Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(навчально-науковий інститут)

факультет інформаційних технологій
(факультет)

Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи ступеня бакалавра

(бакалавра, магістра)

Здобувача вищої освіти Гуліча Ігора Петровича
(ПІБ)

академічної групи 126-21-2
(шифр)

спеціальності 126 «Інформаційні системи та технології»
(код і назва спеціальності)

за освітньо-професійною програмою «Інформаційні системи та технології»

(офіційна назва)

на тему Розробка кросплатформної інтерактивної навчальної платформи для НУШ

(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	к.т.н., доц. Соколова Н.О.			
розділів:				

Рецензент	к.т.н., доц.каф ПЗКС Клименко А.В.			

Нормоконтролер	проф.Коротенко Г.М.			

Дніпро
2025

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних технологій та комп'ютерної інженерії
(повна назва)

_____ __В.В. Гнатушенко__
(підпис) (ініціали та прізвище)

«_____» _____20___року

ЗАВДАННЯ

на кваліфікаційну роботу
ступеня _бакалавра_

(бакалавра, магістра)

здобувача вищої освіти ___Гуліча І.П.____ **академічної групи**___126-21-2____
(прізвище та ініціали) (шифр)

спеціальності_____126 «Інформаційні системи та технології»_____

спеціалізації за освітньою-професійною програмою_____
(за наявності)

на тему Розробка кросплатформної інтерактивної навчальної платформи для НУШ

затверджену наказом ректора НТУ «Дніпровська політехніка» від 05.05.2025 № 336-с

Розділ	Зміст	Термін виконання
Розділ 1. Теоретичні засади створення інтерактивних навчальних платформ у контексті НУШ	1. Аналіз вимог НУШ до цифрових освітніх рішень. 2. Вивчення сучасних підходів до інтерактивного навчання. 3. Порівняльний аналіз ігрових рушіїв (Unity, Unreal Engine, Godot). 4. Обґрунтування вибору рушія Godot. 5. Аналіз прикладів реалізації подібних освітніх застосунків.	03.02.2025 – 30.04.2025
Розділ 2. Проєктування та реалізація інтерактивного навчального музею у Godot	1. Розробка концепції та структури навчального додатка у форматі 2.5D. 2. Проєктування сцени, керованого персонажа, зон взаємодії. 3. Реалізація інтерфейсу користувача та логіки взаємодії з об’єктами. 4. Інтеграція мультимедійного контенту (звук, відео, ілюстрації). 5. Тестування функціоналу та підготовка інструкції користувача.	03.02.2025 – 03.06.2025

Завдання видано _____ ___Соколова Н. О.
(підпис керівника) (ініціали та прізвище)

Дата видачі 03.02.2025

Дата подання до екзаменаційної комісії _____

Прийнято до виконання _____ _____Гуліч І.П.____
(підпис здобувача вищої освіти) (ініціали та прізвище)

РЕФЕРАТ

Пояснювальна записка: 87 стор., 32 рис., 4 табл., 1 додаток, 22 джерела. Об’єкт розробки: програмне забезпечення для створення інтерактивного освітнього музею.

Мета кваліфікаційної роботи: створення кросплатформної інтерактивної навчальної платформи з використанням рушія Godot, орієнтованої на потреби Нової української школи.

У вступі обґрунтовано актуальність цифрових освітніх рішень і наведено аналіз ігрових рушіїв. У першому розділі висвітлено теоретичні основи: визначено поняття цифрового музею, особливості 2D, 2.5D, 3D-середовищ, розглянуто можливості Godot, а також освітню цінність гейміфікації та інтерактивності. Другий розділ присвячено реалізації навчального музею: створено структуру сцен з гравцем, колізіями, зонами Area3D; реалізовано логіку взаємодії, мультимедійні елементи, інтерфейс і локалізацію; проведено тестування на різних платформах.

Практичне значення роботи полягає у створенні доступного, адаптивного й масштабованого інструменту для інтерактивного подання навчального матеріалу, який може бути використаний у школах, гуртках та міжпредметних освітніх проєктах.

Список ключових слів: ІНТЕРАКТИВНИЙ МУЗЕЙ, 2.5D, РУШІЙ GODOT, ЦИФРОВА ОСВІТА, СЦЕНА, СКРИПТ, КАМЕРА, AREA3D, ІНТЕРФЕЙС, НОВА УКРАЇНСЬКА ШКОЛА.

ABSTRACT

Explanatory note: 87 pages, 32 figures, 4 tables, 1 appendix, 22 sources. Object of development: software for creating an interactive educational museum. Purpose of the qualification work: development of a cross-platform interactive learning environment using the Godot engine, tailored to the needs of the New Ukrainian School.

The introduction substantiates the relevance of digital educational solutions and analyzes existing game engines. The first section presents the theoretical framework: the concept of a digital museum, characteristics of 2D, 2.5D, and 3D environments, features of the Godot engine, and the pedagogical value of gamification and interactivity. The second section focuses on the implementation of the virtual museum: scene structure with player, collisions, Area3D zones; implementation of interaction logic, multimedia, user interface, and localization; testing across multiple devices.

The practical significance lies in the development of an accessible, adaptable, and scalable educational tool for interactive knowledge delivery, applicable in schools, extracurricular education, and interdisciplinary learning.

List of keywords: INTERACTIVE MUSEUM, 2.5D, GODOT ENGINE, DIGITAL EDUCATION, SCENE, SCRIPT, CAMERA, AREA3D, USER INTERFACE, NEW UKRAINIAN SCHOOL.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

2D - Two-Dimensional (двовимірна графіка)

2.5D - Two and a Half Dimensional (проміжний формат між 2D та 3D)

3D - Three-Dimensional (тривимірна графіка)

API - Application Programming Interface (інтерфейс прикладного програмування)

AR - Augmented Reality (доповнена реальність)

XR - Extended Reality (розширена реальність)

UI - User Interface (інтерфейс користувача)

UX - User Experience (користувацький досвід)

ПК - Персональний комп’ютер

НУШ - Нова українська школа

СУБД - Система управління базами даних

GDScript - Godot Script (мова програмування в рушії Godot)

HTML5 - HyperText Markup Language v5 (мова розмітки для веб)

FPS - Frames Per Second (кількість кадрів за секунду)

VR - Virtual Reality (віртуальна реальність)

STEM - Science, Technology, Engineering, Mathematics (освітній напрям)

MIT - Massachusetts Institute of Technology (також: тип ліцензії MIT)

AI - Artificial Intelligence (штучний інтелект)

ВСТУП

Цифровізація освіти є одним із пріоритетів реформування освітньої галузі в Україні, зокрема в межах реалізації концепції Нової української школи (НУШ). НУШ акцентує увагу на розвитку ключових компетентностей учнів, зокрема критичного мислення, уміння співпрацювати, вміння навчатися упродовж життя, а також застосуванні сучасних інформаційно-комунікаційних технологій у навчальному процесі. Саме в цьому контексті інтерактивні освітні платформи відіграють важливу роль як засіб залучення учнів до активного пізнання, взаємодії з контентом і розвитку міждисциплінарних зв'язків.

Сьогодні інтерактивні інструменти навчання - це вже не лише мультимедійні презентації чи тестові системи, а повноцінні програмні продукти, які поєднують гейміфікацію, адаптивне навчання, візуалізацію та можливість моделювання складних процесів у зрозумілій формі. Інтерактивні навчальні додатки перетворюють пасивне засвоєння матеріалу на активне конструювання знань, що цілком відповідає підходам, закладеним у стандарті НУШ.

Одним із викликів, з якими стикаються розробники такого програмного забезпечення, є потреба у доступних, гнучких та водночас потужних технічних засобах для реалізації інтерактивного контенту. Традиційно створення освітніх платформ вимагало значних ресурсів, проте з появою ігрових рушіїв з відкритим кодом, таких як Godot, відкрилися нові можливості для освітніх ініціатив, навіть за обмеженого бюджету. Godot відзначається простотою освоєння, кросплатформеністю, підтримкою 2D і 3D-графіки, що робить його вдалим вибором для створення навчальних симуляцій, мініігор, віртуальних музеїв тощо. Більш того, завдяки схожості мови сценаріїв GDScript із Python, Godot можна використовувати не лише як інструмент для створення навчального контенту, а й як освітнє середовище для навчання програмуванню. Учні, які ознайомляться з основами GDScript у навчальному додатку, зможуть надалі легше переходити до вивчення більш складних мов, розуміючи основи логіки, умовних операторів, циклів, структурованого кодування.

Актуальність теми визначається необхідністю створення доступних українських рішень для цифрового освітнього простору, які враховують специфіку шкільної освіти, технічні обмеження багатьох навчальних закладів та потребу в інтерактивних формах навчання.

Мета роботи:

Розробити кросплатформну інтерактивну навчальну платформу з використанням рушія Godot, що відповідатиме вимогам НУШ, буде доступною для використання в умовах обмежених технічних ресурсів та забезпечить активну залученість учнів до навчального процесу.

Завдання роботи:

- Провести аналіз сучасних підходів до створення інтерактивних навчальних додатків у контексті НУШ.
- Дослідити можливості сучасних рушіїв (Unity, Unreal, Godot) для створення навчального контенту.
- Обґрунтувати вибір рушія Godot для створення освітньої платформи.
- Розробити прототип кросплатформного інтерактивного додатку в середовищі Godot.
- Продемонструвати функціональність основних елементів системи: навігація, UI, інтерактивність, мультимедійний контент.
- Провести оцінку відповідності платформи педагогічним та технічним критеріям.

Об’єкт дослідження:

Процес використання цифрових технологій для реалізації інтерактивного навчання в початковій і середній школі.

Предмет дослідження:

Методи, технології та інструменти створення кросплатформних інтерактивних навчальних додатків з використанням рушія Godot.

РОЗДІЛ 1
АНАЛІЗ СТАНУ ОБЛАСТІ РІШЕННЯ ЗАВДАННЯ

1.1 Сучасні підходи до цифрового навчання в новій українській школі

Реформа української освіти, відома під назвою «Нова українська школа» (НУШ), передбачає глибоку трансформацію змісту, методів і форм навчання. Одним із ключових напрямів цієї трансформації є впровадження цифрових технологій в освітній процес. Сучасне покоління учнів - це «цифрові аборигени», які з раннього віку активно використовують гаджети, мобільні застосунки, відеоігри й соціальні платформи. Це створює як нові виклики для системи освіти, так і нові можливості, зокрема для розробки інтерактивних навчальних середовищ.

Цифрове навчання у рамках НУШ базується на концепції компетентнісного підходу, що передбачає формування в учнів здатностей самостійно здобувати знання, працювати з інформацією, вирішувати реальні задачі та ефективно взаємодіяти з іншими. Таке навчання передбачає не лише оволодіння фактичними знаннями, але й розвиток критичного мислення, креативності, ініціативності. Досягнути цього без використання сучасних цифрових інструментів практично неможливо.

У Державному стандарті початкової освіти за 2018 рік прямо вказа на необхідність формувати цифрову компетентність як одну з десяти ключових. Це означає, що школярі мають не просто вміти користуватися пристроями, а й свідомо працювати з цифровими ресурсами, дотримуватись цифрової етики, розуміти основи програмування, мати навички безпечного користування інтернетом.

На практиці реалізація цифрової трансформації передбачає використання як готових освітніх платформ (типу «Всеукраїнська школа онлайн», «Мій клас», «Classtime» тощо), так і створення унікальних локальних цифрових рішень. Особливої актуальності набувають розробки, адаптовані до

українських реалій: врахування мовного контексту, освітніх програм, вікових особливостей учнів. Саме в цьому напрямі можуть бути корисні ігрові рушії, які дозволяють створювати візуально привабливі, гнучкі та легко адаптовані додатки для різних вікових груп.

Розробка інтерактивних освітніх застосунків не повинна розглядатись лише як інструмент гейміфікації. Успішні приклади таких додатків демонструють, що вони здатні формувати мотивацію до навчання, підвищувати рівень засвоєння інформації, а також слугувати інструментом для діагностики знань, інклюзивної освіти та міждисциплінарного навчання.

1.1.1 Принципи інтерактивного навчання та їх реалізація в НУШ

Під впливом цифрової трансформації, що охоплює всі сфери суспільного життя, українська освіта також зазнає змін, головною метою яких є підвищення якості навчання та формування ключових компетентностей учнів. Особливе місце у цих змінах займає інтерактивне навчання - підхід, що базується на активній участі здобувачів освіти в освітньому процесі. В умовах реалізації концепції Нової української школи (НУШ), інтерактивність стає не додатковою опцією, а невід'ємним компонентом сучасного уроку.

НУШ визначає, що сучасне навчання має бути діяльнісним [1], компетентнісним та орієнтованим на інтереси дитини. Це означає, що освітній процес має будуватися на залученні учнів до досліджень [2], обговорень, творчих завдань і командної роботи. Застосування інтерактивних методів дає можливість учням не лише засвоювати нові знання, а й застосовувати їх у практичних ситуаціях, рефлексувати над власним досвідом, розвивати соціальні навички, критичне мислення, здатність працювати в команді.

Ключові принципи інтерактивного навчання, які активно впроваджуються в освітню практику відповідно до вимог НУШ:

- **Діяльнісний підхід**: учні виконують практичні дії, пов'язані з предметом вивчення. Це можуть бути досліди, моделювання ситуацій,

міні-проєкти, інтерактивні вправи тощо. Цей підхід відповідає принципу «навчання через дію» та сприяє формуванню стійких знань і навичок.

- **Співпраця та партнерство**: роль учителя змінюється - він більше не джерело істини, а фасилітатор, наставник, який організовує взаємодію між учнями. Це дає можливість учням навчатися один в одного, обмінюватися досвідом, формувати навички взаємоповаги та командної роботи.
- **Індивідуалізація та адаптивність**: враховуються рівень підготовки, інтереси, потреби й особливості кожного учня. Це реалізується через адаптивні платформи, індивідуальні завдання, зміну темпу та рівня складності матеріалу.
- **Рефлексивність**: кожен учасник навчання має змогу оцінити свої дії, зробити висновки, побачити власний прогрес. Це досягається за допомогою ведення щоденників рефлексії, участі в дискусіях, обговорення результатів спільної діяльності.
- **Зворотний зв'язок у режимі реального часу**: дає можливість учню відразу дізнатися про результат виконання завдання, а вчителю - вчасно скорегувати процес. Більшість цифрових платформ автоматично генерують фідбек, аналізуючи відповіді учня.

На практиці інтерактивні методи реалізуються як за допомогою класичних інструментів [3] (робота в групах, дебати, кейс-методи), так і через сучасні технології. Особливої популярності набули такі платформи, як Kahoot, ClassDojo, Quizizz [4], які дозволяють проводити вікторини, опитування та навчальні ігри в реальному часі. Серед українських рішень - платформи «МійКлас», «Всеукраїнська школа онлайн», «На Урок», що пропонують інтерактивні вправи відповідно до навчальної програми МОН.

У деяких школах уже реалізовано підходи з використанням віртуальної та доповненої реальності, наприклад, 3D-тури історичними об'єктами чи біологічними лабораторіями. Такі рішення активно підтримуються проєктами

грантової підтримки на кшталт «Digital Learning», а також місцевими ініціативами.

Застосування інтерактивного навчання не тільки сприяє якісному засвоєнню знань, а й значно покращує мотивацію учнів. Усі ці аспекти відповідають духу НУШ та формують підґрунтя для інтеграції сучасних інструментів, зокрема ігрових рушіїв, у навчальний процес, що детальніше буде розглянуто в наступних підрозділах.

1.1.2 Використання гейміфікації в освіті

Гейміфікація (англ. *gamification*) - це процес застосування ігрових елементів та механік у неігровому середовищі з метою підвищення мотивації, залученості та ефективності діяльності. В освіті гейміфікація стає не лише модним трендом, а дієвим інструментом, здатним трансформувати традиційне навчання у більш динамічний, цікавий і результативний процес. Такий підхід особливо добре узгоджується з вимогами Нової української школи, що орієнтована на розвиток учня як активного, креативного і самостійного суб’єкта навчання.

Однією з основних переваг гейміфікації є її здатність утримувати увагу учнів [5]. Завдяки наявності чітких цілей, рівнів складності, системи балів і нагород, гейміфіковані завдання перетворюють навчання на цікаву пригоду. Досвід показує, що навіть складний або «нудний» матеріал сприймається легше, якщо він подається в ігровій формі. Це особливо актуально в молодшій школі, де навчальний процес часто ґрунтується на емоційній складовій.

В освітньому середовищі України вже є численні приклади ефективного впровадження гейміфікації. Зокрема, платформи Classcraft, Kahoot, Quizizz широко використовуються як для проведення інтерактивних уроків, так і для організації самостійної роботи учнів. Вони дозволяють створювати тести у форматі змагань, а також давати зворотний зв’язок у режимі реального часу. Це відповідає підходу до формувального оцінювання, який активно впроваджується в НУШ.

Серед українських рішень, орієнтованих на гейміфікацію навчання [6], слід відзначити:

- **«На Урок»[7]** - платформа з інтерактивними тестами, рейтингами та сертифікатами;
- **«МійКлас»[8]** - адаптивні завдання з автоматичною перевіркою, підказками та нагородами;
- **Edugames**,**[9] Learning.ua**,**[10] GIOS[11]** - ресурси з ігровими вправами з математики, української мови, логіки тощо, які дозволяють працювати учням у власному темпі.

У контексті впровадження гейміфікації в НУШ важливими є такі елементи гейміфікації, які активно застосовуються в цифрових середовищах:

- **Бали й нагороди** - стимулюють старанність і регулярність виконання завдань;
- **Рівні складності** - забезпечують адаптивність контенту під здібності учня;
- **Віртуальні аватари та профілі** - сприяють емоційному залученню;
- **Ігрові сценарії** - розвивають критичне мислення й творчість через сюжетне навчання.

Багато сучасних вчителів у НУШ впроваджують ці елементи вручну, адаптуючи класичні ігри або створюючи власні навчальні пригоди. Але з розвитком цифрових інструментів з’являється можливість створювати власні гейміфіковані додатки - саме для цього дедалі частіше використовуються ігрові рушії, зокрема Godot.

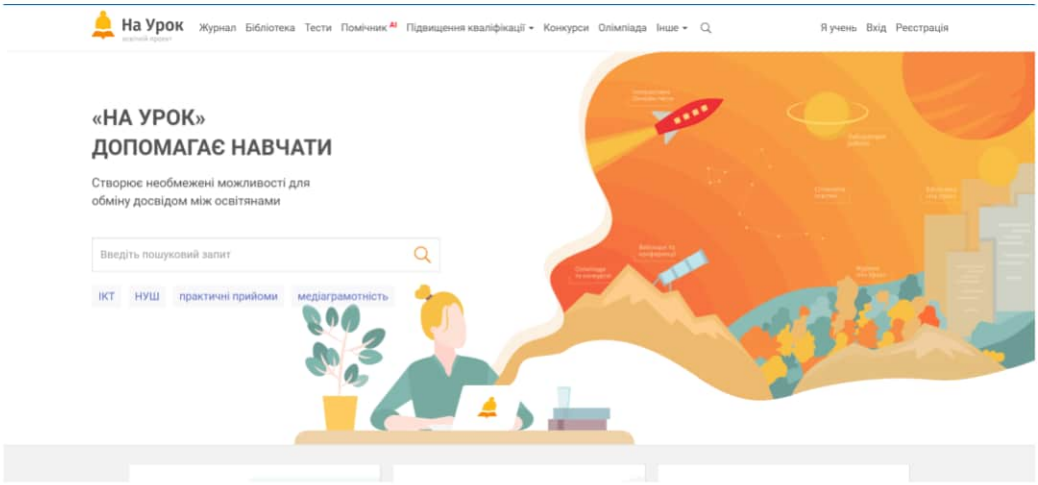
Застосування таких рушіїв дозволяє реалізувати повноцінну логіку навчальної гри: наприклад, учень може досліджувати віртуальне середовище, виконувати завдання, взаємодіяти з навчальними NPC (ігровими персонажами), які пояснюють матеріал чи проводять вікторини. Усе це дає змогу наблизити освітній процес до формату, звичного для сучасного школяра.

