що розширює аудиторію та досяжність гри.

- Широкий спектр готових рішень:

Unity та Unreal Engine мають велику кількість готових рішень та модулів, які значно спрощують процес розробки. Вони включають в себе готові компоненти для реалізації різних геймплейних механік, графічних ефектів, аудіофункцій та багато іншого [13].

- Активна спільнота та підтримка:

Обидва ігрові рушії мають великі та активні спільноти розробників, які надають підтримку, допомогу та різноманітні ресурси для вирішення проблем та вдосконалення навичок.

- Можливості розширення:

Unity та Unreal Engine дозволяють розробникам створювати власні розширення та плагіни для власних потреб або для реалізації специфічних функцій, яких може бракувати в стандартному функціоналі.

Недоліки:

- Вивчення кривої навчання:

Для початківців вивчення Unity або Unreal Engine може виявитися складним завданням через обширний функціонал та можливості цих ігрових рушіїв. Брак структурованого навчального процесу може призвести до втрати часу та плутанини у деталях [11, 13].

- Великі розміри проектів:

Ще одним недоліком є те, що проекти, розроблені на Unity або Unreal Engine, можуть вимагати значної кількості місця на диску через обсяг ресурсів та файлів. Це може ускладнити роботу з проектом та вимагати додаткових зусиль для зберігання та обробки даних.

- Високі вимоги до апаратного забезпечення:

Робота з Unity та Unreal Engine може вимагати потужного комп'ютера з високопродуктивною графічною карткою, що може бути недосяжним для деяких користувачів або компаній з обмеженим бюджетом.

- Висока складність:

Незважаючи на інтуїтивний інтерфейс, процес розробки на Unity та Unreal Engine може бути складним через велику кількість налаштувань, опцій та можливостей, що може призвести до плутанини та затримок у розробці.

- Великий обсяг коду:

Зазвичай проекти на Unity та Unreal Engine мають великий обсяг коду, що може бути складним для розуміння та підтримки, особливо для початківців або невеликих команд розробників.

Розробка гри на рушіях Unity або Unreal Engine пропонує численні переваги, зокрема високоякісні графічні можливості, зручний інтерфейс розробника, використання потужних мов програмування (C# для Unity і C++ для Unreal Engine), а також кросплатформеність, що дозволяє створювати ігри для різних пристроїв. Додаткові переваги включають широкий спектр готових рішень, активну спільноту розробників та можливості для створення власних розширень.

Однак, ці інструменти мають і свої недоліки, такі як складна крива навчання для новачків, високі вимоги до апаратного забезпечення, великі розміри проектів та високий обсяг коду, що може ускладнити розробку та підтримку проектів.

Загалом, Unity та Unreal Engine є потужними інструментами для розробки ігор, які, попри свої складнощі, можуть значно полегшити процес створення якісних та інтерактивних ігрових продуктів, якщо використовувати їх з урахуванням всіх можливостей і обмежень.

Висновки:

Обираючи між ігровим рушієм та веб-технологіями для розробки гри, варто враховувати конкретні потреби та характер проекту. Ігрові рушії надають більше можливостей для створення великих та складних ігор з високоякісною графікою та складною логікою. У той час як розробка веб-гри може бути зручною для простих ігор, які доступні через браузер. Однак у контексті зручності та ефективності розробки, ігрові рушії зазвичай виграють завдяки своїм розширеним можливостям та готовим інструментам.

2.2 Вибір рушія

Ігровий рушій – це спеціалізоване програмне забезпечення, яке надає розробникам інструменти для створення та управління відеоіграми. Рушії об'єднують різноманітні технології, такі як графічний рендеринг, фізичні симуляції, звуковий дизайн, анімації та інтерфейс користувача. Це дозволяє розробникам зосередитися на творчих аспектах гри, не витрачаючи час на розробку базових технічних компонентів з нуля.

Основні компоненти ігрових рушіїв включають графічний рушій, який забезпечує відображення візуальних елементів, фізичний рушій, що моделює реалістичні фізичні явища, та звуковий рушій, який керує відтворенням звукових ефектів і музики. Додатково, рушії надають засоби для анімації персонажів і об'єктів, створення штучного інтелекту для NPC, а також розробки інтерфейсу користувача. Вбудовані скриптові мови дозволяють швидко писати ігрову логіку та сценарії, що полегшує налаштування і управління грою.

