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

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

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬО-ПРОФЕСІЙНОГО СТУПЕНЯ ФАХОВИЙ МОЛОДШИЙ БАКАЛАВР

Тема: *«Мобільний тренажер «Logic Gates»*

Група: *3-011* Спеціальність: *123 «Комп'ютерна інженерія»*

Здобувач освіти (підпис) *Герман КОЗЛОВ* (ім'я, ПРІЗВИЩЕ)

Керівник роботи (підпис) *Оксана ОСАДЧА* (ім'я, ПРІЗВИЩЕ)

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

Кривий Ріг 2025 р.

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

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

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

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ОСВІТИ

Козлову Герману Олексійовичу
(прізвище, ім'я, по батькові)

1.Тема роботи *«Мобільний тренажер «Logic Gates»*

Керівник роботи *Осадча Оксана Георгіївна, викладач вищої категорії*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по коледжу від « 04 » 04 2025 року № 51-ст

2. Строк подання здобувачем освіти роботи з 01.03 по 10.06

3. Вихідні дані до роботи *Інтерактивний додаток для вивчення основ цифрової логіки*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити *аналіз предметної області та середовищ розробки; розробка концепції та проєктування мобільного тренажера; програмна реалізація та тестування додатку*

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

Презентація Microsoft PowerPoint

6. Консультанти розділів роботи (проекту)

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

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	*Узгодження технічного завдання з керівником кваліфікаційної роботи*	*04.04.2025-07.04.2025*	*виконано*
2	*Підбір та вивчення науково-технічної літератури за темою кваліфікаційної роботи*	*08.04.2025-14.04.2025*	*виконано*
3	*Аналіз предметної області та середовищ розробки*	*15.04.2025-21.04.2025*	*виконано*
4	*Концепція та проєктування мобільного тренажера «Logic Gates»*	*22.04.2025-28.04.2025*	*виконано*
5	*Розробка та тестування мобільного тренажера "Logic Gates"*	*29.04.2025-02.05.2025*	*виконано*
6	*Написання та оформлення пояснювальної записки*	*12.05.2025-23.05.2025*	*виконано*
7	*Перевірка на плагіат пояснювальної записки*	*26.05.2025-30.05.2025*	*виконано*
8	*Попередній захист кваліфікаційної роботи*	*02.06.2025-06.06.2025*	*виконано*
9	*Захист кваліфікаційної роботи*		

Здобувач освіти _____ *Герман КОЗЛОВ*
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ *Оксана ОСАДЧА*
(підпис) (ім'я, ПРІЗВИЩЕ)

StrikePlagiarism.com

Дата звіту 6/13/2025
Дата редагування —

Документ прийнятий

Звіт подібності

метадані

Назва організації
Ukrainian national aviation university
Заголовок
Козлов_Герман_Олексійович_3_011_2025_123_ДР
Автор Науковий керівник / Експерт
ГерманОсадча О.
Підрозділ
Криворізький Фаховий коледж

Обсяг знайдених подібностей

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

КП 1

КЦ

25	7904	63038
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

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

Заміна букв	B	0
Інтервали	A→	0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)	a	27

Подібності за списком джерел

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

10 найдовших фраз

Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	РОЗРОБКА ГРИ У ЖАНРІ TOWER DEFENSE З ВИКОРИСТАННЯМ ДВИГУНА UNITY НА МОБІЛЬНУ ПЛАТФОРМУ ANDROID [redacted] 5/23/2022 State University of Telecommunications (HHIIT)	29 0.37 %
2	https://pmtips.com.ua/post/agile-termini-zagalni-poniattia	25 0.32 %

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Мобільний тренажер «*Logic Gates*» містить: 51 сторінку основного тексту, 17 рисунків, 4 таблиці, 19 використаних літературних джерел, 1 додаток.

ЦИФРОВА ЛОГІКА, ОСВІТНЯ ГРА, МОБІЛЬНИЙ ТРЕНАЖЕР, *UNITY*, *C#*, ГЕЙМІФІКАЦІЯ

Мета роботи – розробка мобільного тренажеру «*Logic Gates*» для вивчення основ цифрової логіки, що забезпечує інтерактивну взаємодію користувача з логічними елементами та візуалізацію їх роботи.

У роботі проаналізовано сучасні методології розробки освітніх ігор, проведено порівняльний огляд рушіїв *Unity* та *Unreal Engine*, обґрунтовано вибір *Unity* для реалізації проєкту. Запропоновано концепцію тренажера, сформовано функціональні та нефункціональні вимоги, розроблено архітектуру програмного забезпечення з використанням шаблону *MVC*. Реалізовано основні модулі гри, включаючи навчальні рівні, систему взаємодії з логічними елементами та візуалізацію сигналів. Проведено тестування працездатності додатка на мобільних пристроях. Результати свідчать про відповідність програми поставленим вимогам, її зручність та ефективність для інтерактивного навчання цифрової логіки.

ЗМІСТ

ВСТУП

Сучасний світ характеризується стрімким розвитком інформаційних технологій та їх інтеграцією в усі сфери людської діяльності, включаючи освіту. В умовах цифрової трансформації актуальною є розробка інноваційних програмних продуктів, здатних підвищити ефективність навчання. Одним із перспективних напрямків є створення освітніх ігор, які поєднують навчальний контент та інтерактивні елементи. Актуальність теми обумовлена зростаючим попитом на інтерактивні навчальні інструменти для складних технічних дисциплін. Традиційні методи викладання цифрової логіки та булевої алгебри часто є абстрактними. Розробка мобільного тренажеру, що візуалізує роботу логічних елементів та дозволяє взаємодіяти зі схемами, значно підвищує якість освітнього процесу та мотивацію. Застосування ігрової розробки (*Unity, C#*) відкриває можливості для створення якісних та кросплатформних освітніх продуктів.

Сучасна освітня парадигма переходить від пасивних методів навчання до інтерактивних та діяльнісних підходів. Гейміфікація, що використовує ігрові механіки в неігровому контексті, популярна в освіті для підвищення мотивації, залучення та кращого засвоєння матеріалу через елементи змагання, нагороди, прогресу та негайного зворотного зв'язку. Інтерактивні методи, такі як симуляції та тренажери, надають можливість практично застосовувати теоретичні знання у безпечному віртуальному середовищі. Мобільні освітні застосунки є доступними та гнучкими інструментами, що пропонують портативність, персоналізацію, інтерактивність сенсорних екранів, миттєвий зворотний зв'язок та широку доступність. Технології віртуальної та доповненої реальності відкривають нові горизонти в освітньому процесі, створюючи іммерсивні середовища для симуляцій. Для ефективності освітнього програмного забезпечення критично враховувати когнітивне навантаження, мінімізуючи стороннє навантаження через чітке представлення інформації та інтуїтивний інтерфейс. Мультимедійні принципи навчання Річарда Майєра (принципи мультимедіа, відповідності, когерентності, надмірності) пропонують рекомендації щодо ефективного поєднання тексту,

зображень, аудіо та анімації. Ці принципи враховуються при проєктуванні візуалізації логічних елементів. Аналіз освітніх тенденцій вказує на перехід до студентоцентрованого навчання, де гейміфікація, мобільні технології, інтерактивність та персоналізація є ключовими драйверами. Розробка мобільного тренажера "*Logic Gates*" відповідає цим тенденціям.

Мета роботи – розробка мобільного тренажеру "*Logic Gate*" для вивчення основ цифрової логіки, що забезпечує інтерактивну взаємодію користувача з логічними елементами та візуалізацію їх роботи.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Проаналізувати теоретичні основи цифрової логіки та булевої алгебри, а також вимоги до програмних засобів для їх вивчення.

2. Дослідити можливості сучасних ігрових рушіїв та мов програмування для розробки інтерактивних освітніх застосунків.

3. Розробити архітектуру програмного забезпечення, включаючи структуру класів та модулів, що забезпечують логіку гри, інтерфейс та взаємодію.

4. Реалізувати основні функціональні модулі програми, зокрема: систему управління інтерфейсом (меню, вибір рівня), ігровий процес (розміщення логічних елементів, перевірка рішень) та систему зворотного зв'язку (таймер, індикація перемоги/програшу).

5. Провести тестування розробленого програмного забезпечення та оцінити його відповідність поставленим вимогам.

Об'єктом дослідження є процес інтерактивного навчання основам цифрової логіки за допомогою ігрових засобів.

Предметом дослідження є методи та засоби розробки програмного забезпечення для моделювання логічних елементів та схем.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА СЕРЕДОВИЩ РОЗРОБКИ

Для створення відеоігор необхідний ігровий рушій, що є інтегрованим комплексом інструментів для реалізації ігрової механіки, фізики, логіки та графічної складової. Ігровий рушій забезпечує розробникам середовище для управління об'єктами, їхніми фізичними властивостями, ігровою логікою (наприклад, таймерами, тригерами) та візуальним представленням (текстурами, спрайтами, скайбоксами тощо). Вибір рушія є критично важливим етапом у процесі розробки, оскільки він визначає доступний функціонал, продуктивність та екосистему для проєкту.

1.1 Методології розробки комп'ютерних ігор в контексті освітніх проєктів

Розробка комп'ютерних ігор, особливо освітніх, вимагає застосування специфічних методологій, що дозволяють ефективно управляти складними проєктами, інтегрувати навчальний контент та забезпечувати високу якість кінцевого продукту. Традиційні підходи до розробки програмного забезпечення, такі як каскадна модель (*Waterfall*), хоча і пропонують чітку структуру, часто виявляються занадто жорсткими для динамічного та креативного процесу створення ігор, де вимоги можуть змінюватися, а ітерації гейм-дизайну є ключовими. Тому в сучасній ігровій індустрії та, відповідно, в розробці освітніх ігор, перевага надається гнучким (*Agile*) методологіям [2].

Каскадна модель є лінійною послідовною методологією, де кожен етап проєкту (збір вимог, проєктування, реалізація, тестування, розгортання, підтримка) завершується перед початком наступного. Переваги цієї моделі полягають у її простоті, зрозумілості та можливості чіткого документування на кожному етапі, що полегшує управління проєктами з чітко визначеними та стабільними вимогами. Однак, у контексті розробки ігор, де важлива постійна ітерація, тестування ігрових

механік та швидка адаптація до відгуків користувачів, *Waterfall* демонструє значні недоліки. Виявлення помилок на пізніх етапах може бути надзвичайно дорогим та трудомістким, а жорсткість вимог не дозволяє швидко реагувати на зміни ринку або педагогічних потреб. В освітніх проєктах це може призвести до створення гри, яка не відповідає актуальним навчальним цілям або є нецікавою для цільової аудиторії, оскільки зворотний зв'язок від педагогів та учнів надходить занадто пізно, коли внесення суттєвих змін є вже неможливим або економічно невигідним. Це особливо критично для інноваційних освітніх продуктів, де ефективність навчання та залученість гравця залежать від постійних покращень та тестування інтерактивних механік.

На противагу *Waterfall*, *Agile*-методології акцентують увагу на ітеративній розробці, гнучкості до змін, співпраці зі стейкхолдерами та постійному покращенні. Основні принципи *Agile*, викладені в *Agile Manifesto*, включають: особистості та взаємодії важливіші за процеси та інструменти; працюючий продукт важливіший за вичерпну документацію; співпраця із замовником важливіша за узгодження контракту; готовність до змін важливіша за дотримання початкового плану. Ці принципи ідеально підходять для розробки ігор, де креативність, експерименти та тестування користувацького досвіду є центральними.

Найбільш популярними *Agile*-фреймворками, що застосовуються в розробці ігор, є:

- ***Scrum*:** *Scrum* є ітеративним та інкрементальним підходом до управління проєктами, де робота розбивається на короткі періоди, які називаються спринтами (зазвичай 1-4 тижні). Кожен спринт завершується працюючим фрагментом продукту, який демонструється стейкхолдерам. Ролі (власник продукту, скрам-майстер, команда розробки) та зустрічі (щоденні стендапи, планування спринту, огляд спринту, ретроспектива) чітко визначені. У контексті освітніх ігор, *Scrum* дозволяє інтегрувати педагогів та учнів у процес розробки на ранніх етапах, отримувати швидкий зворотний зв'язок щодо дидактичної ефективності та ігрової механіки, та оперативно вносити зміни. Наприклад, команда може розробити прототип рівня, протестувати його з групою учнів, отримати їхні відгуки щодо

складності та залученості, а потім використати цю інформацію для покращення наступного спринту. Цей підхід сприяє виявленню та усуненню проблем, пов'язаних з навчальним дизайном, на ранніх стадіях, зменшуючи ризики проєкту.

- ***Kanban*:** *Kanban* фокусується на візуалізації робочого процесу, обмеженні кількості одночасних завдань та підвищенні ефективності потоку роботи. Використовуються *Kanban*-дошки з колонками, що відображають різні стани завдань (наприклад, "До виконання", "В роботі", "На тестуванні", "Виконано"). *Kanban* особливо ефективний для проєктів, де є постійний потік завдань і потрібно швидко реагувати на нові пріоритети, що часто трапляється при розробці доповнень, підтримці освітніх платформ або роботі з численними малими ітераціями. Він дозволяє гнучко змінювати пріоритети без необхідності чекати завершення спринту, що може бути корисним у випадках, коли виникають непередбачені педагогічні потреби.