Таким чином, гейміфікація не просто підвищує мотивацію - вона створює нову парадигму навчання, в якій учень не отримує знання пасивно, а самостійно здобуває їх через взаємодію, гру та досвід. Цей підхід повністю відповідає цілям НУШ і є перспективним напрямом розвитку цифрової освіти в Україні.

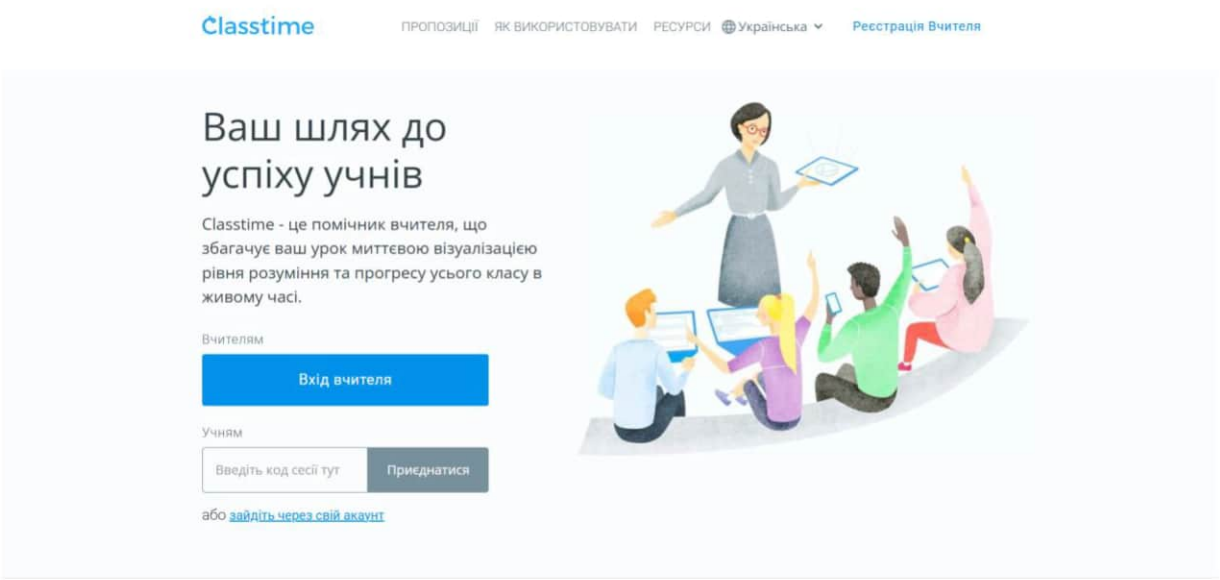
1.1.3 Приклади цифрових практик в українських школах

У межах реалізації концепції Нової української школи дедалі більше навчальних закладів в Україні інтегрують цифрові технології у повсякденний освітній процес. Практика показує, що ефективне поєднання традиційного навчання з інноваційними цифровими рішеннями [12] дозволяє значно підвищити зацікавленість учнів, індивідуалізувати навчання, розвивати навички самостійного пошуку інформації та критичного мислення.

Одним із поширених інструментів в українських школах є інтерактивні онлайн-платформи для створення та проходження тестів, такі як **“На урок”** (рис. 1.1, а), **Classtime** (рис. 1.1, б), **LearningApps** (рис. 1.1, в), **Kahoot** (рис. 1.1, г), що дозволяють учителям готувати динамічні завдання з миттєвою перевіркою. Такі інструменти часто використовуються під час підсумкових уроків, для перевірки знань у форматі гри, а також для дистанційного навчання.



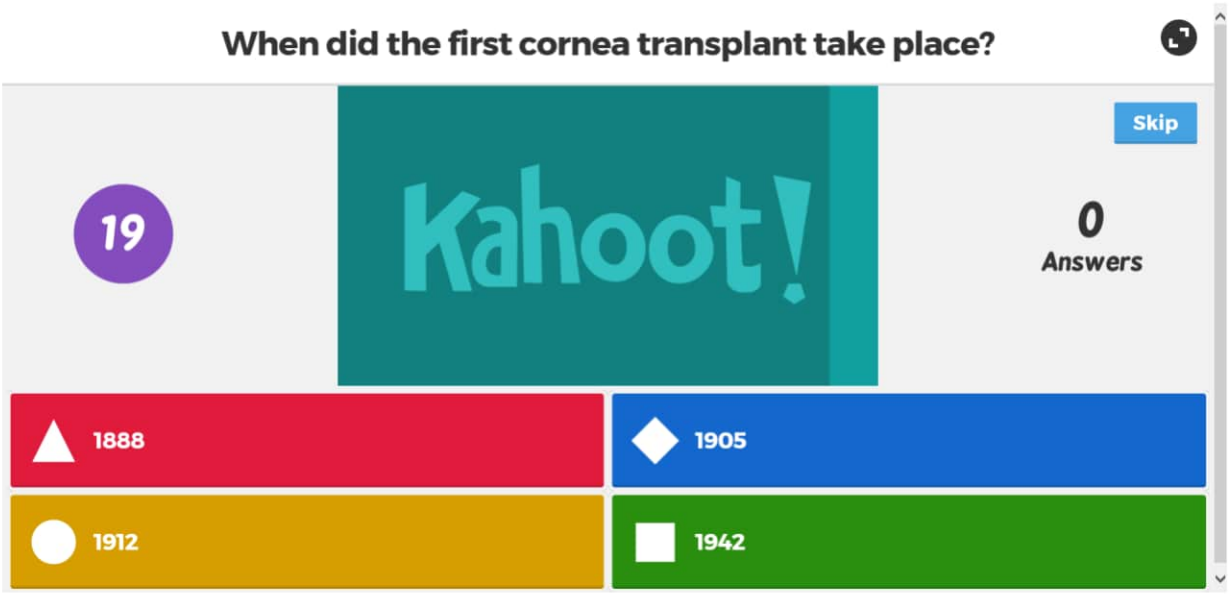
а)


Classtime
ПРОПОЗИЦІЇ
ЯК ВИКОРИСТОВУВАТИ
РЕСУРСИ
Українська
Реєстрація Вчителя
Ваш шлях до успіху учнів
Classtime - це помічник вчителя, що збагачує ваш урок миттєвою візуалізацією рівня розуміння та прогресу усього класу в живому часі.
Вчителям
Вхід вчителя
Учням
Введіть код сесії тут
Приєднатися
або зайдіть через свій акаунт

б)


LearningApps.org
Українська
Перегляд вправ
Створення вправи
Створити колекцію
Реєстрація
Що таке LearningApps.org?
Показати довідку
LearningApps.org
Про нас
Угода / Умови

в)

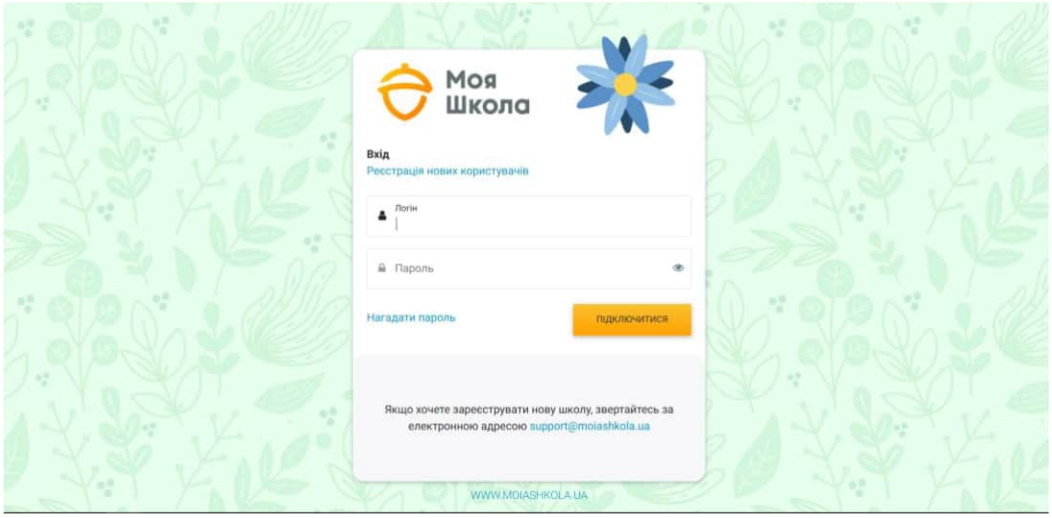


г)

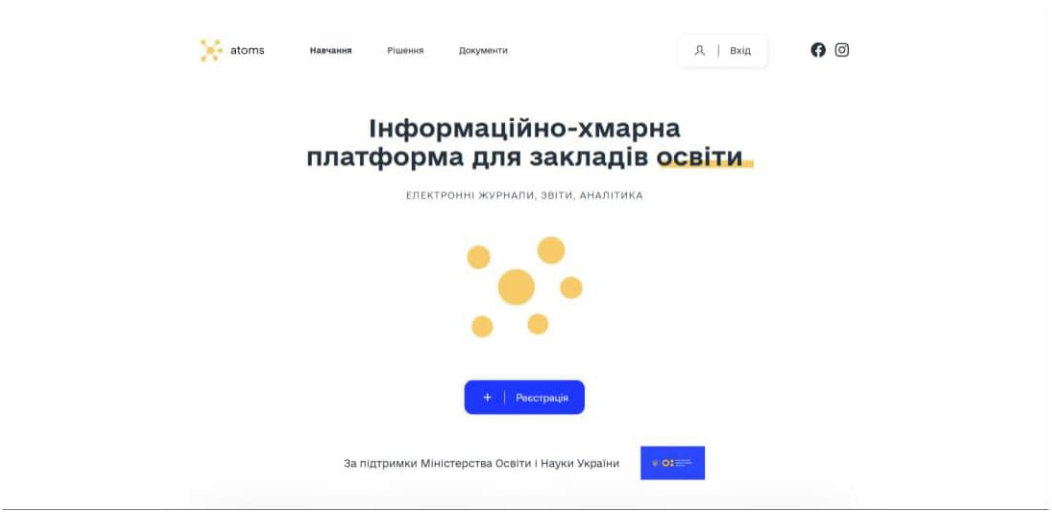
Рисунок.1.1 – Демонстрація платформ:

а) На Урок; б) Classtime; в) LearningApps; г) Kahoot.

Цифрові щоденники та системи управління навчальним процесом на кшталт **“Моя школа”**(рис. 1.2, а), чи **“Атомс”**(рис. 1.2, б), дозволяють не лише вести електронну документацію, але й забезпечують комунікацію між учнями, батьками та вчителями. У таких системах можна надсилати домашні завдання, повідомлення, вести оцінювання та формувати аналітику успішності.



а)

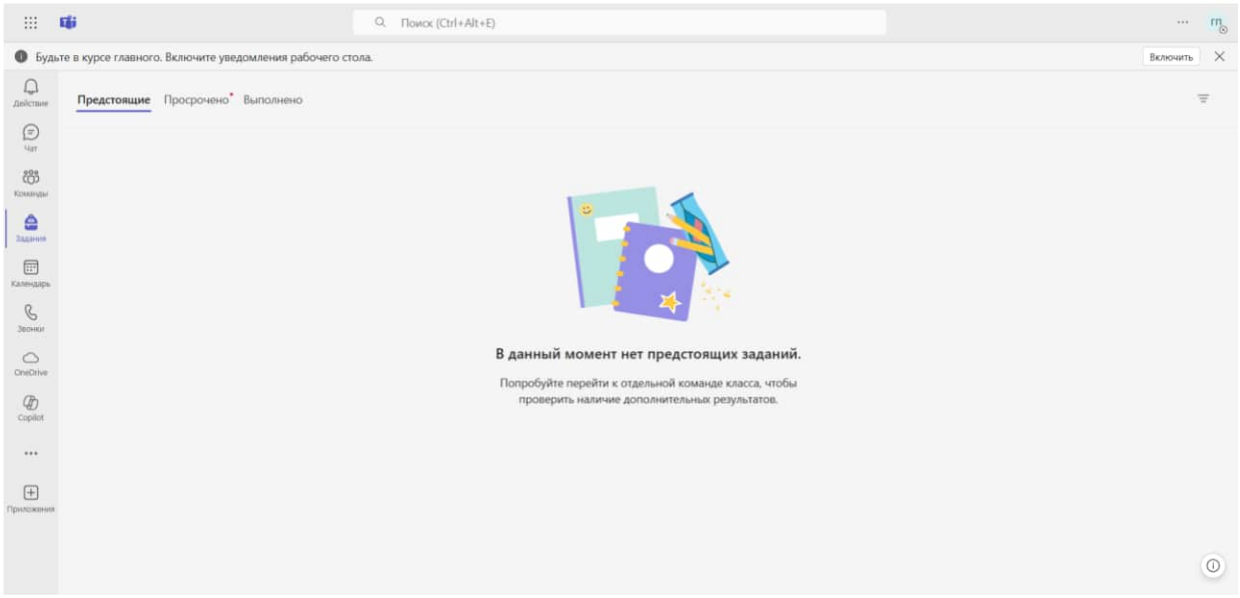


б)

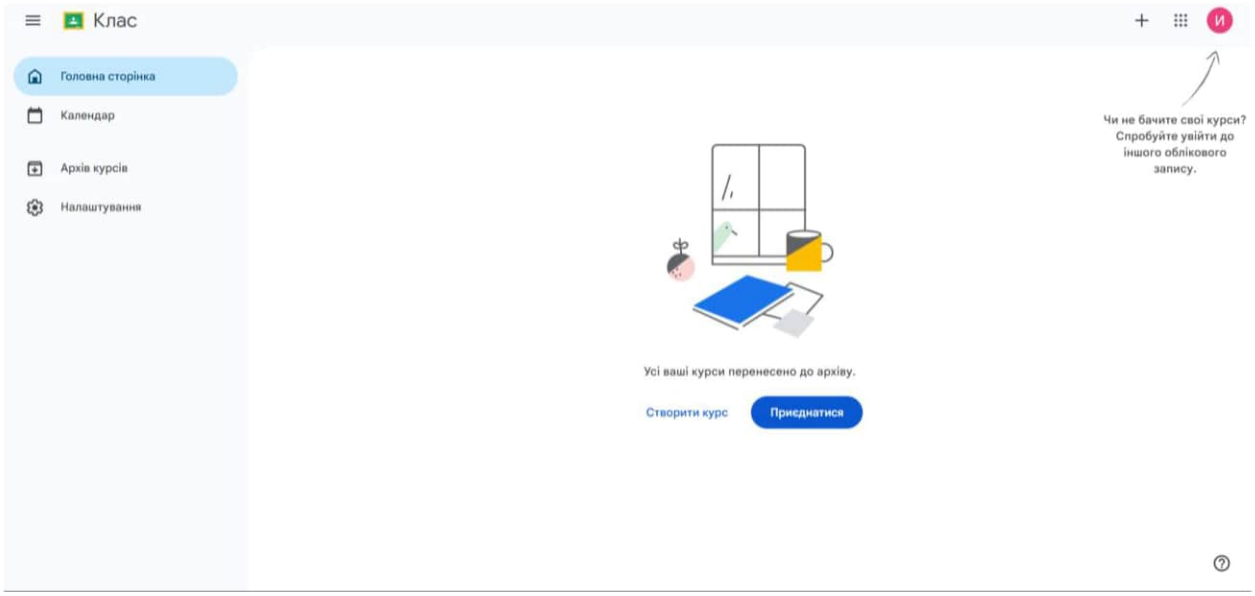
Рисунок.1.2 – Демонстрація цифрових щоденників:

а) Моя школа; б) Атомс.

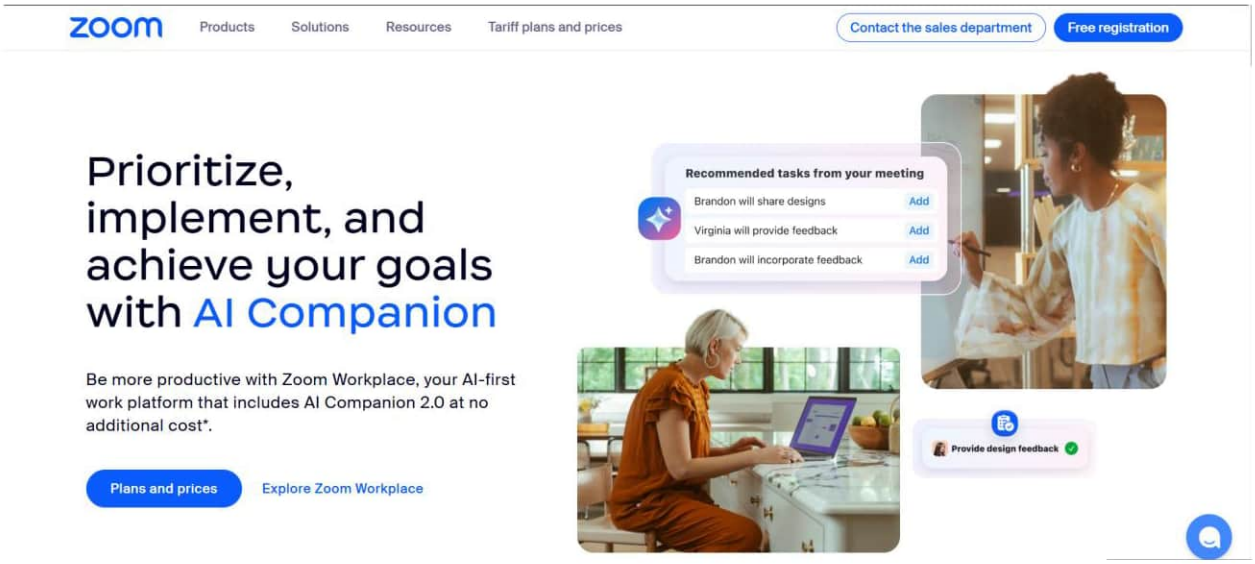
У межах всеукраїнського проєкту “Ноутбук кожному вчителю”, ініційованого Міністерством освіти і науки України, сотні навчальних закладів отримали сучасні пристрої для цифровізації навчання. Це створило основу для активного використання таких ресурсів, як **Microsoft Teams** (рис. 1.2, а)**,** **Google Classroom** (рис. 1.2, б), **Zoom** (рис. 1.2, в), **Meet** (рис. 1.2, г)**,** що особливо проявило себе під час карантинних обмежень [13].



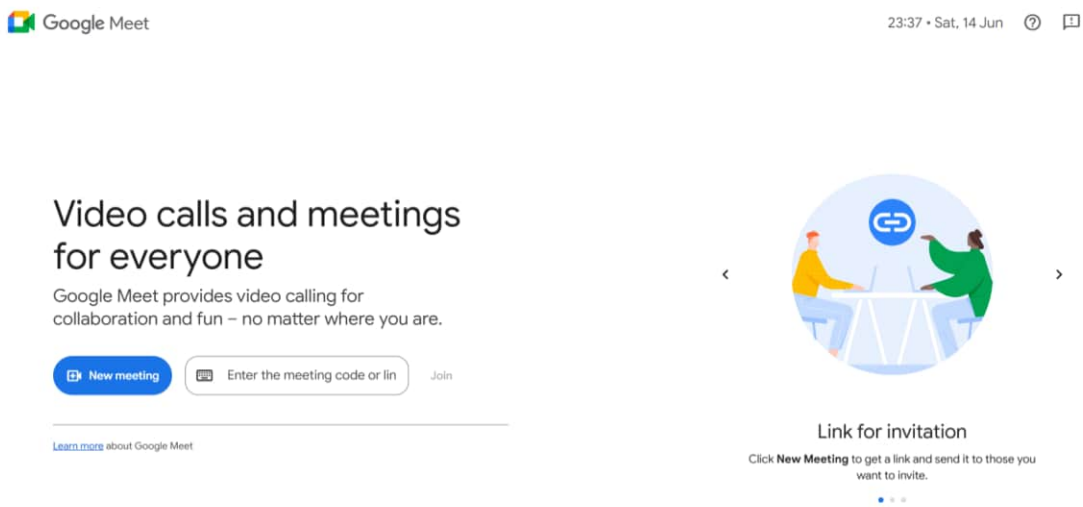
а)


Клас
Головна сторінка
Календар
Архів курсів
Налаштування
Чи не бачите свої курси? Спробуйте увійти до іншого облікового запису.
Усі ваші курси перенесено до архіву.
Створити курс
Приєднатися

б)


zoom
Products
Solutions
Resources
Tariff plans and prices
Contact the sales department
Free registration
Prioritize, implement, and achieve your goals with AI Companion
Be more productive with Zoom Workplace, your AI-first work platform that includes AI Companion 2.0 at no additional cost*.
Plans and prices
Explore Zoom Workplace
Recommended tasks from your meeting
Brandon will share designs
Add
Virginia will provide feedback
Add
Brandon will incorporate feedback
Add
Provide design feedback

в)



г)

Рисунок.1.3 – Демонстрація навчальних платформ:
а) Microsoft Teams; б) Google Classroom;
в) Zoom; г) Meet.