Використання ігрових рушіїв значно прискорює процес розробки ігор, знижує витрати та дозволяє випускати ігри на різні платформи, такі як ПК, консолі, мобільні пристрої та веб. Популярні рушії мають великі спільноти, що діляться знаннями та ресурсами, полегшуючи навчання та вирішення проблем. Завдяки постійній підтримці та оновленням, розробники отримують доступ до нових функцій і покращень, що робить ігрові рушії незамінними в сучасній індустрії відеоігор.

Порівняння Unity, Unreal Engine та GameMaker

Unity

Переваги:

- Низький поріг входу:

Unity є дуже доступним для новачків завдяки своєму інтуїтивно зрозумілому інтерфейсу та великій кількості навчальних ресурсів. Різноманітні туторіали, офіційна документація та онлайн-курси дозволяють швидко освоїти базові концепції та інструменти рушія, що робить його ідеальним для тих, хто тільки починає свою кар'єру в розробці ігор.

- Мова програмування C#:

Використання C# як основної мови програмування спрощує процес розробки гри. C# має дружній синтаксис і багатий набір функцій, що дозволяє легко реалізовувати ігрову логіку. Крім того, C# добре інтегрується з HTML/CSS, що полегшує створення веб-інтерфейсів та елементів UI у грі [12, 14].

- Активна спільнота та Asset Store:

Unity має велику та активну спільноту розробників, що забезпечує швидкий обмін знаннями та допомогу у вирішенні проблем. Крім того, Asset Store пропонує широкий асортимент готових рішень, моделей, текстур, скриптів та інших ресурсів, що значно прискорює процес розробки.

- Платформна незалежність:

Unity дозволяє розробляти ігри для різних платформ, включаючи ПК, мобільні пристрої, консолі та веб. Це дає розробникам можливість охопити широку аудиторію та адаптувати свої ігри під різні девайси.

Недоліки:

- Великі розміри проектів:

Проекти в Unity можуть мати значний обсяг через велику кількість ресурсів, що впливає на час завантаження та загальний розмір файлів гри. Це може створювати незручності при розповсюдженні та оновленні ігор.

- Оптимізація для великої кількості платформ:

Розробка гри для різних платформ може вимагати додаткової роботи з оптимізації. Кожна платформа має свої особливості, що потребують врахування під час розробки, що може збільшити загальний час і зусилля, витрачені на проект [10, 13].

Unreal Engine

Переваги:

- Графічні можливості:

Unreal Engine славиться своїми потужними графічними можливостями, які дозволяють створювати реалістичні ігрові світи з високою деталізацією. Завдяки передовим технологіям рендерингу, Unreal Engine забезпечує високоякісну графіку, яка виглядає вражаюче на всіх платформах [11].

- Графічний інтерфейс Blueprint:

Використання Blueprint, графічного інтерфейсу для створення складних систем логіки без програмування, дозволяє розробникам швидко створювати прототипи та реалізовувати ігрові механіки. Це особливо корисно для тих, хто не має глибоких знань у програмуванні, але хоче реалізувати складні ідеї.

- Безкоштовні інструменти та матеріали:

Unreal Engine надає широкий вибір безкоштовних інструментів та матеріалів, включаючи текстури, моделі, анімації та інші ресурси, які можна використовувати у своїх проектах. Це дозволяє значно зекономити час та зусилля на створення базових елементів гри.

Недоліки:

- Складне навчання:

Освоєння Unreal Engine може бути складним для новачків, особливо для тих, хто тільки починає вивчати розробку ігор. Велика кількість функцій та інструментів може здаватися переважною, що потребує значного часу та зусиль для їх вивчення.

- Великі розміри проектів:

Проекти в Unreal Engine можуть потребувати значного місця на диску через велику кількість ресурсів та високоякісну графіку. Це може ускладнити розповсюдження гри та збільшити час завантаження [11].

- Вимогливість до обладнання:

Unreal Engine може бути досить вимогливим до апаратних ресурсів комп'ютера. Це означає, що для комфортної роботи з рушієм розробникам може знадобитися потужне обладнання, що не завжди доступно новачкам.

GameMaker

Переваги:

- Простий вивчений концепт:

GameMaker є ідеальним для новачків, оскільки дозволяє швидко оволодіти основами розробки ігор. Інтуїтивно зрозумілий інтерфейс та прості інструменти допомагають швидко створювати ігри без необхідності вивчення складних програмних концепцій [15].