- ***eXtreme Programming* (*XP*):** *XP* є однією з найрадикальніших *Agile*-методологій, що акцентує увагу на високій якості коду, постійному тестуванні, парному програмуванні та простих рішеннях. Для розробки освітніх ігор це означає створення надійного та легко модифікованого коду, що важливо для тривалої підтримки та оновлення навчального контенту, а також для масштабованості проєкту. Принципи *XP*, такі як *Test-Driven Development* (*TDD*) та *Continuous Integration* (*CI*), забезпечують стабільність та якість програмного продукту з самого початку.

Ітеративний дизайн (*Iterative Design*)

Незалежно від обраного фреймворку, ітеративний дизайн є центральним принципом розробки ігор. Це процес, який включає повторювані цикли проєктування, прототипування, тестування та вдосконалення. Для освітніх ігор це критично важливо, оскільки дозволяє постійно перевіряти, чи гра дійсно досягає своїх навчальних цілей, чи вона є цікавою для цільової аудиторії, і чи вона функціонує належним чином. Прототипування дозволяє швидко перевіряти гіпотези щодо гейм-дизайну та дидактичної ефективності, мінімізуючи витрати на розробку невдалих рішень.

Специфіка розробки освітніх ігор

При виборі методології для освітніх ігор необхідно враховувати додаткові фактори:

- Педагогічний дизайн: інтеграція педагогічного дизайну у процес розробки є ключовою. Це означає, що навчальні цілі, методи оцінювання та дидактичні принципи мають бути враховані на кожному етапі, а не лише на початковому. Команда розробки має співпрацювати з експертами з педагогіки, щоб забезпечити методичну коректність гри.

- Цільова аудиторія: вікові особливості та рівень знань учнів сильно впливають на дизайн гри та методики тестування. Гра для дошкільнят матиме абсолютно інший інтерфейс та механіки, ніж гра для старшокласників чи студентів.

- Зворотний зв'язок від викладачів та учнів: постійна взаємодія з педагогами та безпосереднє тестування гри учнями дозволяє забезпечити відповідність гри навчальній програмі та її практичну цінність у реальному освітньому середовищі. Цей зворотний зв'язок є основою для ітеративних покращень.

Agile-методології, зокрема *Scrum* та *Kanban,* є найбільш придатними для розробки освітніх комп'ютерних ігор завдяки їхній гнучкості, ітеративному підходу та здатності швидко реагувати на зміни. Вони дозволяють інтегрувати педагогічний дизайн та зворотний зв'язок від цільової аудиторії на ранніх етапах, що є критично важливим для створення ефективних та привабливих навчальних продуктів. Комбінація цих методологій з ітеративним дизайном забезпечує високу якість продукту та його відповідність освітнім потребам.

1.2 Огляд ігрового рушія *Unity*

Unity — це багатоплатформний інструмент та ігровий рушій, що широко використовується для розробки відеоігор та інших інтерактивних застосунків. Він підтримує створення дво- та тривимірної графіки та функціонує на широкому спектрі платформ, включаючи настільні комп'ютери, мобільні пристрої, ігрові консолі, а також пристрої віртуальної та доповненої реальності [28]. *Unity* підтримує такі графічні *API,* як *DirectX* та *OpenGL* .

1.2.1 Історія та розвиток *Unity*

Щоб краще зрозуміти поточні можливості та напрямки розвитку, необхідно звернутися до його історичного шляху. Основні етапи еволюції рушія *Unity* подано у таблиці 1.1.

Таблиця 1.1 – Історія розвитку *Unity*

Версія	**Рік виходу**	**Основні нововведення**
***Unity* 1.0**	2005	Запуск на *WWDC* для *Mac OS X*; підтримка лише *Mac*; фіналіст *Apple Design Awards*.
***Unity* 2.0**	2007	~50 нових функцій: *DirectX*, реальні тіні, відео, мережа на *UDP*, версійність, підтримка *iPhone* (2008), редактор для *Windows* (2009).
***Unity* 3.0**	2010	Підтримка *Android*, *Beast*, *Deferred Rendering*, редактор дерев, *UV*-розгортання, аудіофільтри.
***Unity* 4.0**	2012	*DirectX* 11, *Flash*, анімація *Mecanim*, перегляд редактора *Linux*, *SDK* для *Facebook*.
***Unity* 5.0**	2015	Реальне глобальне освітлення, *WebGL*, *PhysX* 3.3, кінематографічні ефекти, *Unity Cloud*, експериментальний редактор для *Linux*.
***Unity* 5.6**	2017	Підтримка *Nintendo Switch*, *Daydream*, *Vulkan API*, *Facebook Gameroom*, 4*K VR* відео.
***Unity* 2017**	2017	*Timeline*, *Cinemachine*, аналітика, реальний рендеринг, інтеграція з *Autodesk*.
***Unity* 2018**	2018	*HDRP*/*URP* (*SRP*), *ML*-інструменти, *Magic Leap*, *Unity Hub*, шаблони новачків.
***Unity* 2020**	2020	*Unity MARS* (*AR*), *Unity Forma* (авто, ритейл), підтримка *Apple Silicon*.
***Unity* 2021**	2021	*Bolt* (візуальне скриптування), нова мережа, *Il2cpp*, *SSGI*, *Deferred URP*.
***Unity* 2022**	2022	Продуктивність, оновлення 2*D* (*PSD*, фізика, тесселяція), *Unity Hub* 3.0.

Ця таблиця узагальнює історію розвитку ігрового рушія *Unity*, від його першого випуску у 2005 році до запланованої версії *Unity* 6 у 2024 році. Вона візуалізує ключові етапи еволюції рушія, демонструючи, які нові функції та покращення були додані в кожній значній версії, а також які важливі події чи зміни

відбувалися в його житті (наприклад, зміна системи нумерації версій, співпраця з іншими компаніями, вплив на ринок).

1.2.2 Підтримувані платформи

Unity є кросплатформним рушієм. Редактор *Unity* підтримується на *Windows*, *macOS* і *Linux*. Ігрова платформа дозволяє збирати ігри для понад 19 різних платформ, включаючи мобільні пристрої, ПК, консолі та пристрої віртуальної реальності. Детальний опис його функціональних можливостей та список підтримуваних платформ представлено у таблиці 1.2.

Таблиця 1.2 – Функціональність та платформи

Категорія	**Опис**
Мови програмування	*C#* (основна), раніше *Boo* та *UnityScript* (до 2017).
Графіка	Підтримка 2*D*/3*D, bump*/*reflection*/*parallax mapping, SSAO*, динамічні тіні, постобробка, текстурне рендерування.
Рендер пайплайни	*HDRP* (висока якість), *URP* (універсальний), *Built-in* (застарілий).
Інструменти	*Drag* & *Drop, Unity Hub*, *Timeline*, *Cinemachine*, *Bolt*, *Asset Store*.
Платформи	*Windows, macOS, Linux, Android, iOS, WebGL , AR*/*VR*, консолі, *Magic Leap*.
Unity Asset Store	Запущено у 2010, до 2018 – ~40 млн завантажень. Доступ для купівлі/продажу активів.

1.2.3 Застосування *Unity*

Від культових мобільних ігор до передових розробок у галузі штучного інтелекту та кіновиробництва, *Unity* зарекомендував себе як універсальна платформа, що відкриває безмежні можливості для створення інтерактивного контенту.

Відеоігри: *Unity* використовується для створення широкого спектру ігор, включаючи такі відомі проєкти, як *Pokémon Go, Monument Valley, Call of Duty*:

Mobile, *Beat Saber* та *Cuphead* . Одними з перших комерційних проєктів на *Unity* у 2007 році були *Splume* (*Flashbang Studios*), *Magical Flying Pink Pony Game* (*Starscene Software*) та *Global Conflicts*: *Palestine* (*Serious Games Interactive*). Станом на 2018 рік, *Unity* використовувався для створення приблизно половини всіх мобільних ігор на ринку та 60% контенту в галузях доповненої та віртуальної реальності, включаючи близько 90% продуктів для нових платформ, таких як *Microsoft HoloLens* та *Samsung Gear VR*. Технології *Unity* домінують на ринку віртуальної реальності, лежачи в основі більшості *VR*/*AR*-досвідів .

Машинне навчання: *Unity* надає інструменти для дослідників у сфері глибинного навчання з підкріпленням, дозволяючи навчати агентів у змодельованих середовищах. *Unity Machine Learning Agents* можуть діяти як віртуальні персонажі або роботи, щоб розробляти творчі стратегії взаємодії з симульованим навколишнім світом. Програмне забезпечення використовується, зокрема, для розробки роботів і систем автономного керування, таких як самокеровані автомобілі .

Промисловість поза іграми: У 2010-х роках *Unity Technologies* розширила використання свого рушія за межі ігрової сфери, застосовуючи його як платформу реального часу для потреб кіноіндустрії та автомобілебудування. Першим експериментом *Unity* у кіно став короткометражний фільм "*Adam*" про робота, який тікає з в'язниці . Згодом *Unity* співпрацювала з режисером Нілом Бломкампом і його студією *Oats Studios*, яка використала інструменти рушія, зокрема *Cinemachine* і рендеринг у реальному часі, для створення двох *CGI*-фільмів — "*Adam*: *The Mirror*" і "*Adam*: *The Prophet*" . Під час конференції *Unite Europe* 2017 в Амстердамі *Unity* зосередилася на підтримці кіновиробництва, представивши новий інструмент *Cinemachine* в *Unity* 2017.1. У 2018 році *Disney Television Animation* створила три короткометражні фільми під назвою "*Baymax Dreams*" за допомогою *Unity*. Також *Unity* використовувався *Disney* для створення фонових сцен у фільмі "Король Лев" 2019 року . Автомобільні компанії застосовують технології *Unity* для віртуального моделювання автомобілів у повний розмір, проєктування виробничих ліній і навчання працівників . Компанія також розробляє

рішення для архітектури, інженерії та будівництва . Рушій *Unity* використовується компанією *DeepMind* для тренування штучного інтелекту . Крім того, *Unity Technologies* активно впроваджує свої рішення в галузях архітектури, інженерії та будівництва.

1.3 Огляд ігрового рушія *Unreal Engine*

Unreal Engine 5 (*UE*5) — це найновіша версія рушія *Unreal Engine*, розроблена компанією *Epic Games*. Вперше його було представлено у травні 2020 року, а офіційний реліз відбувся у квітні 2022 року [26]. *UE*5 містить численні оновлення та нові функції, серед яких:

- *Nanite*: система віртуалізованої геометрії, що автоматично регулює рівень деталізації 3*D*-моделей (мешів), зберігаючи високу якість графіки без втрати продуктивності.
- *Lumen*: система динамічного глобального освітлення та відбиттів, яка працює як на програмному забезпеченні, так і з апаратним прискоренням за допомогою трасування променів (*ray tracing*) .

Ці інструменти дозволяють створювати більш реалістичні та деталізовані світи в режимі реального часу.

1.3.1 Історія та розвиток *Unreal Engine*

Представлена таблиця слугує для систематичного та хронологічного відображення ключових етапів еволюції ігрового рушія *Unreal Engine*. Її основна мета полягає в деталізації послідовних версій рушія, починаючи від його першого випуску до сучасних ітерацій.

Таблиця 1.3 – Основні події історії *Unreal Engine*

Версія *Unreal Engine*	**Рік випуску**	**Ключові особливості / Покращення**	**Примітки / Вплив**
***Unreal Engine* 1 (*UE*1)**	1998	Програмне відтворення, згодом апаратне прискорення; виявлення зіткнень; кольорове освітлення; обмежена фільтрація текстур; вбудований редактор *UnrealEd*.	Дебют у грі *Unreal*. Підтримка *Windows, Linux, Mac, Unix, Dreamcast, PS*2. Початок ліцензування рушія іншим студіям.
***Unreal Engine* 2 (*UE*2)**	2002	Повноцінний апаратний рендеринг; підтримка *DirectX* 8; перші піксельні та вершинні шейдери; кінематичний редактор; системи частинок; скелетна анімація.	Дебют у *America's Army*. Розширена підтримка *Xbox, PS*2, *GameCube, PSP*. Покращена графіка та реалістичне освітлення.
***Unreal Engine* 3 (*UE*3)**	2006	Підтримка багатопоточності; *DirectX* 9; програмовані шейдери; руйновані середовища; динаміка м’яких тіл; симуляція натовпів.	Дебют у *Gears of War*. Один із перших рушіїв нового покоління. Пізніше додано підтримку *iOS, Steamworks*, глобального освітлення. Безкоштовний *Unreal Development Kit* (*UDK*).
***Unreal Engine* 4 (*UE*4)**	2014	Фізично засновані матеріали (*PBR*); візуальне програмування *Blueprints*; оновлений інтерфейс; *Niagara*; *Volumetric Fog*.	Модель ліцензування: підписка + роялті. Підтримка *VR*, мобільних платформ, нових консолей. Запуск *Unreal Marketplace* та *Dev Grants*. Важливий для інді-розробників.
***Unreal Engine* 5 (*UE*5)**	2022	*Nanite* (віртуалізація геометрії); *Lumen* (глобальне освітлення); *Virtual Shadow Maps*; *World Partition*; *MetaHumans*; оновлений редактор.	Випущено 5 квітня 2022. Революційні інструменти для створення великих відкритих світів і фотореалістичного контенту. Застосування в іграх, фільмах, телебаченні.