Окремо варто згадати і про локальні ініціативи, що реалізуються за підтримки місцевих органів влади або в межах грантових програм. Так, наприклад, у місті Львів було реалізовано пілотний проєкт створення “класу віртуальної реальності”, де учні можуть проходити інтерактивні лабораторії з фізики та хімії за допомогою VR-шоломів.

Ще один приклад - цифрова платформа **“Єдина школа”**, яка охоплює великий спектр функціональності: від електронних журналів до системи дистанційного навчання з інтегрованими відеоуроками та тренажерами.

Інноваційним підходом є також використання ігрових технологій для створення навчального середовища. Наприклад, у кількох школах Київської та Дніпропетровської областей вчителі вже тестують освітні проєкти, створені на рушіях типу **Unity** або **Godot**, де діти проходять навчальні квести, розв’язують задачі у форматі гри або досліджують інтерактивні музеї.

Таким чином, приклади цифрових практик в українських школах свідчать про поступовий, але впевнений перехід до інноваційних форм

навчання. Ці практики формують передумови для створення сучасних, технологічно насичених освітніх платформ, які можуть бути адаптовані до індивідуальних потреб здобувачів освіти.

1.2 Середовища розробки інтерактивних освітніх додатків

У сучасному освітньому процесі цифрові технології відіграють ключову роль, і одним із головних завдань розробника є вибір інструменту для створення якісного інтерактивного контенту. Серед широкого спектру доступних засобів особливої уваги заслуговують рушії для розробки застосунків, які активно використовуються як у сфері геймінгу, так і в освітньому середовищі. Найпоширенішими сьогодні є Unity, Unreal Engine та Godot - потужні кросплатформені середовища, що підтримують 2D та 3D графіку, мають розвинуту екосистему та спільноту користувачів.

Ці рушії дозволяють реалізовувати складні механіки, ефекти, взаємодії з користувачем та адаптацію під різні платформи, що робить їх ефективним інструментом для розробки освітніх рішень. Їх використання у навчальному контексті забезпечує залучення учнів через ігрові сценарії, симуляції, інтерактивні модулі та візуалізацію складних понять.

Однією з переваг використання таких рушіїв є можливість швидкої розробки прототипів, що дає змогу педагогам або командам з обмеженим технічним досвідом створювати повноцінні освітні продукти. Наприклад, у початковій школі це можуть бути прості 2D-квести, які включають візуалізацію навчального матеріалу та завдання у форматі гри. У старших класах ідеться про створення навчальних симуляцій, лабораторій або віртуальних музеїв.

Не менш важливою перевагою є підтримка інтерактивних сценаріїв, які дозволяють організувати навчання за принципами конструктивізму, коли учень не просто споживає інформацію, а активно взаємодіє з навчальним середовищем, експериментує та приймає рішення. Такі підходи відповідають

методології НУШ, де основою виступають діяльнісне навчання та формування ключових компетентностей.

Ігрові рушії також підтримують мультимедійні формати: зображення, відео, звук, що є важливим у контексті побудови багатоканального сприйняття інформації. Наприклад, у застосунках можна озвучити завдання, додати пояснення у відеоформаті або візуалізувати складні математичні формули через інтерактивну анімацію.

Особливо варто підкреслити можливість експорту проєктів на різні платформи: Windows, Android, iOS, Web. Це дає змогу забезпечити доступ до навчального контенту з будь-якого пристрою, включаючи смартфони, планшети, інтерактивні панелі або домашні ПК. Для НУШ, де важливими є принципи інклюзії, гнучкості та доступності, це має стратегічне значення.

На основі аналізу доступних рішень, у наступних підрозділах буде проведено порівняння трьох найбільш вживаних рушіїв з урахуванням їхньої доцільності для створення інтерактивних навчальних додатків, з подальшим обґрунтуванням вибору рушія Godot для реалізації практичного проєкту дипломної роботи.

1.2.1 Порівняльний аналіз ігрових рушіїв: Unity, Unreal Engine, Godot

У процесі створення інтерактивних освітніх додатків вибір рушія відіграє критичну роль, адже саме рушій визначає функціональні можливості, вимоги до обладнання, гнучкість у реалізації логіки та інтеграції мультимедіа. На сучасному ринку існує декілька лідерів серед платформ для розробки додатків - це Unity, Unreal Engine та Godot. У цьому підрозділі проаналізуємо їх переваги, обмеження та потенціал для освітнього застосування.

Unity (рис 1.4, а) є одним із найпоширеніших рушіїв, що підтримує як 2D, так і 3D графіку. Основними перевагами Unity є:

- Потужна підтримка мобільних та веб-платформ [14];
- Широкий вибір шаблонів для освітніх, ігрових та бізнес-додатків;

- Мова програмування C#, що популярна серед новачків;
- Велика спільнота, що забезпечує наявність великої кількості туторіалів.

Проте слід зазначити, що Unity може бути складним для освоєння початківцями, зважаючи на багатофункціональність середовища. Також є нюанси з ліцензуванням - комерційне використання обмежене доходами.

Unreal Engine (рис 1.4, б) - рушій, орієнтований на високоякісну 3D-графіку та складні інтерактивні сцени. Його особливості:

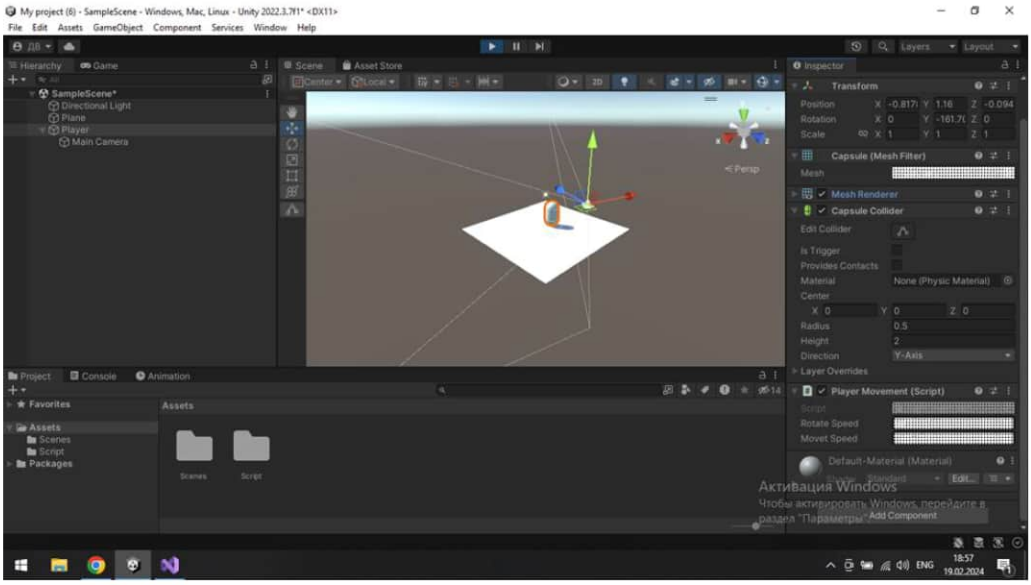
- Вражаючі візуальні ефекти, завдяки рушію на базі C++ та Blueprint-системи;
- Високі системні вимоги, що обмежує використання в умовах шкільних комп'ютерних класів;
- Менш зручний інтерфейс для освітніх потреб.

Використання Unreal може бути виправданим для вузьких випадків, де потрібна повноцінна 3D-візуалізація, зокрема - у STEM-напрямах чи віртуальних лабораторіях.

Godot (рис 1.4, в) вирізняється серед рушіїв тим, що є повністю безкоштовним [15], має відкритий код та активно розвивається спільнотою. Серед його переваг:

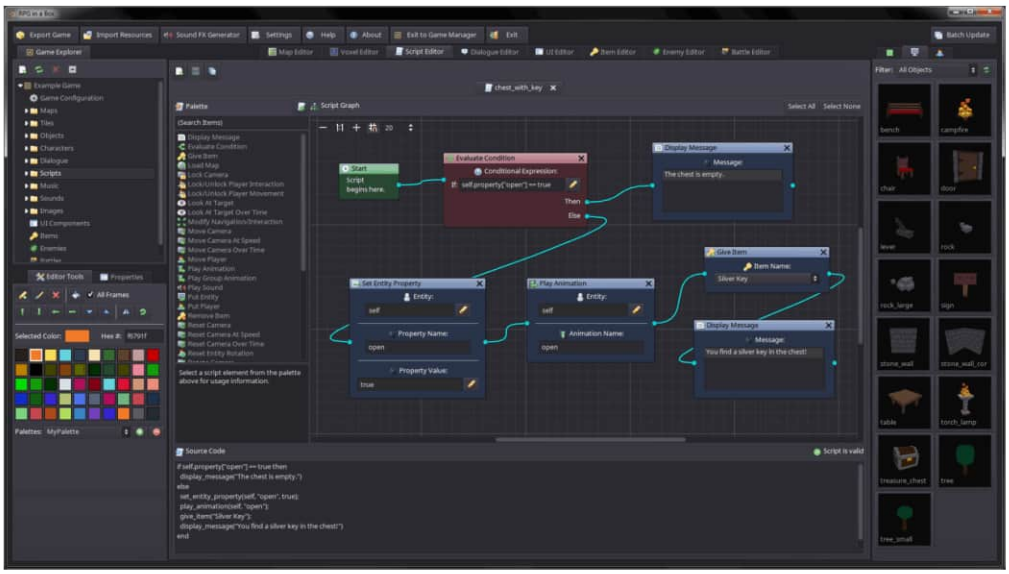
- Мала вага, швидкий запуск;
- GDScript, подібний до Python - ідеальний для навчання програмуванню;
- Зручність у роботі з 2D (і 2.5D);
- Можливість запуску на слабких ПК;
- Підтримка української спільноти та документації.

Godot активно використовується саме в аматорських, інди - та освітніх проєктах, ідеально підходить для реалізації НУШ-підходів.


My project (8) - SampleScene - Windows, Mac, Linux - Unity 2022.3.7f1* <DX11>
File Edit Assets GameObject Component Services Window Help
Hierarchy
Game
SampleScene*
Directional Light
Plane
Player
Main Camera
Scene
Asset Store
Persp
Inspector
Transform
Position
Rotation
Scale
Capsule (Mesh Filter)
Mesh
Mesh Renderer
Capsule Collider
Edit Collider
Is Trigger
Provides Contacts
Material
None (Physic Material)
Center
Radius
Height
Direction
Y-Axis
Layer Overrides
Player Movement (Script)
Rotate Speed
Movet Speed
Default-Material (Material)
Add Component
Project
Console
Animation
Favorites
Assets
Scenes
Script
Packages
Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".
ENG
18:57
19.02.2024

а)

б)



в)

Рисунок.1.4 - Порівняння інтерфейсів середовищ:

а) Unity; б) Unreal Engine; в) Godot.

Кожен із проаналізованих рушіїв має свої сильні сторони. Однак у контексті початкової та середньої освіти, де важлива простота, кросплатформеність, легкість навчання та обмеженість ресурсів - Godot виступає оптимальним рішенням. Його відкритість, спільнота, підтримка 2D та адаптація до слабких ПК дозволяють створювати якісні освітні додатки навіть в умовах обмеженого бюджету.

1.2.2 Переваги 2D і 2.5D середовищ для навчального процесу

У створенні сучасних освітніх цифрових продуктів важливо не лише передати навчальний матеріал, а й зробити його візуально привабливим, доступним, гнучким до адаптації та сумісним з апаратною базою більшості навчальних закладів. У цьому контексті формати 2D та 2.5D займають особливо вигідне положення, поєднуючи технологічну простоту із широкими педагогічними можливостями.

2D-графіка вже давно зарекомендувала себе у навчальних додатках завдяки зрозумілості, універсальності та невибагливості до ресурсів [16]. Вона дозволяє створювати яскраві й інтуїтивно зрозумілі інтерфейси, адаптовані під

потреби учнів молодшого і середнього шкільного віку. Крім того, 2D-формат забезпечує високу продуктивність навіть на застарілих пристроях, що особливо актуально для українських шкіл у регіонах з обмеженим технічним забезпеченням.

Формат 2.5D виступає компромісним рішенням між 2D та повноцінним 3D. Він забезпечує відчуття глибини та об'єму сцени за допомогою проєктування двовимірних спрайтів у просторі або використання спрощених тривимірних моделей. Такий підхід дозволяє значно підвищити залученість учнів, не вдаючись до складного 3D-моделювання чи ресурсомісткого рендерингу [17].

Використання 2.5D особливо ефективне у форматі інтерактивного віртуального музею, де учень пересувається кімнатами або коридорами, обирає експонати для дослідження, взаємодіє з мультимедійним контентом (текст, зображення, відео). Завдяки цьому забезпечується не лише засвоєння знань, а й розвиток візуального мислення, самостійного прийняття рішень і мотивації до навчання.

Основні переваги 2D/2.5D-середовищ у навчальному процесі:

- **Низьке обчислювальне навантаження**, що дозволяє запускати застосунки на слабких пристроях без втрати функціональності.
- **Гнучкість у проєктуванні**: 2.5D дозволяє імітувати простір без складного програмування, при цьому зберігаючи логіку навігації, камер і взаємодії.
- **Простота у створенні контенту**: для реалізації експозицій достатньо використовувати зображення, шрифти, анімації та готові шаблони рушія, як-от Godot.
- **Інтуїтивна навігація для учнів**: спрощені маршрути не викликають дезорієнтації, як це іноді трапляється у повному 3D.
- **Можливість масштабування**: такі додатки легко доповнювати новими залами, тестами, матеріалами, не змінюючи ядра структури.

Особливо важливо, що можливості рушія Godot та формат 2D/2.5D повністю відповідає принципам Нової української школи (НУШ) (рис. 1.5) доступність, інклюзивність, міждисциплінарність і цифрова грамотність. У цифровому віртуальному музеї учень може самостійно обирати маршрут, досліджувати експонати у власному темпі, працювати в режимі індивідуального або групового вивчення.

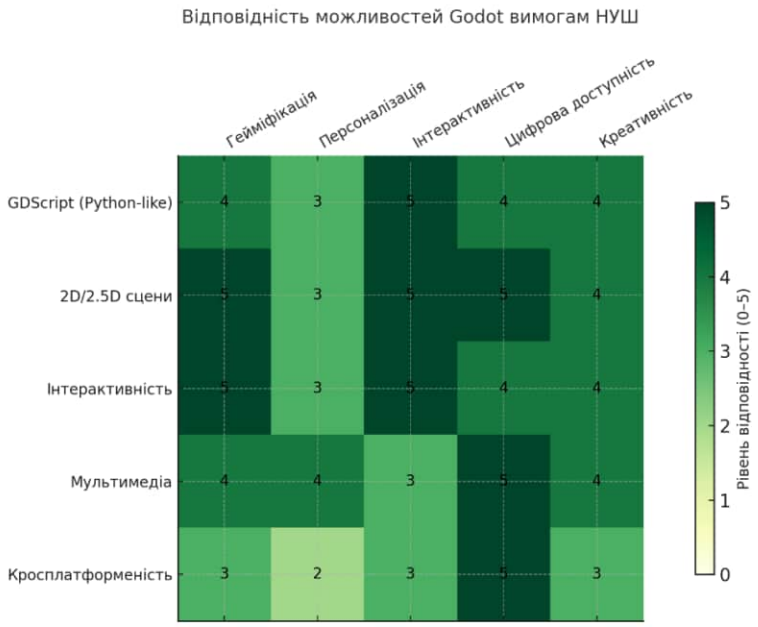


Рисунок. 1.5 – Відповідність концепції віртуального музею принципам НУШ

При цьому слід зазначити, що 2D і 2.5D-середовища не лише сприяють формуванню візуального мислення, а й відкривають простір для інтеграції з іншими предметними галузями – літературою, історією, мистецтвом. Створюючи цифрову галерею, учні не лише споживають контент, але й можуть брати участь у його формуванні, розміщуючи власні роботи або коментарі, що підсилює ефект занурення.

Для наочності нижче подано порівняння форматів 2D, 2.5D та 3D у контексті освітніх застосунків.(рис. 1.6)

Порівняння форматів візуалізації для освітніх додатків

Формат	Переваги	Недоліки	Вимоги до ПК
2D	Проста розробка; Низькі вимоги; Підтримка старих ПК	Обмежена глибина; Менше залучення	Інтегрована графіка; Мінімальні ресурси
2.5D	Ілюзія глибини; Помірні ресурси; Краще занурення	Середня складність; Обмежена перспектива	Середній ПК; Дискретна графіка бажана
3D	Повне занурення; Реалістичність; Багато інтерактиву	Складна розробка; Вищі витрати; Вимоги до ПК	Потужний ПК; GPU; 8+ ГБ ОЗП

Рисунок. 1.6 - Схематичне порівняння форматів 2D, 2.5D та 3D у навчальному застосунку

На завершення варто згадати успішні приклади реалізації подібних проєктів. Зокрема, **Doom: The Gallery Experience** (рис. 1.7, а) демонструє, як навіть старі рушії можуть бути адаптовані до освітніх цілей через заміну бойових механік на інтерактивну навігацію. У свою чергу, **Museum of All Things** (рис. 1.7, б) на Godot ілюструє потенціал 3D-формату в умовах спрощеного управління й відсутності потреби у складних сценаріях (рис. 1.6). У таблиці нижче наведено порівняльні характеристики обох проєктів.(таб. 1.1)

а) б)

Рисунок. 1.7 – Приклади реалізації інтерактивних музеїв:

а) *Doom: The Gallery Experience* – момент взаємодії з експонатом;

б) *Museum of All Things* – навігація 3D-простором і дослідження об’єктів.

Таблиця 1.1 - Порівняльна характеристика двох віртуальних музеїв за ключовими параметрами

Назва проєкту	**Рушій**	**Формат**	**Тип взаємодії**	**Контент**	**Наявність сюжету**	**Адаптація до освіти**
Doom: The Gallery	GZDoom	2.5D	Наведення / підхід	Зображення, опис	Відсутній	Висока
Museum of All Things	Godot	3D	Навігація, кліки	3D-експонати	Відсутній	Висока

Таким чином, використання 2D та 2.5D-графіки у створенні освітніх застосунків, зокрема в інтерактивних віртуальних музеях, дає змогу поєднати технічну ефективність із сучасними педагогічними підходами, орієнтованими на гнучке, наочне та інклюзивне навчання.

1.2.3 Рекомендації щодо вибору рушія для освітніх цілей

Вибір рушія для створення інтерактивних навчальних додатків - це не лише технічне рішення, а й стратегічний крок у розбудові цифрової освіти. В умовах впровадження Нової української школи (НУШ), де акцент робиться на

розвиток цифрових компетентностей, гнучкість навчального процесу та індивідуалізацію підходів, рушій повинен відповідати не лише вимогам технічної реалізації, а й бути максимально адаптованим до освітнього середовища.