- Вбудована мова програмування GML:

Власна мова програмування GameMaker Language (GML) є легкою для вивчення та використання, що дозволяє розробникам швидко створювати ігри без глибоких знань програмування. GML забезпечує гнучкість та простоту у реалізації ігрових механік.

- Швидкість розробки:

GameMaker дозволяє швидко розробляти прості ігри завдяки своїм зручним інструментам та середовищу розробки. Це робить його чудовим вибором для створення прототипів та невеликих проектів, де важлива швидкість реалізації.

Недоліки:

- Обмежені можливості:

GameMaker може бути обмеженим у можливостях порівняно з більш потужними рушіями, такими як Unity або Unreal Engine. Це може стати перешкодою для розробників, які прагнуть створювати складніші та більш деталізовані ігри.

- Обмежені засоби для графіки:

Основні засоби для роботи з графікою в GameMaker є обмеженими, що може вплинути на якість візуальних ефектів та загальну презентацію гри. Розробники, які прагнуть створити високоякісну графіку, можуть зіткнутися з певними обмеженнями.

- Менша активність спільноти:

Спільнота розробників GameMaker є меншою та менш активною порівняно з Unity та Unreal Engine. Це може ускладнити пошук допомоги та ресурсів, необхідних для вирішення технічних питань та проблем під час розробки.

В контексті створення карткової гри для новачка, вибір Unity привабливим варіантом. Причини цього вибору пов’язані не лише з технічними можливостями цього рушія, але й з особливостями процесу навчання та розробки гри. Unity відзначається легкістю у використанні, що є критично важливим аспектом для початківців. Інтерфейс Unity інтуїтивний, а процес створення гри організований у логічному порядку, що дозволяє новачкам швидко освоїти основи розробки та продовжити свій творчий процес без значних перешкод [10, 13].

Однією з ключових переваг Unity є використання мови програмування C#. Для початківців це означає роботу з мовою, яка має дружній до користувача синтаксис та потужний функціонал, що дозволяє легко реалізовувати свої ідеї без необхідності глибокого занурення у складні деталі програмування. Це значно полегшує процес навчання і дозволяє зосередитися на творчих аспектах розробки гри [12, 14].

Unity також має велику та активну спільноту розробників, що є надзвичайно корисним для новачків. Наявність обширної документації, численних навчальних матеріалів та активність на форумах забезпечують швидке вирішення будь-яких питань чи проблем, що можуть виникнути під час розробки гри. Це створює сприятливе середовище для навчання та розвитку, дозволяючи новачкам отримувати підтримку та поради від більш досвідчених розробників.

У випадку розробки карткових ігор, де важливі елементи графіки та дизайну, Unity забезпечує велику гнучкість та зручність в роботі з 2D графікою. Це дозволяє розробникам швидко візуалізувати свої ідеї та легко реалізовувати їх у вигляді гри. Інструменти Unity для роботи з графікою надають широкі можливості для створення якісних візуальних елементів, що робить процес розробки більш приємним та продуктивним.

Крім того, Unity надає можливість легко розгортати гру на різних платформах, що є важливим для широкого кола користувачів. Це означає, що ваша карткова гра може бути доступною як на ПК, так і на мобільних платформах, забезпечуючи більше можливостей для розповсюдження та взаємодії з аудиторією. Така кросплатформенність дозволяє досягти більшої аудиторії та робить гру доступною для користувачів з різними пристроями, що є важливою перевагою у сучасному ігровому середовищі.

Висновки:

При аналізі всіх факторів, Unity виявляється оптимальним вибором для створення гейміфікованої системи підтримки вивчення технологій HTML/CSS. Перш за все, низький поріг входу робить Unity дуже доступним для початківців, дозволяючи швидко освоїти основи розробки. Крім того, використання мови програмування C# спрощує розробку та взаємодію з HTML/CSS. Активна спільнота та наявність Asset Store забезпечують доступ до безлічі ресурсів та готових рішень для полегшення

процесу розробки. Платформна незалежність Unity дає можливість розробляти гри для різних платформ, включаючи веб, забезпечуючи гнучкість у виборі дистрибуції. Всі ці переваги роблять Unity ідеальним інструментом для створення гейміфікованих систем підтримки навчання технологій HTML/CSS, спрямованих на широку аудиторію, зокрема на початківців у програмуванні та веб-розробці.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ГЕЙМІФІКОВАНОЇ СИСТЕМИ

3.1 Розробка та реалізація проекту

У розділі проведено детальний аналіз практичних аспектів створення гейміфікованої системи підтримки вивчення HTML/CSS з використанням рушія Unity. Звертаючись до теоретичних засад гейміфікації та їхнього застосування у сфері освіти, розглянуті особливості інтеграції веб-технологій у гейміфікований контекст. Основна увага приділена методам розробки інтерактивних навчальних ігор, враховуючи адаптацію HTML та CSS для ігрового середовища, а також створення ефективного інтерфейсу та завдань, спрямованих на засвоєння відповідного матеріалу.