1.3.2 Особливості *Unreal Engine*

Однією з ключових особливостей *Unreal Engine* 5 є *Nanite* — система віртуалізованої геометрії, що дозволяє розробникам використовувати фотограмметрію та інші детально опрацьовані 3*D*-моделі в іграх без значного впливу на продуктивність [25]. Раніше художники створювали кілька моделей з різними рівнями деталізації (*LoD*) і генерували нормал-мапи для передачі дрібних деталей. *Nanite* автоматично керує рівнями деталізації, масштабуючи моделі залежно від дистанції в кадрі, роздільної здатності екрана та вимог до продуктивності. Вона використовує ієрархічну структуру, яка дозволяє різним

частинам однієї сітки рендеритися з різними рівнями деталізації. *Nanite* сумісна з багатьма 3*D*-форматами моделей, включаючи *ZBrush*-скульпти та *CAD*-моделі, що дає змогу імпортувати кіноякісні об'єкти без ручної оптимізації . За словами Браяна Каріса з *Epic Games*, однією з головних інновацій *Nanite* є її здатність шити краї між різними рівнями деталізації без появи розривів . У початковому релізі *Nanite* працювала лише зі статичними сітками [25]. *Unreal Engine* 5 також використовує надшвидке твердотільне сховище сучасного обладнання для потокового завантаження об'єктів у пам'ять у міру потреби . Генеральний директор *Epic Games* Тім Суїні наголосив, що така швидкість дозволяє виводити на екран геометрію, яка фізично не вміщується в пам'яті, — це усуває традиційні екрани завантаження і забезпечує плавні переходи між рівнями деталізації . Крім того, рушій пропонує систему "*World Partition*", яка ділить великі карти на менші секції, зменшуючи обсяг даних, що потребують одночасного завантаження [25].

Lumen — це система глобального освітлення та відбиттів на основі трасування променів, яка динамічно реагує на зміни у сцені та освітленні в режимі реального часу . Вона усуває необхідність попереднього розрахунку світлових карт і забезпечує автоматичне налаштування світла, тіней та відбиттів. *Lumen* підтримує як програмне, так і апаратне трасування променів . Програмне трасування, що використовує *Mesh Distance Fields*, забезпечує широку сумісність і високу швидкість, але знижує точність. Апаратне трасування пропонує точніші результати та підтримує додаткові типи геометрії, наприклад, анімовані моделі [25]. Система *Surface Cache* зменшує обчислювальне навантаження, необхідне для обрахунку освітлення. Якщо *Lumen* вимкнено, рушій використовує альтернативу — *Signed Distance Field Ambient Occlusion* — менш точну, але продуктивну систему .

Virtual Shadow Maps — це новий метод побудови тіней, який забезпечує стабільне, високоякісне затінення, що підходить для високо полігональних об'єктів і великих динамічно освітлених світів . На відміну від класичних методів тіней, ця система пропонує вищу роздільну здатність і усуває проблеми з каскадами та появою/зникненням тіней при зміні положення камери [25].

Інші функції: *UE*5 використовує систему *Niagara* для симуляції рідин і частинок та власний фізичний рушій *Chaos*, який замінив *PhysX* [25]. Починаючи з версії *UE*5.2, рушій також має нову систему створення матеріалів *Substrate*, яка дозволяє більш гнучко та модульно створювати матеріали .

Unreal Engine 5 також використовує технології, отримані після придбань та партнерств *Epic Games*:

Nanite інтегрує можливості бібліотеки *Quixel*, найбільшої у світі фотограмметричної колекції на 2019 рік .

MetaHuman Creator дозволяє швидко створювати реалістичних персонажів на основі технологій компаній 3*Lateral*, *Cubic Motion* та *Quixel* [25].

У партнерстві з *Cesium*, *Epic* запропонувала безкоштовний плагін, який надає 3*D*-геопросторові дані, що дозволяє відтворювати будь-яку частину поверхні Землі.

RealityCapture, продукт, який створює 3*D*-моделі об'єктів на основі фотографій з різних кутів, став частиною *Unreal* після придбання компанії *Capturing Reality* .

Підтримка *middleware*-інструментів від *Epic Games Tools* .

Починаючи з версії *UE* 5.5, *Epic Games* додала внутрішній шар підтримки *WebRTC*, що дозволило плагіну *Pixel Streaming* 2 поставлятися разом з *Unreal Engine* .

Unreal Engine 5 отримав високу оцінку за те, що дозволяє розробникам створювати ігри з високоякісними візуальними активами завдяки *Nanite* та з реалістичним освітленням завдяки *Lumen* [25]. Ігрові медіа відзначили успішне використання *UE*5 у численних проєктах з 2023 по 2025 роки, включаючи *Black Myth*: *Wukong*, *Remnant* 2, *Senua's Saga*: *Hellblade II* та *Avowed* . *Unreal Engine* 5 також отримав похвалу за те, що дає змогу незалежним розробникам і невеликим студіям (так званим "*AA*" студіям) досягати рівня графічної якості, подібного до того, який демонструють великі *AAA*-компанії .

Разом з тим, рушій зазнав критики за певні проблеми з продуктивністю, зокрема:

- пригальмовування під час компіляції шейдерів (*shader compilation stutter*);
- пригальмовування під час переміщення (*traversal stutter*), — тобто стрибки часу кадру, які виникають, коли гравець активує завантаження нових об'єктів у грі.

1.4 Порівняльний аналіз *Unity* та *Unreal Engine*

Обидва пропонують потужні інструменти для створення ігор різної складності та жанрів, але мають суттєві відмінності у своїй філософії, архітектурі та цільовій аудиторії. Розуміння цих відмінностей є життєво важливим для раціонального вибору рушія під конкретний проект. Детальне порівняння *Unity* та *Unreal Engine* за основними критеріями, що дозволяє оцінити їхні переваги та недоліки, представлено у таблиці 1.4.

Таблиця 1.4 – Порівняння *Unity* та *Unreal Engine*

Характеристика	***Unity***	***Unreal Engine* (*UE*5)**
Розробник	*Unity Technologies*	*Epic Games*
Дата першого релізу	2005	1998 (*UE*1), *UE*5 — 2022
Мова програмування	*C#*	*C++*, *Blueprints*
Підтримка платформ	19+ платформ (мобільні, ПК, консолі, *VR*, *WebGL*)	ПК, консолі, мобільні, *VR*/*AR*, *Web* (експериментально)
Основні рендер-рушії	*URP*, *HDRP*, *Legacy Renderer*	*Nanite*, *Lumen*, *Virtual Shadow Maps*
2*D* підтримка	Потужна підтримка 2*D*	Обмежена 2*D*-підтримка
Графіка	Добра, але менш фотореалістична	Високо Фотореалістична
Простота використання	Дружній до новачків	Складніший, але потужний
Візуальне програмування	Обмежене (*Bolt*)	*Blueprints* — вбудована система
Ліцензія та вартість	Безкоштовно до $100*K*/рік, далі — підписка	Безкоштовно до $1 млн, потім 5% роялті
Asset Store/*Marketplace*	*Unity Asset Store*	*Unreal Marketplace*
Використання в іграх	Мобільні, інді, *AR*/*VR*	*AAA*-ігри, кінематографічні проєкти

Продовження таблиці 1.4

Використання поза іграми	Автомобілі, архітектура, медицина	Кіно, архітектура, віртуальне виробництво
Підтримка *VR*/*AR*	Потужна (*HoloLens, ARKit, ARCore*)	Потужна, у промислових рішеннях
Головна перевага	Кросплатформеність, простота	Передова графіка
Головний недолік	Менш потужна графіка	Вища складність, ресурсоємність

Для розробки "*Logic Gates*" був обраний *Unity*. Основними критеріями, що зумовили цей вибір, стали: простота використання та добра підтримка 2*D*-графіки. *Unity* пропонує інтуїтивно зрозуміле середовище розробки, що є перевагою для проектів, де пріоритетом є швидка імплементація та тестування логіки гри. Крім того, наявність широкого спектру інструментів для роботи з 2*D*-світами, а також велика спільнота та доступність ресурсів в *Asset Store*, роблять *Unity* оптимальним вибором для цього типу проєкту.

РОЗДІЛ 2
КОНЦЕПЦІЯ ТА ПРОЄКТУВАННЯ МОБІЛЬНОГО ТРЕНАЖЕРА «*LOGIC GATES*»

2.1 Огляд існуючих програмних продуктів для навчання цифровій логіці

Перед початком розробки нового програмного продукту, необхідно провести детальний аналіз існуючих рішень. Це дозволяє виявити їхні сильні та слабкі сторони, визначити незаповнені ніші та уникнути дублювання вже реалізованого функціоналу.

Існує низка програмних засобів, призначених для навчання цифровій логіці:

- Десктопні симулятори: До таких належать *Logisim, CircuitVerse, Falstad Circuit Simulator*. Вони зазвичай пропонують широкий функціонал для проєктування та симуляції складних цифрових схем, мають великі бібліотеки елементів. Однак, вони вимагають інсталяції на ПК, не завжди мають інтуїтивний інтерфейс для новачків і не є мобільними.
- Веб-симулятори: Деякі онлайн-платформи (наприклад, *CircuitVerse*) пропонують веб-версії симуляторів. Вони доступні з будь-якого пристрою з браузером, але часто вимагають стабільного інтернет-з'єднання та можуть бути неоптимізованими для мобільних екранів.
- Мобільні додатки: Існують мобільні додатки, такі як "*Logic Gate Simulator*", "*Digital Logic Gates*", "*Boolean Algebra Solver*". Вони зазвичай орієнтовані на конкретні аспекти (наприклад, симуляцію окремих вентилів, або вирішення булевих рівнянь). Часто їм бракує комплексного підходу, гейміфікації, або вони мають застарілий інтерфейс та обмежений функціонал.

Висновок з огляду: Існуючі рішення, хоч і надають певний функціонал, часто мають такі недоліки:

- Брак гейміфікованих елементів для підвищення мотивації.
- Недостатня інтерактивність та візуалізація процесів.

- Обмежена доступність (лише десктопні або неоптимізовані для мобільних).
- Відсутність персоналізованих навчальних траєкторій або адаптивних завдань.
- Застарілий дизайн інтерфейсу та досвід користувача.

Розробка мобільного тренажера "*Logic gates*" має на меті заповнити ці прогалини, надавши комплексний, інтерактивний, гейміфікований та візуально привабливий інструмент для вивчення цифрової логіки на мобільних пристроях.

2.2 Концепція мобільного тренажера "*Logic Gates*"

На основі проведеного аналізу освітніх тенденцій та існуючих рішень, формується детальна концепція мобільного тренажера "*Logic gates*", яка визначає його функціональні та нефункціональні вимоги, а також основні сценарії використання.

2.2.1 Мета та основні ідеї додатка

Мета: Забезпечити ефективне, інтерактивне та цікаве вивчення та закріплення принципів роботи цифрових логічних елементів для широкого кола користувачів – від школярів до студентів технічних спеціальностей.

Основні ідеї:

- Навчання через практику: замість пасивного споглядання, користувач активно взаємодіє з віртуальними логічними елементами.
- Візуалізація: чітке та динамічне відображення потоків сигналів та логічних станів.
- Гейміфікація: впровадження елементів гри (рівні, завдання, система оцінок, підказки) для підвищення мотивації.
- Персоналізація: можливість адаптації складності завдань та темпу навчання.
- Доступність: можливість використання на більшості сучасних мобільних пристроїв.

2.2.2 Функціональні вимоги до тренажера

Функціональні вимоги описують, що саме система повинна робити:

1) Режим "Вивчення":

- відображення детальної інформації про кожен з семи базових логічних елементів (І, АБО, НЕ, ВИКЛ. АБО, І-НЕ, АБО-НЕ).
- представлення їх графічних позначень (за стандартом *IEC* або *ANSI*, за вибором).
- короткий опис принципу роботи.

2) Візуалізація сигналів: чітке візуальне відображення логічних станів ("0" і "1") на входах, виходах елементів та з'єднувальних лініях (наприклад, різні кольори або анімація для "0" та "1").

2.2.3 Нефункціональні вимоги

Нефункціональні вимоги описують, *як* система повинна працювати, а не *що* вона повинна робити.

1) Зручність використання (*Usability*):

- Інтуїтивно зрозумілий інтерфейс: мінімальна крива навчання для нових користувачів. Елементи керування повинні бути легко зрозумілими без додаткових інструкцій.
- Зрозумілий зворотний зв'язок: чітке сповіщення користувача про успіх або помилку, а також візуальні підказки.

2) Продуктивність: Додаток повинен працювати без помітних затримок та зависань навіть на пристроях середнього рівня. Час завантаження має бути мінімальним.