Умови та виклики української освіти

Школи України часто працюють в умовах обмежених ресурсів: це і застаріле обладнання, і нестача доступу до якісного інтернету, а іноді й обмеження у професійній підготовці педагогів у сфері цифрових технологій. Саме тому одним із ключових факторів вибору рушія має бути **доступність** як у фінансовому, так і в навчальному плані. В ідеалі рушій має бути:

- безкоштовним для використання;
- з мінімальними системними вимогами;
- з інтуїтивно зрозумілим інтерфейсом;
- з достатньою документацією та підтримкою українською або англійською мовами.

Особливо актуальними є open-source рішення, які не вимагають ліцензійних внесків або спеціалізованого обладнання - саме такі інструменти дозволяють забезпечити рівний доступ до цифрової освіти як у великих містах, так і у сільських громадах.

Аналіз ключових критеріїв вибору рушія

До основних критеріїв, за якими варто обирати рушій для освітніх проєктів, відносяться:

- **Ліцензія**: бажано - безкоштовна та відкрита (open-source).
- **Мова програмування**: проста у вивченні, бажано схожа на Python або JavaScript.
- **Підтримка 2D/2.5D**: зручність роботи з об'єктами у площині, що актуально для більшості навчальних додатків.
- **Підтримка мультимедіа**: інтеграція зображень, відео, звуку.
- **Можливість кросплатформеного експорту**: для Android, Windows, Web.

- **Активна спільнота**: наявність форумів, туторіалів, шаблонів [18].

У таблиці нижче наведено порівняння рушіїв за ключовими критеріями (таб. 1.2)

Таблиця.1.2 - Порівняльна таблиця рушіїв Unity, Unreal Engine та Godot за освітніми критеріями

Критерій	**Godot**	**Unity**	**Unreal Engine**
Ліцензія	Open-source (MIT), повністю безкоштовна	Безкоштовна для освітніх потреб	Умовно безкоштовна (до $1 млн доходу на рік)
Мова програмування	GDScript (схожий на Python), C#, VisualScript	C#, Visual Scripting	C++, Blueprints
Системні ресурси	Низькі вимоги, працює на слабких пристроях	Середні системні вимоги	Високі вимоги до «заліза»
Підтримка 2D / 3D графіки	Відмінна підтримка 2D, базова — для 3D	Надійна підтримка як 2D, так і 3D	Основний акцент на 3D, обмежена робота з 2D
Документація та навчальні матеріали	Докладна, доступна англійською, спільнотою оновлюється	Вичерпна документація, багато курсів та гідів	Широка база знань, орієнтована на професіоналів
Активність спільноти	Активна, зростаюча спільнота	Велика міжнародна спільнота	Професійна спільнота, велика кількість прикладів
Легкість навчання	Висока — зручний інтерфейс та простий синтаксис	Середній рівень складності	Складна крива навчання, більше підходить для досвідчених користувачів

Успішні приклади використання рушіїв в освітніх цілях

- **Funexpected Math** *(Godot)* - українсько-європейський мобільний застосунок, що пропонує математичні головоломки у форматі гри для молодших школярів. Він побудований на рушії Godot і підтримує просту інтерактивну взаємодію на планшетах.

- **Crystal Island** *(Unity)* - інтерактивне середовище, розроблене в рамках наукового дослідження для навчання природничим наукам. Гравець має розгадувати проблеми на основі аналізу даних, які відповідають навчальним стандартам США.
- **Island Saver** *(Unity)* - освітньо-ігровий додаток для вивчення основ фінансової грамотності. Гравці «рятують» острови від сміття, керують віртуальними фінансами, накопичують та витрачають монети. Розроблено студією NatWest (рис. 1.7) .
- **Mathland** *(Unreal Engine)* - інтерактивна гра, створена для вивчення математики в початкових класах. Відзначається яскравою 3D-візуалізацією, проте через технічну складність має обмежене використання на слабких пристроях.
- **HEROES Simulation** *(Unreal Engine)* - система тренувань для екстрених служб [19], яка може слугувати основою для створення освітніх платформ у старшій школі (наприклад, курс «Захист України»).
- **Coding for Kids** *(Godot)* - онлайн-курс, який навчає школярів створенню простих ігор. Застосунок побудований на рушії Godot і використовується в США та Польщі як частина позашкільної IT-освіти.

Рисунок.1.8 — Візуалізація інтерфейсу освітньої гри Island Saver

1.3 Обґрунтування вибору рушія Godot для освітніх проєктів

Поява інструментів з відкритим кодом дала поштовх розвитку освітніх застосунків навіть за умов обмежених ресурсів. У цьому контексті рушій Godot виявляється винятково придатним для реалізації інтерактивних освітніх платформ у межах Нової української школи (НУШ). Його архітектура, відкритість та простота у використанні відповідають сучасним вимогам до освітніх технологій.

Godot має низку переваг, які відрізняють його серед інших рушіїв:

- **Безкоштовність і відкритий код.** Це дозволяє використовувати Godot у навчальних закладах без ризику порушення ліцензій, а також дає змогу вивчати та змінювати рушій на рівні ядра.
- **Простота у вивченні.** GDScript — мова сценаріїв, що використовує синтаксис, подібний до Python. Це дозволяє учням поступово перейти до вивчення більш потужних мов без додаткового бар’єру входу.
- **Потужна підтримка 2D та 2.5D.** Це особливо актуально для створення навчальних ігор, які не потребують високої продуктивності або складної тривимірної графіки.
- **Кросплатформеність.** Готові додатки можна експортувати на Windows, Android, Linux та Web без суттєвих змін у коді.

Godot дозволяє реалізовувати навчальні концепти у вигляді інтерактивних віртуальних просторів, симуляцій, міні-ігор і цифрових виставок. Завдяки відкритій структурі рушій може бути адаптований для навчання як самостійний об'єкт вивчення: учні можуть вивчати архітектуру рушія, структуру проєкту, організацію ресурсів, створення логіки через GDScript тощо.

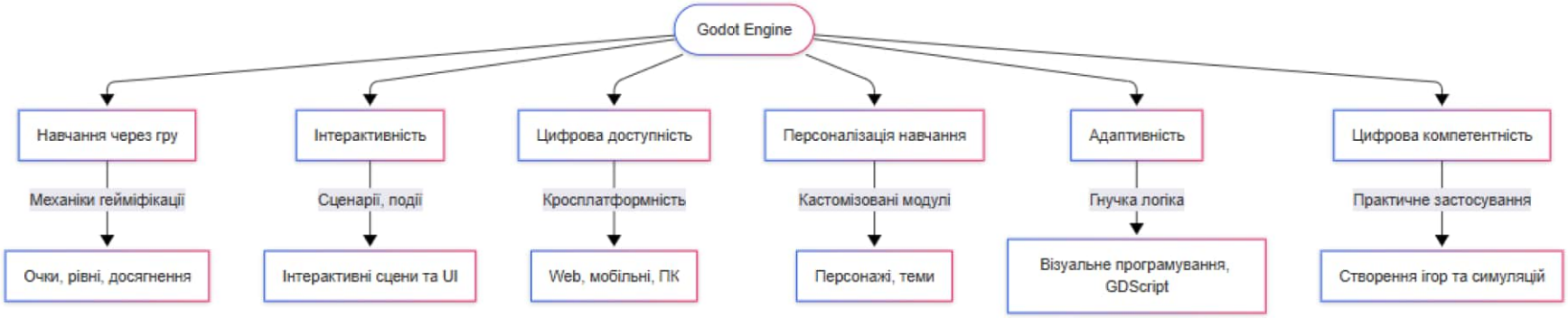


Рис.1.9 - Схематичне зіставлення функцій Godot із вимогами НУШ (гейміфікація, персоналізація, цифрова доступність, інтерактивність)

Крім того, важливо враховувати, що Godot активно використовується ентузіастами, вчителями інформатики, а також незалежними розробниками в усьому світі. Зростання кількості безкоштовних прикладів, шаблонів і документації дозволяє ефективно впроваджувати Godot навіть у звичайні загальноосвітні школи.

Саме тому в наступних підпунктах буде детальніше розглянуто архітектуру Godot, його мову програмування GDScript та аналіз вже реалізованих освітніх проєктів, створених із використанням цього рушія.

1.3.1 Особливості архітектури та відкритого коду Godot

Рушій Godot є унікальним у світі програмного забезпечення для розробки інтерактивних застосунків завдяки своїй відкритій архітектурі та повністю відкритому коду. Це означає, що користувачі - від аматорів до професійних розробників - мають вільний доступ до всіх внутрішніх компонентів рушія, можуть вивчати його зсередини, змінювати під свої потреби та використовувати без жодних ліцензійних обмежень. Це є особливо важливим у контексті шкільної освіти, де доступ до безкоштовного, легального та адаптивного інструментарію є критично важливим.

На відміну від багатьох комерційних продуктів (наприклад, Unity або Unreal Engine), які потребують ліцензій або передбачають відрахування у разі комерційного використання, Godot розповсюджується під ліцензією MIT. Це не лише забезпечує фінансову доступність для навчальних закладів, а й відкриває можливість реального занурення в архітектуру рушія. Учні, які цікавляться програмуванням, можуть не лише створювати ігри чи навчальні симуляції, а й вивчати, як працює сам рушій, змінювати або доповнювати його функціональність, що є безцінним у контексті формування глибоких IT-компетентностей.

Ще однією важливою перевагою архітектури Godot є її модульність. У рушії використовується система "сцен" та "нодів" (вузлів), що дозволяє

будувати складні інтерфейси, логіку взаємодії та анімації з окремих повторно використовуваних блоків. Це не тільки зручно в практиці розробки, а й відповідає принципам об'єктно-орієнтованого мислення, що входить до сучасних освітніх стандартів.

Кожен вузол у Godot виконує певну функцію - наприклад, Node2D використовується для позиціонування об'єктів у 2D-просторі, Sprite для виводу графіки, CollisionShape2D для обробки фізики, AudioStreamPlayer для програвання звуку тощо. Побудова ієрархії з таких нодів дозволяє конструювати повноцінну логіку проєкту без потреби у складних структурах коду.

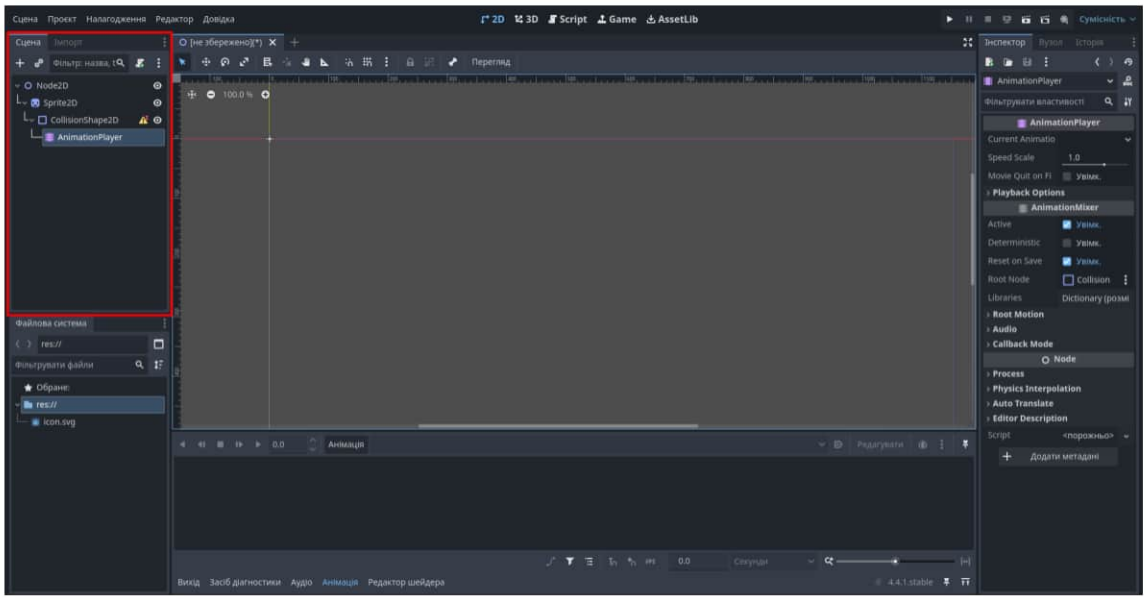


Рисунок.1.10 - Приклад базової структури сцени у Godot: головна сцена з персонажем, колізіями, анімацією та звуком

Ще однією визначною рисою Godot є вбудований редактор коду, що інтегрований у сам рушій. Він підтримує підсвічування синтаксису, автозаповнення, рефакторинг та відлагодження, що дозволяє зосередитись на навчанні програмуванню без потреби додатково налаштовувати середовище розробки.

Інструменти для візуального редагування, налаштування фізики, анімацій та інтерфейсів користувача вбудовані безпосередньо в редактор, що значно пришвидшує навчання. Навіть новачки можуть за короткий час

створити повноцінну інтерактивну сцену або гру, експериментуючи з різними підходами та ефектами.

Таким чином, відкритість коду, зручна архітектура та адаптованість до освітніх потреб роблять Godot ідеальним середовищем як для розробки навчальних застосунків, так і для використання його в якості платформи для викладання основ програмування, логіки, дизайну взаємодії та мультимедійного виробництва.

1.3.2 Освітній потенціал GDScript як підготовки до Python

Одним із вагомих факторів, що обумовлює доцільність використання рушія Godot у навчальному середовищі, є мова програмування GDScript, яка використовується як основний інструмент створення логіки додатків. GDScript спеціально розроблена для Godot і має синтаксис, який у багатьох аспектах нагадує популярну мову Python. Завдяки цьому вона є зручною для початківців і особливо ефективною в освітніх умовах, де головна мета полягає не лише в отриманні результату, а й у формуванні базових навичок програмування.

У навчальному контексті така схожість між GDScript і Python дає змогу розглядати використання Godot не лише як засіб для створення інтерактивних продуктів, а й як платформу для підготовки учнів до подальшого вивчення повноцінних мов програмування. Це важливо з огляду на той факт, що Python активно використовується в багатьох галузях: від веброзробки до аналізу даних і машинного навчання. Таким чином, ознайомлення з базовими конструкціями GDScript (цикли, умови, функції, об’єкти) дозволяє учням закласти фундамент для подальшого розвитку в IT-напрямі.

Наприклад, конструкція умовного оператора в GDScript виглядає так:

```
if score >= 12:
    print("Вітаємо! Ви успішно пройшли тест.")
else:
    print("Спробуйте ще раз.")
```

Цей приклад майже повністю аналогічний відповідному синтаксису в Python, що робить перехід між мовами максимально плавним і природним.

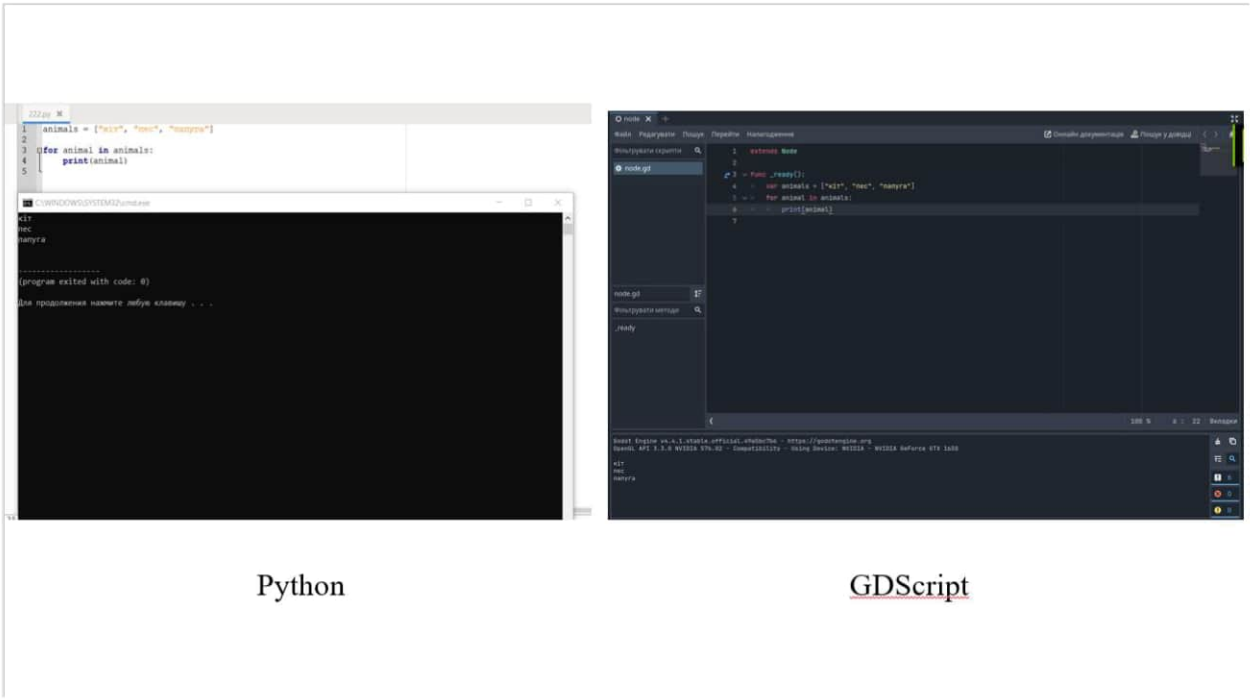
Для ще більшої наочності у навчальному процесі доцільно демонструвати учням порівняння схожих конструкцій на обох мовах. Нижче наведено приклад оголошення масиву та циклу в GDScript:

```
var animals = ["кіт", "пес", "папуга"]

for animal in animals:
    print(animal)
```

Аналогічний код у Python:

```
animals = ["кіт", "пес", "папуга"]

for animal in animals:
    print(animal)
```

Як видно з прикладу, для учнів та студентів це може стати ідеальним мостом до складніших мов, а також сформувати позитивний досвід перших кроків у сфері програмування. (рис. 1.10)



Python GDScript

Рисунок.1.11 - Схожість синтаксису GDScript і Python у базових конструкціях програмування

Крім технічної схожості, важливим є й дидактичний аспект: простота, передбачуваність та лаконічність GDScript дозволяють учням зосередитись не

на запам’ятовуванні складного синтаксису, а на логіці розв'язання задач, структурі програм та концептуальному розумінні.

Таким чином, Godot стає не просто інструментом створення ігор чи навчальних платформ, а повноцінною освітньою системою, що сприяє формуванню алгоритмічного мислення, розвитку логіки та готує ґрунт для глибшого вивчення ІТ.

1.3.3 Кейс-аналіз прикладів освітніх застосунків, створених у Godot

Попри репутацію рушія Godot як інструменту здебільшого для інді-розробників та ентузіастів, останніми роками він усе частіше застосовується у сфері освіти - як у школах, так і на рівні університетів. Простота використання, низькі вимоги до ресурсів і відкритий вихідний код роблять його особливо привабливим для педагогів, які прагнуть поєднувати навчання з інноваційними цифровими підходами.