У контексті використання Unity для розробки гейміфікованої системи, ретельно проаналізовано конкретні компоненти та елементи, які сприяють активному навчанню та успішному засвоєнню HTML/CSS. Окрема увага приділена методам мотивації студентів та педагогічно обґрунтованим стратегіям навчання, спрямованим на забезпечення ефективного процесу освоєння навчального матеріалу.

В результаті аналізу визначені ключові аспекти успішної гейміфікованої системи навчання HTML/CSS, а також розроблені та втілені в практику відповідні практичні компоненти проекту. Це охоплює створення структури гри, розробку навчальних завдань та інтерактивних вправ, а також оптимальне проектування інтерфейсу з метою досягнення найвищої ефективності навчання. Такий науковий підхід дозволяє практично реалізувати концепції гейміфікації у навчальному процесі та досягти успішного вивчення матеріалу з HTML/CSS.

Створення зображень за допомогою штучного інтелекту

- Створення фону:

Використання штучного інтелекту для створення графічного фону (Рис. 3.1), за допомогою геометричних елементів та колірних схем, з метою створення естетичного візуального враження. [16].

Рисунок 3.1 – Фон гри

- Створення карти та столу:

Використання алгоритмів для автоматичного створення ігрового столу (Рис. 3.2) і карт (Рис. 3.3) з урахуванням дизайну та естетики, з метою створення привабливого вигляду ігрового середовища.

Рисунок 3.2 – Стіл для карт

Стіл, який буде використовуватись для розміщення карт та колоди на ньому.

Рисунок 3.3 – Ігрові картки горілиць та долілиць відповідно

Додавання зображень як об'єктів

- Створення об'єкту GameManager:

Створення пустого об'єкта "GameManager" для керування основною логікою гри, зокрема управління картами та іншими елементами.

- Створення об'єктів cardSlots, deck:

Створення об'єктів "cardSlots" та "deck" для відображення та організації місць для карт та колоди.

- Реалізація логіки та управління

Написання коду на мові програмування C#:

Реалізація логіки гри та взаємодії об'єктів за допомогою мови програмування C# в середовищі Unity (додаток А).

Опис скрипту, що реалізує механізм проходження рівня (Рис. 3.4).

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class ButtonController : MonoBehaviour
{
    public GameObject hiddenObject;
    public Button[] buttons;
    public Button[] incorrectButtons;


    private int activeButtonCount = 0;
    private List<Button> clickedIncorrectButtons = new List<Button>();
```

Рисунок 3.4 – Оголошення змінних та компонентів

У цьому блоці оголошуються основні змінні класу:

- `hiddenObject`: об'єкт, який спочатку прихований і стане видимим після натискання певної кількості правильних кнопок.
- `buttons`: масив правильних кнопок.
- `incorrectButtons`: масив неправильних кнопок.
- `activeButtonCount`: лічильник активних (натиснутих) правильних кнопок.
- `clickedIncorrectButtons`: список неправильних кнопок, які були натиснуті.

Метод `Start` викликається один раз на початку виконання скрипту (Рис. 3.5).

```
void Start()
{
    hiddenObject.SetActive(false);

    foreach (Button button in buttons)
    {
        button.onClick.AddListener(() => ToggleButton(button));
    }

    foreach (Button incorrectButton in incorrectButtons)
    {
        incorrectButton.onClick.AddListener(() => ToggleIncorrectButton(incorrectButton));
    }


}
```

Рисунок 3.5 – Метод `Start`

1. `hiddenObject.SetActive(false)`: приховує об'єкт при запуску.
2. Для кожної кнопки з масиву `buttons` додається обробник подій, який викликає метод `ToggleButton` при натисканні на кнопку.
3. Для кожної кнопки з масиву `incorrectButtons` додається обробник подій, який викликає метод `ToggleIncorrectButton` при натисканні на кнопку.

Метод `ToggleButton` змінює стан активності кнопки (Рис. 3.6).