3) Надійність: Додаток повинен бути стабільним та без помилок, здатним коректно обробляти різні вхідні дані та сценарії використання. Непередбачені завершення роботи неприпустимі.

4) Адаптивність: Інтерфейс повинен автоматично адаптуватися до різних розмірів екранів мобільних пристроїв та їх орієнтацій (портретна/ландшафтна) без втрати функціоналу або візуальної привабливості.

5) Масштабованість: Архітектура має бути розроблена таким чином, щоб дозволяти легке додавання нових логічних елементів, типів завдань, розширення рівнів складності або впровадження нових функціональних можливостей (наприклад, збереження власних схем).

2.2.4 Сценарії використання та цільова аудиторія

Цільова аудиторія:

- Учні старших класів: для першого ознайомлення з основами цифрової логіки.
- Студенти технічних спеціальностей: для закріплення теоретичних знань та отримання практичних навичок з цифрової схемотехніки, електроніки, комп'ютерних наук.
- Люди, що займаються самоосвітою: для всіх, хто цікавиться електронікою та програмуванням.

2.3 Проєктування архітектури та інтерфейсу користувача

На етапі проєктування визначається внутрішня структура програмного продукту та його зовнішній вигляд, що забезпечує логічне та ефективне функціонування.

2.3.1 Загальна архітектура програмного продукту

Для забезпечення чіткої структури, гнучкості та легкого масштабування, архітектура мобільного тренажера "*Logic gates*" буде реалізована з використанням шаблону Модель-Вид-Контролер (*MVC*), представленому на рисунку 2.1.

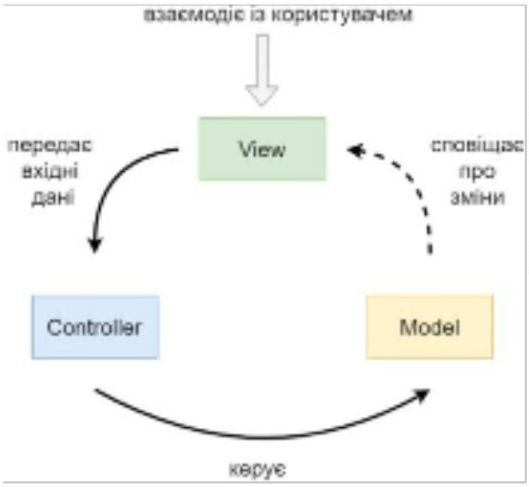

Рисунок 2.1 – Шаблон Модель-Вид-Контролер

Цей шаблон дозволяє ефективно розділити логіку застосунку на три взаємопов'язані компоненти, що сприяє чистішому коду та полегшує тестування.

1) Модель (*Model*): Це ядро застосунку, що містить дані та основну бізнес-логіку. Для "*Logic gates*" Модель включатиме:

- дані про всі типи логічних елементів (ідентифікатори, таблиці істинності, візуальні властивості);
- представлення поточної логічної схеми (з'єднання, стани входів/виходів);
- дані про завдання (текст завдання, очікувані входи/виходи, правильне рішення).

2) Вид (*View*): Цей компонент відповідає за відображення Моделі користувачу. В *Unity* Вид буде представлений ігровими об'єктами (*GameObject*), що візуалізують:

- самі логічні елементи на ігровому полі (спрайти або 3*D*-моделі);
- з'єднувальні лінії та індикатори сигналів;
- елементи інтерфейсу користувача (кнопки, текстові поля для завдань, панелі статистики, меню);

3) Контролер (*Controller*): Цей компонент є посередником між Моделлю та Видом. Він приймає вхідні дані від користувача (наприклад, кліки, перетягування), обробляє їх, оновлює Модель та дає вказівки Виду щодо оновлення відображення.

Для "*Logic gates*" Контролер виконуватиме:

- обробку подій користувача (перетягування елементів, встановлення з'єднань, натискання кнопок);
- запуск процесу перевірки рішення, отримання результатів від Моделі;
- щновлення інтерфейсу користувача на основі змін у Моделі (наприклад, зміна кольору ліній при зміні сигналу).

Діаграма компонентів (Додаток А) відображатиме взаємодію між основними модулями: Модуль логічних елементів, Модуль генерації завдань, Модуль *UI*, Модуль збереження даних.

2.4 *C#* як основа архітектури додатку: функціональні можливості та реалізація

C# — це сучасна, об'єктно-орієнтована мова програмування [1], розроблена компанією *Microsoft* [2] у рамках її ініціативи *.NET* [3]. Вона поєднує в собі найкращі риси таких мов, як *C++*, *Java* та *Delphi*, пропонуючи розробникам потужний та гнучкий інструмент для створення широкого спектру додатків.

Основні характеристики та сфери застосування:

1) Об'єктно-орієнтована: *C#* повністю підтримує принципи ООП, такі як інкапсуляція, успадкування та поліморфізм. Це сприяє створенню модульного, повторно використовуваного та легко підтримуваного коду. Більше про це можна дізнатись в офіційній документації *C#* [1].

2) Типобезпечна: *C#* є строго типізованою мовою, що допомагає виявляти багато помилок на етапі компіляції, а не під час виконання, підвищуючи надійність додатків.

3) Кросплатформна: Завдяки платформі *.NET* (особливо з появою *.NET Core* / *.NET* 5+) [4], *C#* дозволяє розробляти додатки, які можуть працювати на різних операційних системах, включаючи *Windows*, *Linux* та *macOS*.

4) Гнучкість та потужність: Мова підтримує такі концепції, як делегати [5], події [6], *LINQ* (*Language Integrated Query*) [7] для роботи з даними, асинхронне

програмування (*async*/*await*) [8] та багато іншого, що робить її дуже потужною для складних завдань.

5) Широкий спектр застосування: *C#* використовується для розробки:

- настільних додатків: за допомогою *.NET MAUI* [9] (для кросплатформних) або *WPF*/*Windows Forms* (для *Windows*);
- веб-додатків: за допомогою *ASP.NET Core* [11] для створення високопродуктивних веб-сайтів та *API*;
- ігор: *Unity*, один з найпопулярніших ігрових рушіїв у світі, використовує *C#* як основну мову сценаріїв [13];
- мобільних додатків: за допомогою *.NET MAUI* або *Xamarin* [14] (який тепер є частиною *.NET MAUI*);
- хмарних сервісів: інтеграція з *Azure* [15] та іншими хмарними платформами;
- штучного інтелекту та машинного навчання: завдяки бібліотекам, таким як *ML.NET* [16].

C# є невід'ємною частиною платформи *.NET*, яка надає велику бібліотеку класів (*Base Class Library)* [17], середовище виконання (*Common Language Runtime - CLR*) та інструменти розробки. *CLR* керує виконанням *C#*-коду, забезпечуючи такі функції, як управління пам'яттю (збирання сміття) [18], безпека типів та обробка винятків.

Популярність. *C#* стабільно входить до топ-10 найпопулярніших мов програмування [19] згідно з різними індексами (наприклад, *TIOBE*, *PYPL*). Це пов'язано з її універсальністю, підтримкою з боку великої корпорації (*Microsoft*) та активною спільнотою розробників.

2.5 *UI /UX* дизайн

2.5.1 Ключові принципи *UX* дизайну

Досвід користувача (*UX*) охоплює всі аспекти взаємодії користувача з продуктом. Для освітніх ігор це означає, що кожен етап – від першого запуску гри

до проходження останнього рівня – має бути спланований таким чином, щоб забезпечити позитивний, навчальний та безперешкодний досвід.

Зрозумілість (*Clarity*): Інтерфейс має бути максимально зрозумілим, щоб учень одразу розумів, що відбувається на екрані, які елементи інтерактивні, і як виконувати завдання. Використання звичних іконок, чітких текстових підписів та інтуїтивно зрозумілих візуальних метафор є ключовим. Наприклад, кнопки "Пуск" або "Пауза" мають виглядати відповідно до загальноприйнятих стандартів. Важливо уникати двозначності та надмірної складності, особливо для молодших вікових груп або новачків.

Доступність (*Accessibility*): Освітні ігри мають бути доступними для широкого кола користувачів, включаючи тих, хто має особливі потреби (наприклад, порушення зору, слуху, когнітивні особливості). Це включає забезпечення можливості налаштування розміру шрифту, контрастності (достатній контраст тексту до фону), використання альтернативних методів введення (наприклад, керування клавіатурою, якщо миша незручна), підтримку скрін-рідерів для слабозорих користувачів та уникнення кольорів, які можуть бути проблемними для людей з дальтонізмом. Дотримання рекомендацій *WCAG* (*Web Content Accessibility Guidelines*) є рекомендованим стандартом для створення інклюзивних цифрових продуктів.

Консистентність (*Consistency*): Елементи інтерфейсу, їхнє розташування, кольорова гама, шрифти та анімації мають бути однаковими по всій грі. Це допомагає учням швидше звикнути до інтерфейсу та зосередитися на навчальному контенті, а не на розумінні того, як працює гра. Наприклад, кнопка "Назад" завжди має знаходитися в одному і тому ж кутку екрану, а успішні дії завжди позначатися одним і тим же кольором.

Зворотний зв'язок (*Feedback*): Кожна дія гравця повинна мати чіткий і негайний зворотний зв'язок. Це може бути візуальний ефект (зміна стану кнопки, анімація успіху/помилки, миготіння), звуковий ефект (короткий звук підтвердження, сигнал помилки) або текстове повідомлення. Зворотний зв'язок є критично важливим для навчання, оскільки він дозволяє учневі зрозуміти наслідки

своїх дій та коригувати їх. У контексті гри "*Drag*", поява "хрестика" при помилці та "прилипання" об'єкта до цільової зони при успіху є яскравими прикладами ефективного зворотного зв'язку.

Ефективність та продуктивність (*Efficiency* & *Performance*): Інтерфейс не повинен уповільнювати гру або відволікати від основного завдання. Анімації та переходи мають бути плавними, час завантаження мінімальним. Це забезпечує безперешкодний навчальний процес і запобігає фрустрації користувача. Оптимізація ресурсів, таких як текстури та спрайти, є ключовою для досягнення високої продуктивності.

Контроль користувача (*User Control*): Користувач повинен відчувати контроль над грою. Можливість скасувати дію, перейти в меню, регулювати звук, змінити налаштування – усе це сприяє кращому *UX* та відчуттю автономності гравця. Надання користувачеві вибору та можливості адаптувати гру під власні потреби підвищує задоволеність.

Естетика та привабливість (*Aesthetics* & *Attractiveness*): Хоча це може здатися другорядним для освітніх ігор, привабливий та естетичний інтерфейс може значно підвищити мотивацію та задоволеність учнів. Приємний візуальний стиль створює позитивне перше враження та заохочує до подальшої взаємодії. Дослідження показують, що естетично привабливі інтерфейси сприймаються як більш функціональні та прості у використанні, навіть якщо це не завжди відповідає дійсності (ефект естетичної зручності).

2.5.2 Особливості *UI* дизайну в освітніх іграх

UI (*User Interface*) – це візуальні елементи, з якими користувач взаємодіє. В освітніх іграх *UI* має бути не тільки функціональним, але й дидактично ефективним.

Мінімалізм та відсутність перевантаження: Освітня гра не повинна перевантажувати учня надмірною кількістю інформації або кнопок на екрані. Кожен елемент має мати чітке призначення, щоб учень міг зосередитися на

навчальному завданні. Принцип "менше – це більше" є особливо актуальним для освітніх застосунків, де когнітивне навантаження має бути оптимальним.

Інтуїтивна навігація: Меню, кнопки "назад", "далі", "вихід" повинні бути легко впізнаваними та розташованими в очікуваних місцях (наприклад, кнопки навігації в нижній частині екрану, меню паузи в кутку). Уникати прихованих функцій або складних комбінацій клавіш.

Візуальна ієрархія: Важливі елементи (навчальне завдання, таймер, індикатор прогресу, основні інтерактивні об'єкти) повинні бути візуально виділені та легко помітні. Це може бути досягнуто за допомогою розміру, кольору, контрасту, тіней або анімації. Чітка візуальна ієрархія допомагає користувачеві швидко сканувати інтерфейс та знаходити потрібну інформацію.

Іконки та символи: Використання універсальних та зрозумілих іконок може значно покращити навігацію та розуміння функціоналу, особливо для дітей або міжнародної аудиторії, зменшуючи потребу в текстових поясненнях. Іконки мають бути простими та легко розпізнаваними.

Типографіка: Вибір шрифтів має враховувати вік цільової аудиторії та читабельність. Для дітей краще використовувати великі, чіткі, несерифні шрифти. Контрастність тексту до фону є критичною для зручності читання та доступності. Розмір шрифту також має бути адаптивним або достатньо великим для комфортного читання на різних пристроях.

Кольорова палітра: Кольори впливають на емоції та сприйняття. В освітніх іграх часто використовуються яскраві, привабливі кольори для залучення уваги, але також важливо уникати надмірно агресивних, відволікаючих або тих, що викликають зорову втому, комбінацій. Колір може використовуватися для індикації статусу (зелений – правильно, червоний – неправильно), диференціації елементів та створення певної атмосфери. Наприклад, пастельні тони можуть створювати спокійну атмосферу, тоді як яскраві – динамічну.