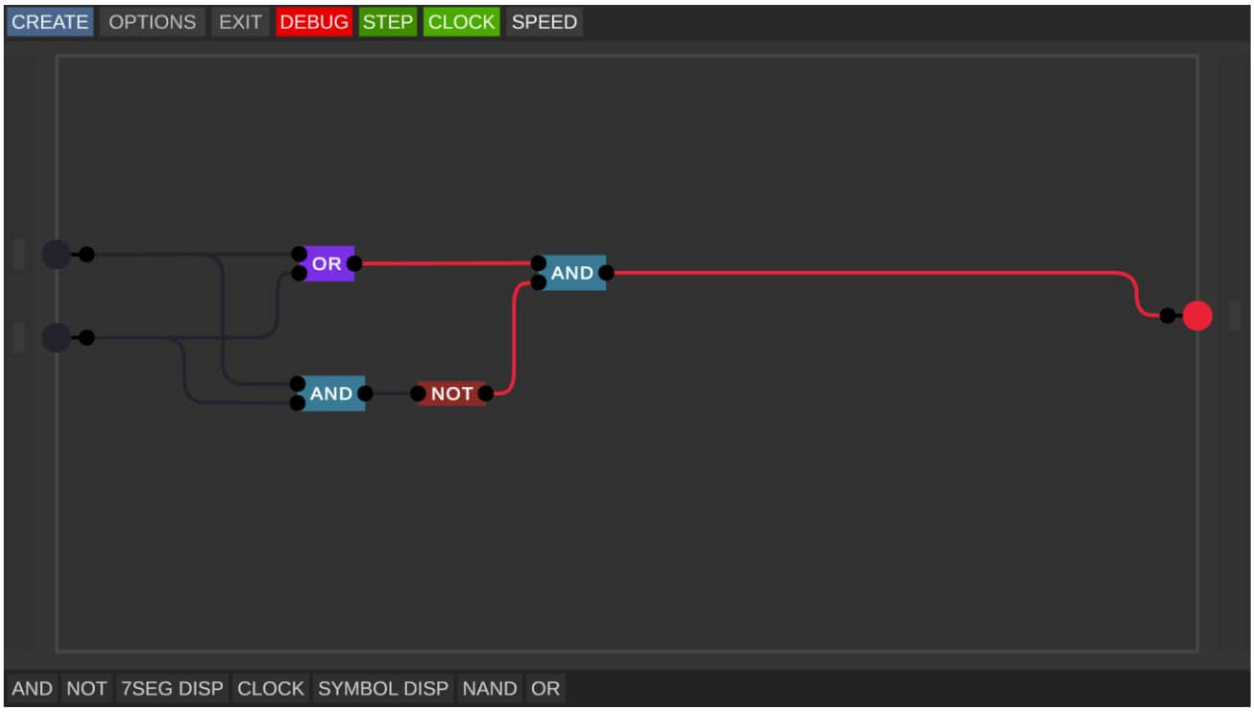
Одним із найяскравіших прикладів є Super Practica (рис. 1.12, а) [20] - освітній проєкт із відкритим кодом, спрямований на вивчення математики через гру. Користувач отримує можливість тренуватись у базових обчисленнях за допомогою простих міні-ігор, які мають зрозумілий візуальний інтерфейс і добре адаптовані для початкової школи. Головна мета - не тільки повторити матеріал, а й заохотити дітей шляхом гейміфікації.

Ще один важливий приклад - Digital Logic Sim 2 (рис. 1.12, б) [21]. Це симулятор для вивчення основ цифрової логіки та електроніки. У програмі можна створювати цифрові схеми, експериментувати з елементами типу AND, OR, NOT, та бачити результат у реальному часі. Застосунок використовують як вчителі інформатики, так і викладачі технічних ВНЗ, адже він допомагає візуалізувати абстрактні поняття. Особливо цінною є його функціональність в офлайн-режимі - що важливо для сільських шкіл.

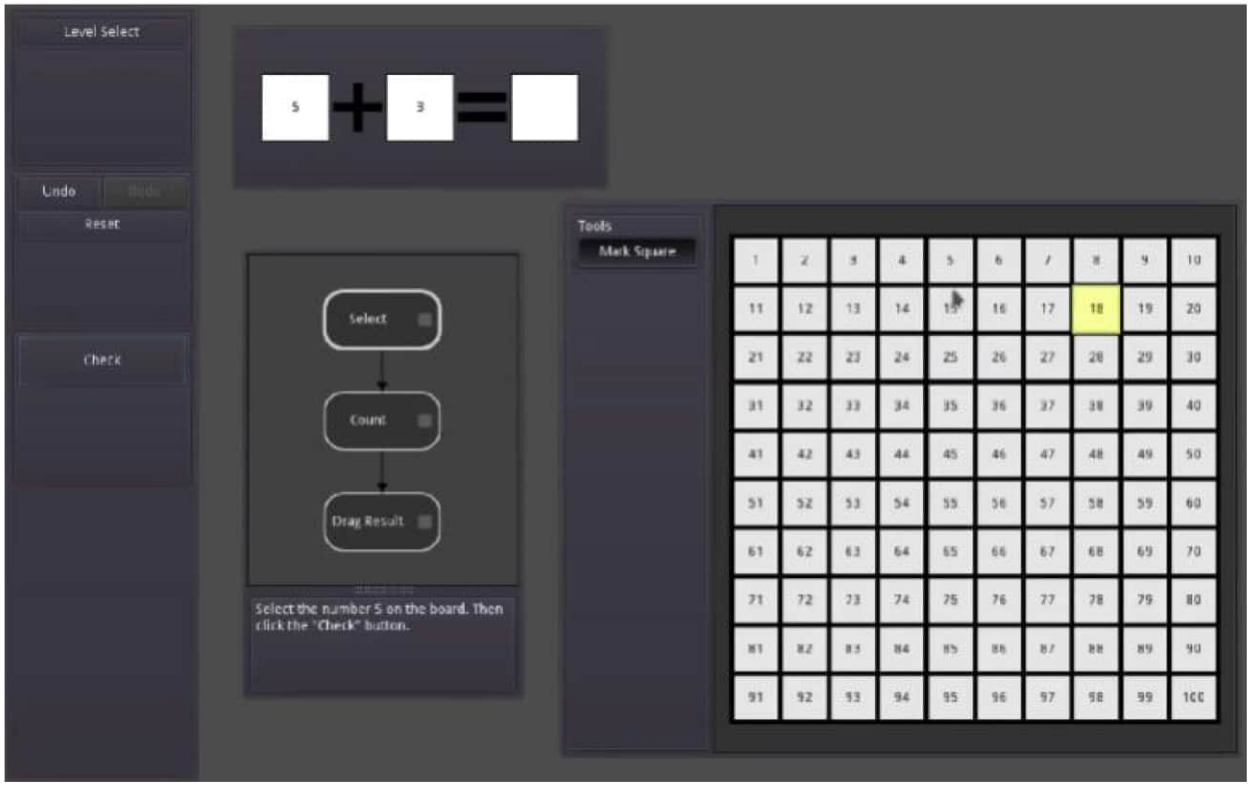
Іншим прикладом є Bioblox 2.5D (рис. 1.12, в) [22] - проєкт, що поєднує ігрову форму з біологічною моделлю процесу докування білків. Попри вузьку спеціалізацію, ця гра доводить, що рушій Godot придатний навіть для складних

візуалізацій у науково-освітньому контексті. Студенти можуть збирати молекули, бачити взаємодію структур у 2.5D-просторі й таким чином краще опановувати біохімію. Також варто згадати менш технічні, але педагогічно цінні проєкти. Наприклад, Project Learn - інтерактивна пригода для молодших школярів, де сюжетна лінія базується на історичних фактах або літературних творах. Гравець переміщується світом, дізнається нову інформацію й відповідає на запитання, що перевіряють розуміння. Проєкт застосовувався як допоміжний інструмент для уроків літератури та історії.

Такі кейси демонструють практичну можливість реалізації освітніх ідей у рушії Godot. Залежно від мети, контент може бути як глибоко предметним, так і розважально-пізнавальним. Особливо важливо, що Godot дозволяє реалізовувати інтерактивність навіть тим, хто не має великого досвіду в програмуванні - наприклад, педагогам, які хочуть створити власний тренажер або демонстраційну програму.



а)


Level Select
5
+
3
=
Undo
Redo
Reset
Check
Tools
Mark Square
Select
Count
Drag Result
Select the number 5 on the board. Then click the "Check" button.

б)


116
Score
116
High Score
2
Lives
0%
Docking %
14
Timer
118
Score
GREAT! +2
118
High Score
2
Lives
100%
Docking %
13
Timer

в)

Рисунок.1.12 - Приклади освітніх застосунків, реалізованих у середовищі Godot

а)Digital Logic Sim 2; б)Super Practica; в)Bioblox 2.5D.

1.4 Висновки по першому розділу

У першому розділі розглянуті сучасні тенденції розвитку цифрової освіти в Україні, зокрема - у контексті впровадження реформи Нової української школи. Детально проаналізовані ключові принципи інтерактивного навчання, важливість гейміфікації, персоналізації, візуалізації знань та використання цифрових інструментів у шкільному процесі.

Окрему увагу приділено порівнянню сучасних рушіїв розробки (Unity, Unreal Engine, Godot), що використовуються для створення інтерактивного освітнього контенту. На основі аналізу функціональних можливостей, ресурсної вимогливості та зручності використання зроблено висновок про доцільність використання Godot у шкільному середовищі.

Наведені також рекомендації щодо вибору рушія для освітніх цілей, з урахуванням технічних можливостей українських шкіл та вимог до цифрової доступності. Встановлено, що Godot - відкритий, легкий, інтуїтивно зрозумілий рушій, який підтримує 2D і 2.5D графіку та є придатним для розробки ігор і симуляцій навіть у складних умовах.

У межах розділу також проведено аналіз уже реалізованих освітніх рішень на рушії Godot. Ці кейси охоплюють як предметно-орієнтовані застосунки (математика, інформатика, біологія), так і міждисциплінарні симуляції, що демонструють широкий спектр можливостей рушія у навчанні.

На основі теоретичного аналізу сформульовано підґрунтя для реалізації власного проєкту інтерактивної навчальної платформи - у формі освітнього музею у стилі 2.5D. Такий підхід дозволяє поєднати освітній зміст із візуально привабливою інтерактивною формою, що відповідає вимогам сучасного учня та ідеології НУШ.

Для реалізації кваліфікаційної роботи необхідно:

1. Оцінити потенціал використання інтерактивного візуального середовища у навчальному процесі.
2. Розробити структуру віртуального простору сцени у Godot.
3. Реалізувати механіку пересування, взаємодії та подачі навчального контенту.
4. Забезпечити мультимедійне наповнення додатку (аудіо, зображення, відео).
5. Тестування функціоналу та підготовка інструкції користувача.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ІНТЕРАКТИВНОГО НАВЧАЛЬНОГО МУЗЕЮ У GODOT

2.1 Вибір концепції та формат візуалізації

Плануючи розробку навчального цифрового ресурсу у вигляді інтерактивного музею, необхідно перш за все визначити концепцію подачі інформації, її структуру, візуальний стиль та формат взаємодії з користувачем. Такий підхід дозволяє не лише організувати зміст відповідно до освітніх цілей, а й забезпечити зацікавленість учнів шляхом гейміфікації, наративної структури та мультимедійного супроводу.

Формат інтерактивного музею було обрано невипадково - подібні рішення дають змогу поєднувати переваги традиційного екскурсійного навчання з можливостями цифрових технологій: гіперпосиланнями, анімацією, аудіосупроводом та інтерфейсами, що реагують на дії користувача. Крім того, віртуальні музеї не потребують спеціального обладнання чи фізичної присутності - ними можна користуватися як у класі, так і вдома.

Особливо актуальним є застосування формату музею у контексті Нової української школи (НУШ), яка акцентує увагу на інтегративності, критичному мисленні, візуальному сприйнятті інформації та розвитку креативних здібностей. Завдяки тому, що віртуальний музей поєднує елементи ігрової форми, наукового змісту та візуалізації, він відповідає педагогічним принципам НУШ і може бути використаний як на уроках (наприклад, історії, літератури, мистецтва), так і у міждисциплінарних навчальних заходах (таблиця.2.1).

У межах дипломного проєкту обрано візуалізацію у стилі 2.5D. Цей підхід дозволяє створити ілюзію тривимірного простору, не вимагаючи складного 3D-моделювання. Таким чином, досягається баланс між візуальною привабливістю та технічною досяжністю - важлива умова для реалізації

подібних проєктів у школах, де технічні можливості часто обмежені. Концепція інтерактивного музею подана на рисунку 2.1.

Таблиця. 2.1 - Зіставлення функцій інтерактивного музею з принципами НУШ

Освітня перевага	**Реалізація в інтерактивному музеї**	**Відповідність принципам НУШ**
Гейміфікація	Взаємодія з експонатами, ігрові завдання	Навчання через діяльність
Персоналізація	Вибір маршруту, контроль за темпом вивчення	Індивідуальна освітня траєкторія
Цифрова доступність	Робота на слабких пристроях, офлайн-режим	Рівність у доступі до ресурсів
Інтерактивність	Тригери, діалоги, реакція на дії учня	Активна участь учня у навчальному процесі
Мультимедійність	Аудіо, відео, зображення, текст	Залучення всіх каналів сприйняття

Рисунок. 2.1 - Концепція побудови інтерактивного музею у форматі 2.5D

Концепція також передбачає адаптивність - можливість розширення музею новими залами, темами або мультимедійним контентом, що відповідає довгостроковим цілям НУШ щодо безперервного вдосконалення навчального середовища. Обраний підхід також забезпечує гнучкість у підготовці матеріалів - нові експонати можуть бути додані без потреби кардинально змінювати структуру проєкту.

Що можна запозичити для власного проєкту:

- З *Doom Gallery* варто перейняти формат 2.5D та використання простих механік пересування без складного коду керування камерою.
- З *Museum of All Things* - візуальний стиль і підхід до структурування експозиції.
- Обидва проєкти уникають надмірного UI-навантаження та базуються на «атмосферному» геймдизайні - що важливо для збереження фокусу на навчальному контенті.

Адаптація під навчальні дисципліни

Концепція віртуального музею може бути ефективно застосована у шкільних курсах:

- **Історія України** - розміщення копій історичних артефактів або експозиції, присвяченої козацтву;
- **Образотворче мистецтво** - візуальна галерея картин українських художників;
- **Література** - створення інтерактивних кімнат за творами Шевченка або Франка, де експонатами слугують фрагменти текстів і візуальні метафори.

2.2 Архітектура сцени та рух користувача

У проєктуванні інтерактивного середовища на рушії Godot важливу роль відіграє правильна організація сцени та реалізація контролю над переміщенням користувача. Це особливо критично у форматі 2.5D, де

необхідно імітувати просторову глибину, зберігаючи при цьому двовимірну логіку координатної площини.

Godot використовує деревоподібну структуру вузлів (nodes), що дозволяє логічно організовувати ігрові об’єкти (рис. 2.2). Кожен вузол виконує окрему функцію: від рендерингу графіки до обробки фізики, звуку чи логіки взаємодії. Такий підхід дозволяє створювати модульну, гнучку та зрозумілу архітектуру, яку легко масштабувати та модифікувати.

У контексті створення навчального музею структура сцени зазвичай має наступний вигляд:

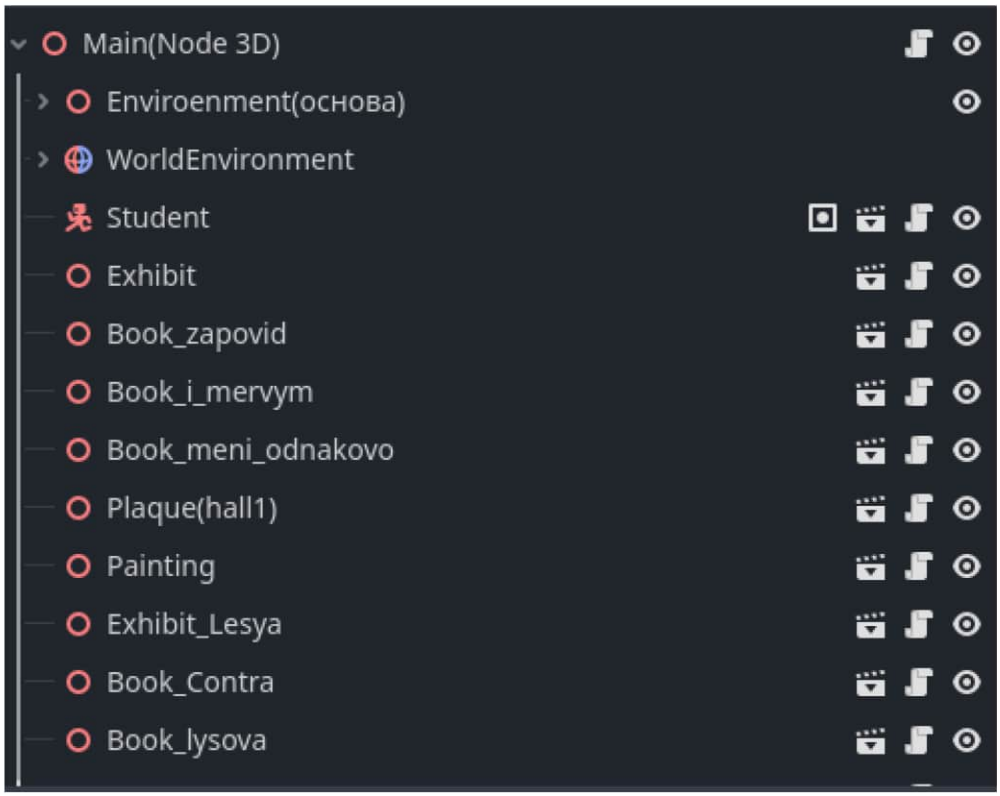


Рисунок. 2.2 - Структура типової сцени інтерактивного музею у форматі 2.5D

У цій структурі MainScene є кореневим елементом усієї локації. Вузол Camera2D забезпечує слідування за гравцем. TileMap використовується для відображення статичного оточення - стін, підлоги, об’єктів інтер’єру. Персонаж гравця (Player) є вузлом типу CharacterBody2D, що дозволяє реалізовувати фізичне пересування з підтримкою колізій.

Усередині Player розташовано вузли, відповідальні за візуальне представлення (Sprite2D), фізичні межі (CollisionShape2D), анімацію (AnimationPlayer) та зону взаємодії (InteractionArea). Остання представлена через Area2D і дозволяє виявляти об’єкти, з якими гравець може взаємодіяти (наприклад, інформаційні стенди чи експонати музею).

Інтерфейс користувача (UI) винесено в окремий шар (CanvasLayer), що дозволяє фіксувати елементи на екрані незалежно від позиції камери. Тут можуть відображатись повідомлення, діалоги, зображення чи відеофрагменти. Для відтворення звуків і музики застосовується AudioStreamPlayer.

Група InteractiveObjects містить усі елементи, з якими гравець може взаємодіяти - це можуть бути як статичні інформаційні таблички, так і мультимедійні екрани або персонажі.

Такий підхід дозволяє логічно структурувати сцену, зберігати чистоту коду та гнучко керувати як ігровою логікою, так і навчальним контентом. У наступних підрозділах буде розглянуто кожен із компонентів цієї структури детальніше - зокрема, принципи компонування простору, організацію переміщення гравця, реалізацію колізій та інтерактивності.

2.2.1 Ієрархія вузлів у Godot

Розглядаючи процес оптимізації структури гри, розробленої на базі ігрового рушія Godot, особливу увагу приділено аналізу ієрархії вузлів, яка є ключовим елементом організації сцени. Ієрархія вузлів визначає взаємодію об’єктів у межах ігрового простору та впливає на ефективність обробки даних у сцені, що є важливим для забезпечення стабільної роботи проєкту.

Ієрархія вузлів у Godot базується на деревоподібній структурі, де кожен вузол є складовою частиною сцени та може мати дочірні вузли (рис. 2.3). Кореневий вузол сцени, наприклад, Node або Node3D, слугує основою для побудови всієї структури. У нашому випадку для сцени main(node_3d).tscn кореневий вузол є Node3D, що забезпечує тривимірний простір для об’єктів музею. Дочірні вузли, такі як Camera3D, MeshInstance3D та CollisionShape3D,

формують геометрію, візуальні елементи та фізичні об’єкти для взаємодії в межах сцени .