```
void ToggleButton(Button button)
{
    button.interactable = !button.interactable;


    if (!button.interactable)
    {
        activeButtonCount++;
    }
    else
    {
        activeButtonCount--;
    }

    CheckHiddenObjectVisibility();
```

Рисунок 3.6 – Метод `ToggleButton`

1. `button.interactable = !button.interactable`: змінює стан кнопки з активного на неактивний або навпаки.
2. Якщо кнопка стає неактивною, збільшується `activeButtonCount`.
3. Якщо кнопка знову стає активною, зменшується `activeButtonCount`.
4. Викликається метод `CheckHiddenObjectVisibility`, щоб перевірити, чи потрібно зробити прихований об'єкт видимим.

Метод `ToggleIncorrectButton` обробляє натискання неправильних кнопок (Рис. 3.7).

```
void ToggleIncorrectButton(Button button)
{

    if (!clickedIncorrectButtons.Contains(button))
    {
        clickedIncorrectButtons.Add(button);
        Debug.Log("Цей варіант не правильний!");
    }
}
```

Рисунок 3.7 – Метод `ToggleIncorrectButton`

1. Якщо кнопка ще не була натиснута (`!clickedIncorrectButtons.Contains(button)`), вона додається до списку `clickedIncorrectButtons`.
2. Виводиться повідомлення у консоль `Debug.Log("Цей варіант не правильний!");`.

Метод `CheckHiddenObjectVisibility` перевіряє кількість активних кнопок і вирішує, чи потрібно зробити прихований об'єкт видимим (Рис. 3.8).

```
    void CheckHiddenObjectVisibility()
    {

        if (activeButtonCount == 5)
        {
            hiddenObject.SetActive(true);
        }
        else
        {
            hiddenObject.SetActive(false);
        }
    }