2.5.3 Взаємодія UI/UX з навчальним контентом

Динамічна адаптація *UI*: Інтерфейс може адаптуватися до прогресу учня. Наприклад, на початкових етапах можуть бути більш помітні підказки або візуальні інструкції, які поступово зникають у міру освоєння матеріалу та зростання рівня складності. Це допомагає підтримувати оптимальний рівень виклику.

Інтеграція навчальних елементів в *UI*: Самі елементи *UI* можуть містити навчальну інформацію або бути частиною головоломки. Наприклад, кнопки можуть бути марковані математичними виразами, а не просто текстом, або візуальні елементи *UI* можуть відображати частину завдання.

Ігрофікація *UI*: Включення елементів гейміфікації безпосередньо в *UI,* таких як шкали прогресу, значки досягнень, віртуальні валюти, системи "зірок" або аватари, дозволяє посилити мотиваційний аспект та відчуття досягнення у гравця.

Ефективний *UI*/*UX* дизайн в освітніх іграх є результатом поєднання принципів ігрового дизайну та дидактичних вимог. Забезпечення зрозумілості, доступності, консистентності, чіткого зворотного зв'язку, ефективності, контролю та естетичної привабливості є ключовим для створення продукту, який не тільки навчає, а й захоплює. Правильно спроектований інтерфейс здатен зменшити когнітивне навантаження, підвищити залученість та сприяти глибшому засвоєнню навчального матеріалу, перетворюючи гру на потужний освітній інструмент

2.4.5 Проєктування інтерфейсу користувача (*UI*) та взаємодії (*UX*)

*Я*кісний *UI*/*UX* дизайн є ключовим для залучення користувачів та ефективності освітнього тренажера.

На етапі проєктування буде створено детальні макети (*wireframes* та *mockups*) для всіх основних екранів додатка:

1) Головне меню: Точки входу в режими "Вивчення", "Практика", "Конструктор", а також доступ до налаштувань та статистики.

2) Екран "Вивчення": Зона для відображення елемента, його таблиці істинності, опису та інтерактивних вхідних перемикачів для демонстрації роботи.

3) Екран "Практика":

- панель завдань: чітке відображення умови завдання (текст, таблиця істинності, схема-зразок).
- робоче поле: простір для перетягування та розміщення логічних елементів.
- панель інструментів: зручний доступ до бібліотеки логічних елементів, інструментів з'єднання/видалення, кнопок "Підказка", "Перевірити", "Скинути".
- індикатори: візуальні індикатори вхідних/вихідних сигналів, прогресу завдання.

4) Екран "Конструктор": аналогічний "Практиці", але з більшою гнучкістю (без завдань, можливість збереження/завантаження).

5) Екран "Статистика": Графіки та чисельні дані про успішність користувача.

6) Екран "Налаштування": Можливість змінювати мову, складність за замовчуванням, інші параметри.

Елементи керування (кнопки, перемикачі, повзунки) будуть розроблені з урахуванням принципів "чіткості", "доступності" та "зворотного зв'язку". Вони повинні мати чіткі візуальні стани (наприклад, при натисканні, наведенні), щоб користувач завжди розумів свої дії.

Візуалізація є ключовою для розуміння логічних процесів:

- Символи: Логічні елементи будуть представлені стандартними графічними символами (наприклад, *ANSI* або *IEC*), щоб забезпечити відповідність академічним стандартам.
- Порти: Чітке позначення вхідних та вихідних портів.
- Стани сигналів: Використання кольорів (наприклад, зелений для логічної "1", червоний для логічного "0") або анімації (пульсація) для відображення поточних логічних станів на входах, виходах та з'єднувальних лініях.
- Анімації: Плавні анімації при перетягуванні елементів, з'єднанні ліній та зміні стану сигналів, щоб зробити взаємодію динамічною та приємною.

Дизайн інтерфейсу буде інтегрований з навчальним контентом для максимальної ефективності:

- Візуальна ієрархія: Важлива інформація (умова завдання, ключові елементи схеми) буде візуально виділена.
- Контекстні підказки: Підказки будуть з'являтися в тому місці інтерфейсу, де користувач має проблеми, мінімізуючи когнітивне навантаження.
- Гейміфіковані елементи: Прогрес-бари, візуальні індикатори досягнень, анімовані "переможні" екрани при успішному вирішенні завдання.
- Інтуїтивне управління: Наприклад, використання жестових взаємодій для масштабування чи панорамування робочого поля.

РОЗДІЛ 3
РОЗРОБКА ТА ТЕСТУВАННЯ МОБІЛЬНОГО ТРЕНАЖЕРА «*LOGIC GATES*»

3.1 *BGTransparencyController.cs*

Призначення: Цей скрипт керує прозорістю фонового зображення (*Background*) в ігровому інтерфейсі. Його основна функція — плавно змінювати прозорість для візуальних ефектів, наприклад, при появі екранів програшу чи виграшу, щоб фон ставав напівпрозорим.

Детальний опис:

public SpriteRenderer bgRenderer;: Змінна для посилання на компонент *SpriteRenderer* фонового зображення. Через цей об'єкт скрипт отримує доступ до властивостей кольору та прозорості фону.

void Start(): Метод, що викликається один раз при старті скрипта. Встановлює початкову прозорість фону на 0*f* (повністю прозорий).

public void SetTransparency(*float alpha*): Публічний метод, який дозволяє іншим скриптам встановлювати значення прозорості фону. Приймає параметр *alpha* (від 0 до 1)

На рисунку 3.1 показана його структура та схематична взаємодія цього скрипта із іншими.

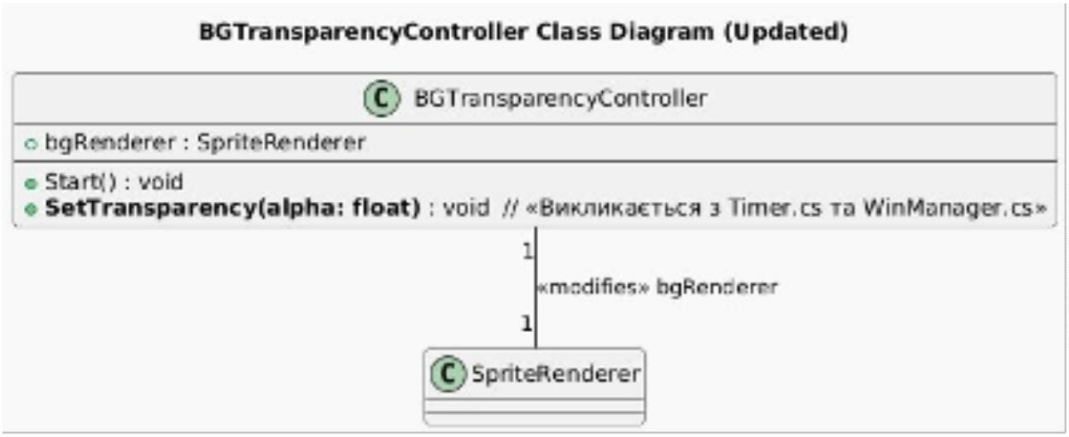

Рисунок 3.1 – Діаграма коду *BGTransparencyController*

3.2 *Button.cs*

Призначення: Простий скрипт для реалізації функціональності кнопки "Перезапустити гру".

Детальний опис:

public void restartGame(): Публічний метод, який викликається при натисканні на кнопку. Завантажує поточну активну сцену, ефективно перезапускаючи рівень.

Використовує *SceneManager*.*LoadScene*(*SceneManager*.*GetActiveScene*().*name*) для досягнення цієї мети.

На рисунку 3.2 показана його структура.

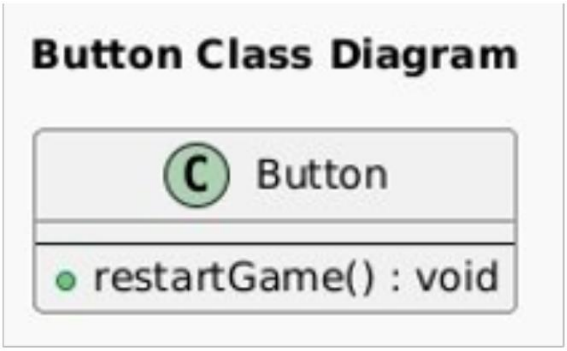

Рисунок 3.2 – Діаграма коду *Button.cs*

3.3 *ButtonWin.cs*

Призначення: Скрипт для кнопки, яка дозволяє перейти до наступного рівня після перемоги.

public void LoadNextLevel(): Публічний метод, що викликається при натисканні на кнопку.

Визначає індекс поточної сцени (*currentSceneIndex*) та розраховує індекс наступної сцени (*nextSceneIndex*).

Виконує перевірку, чи існує наступна сцена у *Build Settings* (*nextSceneIndex* < *SceneManager*.*sceneCountInBuildSettings*).

Якщо наступна сцена існує, вона завантажується за допомогою *SceneManager*.*LoadScene*(*nextSceneIndex*).

В іншому випадку виводиться попередження в консоль (*Debug.LogWarning*), якщо наступна сцена не додана.

На рисунку 3.3 показана його структура. Також на рисунку 3.4 продемонстровано екран перемоги.

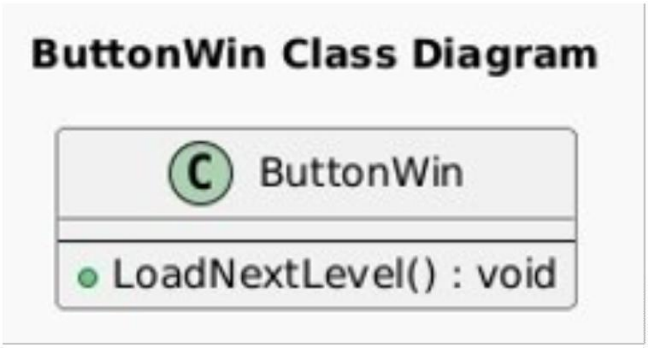

Рисунок 3.3 – Діаграма *ButtonWin*

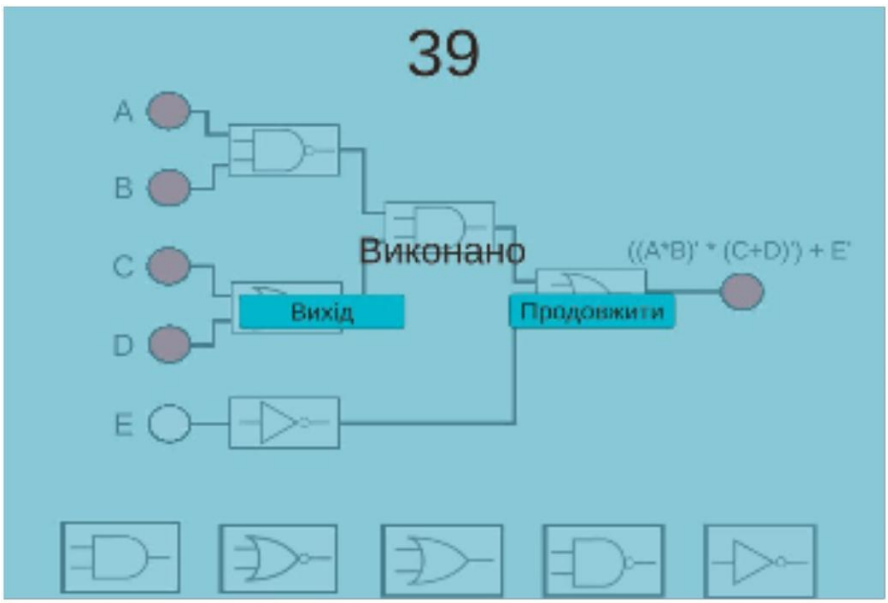

Рисунок 3.4 – Демонстрація екрану перемоги

3.4 *Drag.cs*

Призначення: Цей скрипт відповідає за логіку перетягування об'єктів, їх розміщення на ігровому полі та відповідні наслідки (штрафи за неправильне розміщення та відтворення звуку).

public float SizeX = 1f; public float SizeY = 1f;: Визначає розмір об'єкта після його правильного встановлення.

public Timer timer;: Посилання на компонент *Timer* для віднімання часу у випадку неправильного розміщення.

public GameObject crossObject;: Посилання на ігровий об'єкт, який відображає "хрестик" при неправильному розміщенні.

public AudioClip wrongPlacementSound;: (Додано) Звуковий кліп, який відтворюється при неправильному розміщенні об'єкта.

private AudioSource audioSource;: (Додано) Компонент для відтворення звукових ефектів.

private static Drag selectedObject = null;: Статична змінна, яка відстежує, який об'єкт *Drag* зараз вибраний гравцем. Це гарантує, що лише один об'єкт може бути "перетягнутим" одночасно.

private bool isPlaced = false;: Прапор, що вказує, чи був об'єкт вже розміщений. Розміщені об'єкти ігнорують подальші події перетягування.

void Awake(): (Додано) Метод, що викликається при завантаженні скрипта. Отримує або додає компонент *AudioSource* до ігрового об'єкта.

void Update(): Метод, що викликається кожен кадр. Обробляє події відпускання кнопки миші (лівої кнопки - *Input.GetMouseButtonUp*(0)).