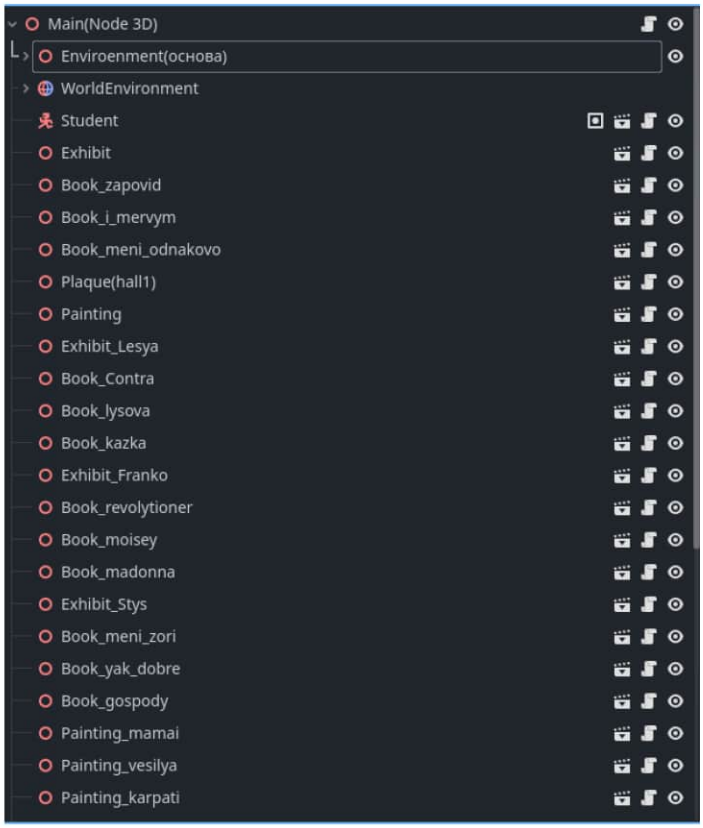


Рисунок. 2.3 - Ієрархія вузлів у сцені main(node_3d).tscn

Аналіз ієрархії показує, що правильне вкладення вузлів впливає на продуктивність рендерингу та логіку взаємодії об’єктів. Наприклад, розміщення Camera3D як дочірнього вузла Node3D забезпечує синхронізацію камери з рухом сцени, що є критичним для плавності відображення (рис. 2.4). Кількість дочірних вузлів та їхній рівень вкладеності також впливають на час обробки подій, що може призводити до затримок у разі надмірної складності структури.

Для оптимізації запропоновано структурувати ієрархію наступним чином:

1. Кореневий вузол Node3D як основа сцени.
2. Вузол Camera3D для управління зором гравця.
3. Вузли MeshInstance3D для геометрії об’єктів музею.

4. Вузли CollisionShape3D для визначення фізичних меж.

Така організація дозволяє мінімізувати накладні витрати на обробку сцени та забезпечити стабільну роботу гри. Результати тестування ієрархії будуть розглянуті в подальших підрозділах для оцінки її впливу на продуктивність.

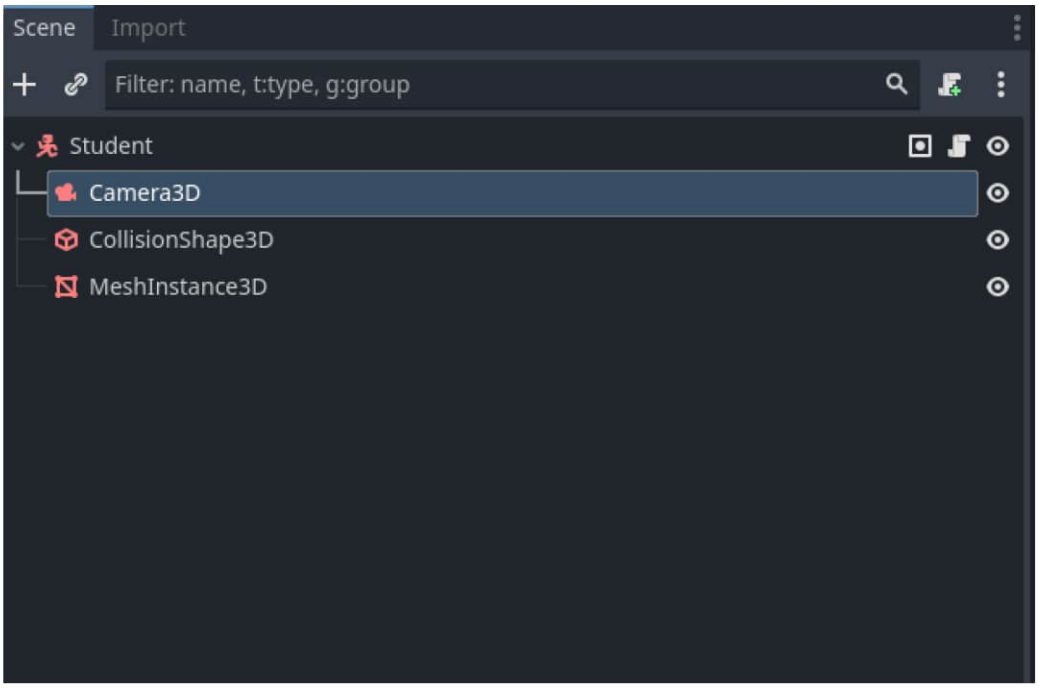


Рисунок. 2.4 - Приклад вкладення Camera3D у кореневий вузол Node3D

2.2.2 Просторове компонування та камери

Розглядаючи оптимізацію структури гри, розробленої на основі ігрового рушія Godot, значну увагу приділено просторовому компонуванню об’єктів та ролі камери в організації ігрового простору. Просторове компонування визначає розміщення та взаємодію тривимірних об’єктів у сцені, тоді як камера забезпечує візуальну перцепцію гравця, що є ключовим для створення зрозумілого ігрового досвіду.

Просторове компонування у сцені реалізовано через використання вузлів MeshInstance3D для представлення геометрії об’єктів музею та CollisionShape3D для визначення їхніх фізичних меж (рис. 2.5). Ці вузли інтегровані в окрему сцену Student, де кореневий вузол CharacterBody3D

слугує основою для моделі гравця. У межах цієї сцени Camera3D виступає як "очі" гравця, а MeshInstance3D та CollisionShape3D забезпечують візуальне зображення та фізичну взаємодію. Розташування об’єктів у тривимірному просторі здійснюється через налаштування координат у властивостях вузлів, що дозволяє створювати реалістичну структуру сцени. Оптимізація компонування полягає у групуванні цих елементів для мінімізації перетинів колізійних об’єктів та ефективного рендерингу.

Камера, представлена вузлом Camera3D, інтегрована в сцену Student як дочірній елемент CharacterBody3D і керується скриптом student.gd, який реалізує рух та орієнтацію гравця. Скрипт базується на класі CharacterBody3D і включає параметри, такі як базова швидкість (speed := 5.0), множник спринту (sprint_multiplier := 1.5), чутливість миші (mouse_sensitivity := 0.002), прискорення (acceleration := 10.0) та гальмування (friction := 8.0). Рух камери по вертикалі (camera_rot_x) обмежено діапазоном від -85° до 85°, що запобігає перевертанню зображення. Горизонтальний поворот гравця реалізовано через функцію rotate_y, а обробка вводу миші та клавіатури (включно зі спринтом при натисканні Shift) забезпечує плавне управління. Функція _physics_process використовує вектор input_dir для визначення напрямку руху, а метод move_and_slide() забезпечує фізичне переміщення з урахуванням зіткнень. Такий підхід оптимізує взаємодію камери з простором сцени, забезпечуючи стабільність ігрового процесу.

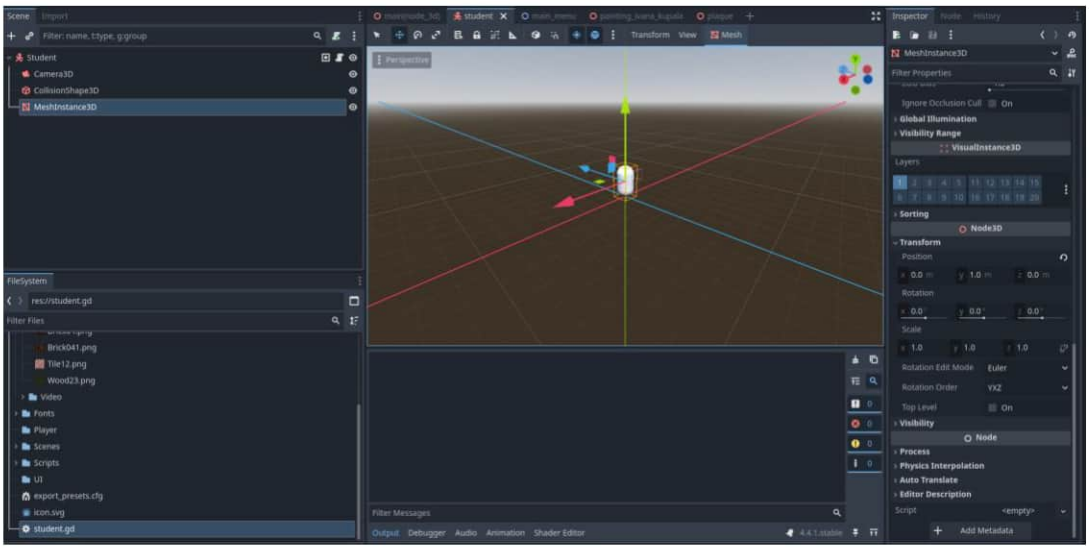


Рисунок. 2.5 - Просторове компонування об’єктів у сцені Student

Аналіз показує, що просторове компонування впливає на час обробки фізичних розрахунків, тоді як налаштування камери через скрипт оптимізує візуальну складову та реакцію на ввід. Запропоновано наступний підхід:

1. Розташування MeshInstance3D та CollisionShape3D у сцені Student для інтеграції моделі гравця.
2. Налаштування Camera3D (рис. 2.6) через скрипт student.gd для плавного слідкування та обмеження поворотів.
3. Перевірка параметрів руху (швидкість, прискорення) для адаптації до різних сценаріїв гри.

Така організація сприяє підвищенню продуктивності сцени та стабільності гри. Результати тестування будуть розглянуті в подальших розділах для оцінки ефективності запропонованих змін.

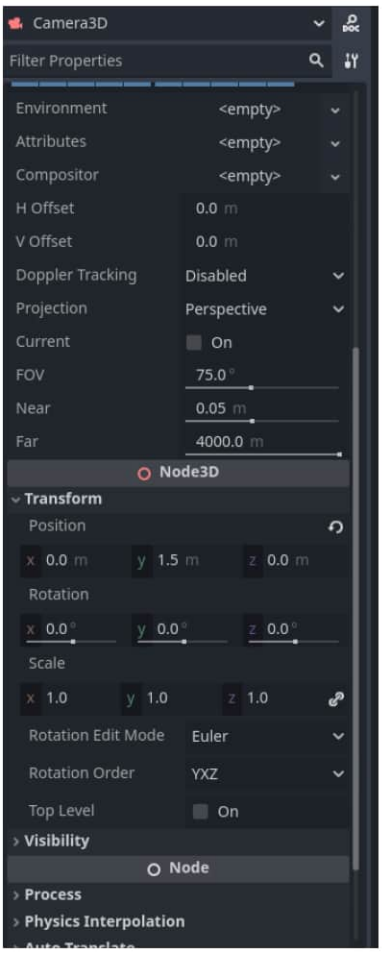


Рисунок. 2.6 - Налаштування Camera3D та її взаємодія з CharacterBody3D

2.2.3 Рух гравця у 2.5D середовищі

Розробка руху гравця у 2.5D середовищі є ключовим елементом створення інтерактивного навчального музею на базі рушія Godot, визначаючи комфортність навігації та залученість учня до освітнього процесу. Основна сцена світу, представлена вузлом Main(Node3D), слугує кореневою структурою, куди інтегровано сцену Student як дочірню через інстанціювання. У цій структурі (рис. 2.7) рух гравця реалізовано через вузол MeshInstance3D, який слугує візуальним представником персонажа, на відміну від CharacterBody3D, що забезпечує фізичну взаємодію.

Рух базується на скрипті student.gd, де координати MeshInstance3D оновлюються залежно від введення користувача. Швидкість переміщення задано через змінну speed := 5.0, а плавність досягається через лінійну інтерполяцію (lerp) з прискоренням acceleration := 10.0 та гальмуванням

friction := 8.0. У 2.5D середовищі рух обмежено площиною XZ (з input_dir.y = 0), що підтримує ілюзію глибини через візуальні ефекти, як-от перспектива. Спринт, активований множником sprint_multiplier := 1.5 при натисканні Shift, додає динаміки, відповідаючи принципам гейміфікації НУШ.

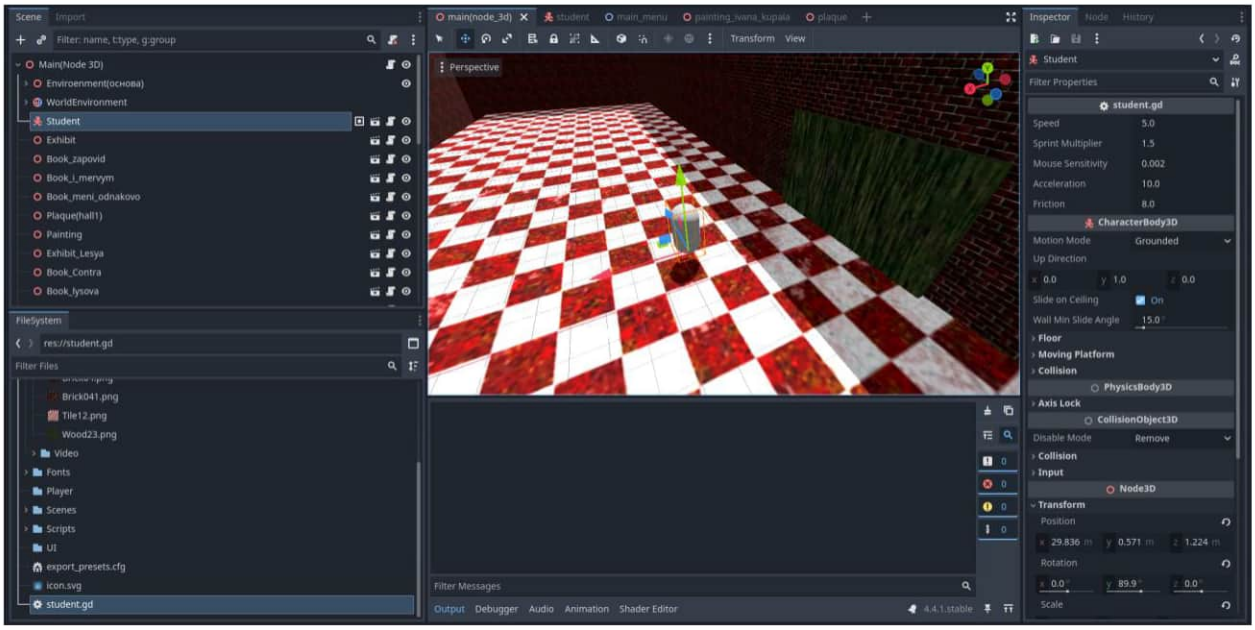


Рисунок. 2.7 - Структура руху гравця з MeshInstance3D у сцені Student всередині Main(Node3D)

Камера, реалізована через Camera3D у сцені Student, синхронізується з рухом MeshInstance3D у межах Main(Node3D). Вертикальний поворот (camera_rot_x) обмежено від -85° до 85°, а горизонтальний - через rotate_y за рухом миші, забезпечуючи інтуїтивне керування. Оскільки MeshInstance3D не має вбудованої фізики, зіткнення з об’єктами CollisionShape3D у Main(Node3D) обробляються через окремі перевірки в скрипті, уникаючи проникнення через стіни.

Оптимізація продуктивності досягається групуванням статичних об’єктів музею в StaticBody3D у сцені Main(Node3D), що зменшує навантаження на слабкі пристрої. Такий підхід підтримує стабільну частоту кадрів, що було перевірено на попередніх етапах. Рух гравця сприяє активному дослідженню експонатів, відповідаючи педагогічним цілям НУШ.

2.2.4 Колізії, обмеження та керування взаємодією

Реалізація колізій, обмежень та керування взаємодією є ключовим елементом створення інтерактивного навчального музею на базі рушія Godot, оскільки ці аспекти забезпечують логічну взаємодію гравця з об’єктами сцени та підтримують ефективність навчального процесу. Основна сцена Main(Node3D) слугує базовою структурою, до якої інтегровано численні дочірні сцени, зокрема Student і exhibit, через інстанціювання. Колізії реалізовано за допомогою вузлів CollisionShape3D (рис. 2.8), асоційованих зі статичними об’єктами, такими як стіни, підлога та експонати, згрупованими в StaticBody3D у відповідних сценах. Ця організація дозволяє ефективно управляти великою кількістю об’єктів, імпортованих як дочірні сцени, включаючи декоративні стіни, інтерактивні стенди та інші елементи.

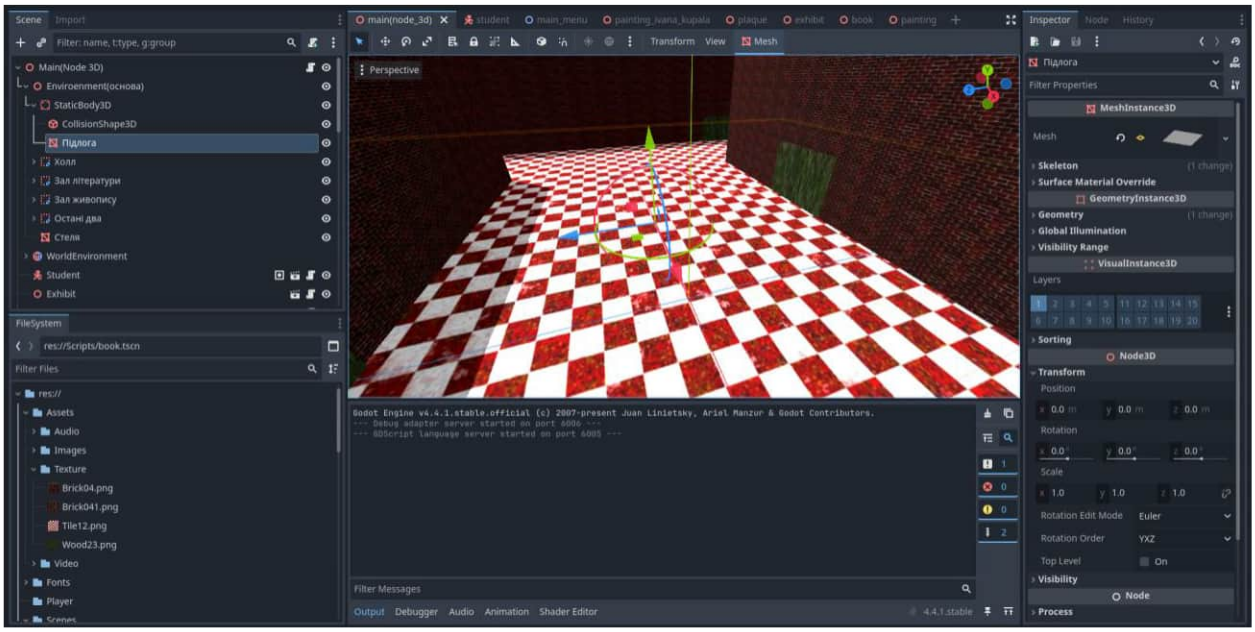


Рисунок. 2.8 - Схема колізій MeshInstance3D з CollisionShape3D у сцені Main(Node3D)

Колізійна система в сценарії student.gd базується на перевірці перетину MeshInstance3D гравця з CollisionShape3D. Функція move_and_collide() коригує позицію гравця, уникаючи проникнення через перешкоди, що

особливо важливо для роботи з імпортованими сценами. Обмеження руху реалізовано шляхом блокування напрямків, коли вектор input_dir стикається з межами CollisionShape3D. Цей підхід забезпечує реалістичне відчуття бар'єрів, таких як стіни чи декоративні елементи, додані через дочірні сцени. Наприклад, у попередніх тестах карта містила десятки об'єктів - від простих стін до складних інтерактивних експонатів, імпортованих із окремих сцен, і система колізій успішно справлялася з їхньою обробкою, уникаючи накладань. Це критично для 2.5D середовища, де ілюзія глибини підтримується візуальними ефектами, такими як тіні та перспектива.