}
```

Рисунок 3.8 – Метод `CheckHiddenObjectVisibility`

1. Якщо кількість активних кнопок дорівнює 5 (`activeButtonCount == 5`), об'єкт `hiddenObject` стає видимим (`hiddenObject.SetActive(true)`).
2. В іншому випадку об'єкт залишається прихованим (`hiddenObject.SetActive(false)`).

Цей код забезпечує механіку гри, де користувач повинен натиснути певну кількість правильних кнопок, щоб отримати правильний результат, і отримують зворотній зв'язок при натисканні неправильних кнопок.

Також реалізовано скрипт переходу між сценами. Лістинг коду знаходиться у додатку Б.

Висновки:

Цей розділ висвітлює всі етапи розробки гри в середовищі Unity, починаючи від створення зображень та об'єктів, закінчуючи скриптами для взаємодії об’єктів.

3.2 Тестування проекту

Для проходження рівня потрібно повторити елемент в лівому верхньому куті за допомогою комбінацій карт(Рис. 3.9).

Рисунок 3.9 – Вигляд поля під час ігрового процесу

Для початку потрібно створити сам блок, тобто div. Для цього натискаємо на картку з створенням цього елементу, зараз це перша карта в списку ( )(Рис. 3.10).

Рисунок 3.10 – Створений блоковий елемент div на панелі гри

Далі по тій самій логіці вибираємо потрібні карти для правильного виконання завдання (Рис. 3.11).

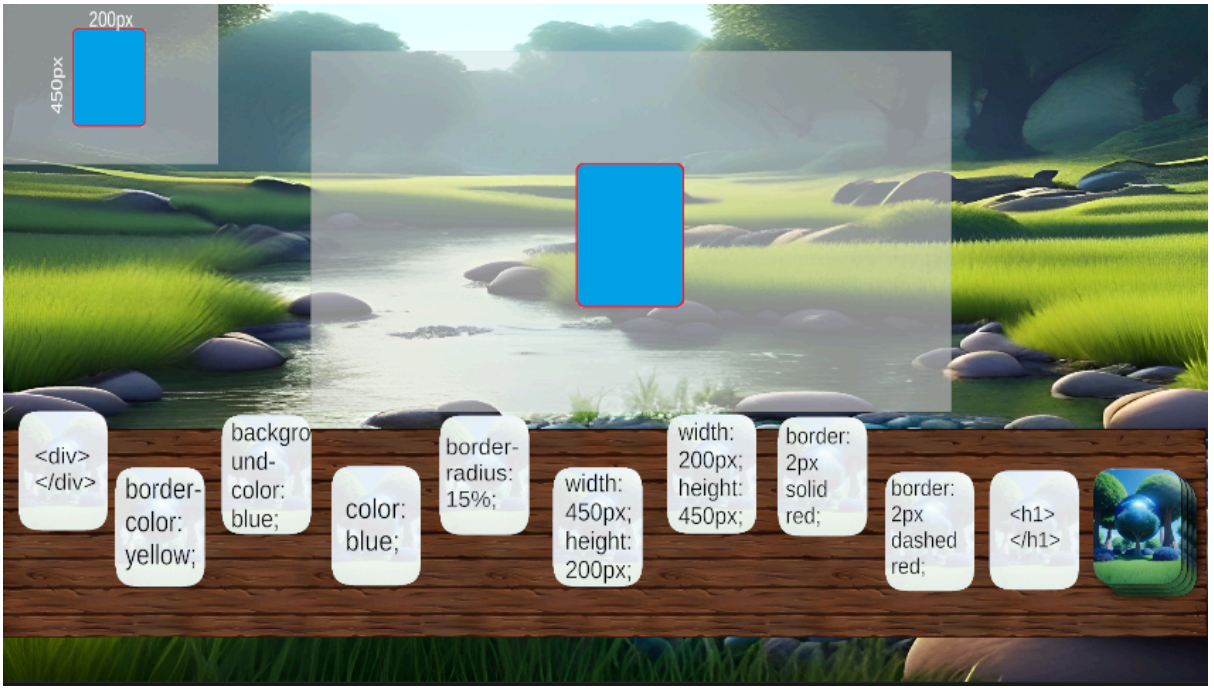

Рисунок 3.11 – Приклад виконання завдання

Підняті карти означають, що вони зараз задіяні для стилізації блокового елементу. Об’єкт повністю ідентичний до елементу, який вказаний у завданні. Це означає, що завдання виконане успішно, і можна переходити до наступного. Програма працює успішно.

ВИСНОВКИ

У процесі розробки та тестування карткової гри для вивчення HTML та CSS були досягнуті певні успіхи, що дозволяють зробити кілька важливих висновків щодо ефективності гейміфікації у навчанні веб-технологій.

Використання гейміфікації у навчанні HTML та CSS виявилося ефективним методом залучення та мотивації користувачів. Гра створює контекст для вивчення, що підвищує інтерес та стимулює активність учасників. Гейміфіковані елементи, такі як нагороди, рівні складності та змагальні аспекти, сприяли підвищенню залученості та утриманню уваги користувачів протягом всього навчального процесу. Це свідчить про те, що інтеграція ігрових механік у навчальні програми може значно покращити мотивацію та зацікавленість студентів.

Розроблена гра надає користувачам інтерактивне навчальне середовище, де вони можуть застосовувати свої знання HTML та CSS у форматі гри. Такий підхід дозволяє користувачам не лише отримувати теоретичні знання, але й використовувати їх на практиці через виконання різноманітних завдань та вправ у грі. Інтерактивність та практична спрямованість гри сприяють більш глибокому засвоєнню навчального матеріалу та розвитку необхідних навичок.

Загальний процес розробки та тестування карткової гри для вивчення HTML та CSS продемонстрував, що гейміфікація може бути ефективним інструментом у навчанні програмування. Створення залучаючого та інтерактивного способу освоєння технологій дозволяє студентам більш ефективно засвоювати матеріал та застосовувати отримані знання на практиці. Гейміфікація допомагає перетворити навчання на захоплюючий процес, що підвищує ефективність освітніх програм у галузі програмування.

Таким чином, результати даної роботи підтверджують доцільність використання гейміфікації у навчанні HTML та CSS. Інтерактивний ігровий підхід не тільки сприяє підвищенню мотивації та зацікавленості студентів, але й забезпечує більш ефективне та практично орієнтоване засвоєння навчального матеріалу. Це відкриває перспективи для подальших досліджень та розробок у галузі гейміфікованого навчання, спрямованого на покращення якості освіти у сфері веб-технологій та програмування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Lehinovych A., Soroka R., Nevmerzhytskyi V., Semianyk M.-I., Izmailov A. Gamified Web Assistant for Support of Learning Process in Secondary School – Zgamifikowany Asystent Webowy do Wspierania Procesu Uczenia w Szkole Średniej. Zeszyty Studenckiego Towarzystwa Naukowego AGH. Wydawnictwo Studenckiego Towarzystwa Naukowego AGH – ISSN 1732-0925; 2024. In print.