Вибір об'єкта: Якщо миша клікає на об'єкті з цим скриптом, він стає *selectedObject*.

Правильне розміщення: Якщо *selectedObject* існує, і клік відбувається на об'єкті з таким же тегом (*hit.CompareTag*(*this.tag*)) і це не сам вибраний об'єкт, тоді створюється копія об'єкта в позиції *hit.transform.position*, розмір копії змінюється, і копія позначається як розміщена (*MarkAsPlacedExternally*()). Після цього *selectedObject* скидається.

Неправильне розміщення/Штраф: Якщо *selectedObject* існує, але клік не призводить до правильного розміщення, віднімається час від таймера (*timer.SubtractTime*(5*f*)). Також відображається "хрестик" у позиції кліка

(*ShowCrossAtMouse*()) і відтворюється звук неправильного розміщення (*audioSource*.*PlayOneShot*(*wrongPlacementSound*);).

public void MarkAsPlacedExternally(): Публічний метод, який позначає об'єкт як розміщений. Він викликається при створенні копії об'єкта після успішного перетягування. Також викликає *WinManager*.*instance*?.*RegisterPlacement*() для реєстрації успішного розміщення.

void ShowCrossAtMouse(): Відображає *crossObject* у позиції курсору миші на 1 секунду.

void HideCross(): Приховує *crossObject*.

*Vector*3 *GetMouseWorldPosition*(): Допоміжний метод для перетворення координат екрану миші в світові координати.

На рисунку 3.5 показана його структура та схематична взаємодія цього скрипта із іншими. Також на рисунку 3.6 продемонстровано спробу розташувати логічний елемент у неправильному місці

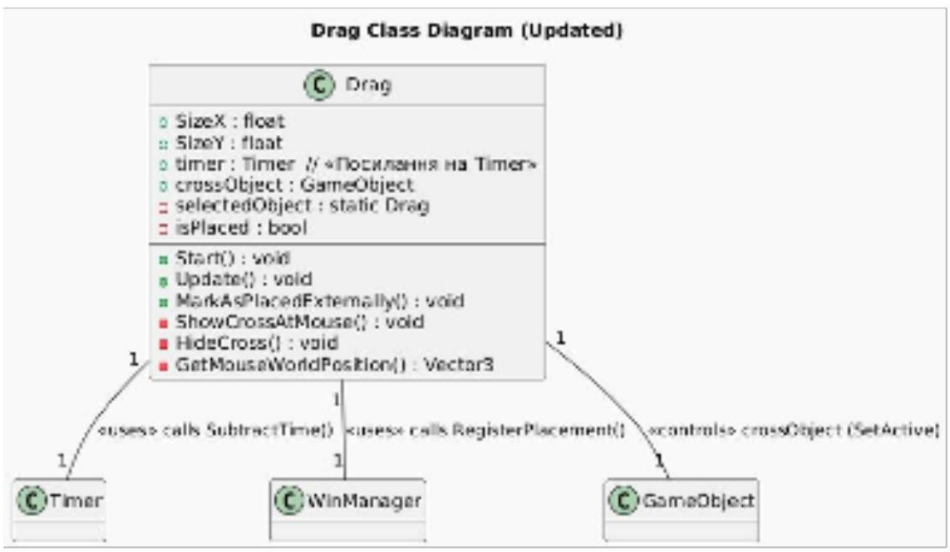

Рисунок 3.5 – Діаграма коду *Drag*

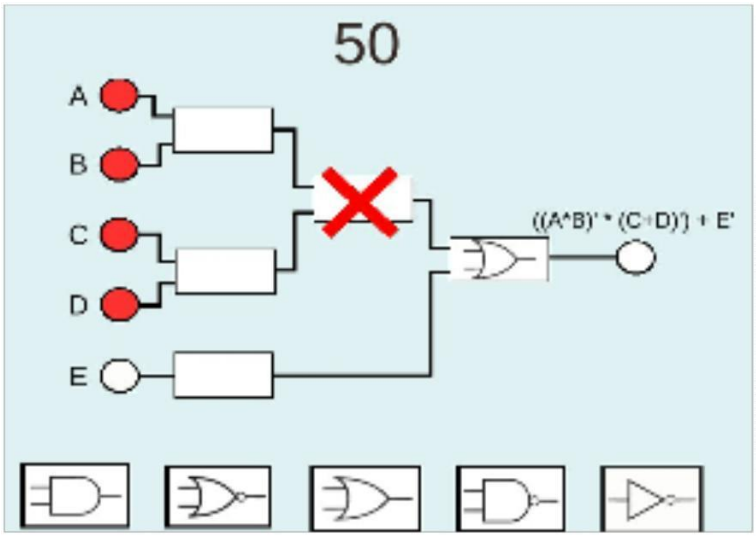

Рисунок 3.6 – Демонстрація спроби розташувати логічний елемент у неправильному місці

3.5 *ExitButton.cs*

Призначення: Простий скрипт для реалізації кнопки "Вихід з гри".

Детальний опис:

public void QuitGame(): Публічний метод, що викликається при натисканні на кнопку.

Виводить повідомлення "Вихід з гри" у консоль.

Має умовну компіляцію (*#if UNITY_EDITOR* / *#else*), що дозволяє коректно завершувати роботу гри як в редакторі *Unity* (*UnityEditor.EditorApplication.isPlaying = false*), так і в зібраній версії (*Application.Quit*()).

3.6 *LevelLoader.cs*

Призначення: Скрипт для завантаження конкретних рівнів за їхнім індексом.

Детальний опис:

public void LoadLevel(*int levelIndex*): Публічний метод, який приймає цілочисельний індекс рівня і завантажує відповідну сцену за допомогою *SceneManager.LoadScene*(*levelIndex*). На рисунку 3.7 показана його структура. Також на рисунку 3.8 продемонстровано інтерфейс обирання рівня.

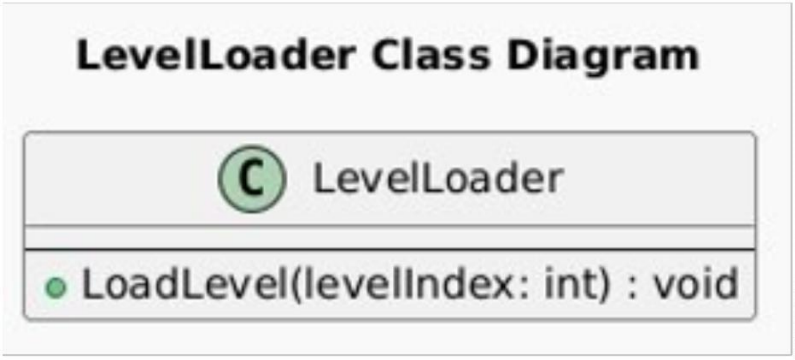

Рисунок 3.7 – Діаграма коду *LevelLoader*

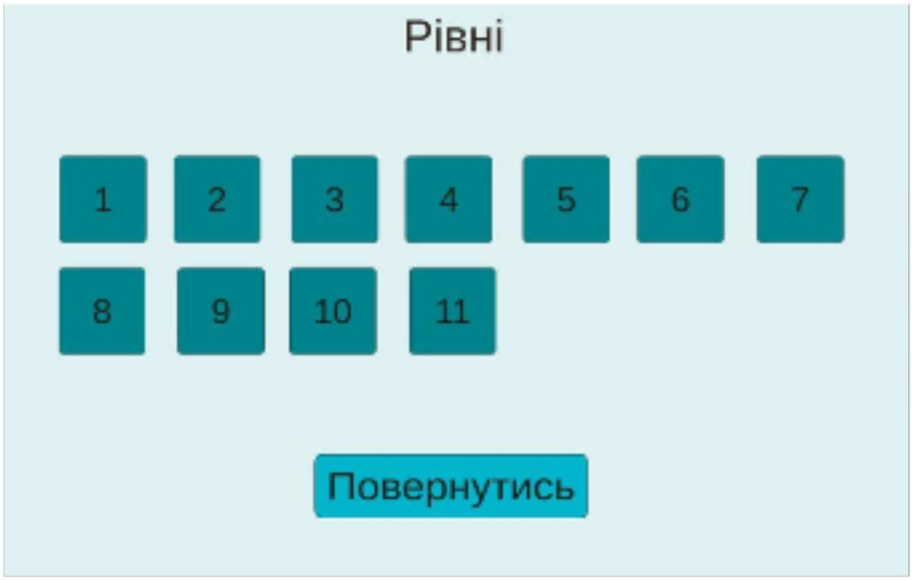

Рисунок 3.8 – Демонстрація інтерфейсу обирання рівня.

3.7 *Timer.cs*

Призначення: Цей скрипт керує відліком часу в грі, відображає його на екрані, обробляє закінчення часу та показує екран програшу.

Детальний опис:

public float timeLeft = 60*f*;: Початковий час, що залишився (за замовчуванням 60 секунд).

private float initialTime;: Зберігає початкове значення *timeLeft* для можливості скидання таймера.

public TMP_Text timerText;: Посилання на текстовий компонент *UI* (*TextMeshPro*) для відображення часу.

public GameObject loseScreen;: Посилання на ігровий об'єкт екрану програшу.

private bool isRunning = true;: Прапор, що вказує, чи працює таймер.

void Start(): Ініціалізує таймер при старті сцени: зберігає початковий час, встановлює *isRunning* в *true* та *Time.timeScale* в 1*f* (нормальна швидкість гри).

void Update(): Оновлюється кожен кадр. Якщо таймер запущений (*isRunning - true*):

Зменшує *timeLeft* на *Time.deltaTime*.

Оновлює текст таймера (*timerText.text*).

Якщо *timeLeft* досягає 0 або менше, зупиняє таймер, виводить повідомлення "Час сплинув!" і викликає *ShowLoseScreen*().

public void SubtractTime(*float amount*): Публічний метод для віднімання певної кількості часу (використовується, наприклад, скриптом *Drag* при неправильному розміщенні).

public void ResetTimer(): Скидає таймер до початкового значення і запускає його знову.

public void StopTimer(): Зупиняє таймер, встановлюючи *isRunning* в *false*.

public bool IsTimeUp(): Повертає *true,* якщо час вийшов.

private void ShowLoseScreen(): Активує *loseScreen,* викликає *BGTransparencyController* для зміни прозорості фону на 0.5*f* і зупиняє час в грі (*Time.timeScale* = 0*f*).

На рисунку 3.9 показана його структура та схематична взаємодія цього скрипта із іншими. Також на рисунку 3.10 продемонстрований екран поразки.