Керування взаємодією реалізовано через вузол Area3D, інтегрований у сцену exhibit, де використано StaticBody3D як основний об'єкт для представлення експонатів. Зона виявлення (InteractionArea) активується при наближенні MeshInstance3D гравця до StaticBody3D з Area3D, викликаючи дії, такі як відображення текстової інформації, відтворення аудіо чи запуск анімацій. Наприклад, при підході до стенду з описом творів українських письменників, імпортованого як сцена exhibit (рис. 2.9), система може відобразити деталі про їхні роботи. Кількість одночасних активних зон обмежено (наприклад, до 5 об'єктів), що зменшує навантаження на процесор і було протестовано на карті з великою кількістю імпортованих сцен.

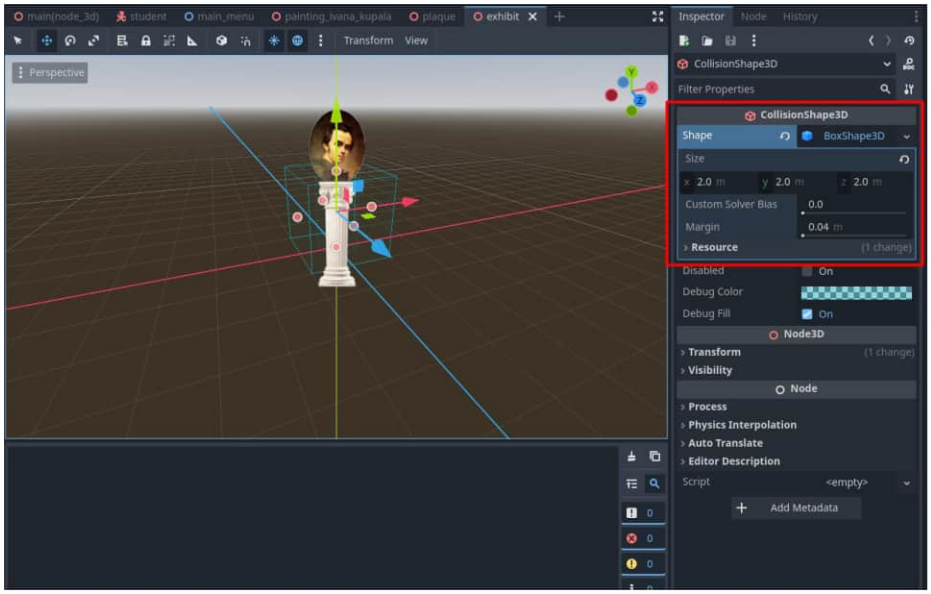


Рисунок. 2.9 - Налаштування зони Area3D у сцені exhibit з StaticBody3D

Система колізій і взаємодії адаптована до роботи з дочірніми сценами, де об’єкти різного типу - стіни, двері, інтерактивні панелі - імпортовані з окремих файлів і потребують точного виявлення. Оптимізація продуктивності досягається завдяки групуванню статичних об’єктів у StaticBody3D у межах кожної сцени, що стабілізує частоту кадрів навіть на пристроях із обмеженими ресурсами, відповідаючи вимогам доступності НУШ. Аналіз показує, що реалізована система забезпечує комфортну навігацію та підтримує інтерактивне дослідження експонатів, сприяючи педагогічним цілям НУШ (рис 2.10).

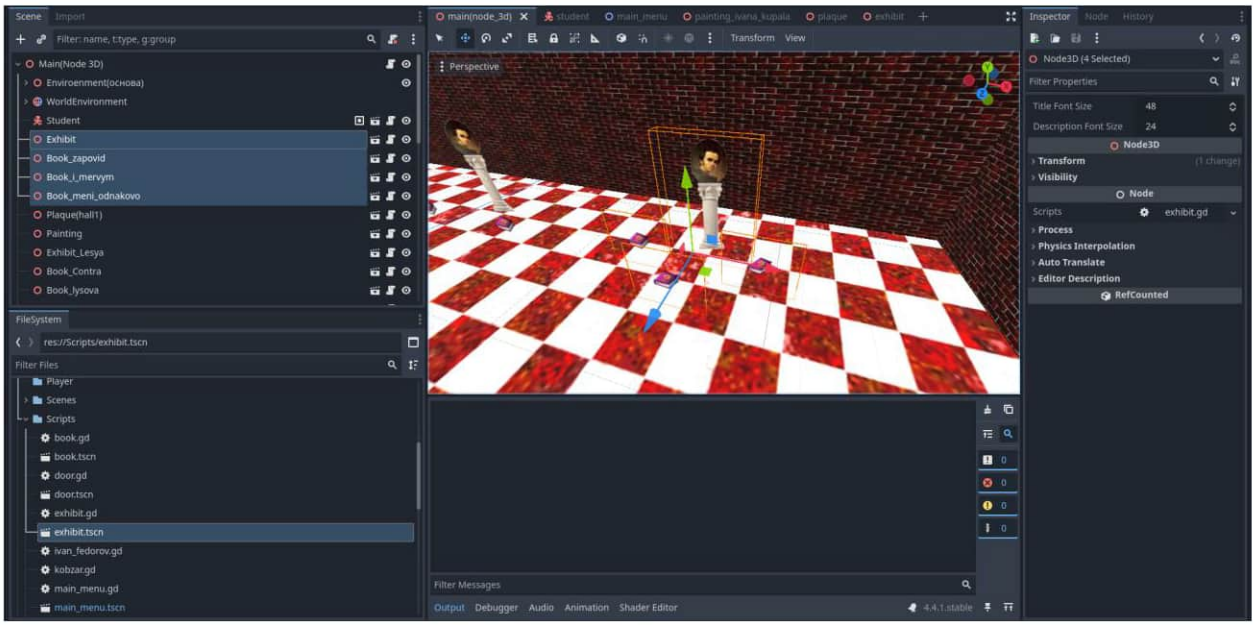


Рисунок. 2.10 - Приклад обробки колізій із кількома імпортованими сценами на карті

2.2.5 Контроль дій

Контроль дій є основою для комфортної взаємодії гравця в інтерактивному навчальному музеї на базі рушія Godot, визначаючи, як учень переміщається та взаємодіє з об’єктами сцени. У цьому підрозділі розглянуто реалізацію механізмів управління через скрипт student.gd, прив’язаний до вузла CharacterBody3D, який забезпечує фізичне переміщення в 2.5D

середовищі. Скрипт адаптовано до потреб учнівського середовища, враховуючи рух, повороти та зручність керування.

Реалізація базується на обробці ввідних даних із клавіатури та миші. Основні параметри визначено як експортовані змінні: speed := 5.0 для базової швидкості, sprint_multiplier := 1.5 для спринту (активується Shift), mouse_sensitivity := 0.002 для чутливості миші, acceleration := 10.0 для прискорення та friction := 8.0 для гальмування. Змінна camera_rot_x := 0.0 контролює вертикальний поворот камери, прив'язаної через @onready var camera = $Camera3D.

У функції _ready() миша захоплюється режимом Input.MOUSE_MODE_CAPTURED для точного керування, що дозволяє уникати виведення курсору за екран. Функція _input(event) обробляє повороти: горизонтальний через rotate_y(-event.relative.x * mouse_sensitivity), а вертикальний - через camera_rot_x (рис. 2.11) , обмежений clamp у діапазоні від -85° до 85° (перетворено в радіани), із синхронізацією camera.rotation.x. Клавіша Esc (ui_cancel) перемикає мишу в видимий режим для паузи.

```
15  func _input(event):
16      if event is InputEventMouseMotion:
17          # Обертання гравця по горизонталі (вліво/вправо)
18          rotate_y(-event.relative.x * mouse_sensitivity)
19
20          # Обертання камери по вертикалі (вгору/вниз)
21          camera_rot_x = clamp(
22              camera_rot_x - event.relative.y * mouse_sensitivity,
23              deg_to_rad(-85), deg_to_rad(85)
24          )
25          camera.rotation.x = camera_rot_x
26
```

Рисунок. 2.11 - Налаштування камери з обмеженням camera_rot_x

У _physics_process(delta) формується вектор input_dir (рис. 2.12) на основі дій: move_forward, move_back, move_left, move_right, прив'язаних до осей transform.basis. Компонента y скидається до 0 для обмеження руху площиною. Спринт множить швидкість на sprint_multiplier, а плавне прискорення досягається через velocity.lerp(target_velocity, acceleration * delta). Гальмування активується, якщо input_dir == Vector3.ZERO, за допомогою velocity.lerp(Vector3.ZERO, friction * delta). Метод move_and_slide() застосовує рух із урахуванням колізій.

```
31  func _physics_process(delta):
32      var input_dir = Vector3.ZERO
33
34      # Ввід напрямку руху
35      if Input.is_action_pressed("move_forward"):
36          input_dir -= transform.basis.z
37      if Input.is_action_pressed("move_back"):
38          input_dir += transform.basis.z
39      if Input.is_action_pressed("move_left"):
40          input_dir -= transform.basis.x
41      if Input.is_action_pressed("move_right"):
42          input_dir += transform.basis.x
43
44      input_dir.y = 0
45      input_dir = input_dir.normalized()
46
```

Рисунок. 2.12 - Схема руху з вектором input_dir у сцені Student

Оптимізація включає нормалізацію input_dir для однакової швидкості, мінімалізацію обчислень через lerp і обмеження поворотів clamp. Перевірка працездатності проходить завдяки створеню тестового маршруту в головній сцені (Node3D) (рис.2.13)

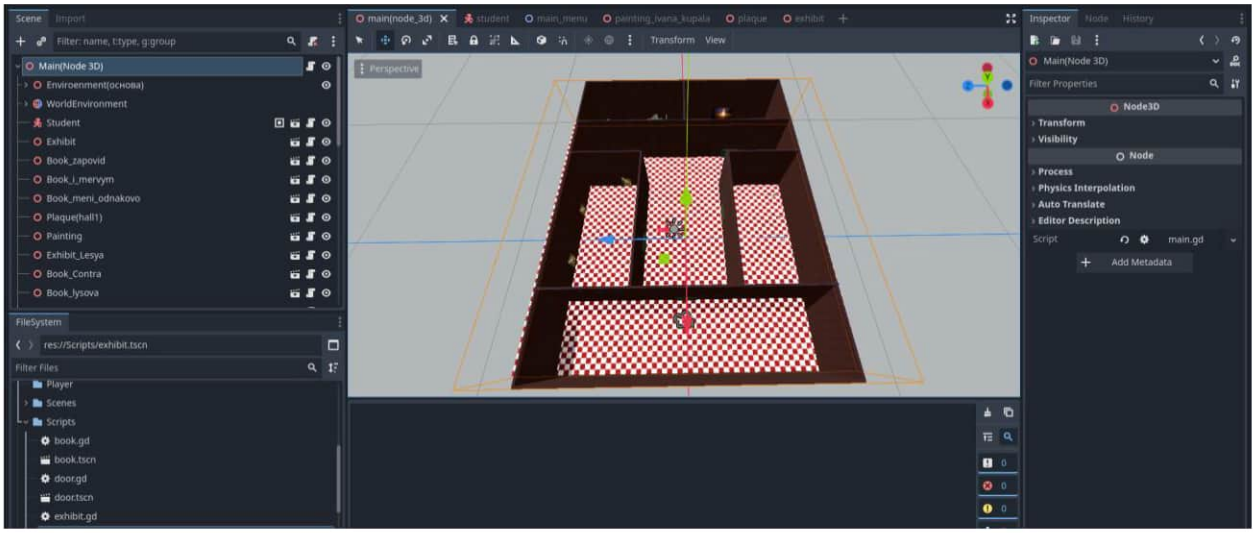


Рисунок. 2.13 - Тестовий маршрут у сцені Main(Node3D)

2.3 Реалізація інтерактивності

Реалізація інтерактивності була визначена як ключовий компонент інтерактивного навчального музею на базі рушія Godot, оскільки вона забезпечувала активну участь учнів у навчальному процесі через взаємодію з віртуальним середовищем. У цьому розділі було детально описано технічні аспекти створення інтерактивних елементів, їх інтеграцію в основну сцену Main(Node3D), де були розміщені експонати (Painting, Exhibit, Book_zapovid, Statue_Kobzar, Wreath_IvanaKupala, Vyshyvanka, Statue_IvanFedorov), та їхній вплив на педагогічні цілі Нової української школи (НУШ).

Інтерактивність була реалізована через вузли Area3D, додані до експонатів у сцені Main(Node3D). Кожна сцена-експонат, наприклад, Painting чи Statue_Kobzar, містила зону Area3D (рис 2.14) з прив'язаною CollisionShape3D, розміри якої налаштовувалися відповідно до об'єкта (наприклад, 2x2x1 м для Painting). Сигнали body_entered і body_exited у скриптах, прив'язаних до Area3D, активувалися при наближенні MeshInstance3D гравця (сцена Student). При вході в зону, наприклад, у Statue_Kobzar, було реалізовано виведення повідомлення про наближення.

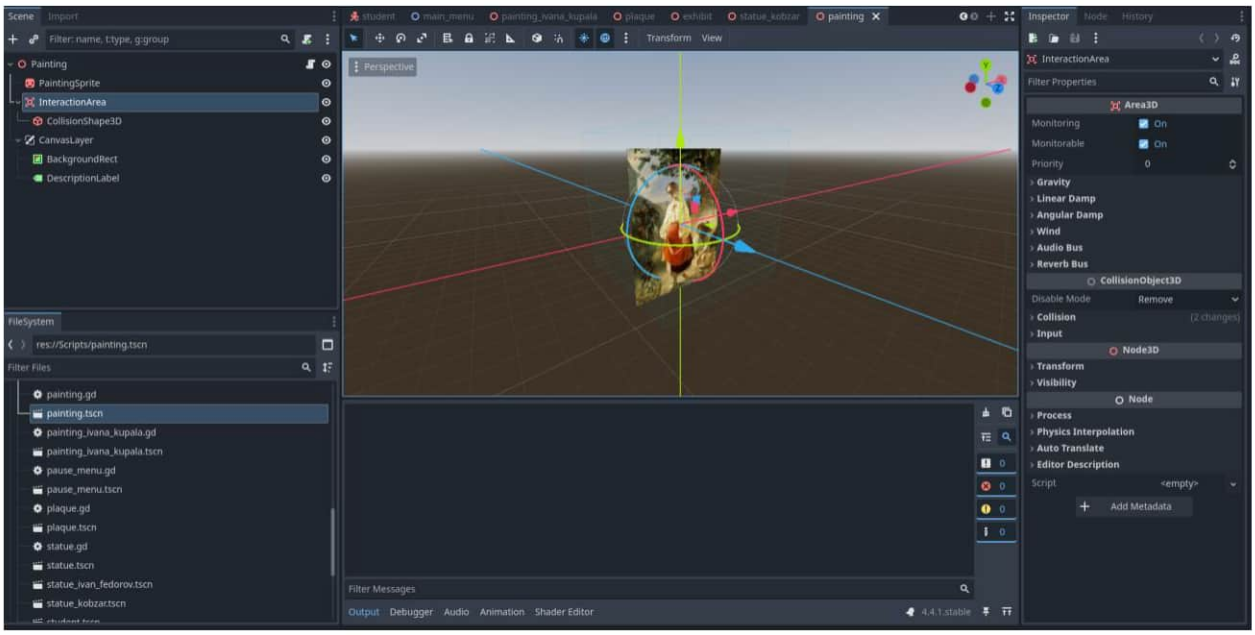


Рисунок. 2.14 - Налаштування зони Area3D у Painting в Main(Node3D)

Приклад коду для Painting:

```
func _on_area_3d_body_entered(body):
    if body.name == "Student":
        print("Гравець увійшов у зону картини")
```

Управління взаємодією було реалізовано через клавішу E (interact) у скрипті student.gd, де RayCast використовувався для сканування зон Area3D у Main(Node3D). Натиснення E викликало дію, наприклад, відкриття вікна в Book_zapovid (рис. 2.15) . Було проведено тестування цього механізму на всіх сценах, адаптуючи зони до форми експонатів.

Для виведення інформації було інтегровано текст через вузол Label у CanvasLayer (наприклад, опис художника в Painting) і зображення через TextureRect (фото вишиванки в Vyshyvanka), використовуючи PNG-файли з роздільною здатністю 512x512 пікселів. Шрифти налаштовувалися як DynamicFont із пакетом Noto Serif (розмір 16-20 pt), із застосуванням кешування для економії ресурсів.

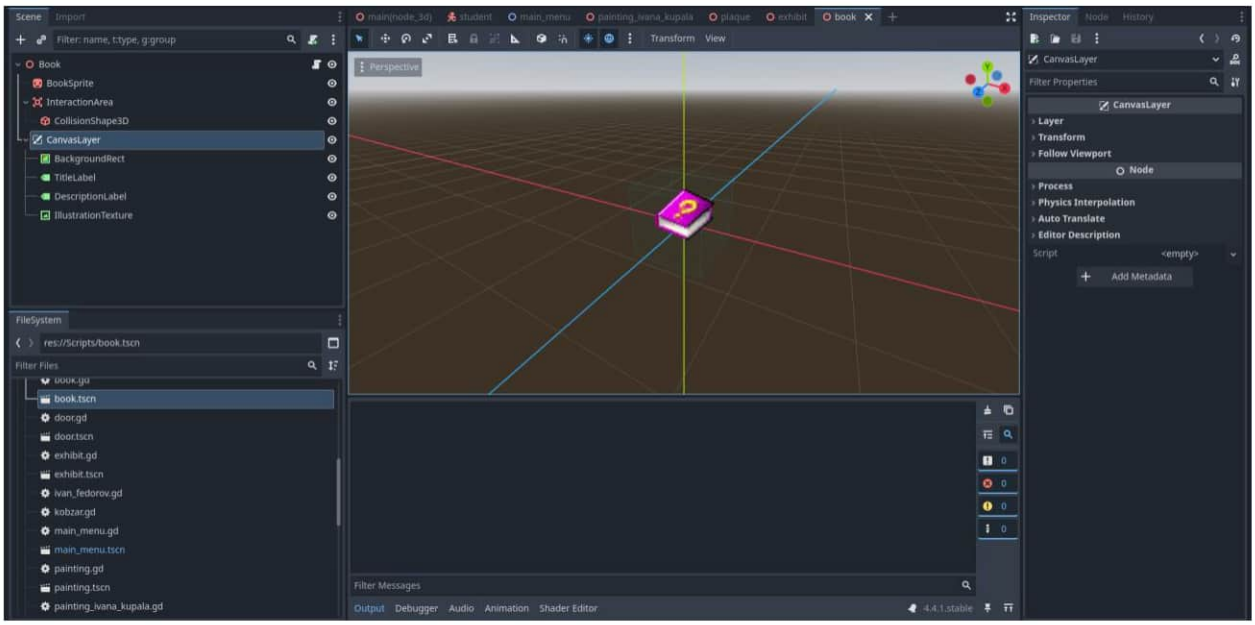


Рисунок 2.15 - Вивід тексту в Book_zapovid

Мультимедіа було реалізовано через AudioStreamPlayer3D (аудіо вірша в Statue_Kobzar (рис. 2.16), MP3 128 kbps, із обмеженням гучності до 50%) і VideoStreamPlayer (відео в Exhibit, WebM 320x240). Відтворення активувалося при вході в зону або натисканні E, із застосуванням асинхронного завантаження для уникнення затримок. Було обмежено одночасне відтворення (до 2 аудіо) для оптимізації.