[2]Yu-kai Chou ,Actionable Gamification: Beyond Points, Badges, and Leaderboards, Kindle Edition, 2015

[3] “hmhco” [електронний ресурс] - https://www.hmhco.com/blog/what-is-gamification-in-education . [Дата звернення: 29.04.2024]

[4] “Coding Fantasy” [електронний ресурс] - https://codingfantasy.com/ . [Дата звернення: 11.04.2024]

[5] “Codepip” [електронний ресурс] - https://codepip.com/ . [Дата звернення: 15.04.2024]

[6] “CodeCombat” [електронний ресурс] - https://codecombat.com/ . [Дата звернення: 20.04.2024]

[7]Graeme Stewart, Introducing JavaScript Game Development : Build a 2D Game from the Ground Up, 2017

[8] Karl Bunyan, Build an HTML5 Game, 2015.

[9] “The Modern JavaScript Tutorial” [електронний ресурс] - https://javascript.info/ . [Дата звернення: 16.04.2024]

[10] Joseph Hocking, Unity in Action, Second Edition, 2018.

[11] “Unreal Engine documentation” [електронний ресурс] - https://docs.unrealengine.com/5.3/en-US/understanding-the-basics-of-unreal-engine/ . [Дата звернення: 18.04.2024]

[12] Wouter van Toll, Learning C# by Programming Games, 2019.

[13] “Unity Documentation” [електронний ресурс] - https://docs.unity.com/ . [Дата звернення: 9.05.2024]

[14] “C# documentation” [електронний ресурс] - https://learn.microsoft.com/en-us/dotnet/csharp/ . [Дата звернення: 19.04.2024]

[15] “GameMaker manual” [електронний ресурс] - https://manual.gamemaker.io/monthly/en/#t=Content.htm . [Дата звернення: 18.04.2024]

[16] “Piscart - Ai art generator” [електронний ресурс] - https://picsart.com/ai-art-generator/ . [Дата звернення: 20.04.2024]

ДОДАТОК А

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class ButtonController2 : MonoBehaviour
{
    public GameObject hiddenObject; // Посилання на об'єкт, який прихований спочатку
    public Button[] buttons; // Масив кнопок
    public Button[] incorrectButtons;


    private int activeButtonCount = 0; // Кількість активних кнопок
    private List<Button> clickedIncorrectButtons = new List<Button>();

    void Start()
    {
         hiddenObject.SetActive(false);
        // Додаємо обробник подій для кожної кнопки
        foreach (Button button in buttons)
        {
            button.onClick.AddListener(() => ToggleButton(button));
        }

         foreach (Button incorrectButton in incorrectButtons)
        {
            incorrectButton.onClick.AddListener(() => ToggleIncorrectButton(incorrectButton));
        }
```

```
    }

    // Функція для тоглінгу стану кнопки
    void ToggleButton(Button button)
    {
        button.interactable = !button.interactable; //
Змінюємо стан активності кнопки


        if (!button.interactable)
        {
            activeButtonCount++;
        }
        else
        {
            activeButtonCount--;
        }

        CheckHiddenObjectVisibility(); // Перевіряємо
видимість об'єкта при кожній зміні стану кнопки
    }

    void ToggleIncorrectButton(Button button)
    {
        // Якщо кнопка ще не була натиснута, додаємо її до
масиву натисканих хибних кнопок
        if (!clickedIncorrectButtons.Contains(button))
        {
            clickedIncorrectButtons.Add(button);
            Debug.Log("Цей варіант не правильний!");
        }
    }

    // Функція для перевірки видимості об'єкта
    void CheckHiddenObjectVisibility()
    {
```

```
        // Якщо всі 5 кнопок активні, робимо об'єкт видимим
        if (activeButtonCount == 7)
        {
            hiddenObject.SetActive(true);
        }
        else
        {
            hiddenObject.SetActive(false);
        }
    }

}
```

ДОДАТОК Б

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;



public class GameManager : MonoBehaviour
{
    // Функція для виходу з гри
    public void QuitGame()
    {
        #if UNITY_EDITOR
            UnityEditor.EditorApplication.isPlaying = false;
        #else
            Application.Quit();
        #endif
    }

    // Функція для зміни сцени за іменем
    public void ChangeScene(string sceneName)
    {
        SceneManager.LoadScene(sceneName);
    }
}
```