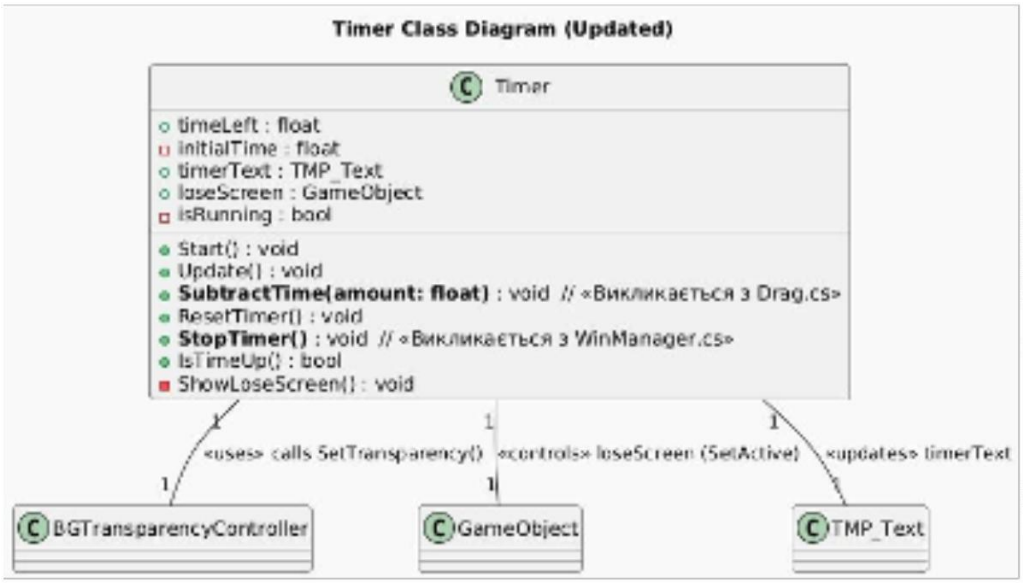

Рисунок 3.9 – Діаграма коду *Timer*

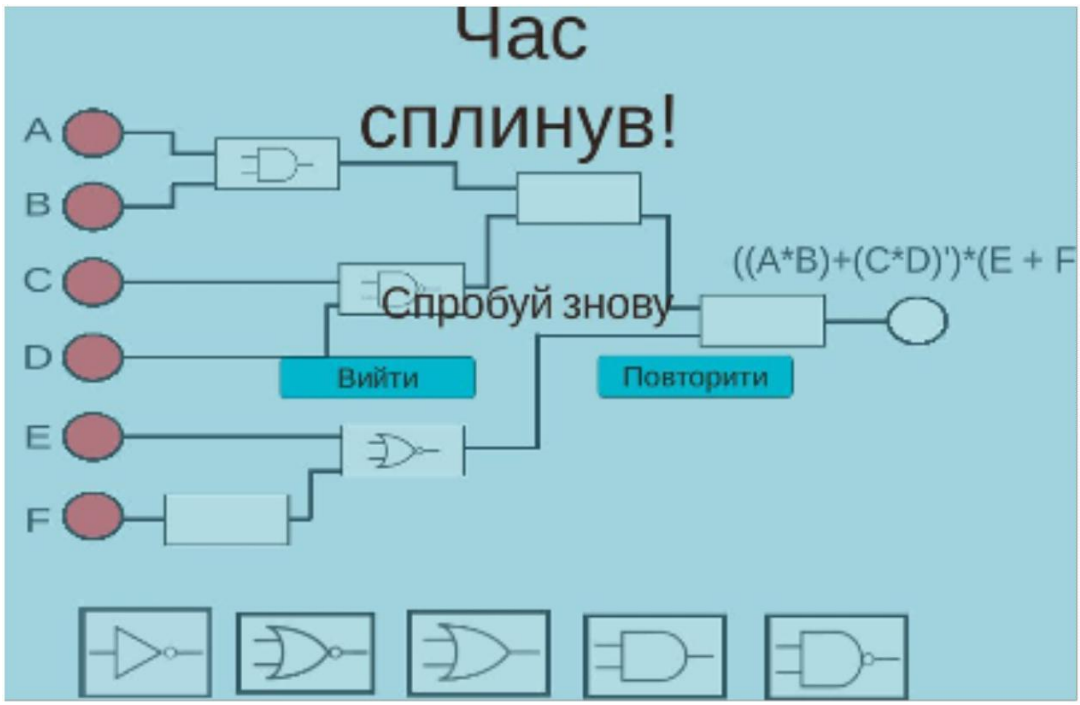

Рисунок 3.10 – Демонстрація екрану програшу

3.8 *UIManager.cs*

Призначення: Цей скрипт керує відображенням різних елементів користувацького інтерфейсу (*UI*) та ігрових об'єктів для навігації між головним меню, вибором рівня та ігровим процесом.

Детальний опис:

public GameObject mainMenuPanel;: Посилання на панель головного меню.

public GameObject levelSelectPanel;: Посилання на панель вибору рівня.

public GameObject gameUI;: Посилання на елементи *UI*, що відображаються під час гри.

public GameObject levelObjects;: Посилання на ігрові об'єкти самого рівня.

public void ShowMainMenu(): Активує панель головного меню та деактивує інші панелі/об'єкти.

public void ShowLevelSelect(): Активує панель вибору рівня та деактивує інші панелі/об'єкти.

public void StartLevel(): Активує ігровий інтерфейс та об'єкти рівня, деактивуючи меню.

public void ReturnToMenu(): Просто викликає *ShowMainMenu*(), щоб повернутися до головного меню.

На рисунку 3.11 показана його структура. Також на рисунку 3.12 продемонстровано головне меню тренажера.

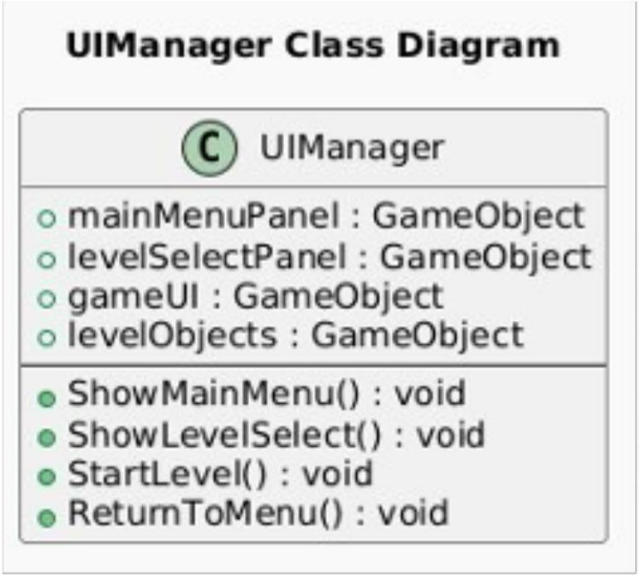

Рисунок 3.11 – Діаграма коду *UIManager*

Рисунок 3.12 – Демонстрація головного меню тренажера

3.9 *WinManager.cs*

Призначення: Цей скрипт відстежує кількість правильно розміщених об'єктів і керує логікою перемоги в грі.

Детальний опис:

public static WinManager instance;: Реалізує шаблон *Singleton*, що дозволяє легко отримати доступ до цього менеджера з будь-якого іншого скрипта.

public int requiredPlacements = 3;: Кількість об'єктів, які необхідно правильно розмістити для перемоги.

private int currentPlacements = 0;: Лічильник поточної кількості правильно розміщених об'єктів.

public GameObject winScreen;: Посилання на ігровий об'єкт екрану перемоги.

public Timer timer;: Посилання на компонент *Timer* для зупинки таймера після перемоги.

private bool hasWon = false;: Прапор, що вказує, чи гравець вже переміг.

public SpriteRenderer exitRenderer;: Посилання на *SpriteRenderer*, колір якого змінюється при перемозі (ймовірно, для підсвічування "виходу").

public AudioClip correctPlacementSound;: (Додано) Звуковий кліп, який відтворюється при правильному розміщенні об'єкта.

private AudioSource audioSource;: (Додано) Компонент для відтворення звукових ефектів.

void Awake(): Метод, що викликається при завантаженні скрипта. Реалізує *Singleton*-логіку (якщо *instance* не існує, призначає *this*, інакше знищує себе). Також отримує або додає компонент *AudioSource*.

void Start(): Ініціалізує лічильник *currentPlacements*, встановлює *hasWon* в *false* і приховує *winScreen* при старті сцени.

public void RegisterPlacement(): Публічний метод, який викликається скриптом *Drag* при успішному розміщенні об'єкта.

Збільшує *currentPlacements*.

Відтворює звук правильного розміщення (*audioSource*.*PlayOneShot*(*correctPlacementSound*);).

*Я*кщо *currentPlacements* досягає *requiredPlacements*, встановлює *hasWon* в *true* і запускає корутину *ShowWinNextFrame*().

System.Collections.IEnumerator ShowWinNextFrame(): Корутина для відображення екрану перемоги. Використовується корутина, щоб забезпечити, що зміна кольору *exitRenderer* відбудеться перед появою екрану перемоги.

- Змінює колір *exitRenderer* на червоний.
- Чекає один кадр (*yield return null*;).
- Активує *winScreen*.
- Викликає *BGTransparencyController* для зміни прозорості фону на 0.7*f*.
- Зупиняє таймер (*timer*.*StopTimer*()).
- Зупиняє час у грі (*Time*.*timeScale* = 0*f*).

На рисунку 3.13 показана його структура та схематична взаємодія цього скрипта із іншими.

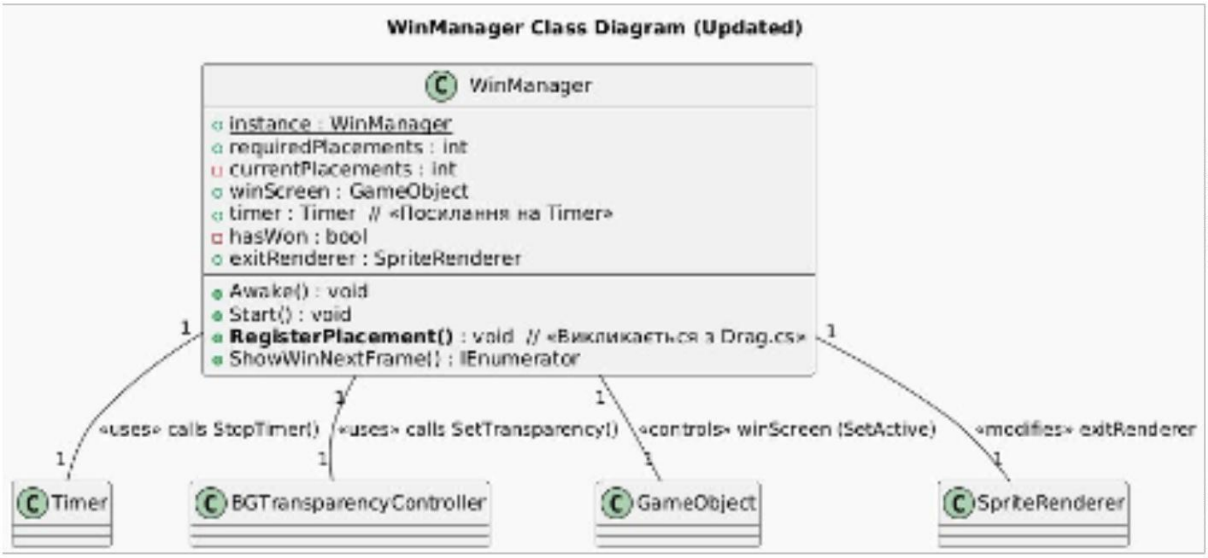

Рисунок 3.13 – Діаграма коду *WinManager*

3.10 Сценарії взаємодії користувача

Ось наведенні схеми які розділяють взаємодію взаємодію гравця з грою на декілька сегментів: Початковий потік (рисунок 3.14), Ігровий процес (рисунок 3.15) та Завершення (див рисунок 3.16)

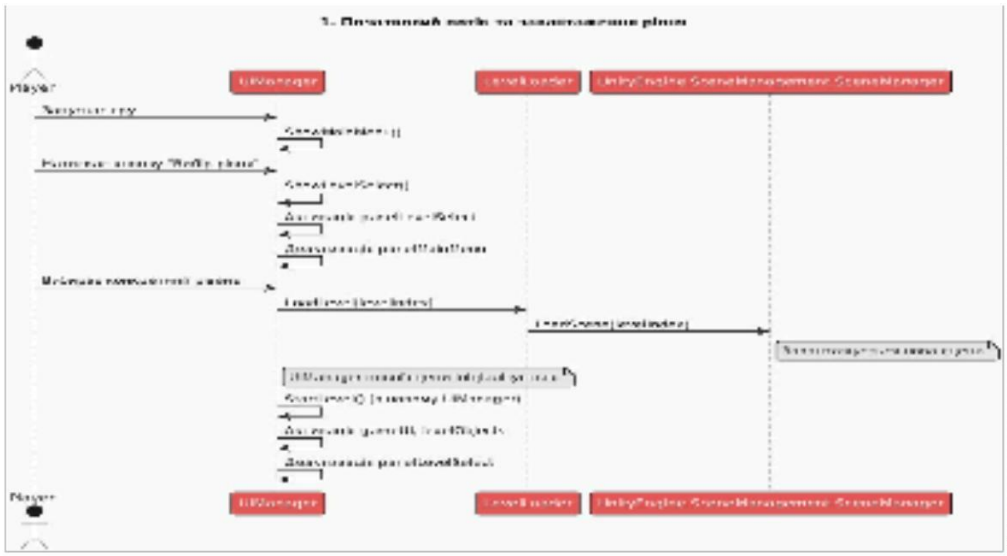

Рисунок 3.14 – Початковий потік та завантаження рівня

Рисунок 3.15 – Ігровий процес Перетягування об'єктів та таймер

Рисунок 3.16 – Умови завершення гри та *UI* кнопок

Ці скрипти працюють разом, створюючи базову ігрову логіку з елементами перетягування, таймером, системою перемоги/програшу, навігацією по меню та доданими звуковими ефектами для кращого зворотного зв'язку з гравцем.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було проведено аналіз сучасних інформаційних технологій, що застосовуються в освітньому процесі, та розроблено мобільний тренажер “*Logic Gates*”, орієнтований на підвищення ефективності навчання основам цифрової електроніки.

Інформаційні технології продемонстрували значний потенціал для трансформації освіти в таких аспектах, як інтерактивність, індивідуалізація навчання, доступність, глобальний обмін знаннями та освітня аналітика. Вони сприяють активному залученню студентів, адаптації навчального матеріалу до індивідуальних потреб, забезпечують гнучкість у здобутті знань, а також відкривають доступ до передових світових освітніх ресурсів. Аналіз даних дозволяє викладачам ефективніше коригувати навчальні програми, що сприяє покращенню якості освіти.

У практичній частині було реалізовано прототип мобільного застосунку, який виконує функції навчального тренажера. Його особливістю є поєднання простоти інтерфейсу з гейміфікованим підходом, що дозволяє знизити когнітивне навантаження на користувача та водночас підвищити мотивацію до навчання. Завдяки адаптивності рівнів, миттєвому зворотному зв’язку, логічному моделюванню та обмеженню за часом, додаток “*Logic Gates*” може бути використаний як ефективний інструмент для закріплення знань у сфері цифрової логіки.

Таким чином, застосування інформаційних технологій у поєднанні з інструментами гейміфікації відкриває нові можливості для модернізації освітнього процесу, підвищення зацікавленості студентів, розвитку критичного мислення та формування навичок *XXI* століття.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Asynchronous programming with async and await (C#) [Електронний ресурс]. Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/csharp/asynchronous-programming/async-and-await (дата звернення: 03.06.2025).

2. ASP.NET Core documentation [Електронний ресурс]. Microsoft Learn. URL: https://learn.microsoft.com/uk-ua/aspnet/core/?view=aspnetcore-8.0 (дата звернення: 03.06.2025).

3. Delegates (C# programming guide) [Електронний ресурс]. Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/delegates/ (дата звернення: 03.06.2025).

4. .NET MAUI documentation [Електронний ресурс]. Microsoft Learn. URL: https://learn.microsoft.com/uk-ua/dotnet/maui/ (дата звернення: 03.06.2025).

5. .NET on Azure documentation [Електронний ресурс]. Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/azure/ (дата звернення: 03.06.2025).

6. EdTech Evidence Exchange: [Електронний ресурс]. 2022. URL:https://edtechevidence.org(дата звернення: 03.06.2025).

7. eLearning Or Traditional Classroom Learning? Exploring The Pros And Cons. eLearning Industry: [Електронний ресурс]. URL: https://elearningindustry.com (дата звернення: 03.06.2025).

8. Global Education Monitoring Report 2023. UNESCO: [Електронний ресурс]. 2023. URL: https://unesco.org (дата звернення: 03.06.2025).

9. Impact of Digital Tools on Learning Outcomes. ResearchGate: [Електронний ресурс]. URL: (PDF) Systematic Review of Digital Tools' Impact on Primary and Secondary Education Outcomes (дата звернення: 03.06.2025).

10. LINQ (Language Integrated Query) (C#) [Електронний ресурс]. Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/ (дата звернення: 03.06.2025).

11. ML.NET documentation [Електронний ресурс]. Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/machine-learning/ (дата звернення: 03.06.2025).

12. Sweller J. Cognitive load during problem solving: Effects on learning // Cognitive Sci

13. System Namespace (.NET API browser) [Електронний ресурс]. Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/api/system?view=net-8.0 (дата звернення: 03.06.2025).

14. TIOBE Index [Електронний ресурс]. TIOBE Software. URL: https://www.tiobe.com/tiobe-index/(дата звернення: 03.06.2025).

15. Traditional Learning Vs. Online Learning. eLearning Industry : [Електронний ресурс]. URL: https://elearningindustry.com(дата звернення: 03.06.2025).

16. Unreal Engine Documentation [Електронний ресурс]. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine(дата звернення: 03.06.2025).

17. Unreal Engine : Wikipedia : [Електронний ресурс]. URL: https://en.wikipedia.org/wiki/Unreal_Engine (дата звернення: 03.06.2025).

18. Unity : Wikipedia : [Електронний ресурс]. URL: Unity (game engine) - Wikipedia (дата звернення: 03.06.2025).

19. Козлов Г. О., Осадча О. Г. Розробка мобільного тренажера “Logic Gates” для підвищення рівня пізнавальної активності здобувачів освіти під час вивчення цифрової електроніки // *Авіація і космонавтика*: тези допов. IV Міжнародна науково-практична конференція. Кривий Ріг, 2025 р. С.193-195.

ДОДАТОК А

Лістинг програми

Drag.cs

```
using UnityEngine;

public class Drag : MonoBehaviour
{
    [Header("Розмір після встановлення")]
    public float SizeX = 1f;
    public float SizeY = 1f;

    [Header("Посилання")]
    public Timer timer;
    public GameObject crossObject; // Хрестик (UI або 2D-об’єкт)
    public AudioClip wrongPlacementSound; // Звук неправильного розміщення
    private AudioSource audioSource; // AudioSource для відтворення звуків

    private static Drag selectedObject = null;
    private bool isPlaced = false;

    private void Awake()
    {
        audioSource = GetComponent<AudioSource>();
        if (audioSource == null)
        {
            audioSource = gameObject.AddComponent<AudioSource>(); // Додаємо
AudioSource, якщо його немає
        }
    }
```

```
    private void Start()
    {
        if (crossObject != null)
            crossObject.SetActive(false);
    }

    void Update()
    {
        if (isPlaced) return;

        if (Input.GetMouseButtonUp(0))
        {
            Vector3 mousePos = GetMouseWorldPosition();
            Collider2D hit = Physics2D.OverlapPoint(mousePos);

            if (hit != null && hit.gameObject == this.gameObject)
            {
                // Вибір цього об'єкта
                selectedObject = this;
                Debug.Log($"{name} вибрано");
            }
            else if (selectedObject == this && hit != null && hit.CompareTag(this.tag) &&
hit.gameObject != this.gameObject)
            {
                // Правильне місце — створюємо копію
                GameObject newCopy = Instantiate(gameObject, hit.transform.position,
Quaternion.identity);
                newCopy.transform.localScale = new Vector2(SizeX, SizeY);

                // Відмічаємо копію як встановлену
                Drag dragCopy = newCopy.GetComponent<Drag>();
```

```
            if (dragCopy != null)
            {
                dragCopy.MarkAsPlacedExternally();
            }

            selectedObject = null;
        }
        else if (selectedObject == this)
        {
            // Неправильне місце — штраф
            Debug.Log($"{name}: неправильна ціль або клацнув сам на себе");

            if (timer != null)
                timer.SubtractTime(5f);

            ShowCrossAtMouse(); // показати хрестик

            // Відтворення звуку неправильного розміщення
            if (audioSource != null && wrongPlacementSound != null)
            {
                audioSource.PlayOneShot(wrongPlacementSound);
            }

            selectedObject = null;
        }
    }
}

public void MarkAsPlacedExternally()
{
```

```
    isPlaced = true;
    if (crossObject != null)
        crossObject.SetActive(false);

    Debug.Log($"{name} копію встановлено");
    WinManager.instance?.RegisterPlacement();
}

void ShowCrossAtMouse()
{
    if (crossObject != null)
    {
        Vector3 mouseWorldPos = GetMouseWorldPosition();
        mouseWorldPos.z = 0f;

        // Якщо це UI-елемент: використай Input.mousePosition
        if (crossObject.GetComponent<CanvasRenderer>() != null)
        {
            crossObject.transform.position = Input.mousePosition;
        }
        else
        {
            crossObject.transform.position = mouseWorldPos;
        }

        crossObject.SetActive(false); // оновлення позиції
        crossObject.SetActive(true);

        CancelInvoke(nameof(HideCross));
        Invoke(nameof(HideCross), 1f);
```

```
        }
    }

    void HideCross()
    {
        if (crossObject != null)
            crossObject.SetActive(false);
    }

    Vector3 GetMouseWorldPosition()
    {
        Vector3 pos = Camera.main.ScreenToWorldPoint(Input.mousePosition);
        pos.z = 0f;
        return pos;
    }
}
```

WinManager.cs

```
using System;
using UnityEngine;

public class WinManager : MonoBehaviour
{
    public static WinManager instance;

    public int requiredPlacements = 3;
    private int currentPlacements = 0;

    public GameObject winScreen;
    public Timer timer;
```

```
private bool hasWon = false;
public SpriteRenderer exitRenderer;

void Awake()
{
    if (instance == null) instance = this;
    else Destroy(gameObject);
}

void Start()
{
    currentPlacements = 0;
    hasWon = false;
    if (winScreen != null) winScreen.SetActive(false);
}

public void RegisterPlacement()
{
    if (hasWon) return;

    currentPlacements++;
    Debug.Log("Зареєстровано: {currentPlacements}/{requiredPlacements}");

    if (currentPlacements >= requiredPlacements)
    {
        hasWon = true;
        StartCoroutine(ShowWinNextFrame());
    }
}
```

```
    System.Collections.IEnumerator ShowWinNextFrame()
    {
        if (exitRenderer != null)
        {
            exitRenderer.color = new Color32(0xFF, 0x31, 0x31, 0xFF); // FF3131
        }

        yield return null;

        // Потім виводимо екран перемоги
        if (winScreen != null) winScreen.SetActive(true);

        // Прозорість BG
        FindObjectOfType<BGTransparencyController>()?.SetTransparency(0.7f);

        if (timer != null) timer.StopTimer();
        Time.timeScale = 0;
    }

}
```

LevelLoader.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class LevelLoader : MonoBehaviour
{
    public void LoadLevel(int levelIndex)
    {
        SceneManager.LoadScene(levelIndex);
    }
```

```
}
```

UIManager.cs

```
using UnityEngine;

public class UIManager : MonoBehaviour
{
    public GameObject mainMenuPanel;
    public GameObject levelSelectPanel;
    public GameObject gameUI;
    public GameObject levelObjects;

    public void ShowMainMenu()
    {
        mainMenuPanel.SetActive(true);
        levelSelectPanel.SetActive(false);
        gameUI.SetActive(false);
        levelObjects.SetActive(false);
    }

    public void ShowLevelSelect()
    {
        mainMenuPanel.SetActive(false);
        levelSelectPanel.SetActive(true);
        gameUI.SetActive(false);
        levelObjects.SetActive(false);
    }

    public void StartLevel()
    {
```

```
        mainMenuPanel.SetActive(false);
        levelSelectPanel.SetActive(false);
        gameUI.SetActive(true);
        levelObjects.SetActive(true);
    }

    public void ReturnToMenu()
    {
        ShowMainMenu();
    }
}
```

Timer.cs

```
using UnityEngine;
using UnityEngine.UI;
using TMPro;



public class Timer : MonoBehaviour
{
    public float timeLeft = 60f;
    private float initialTime;
    public TMP_Text timerText;
    public GameObject loseScreen;
    private bool isRunning = true;

    void Start()
    {
        initialTime = timeLeft;
```

```
    isRunning = true;
    Time.timeScale = 1f;
}

void Update()
{
    if (!isRunning) return;

    if (timeLeft > 0)
    {
        timeLeft -= Time.deltaTime;
        timerText.text = Mathf.Ceil(timeLeft).ToString();
    }
    else
    {
        timeLeft = 0;
        timerText.text = "Час сплинув!";
        isRunning = false;
        ShowLoseScreen();
    }
}

public void SubtractTime(float amount)
{
    timeLeft -= amount;
    if (timeLeft < 0)
    {
        timeLeft = 0;
        isRunning = false;
    }
```

```
}

public void ResetTimer()
{
    timeLeft = initialTime;
    isRunning = true;
    Time.timeScale = 1f;
}

public void StopTimer()
{
    isRunning = false;
}

public bool IsTimeUp()
{
    return timeLeft <= 0;
}
private void ShowLoseScreen()
{
    if (loseScreen != null)
    {
        loseScreen.SetActive(true);

        // Зміна прозорості BG
        FindObjectOfType<BGTransparencyController>()?.SetTransparency(0.5f);

        Time.timeScale = 0f;
    }
}
```

```
}
```

ButtonWin.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class ButtonWin : MonoBehaviour
{
    public void LoadNextLevel()
    {
        int currentSceneIndex = SceneManager.GetActiveScene().buildIndex;
        int nextSceneIndex = currentSceneIndex + 1;

        // Перевірка: чи існує наступна сцена
        if (nextSceneIndex < SceneManager.sceneCountInBuildSettings)
        {
            SceneManager.LoadScene(nextSceneIndex);
        }
        else
        {
            Debug.LogWarning("Наступна сцена не додана до Build Settings.");
        }
    }
}
```

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

РЕЦЕНЗІЯ

на кваліфікаційну роботу

випускника спеціальності: 123 «Комп'ютерна інженерія»

відділення: комп'ютерної та програмної інженерії

циклова комісія: комп'ютерних систем та мереж

Герман КОЗЛОВ
(ім'я, прізвище)

1. Актуальність теми: тема кваліфікаційної роботи «Мобільний тренажер «Logic Gates» є актуальною, оскільки мобільні додатки для навчання стають все більш популярними та ефективними.

2. Кваліфікаційна робота відповідає темі, затвердженій наказом.

3. Завдання на виконання кваліфікаційної роботи виконано у повному обсязі.

4. У роботі проведено детальний аналіз предметної області та існуючих аналогів, що дозволило визначити сильні та слабкі сторони подібних рішень та сформулювати власні вимоги до функціоналу тренажера. Розроблений додаток має інтуїтивно зрозумілий інтерфейс, функціонал реалізовано коректно, що дозволяє ефективно моделювати роботу логічних вентилів.

5. Пояснювальна записка до кваліфікаційної роботи написана логічно, послідовно та містить вичерпну інформацію про всі етапи роботи – від постановки задачі до тестування та впровадження. Матеріал викладено грамотною технічною мовою. Оформлення роботи відповідає встановленим вимогам.

6. Хоча функціонал тренажера є достатнім для базового вивчення, можливо, варто було б розглянути інтеграцію більш складних завдань або режимів симуляції для просунутих користувачів.

7. Кваліфікаційна робота заслуговує оцінку «відмінно».

Рецензент викладач вищої категорії
(науковий ступінь, посада)

«____»_____ 2025 р. _____ Світлана ДАЦЕНКО
(підпис) (ім'я, прізвище)

З рецензією ознайомлений _____ Герман КОЗЛОВ
(підпис) (ім'я, прізвище)