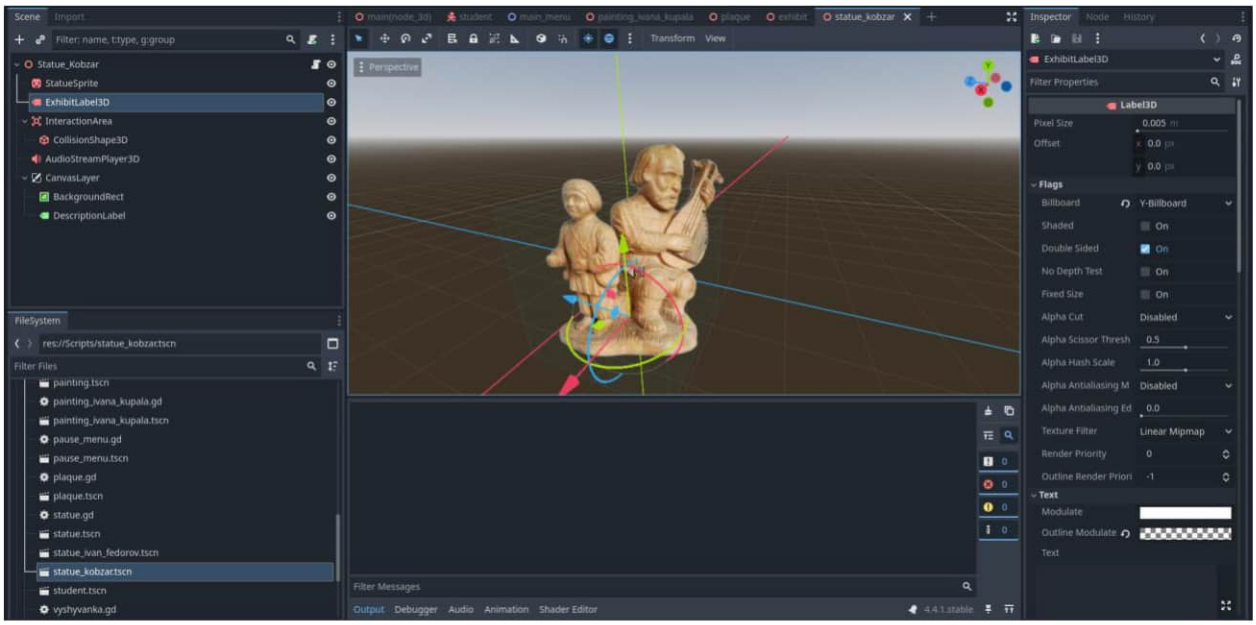


Рис. 2.16 - Відтворення аудіо в Statue_Kobzar

Інтеграція в Main(Node3D) базувалася на групуванні об’єктів у StaticBody3D для забезпечення стабільності. Кількість зон взаємодії було обмежено (до 5) для уникнення перевантаження. Ця система сприяла реалізації педагогічних цілей НУШ, мотивуючи учнів до дослідження через інтерактивні дії, такі як вікторини в Exhibit.

2.3.1 Зони взаємодії та тригери

Одним із основних принципів побудови інтерактивного освітнього простору є створення зон взаємодії, які забезпечують адресне реагування системи на дії користувача. У межах проєкту інтерактивного музею всі ключові об'єкти - книги, таблички, картини, двері - реалізовані як активні вузли з просторовими зонами реагування. Для цього у Godot використовується вузол Area3D разом із пов’язаними сигналами body_entered, body_exited та обробкою подій введення через функцію _input().

Кожен експонат є самостійною сценою (PackedScene), яка включає:

- InteractionArea (Area3D) із CollisionShape3D;
- окремий візуальний або текстовий блок у CanvasLayer;
- логіку взаємодії у скрипті (.gd).

Приклад 1. Книга (book.tscn)

Книга містить назву, розгорнутий опис і ілюстрацію. У скрипті book.gd реалізовано логіку, що дозволяє гравцеві, перебуваючи всередині зони, натиснути клавішу interact (E), щоби побачити вміст:

```
if player_inside and event.is_action_pressed("interact"):
      if title_label.modulate.a > 0:
            # Приховати інтерфейс
      else:
            # Показати інтерфейс з даними книги
```

Зміна прозорості елементів UI (назва, опис, фон, ілюстрація) здійснюється плавно через Tween, що створює зручний візуальний перехід. Окремо реалізовано перевірки на можливе перекриття елементів інтерфейсу в

режимі відладки (is_debug_build()), що спрощує тестування при додаванні нових експонатів.

Приклад 2. Табличка-зала (plaque.tscn)

На перший погляд, цей об’єкт виконує лише інформативну функцію - подає назву зали. Однак, як і книга, він вимагає активного натискання клавіші E, щоб користувач зміг прочитати детальний опис зали:

```
if player_inside and event.is_action_pressed("interact"):
      if description_label.modulate.a > 0:
            # Приховати опис
      else:
            # Показати опис
```

У момент входу гравця в зону табличка змінює текст на підказку "Натисни E", а при виході - повертається до початкової назви зали. Така динаміка підказок забезпечує природну інтуїтивну взаємодію. Схематично це представлено на рисунку 2.17.

Загальна структура зони взаємодії

Типова структура сцени експоната виглядає так:

```
[Експонат] (Node3D)
├── InteractionArea (Area3D)
│   └── CollisionShape3D
└── CanvasLayer
    ├── Назва/Опис (Label/TextureRect)
    └── BackgroundRect (ColorRect)
```

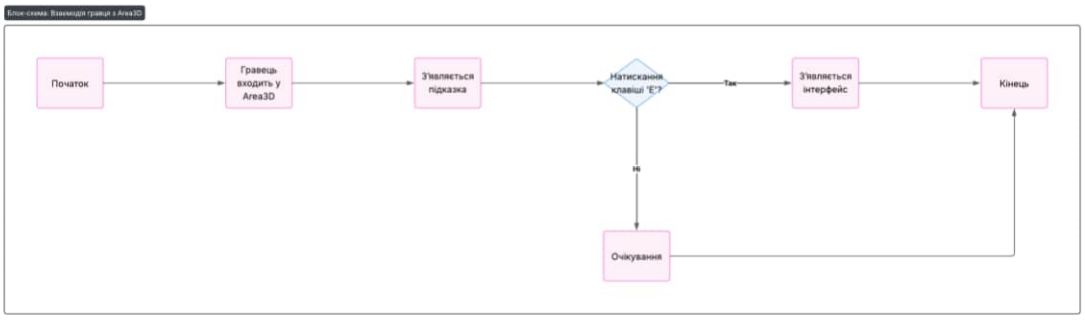


Рисунок. 2.17 - Схема взаємодії гравця з об'єктом через зону Area3D та клавішу дії

Типи взаємодій

Хоча всі експонати реагують на клавішу interact, за характером вмісту та способами активації їх можна класифікувати наступним чином:

- Текстові - таблички з поясненнями або назвами залів.
- Ілюстративні - книги й картини з описом та ілюстраціями.
- Мультимедійні - експонати, що після тексту пропонують переглянути відео або прослухати звук (наприклад, через натискання додаткової клавіші z чи p).
- Функціональні - автоматичні двері, що відкриваються при вході гравця в зону та закриваються після його виходу.

Автоматичні двері (door.tscn) мають власну зону DetectionArea (типу Area3D), до якої підключено сигнали body_entered та body_exited. При вході гравця викликається метод open_door(), який:

- піднімає дверне полотно анімаційно за допомогою Tween;
- вимикає колізію, дозволяючи пройти далі;
- вмикає звуковий ефект (якщо в сцені наявний AudioStreamPlayer3D).

Після виходу гравця з тригерної зони виконується перевірка, чи вона порожня (has_overlapping_bodies()), і якщо так - запускається метод close_door(), який:

- опускає двері у початкове положення;
- вмикає колізію;
- відтворює звук закриття.

Це забезпечує безперервну логіку переміщення між просторами музею без потреби в натисканні клавіш, що добре підходить саме для переходів, а не для змістових експонатів.

2.3.2 Вивід текстової та графічної інформації

Одним із центральних завдань віртуального музею є надання користувачеві доступу до інформації про експонати у зручній і візуально привабливій формі. Для цього в проєкті реалізовано систему виводу текстового і графічного контенту, яка є частиною інтерактивної логіки кожного експоната. Такий підхід дозволяє не лише ознайомлювати користувача з навчальними матеріалами, але й формувати логічну структуру навігації простором музею.

Візуальні компоненти інтерфейсу

Для реалізації виводу інформації використовується комбінація стандартних вузлів Godot, зокрема:

- Label3D - для виводу простих написів у просторі (наприклад, назва зали);
- Label - для розміщення тексту в 2D-шарі інтерфейсу;
- TextureRect - для демонстрації зображень або ілюстрацій;
- ColorRect - як напівпрозоре тіло для покращення читабельності;
- CanvasLayer - для розміщення всіх вищезазначених елементів незалежно від основної камери.

Усі ці вузли групуються в межах сцени експоната або інформативного блоку. Наприклад, у book.tscn структура виглядає наступним чином:

```
Book (Node3D)
├── InteractionArea (Area3D)
└── CanvasLayer
    ├── TitleLabel (Label)
    ├── DescriptionLabel (Label)
    ├── IllustrationTexture (TextureRect)
    └── BackgroundRect (ColorRect)
```

Текстові елементи

У межах сцени книги або таблички виводиться два основних типи тексту: заголовок і опис. Вони формуються через вузли Label, які підтримують стилізацію (розмір шрифту, колір, прозорість) та задаються через експортовані змінні скриптів:

```
@export var book_title = "Book Title"
@export var book_description = "This is a placeholder text for the book!"
```

У функції _ready() задаються розміри шрифту:

```
title_label.add_theme_font_size_override("font_size", title_font_size)
description_label.add_theme_font_size_override("font_size",
description_font_size)
```

При натисканні клавіші interact текст поступово з’являється через Tween, що керує прозорістю (modulate.a):

```
tween.tween_property(title_label, "modulate:a", 1, 0.3)
tween.tween_property(description_label, "modulate:a", 1, 0.3)
```

У режимі розробки реалізовано систему налагодження, яка попереджає про можливе перекриття тексту з іншими елементами (наприклад, ілюстрацією), що підвищує якість візуального контенту:

```
if desc_rect.intersects(illus_rect):
      print("ПОМИЛКА: Опис перекриває ілюстрацію")
```

Графічні компоненти

Для демонстрації зображень використовується вузол TextureRect, до якого прив’язується текстура, що експортується з редактора:

```
@export var illustration: Texture2D
```

Як і з текстом, для зображення застосовується поступова поява та зникнення з допомогою Tween, що забезпечує єдність стилю анімації:

```
tween.tween_property(illustration_texture, "modulate:a", 1, 0.3)
```

У випадку відсутності ілюстрації, текстура автоматично обнуляється:

```
if illustration:
      illustration_texture.texture = illustration.duplicate()
else:
      illustration_texture.texture = null
```

Вивід мультимедійного вмісту

У деяких експонатах після виводу опису пропонується переглянути відео або прослухати аудіо. Такий варіант подачі реалізується поетапно: спочатку відображається текст із інструкцією натиснути клавішу (z, p тощо), після чого активується відтворення відповідного ресурсу (через VideoPlayer або AudioStreamPlayer), інтегрованого в сцену. Це дозволяє уникнути одночасного перевантаження користувача кількома каналами сприйняття.

Підхід до дизайну

Оформлення всіх текстових і графічних елементів витримано у єдиному візуальному стилі:

- напівпрозоре тло (ColorRect) забезпечує контрастність незалежно від фону сцени;
- фіксовані розміри шрифтів дозволяють зберігати читабельність;
- плавна анімація появи підвищує сприйняття контенту та усуває візуальні “стрибки”.

Такий підхід формує чітку, доступну й послідовну подачу інформації, що особливо важливо для освітнього продукту. Оскільки подання відбувається не одразу, а лише після ініціативи користувача, зберігається концентрація уваги та створюється ефект поступового розкриття знань.

2.3.3 Робота з мультимедіа (аудіо, відео, зображення, шрифти)

Важливою складовою інтерактивного навчального середовища є мультимедійна подача матеріалу. У проєкті віртуального музею реалізовано можливість **озвучення окремих експонатів** та **відтворення відео**, що розширює канали сприйняття і підсилює емоційне занурення користувача в тему. Аудіо- та відеоінтеграція виконуються у контексті вже знайомої системи зон взаємодії, з опорою на AudioStreamPlayer3D і VideoStreamPlayer.

Аудіо: приклад Statue_Kobzar

Сцена Statue_Kobzar (рис. 2.18) є типовим прикладом експоната з аудіосупроводом. Об’єкт містить просторову модель або зображення кобзаря, короткий текстовий опис, а також функцію відтворення фрагмента кобзарської пісні.

У скрипті kobzar.gd визначено змінні для тексту та аудіофайлу:

```
@export var description_text = "... Натисни клавішу З щоб послухати"
@export var audio_stream: AudioStream
```

Аудіоплеєр (audio_player) активується лише тоді, коли користувач, перебуваючи в зоні взаємодії, натискає клавішу З (латинська P). Реалізовано це у функції _input():

```
if player_inside and event.is_action_pressed("play_music"):
      if audio_player:
            if not audio_player.playing:
                  audio_player.play()
            else:
                  audio_player.stop()
```

Таким чином, користувач самостійно вирішує, чи вмикати звук, що дозволяє уникати нав’язливості, а також відповідає принципам інклюзивного дизайну.

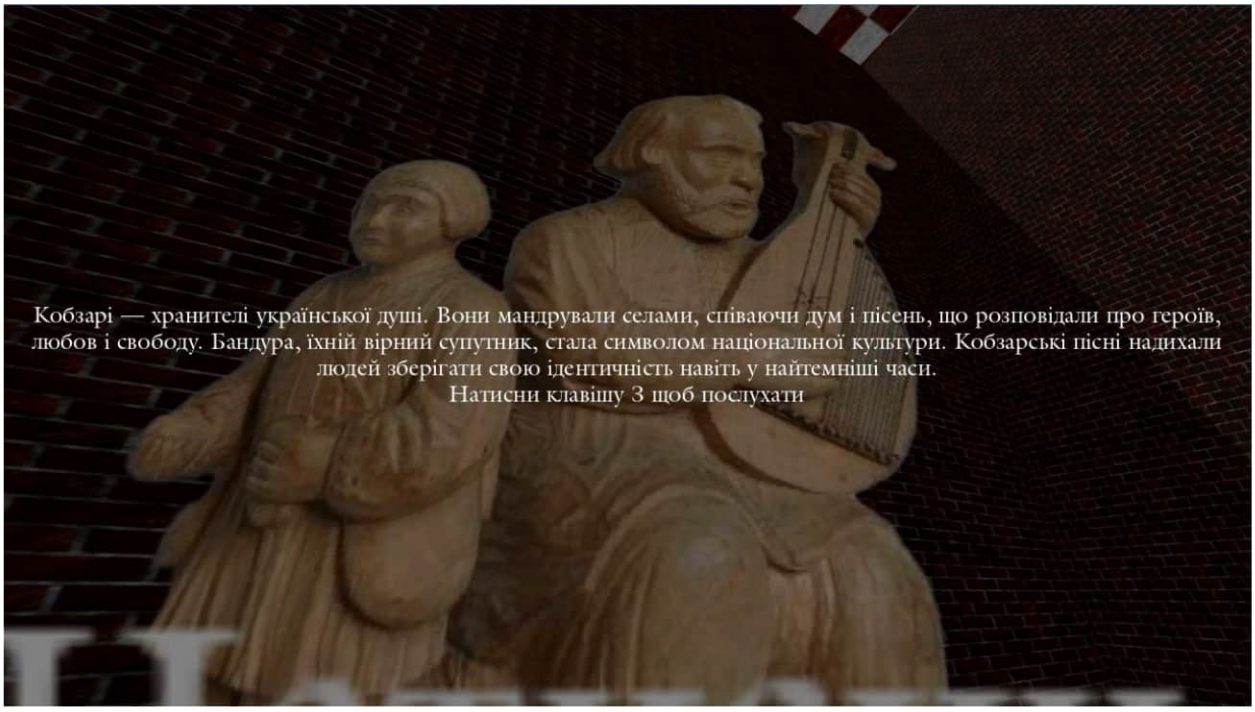


Рис. 2.18 – демонстрація реалізації аудіо в Statue_Kobzar

Відео: приклад Wreath_IvanaKupala

У сцені Wreath_IvanaKupala реалізовано більш комплексну мультимедійну взаємодію: поєднано текстовий опис, зображення (вінок) та відеофрагмент - реконструкцію обряду Івана Купала (рис. 2.19). У скрипті wreath_ivana_kupala.gd зазначено:

```
@export var title_text = "Реконструкція свята Івана Купала"
@export var video_stream: VideoStream
```

Після входу в зону взаємодії та натискання клавіші E гравцеві показується текст із підказкою “Натисни клавішу П щоб подивитися” (латинська G). Активація відео відбувається окремо, через клавішу video_play:

```
if player_inside and event.is_action_pressed("video_play"):
      if video_player:
            if not video_player.is_playing():
                  video_player.play()
            else:
                  video_player.stop()
```

Розмір відеоплеєра встановлюється вручну:

```
video_player.size = Vector2(200, 112)
```

При виході з зони або при повторному натисканні інтерфейс і відео приховуються:

```
video_player.visible = false
if video_player.is_playing():
      video_player.stop()
```

Це дозволяє уникнути конфлікту елементів інтерфейсу та зберегти чистоту візуального простору.

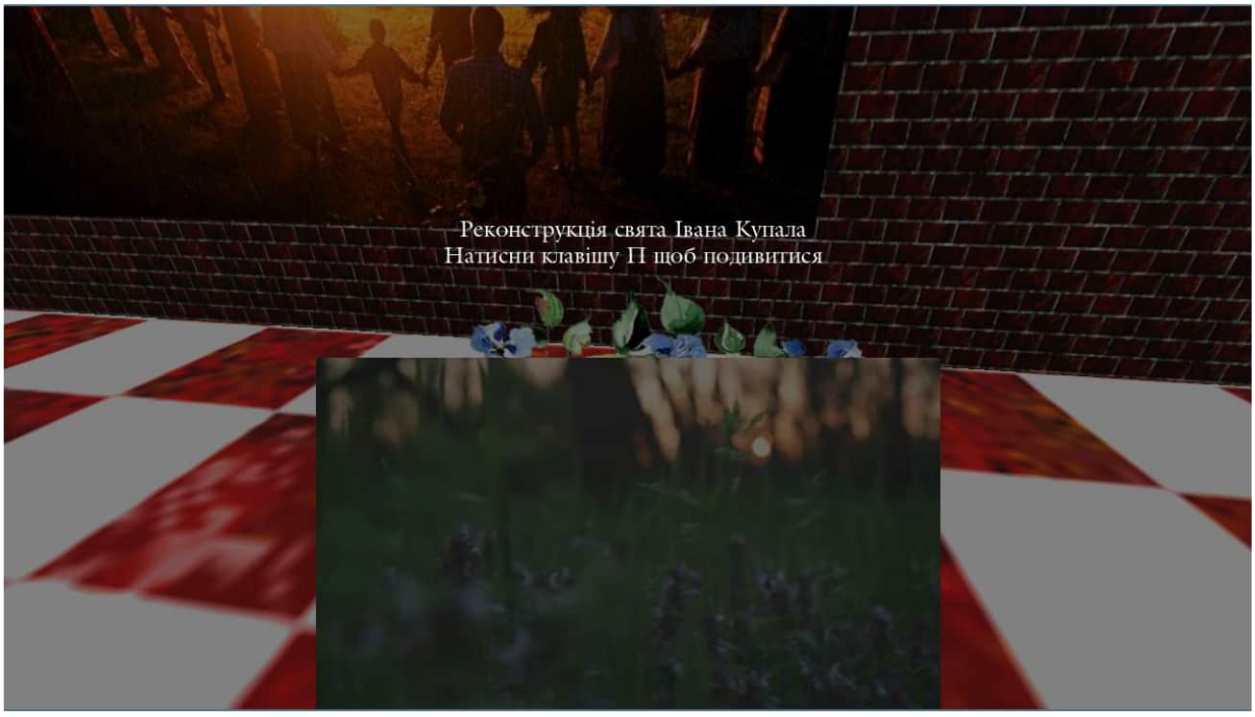


Рисунок. 2.19 - демонстрація реалізації відео в Wreath_IvanaKupala

Організація в сценах: на прикладі Wreath_IvanaKupala

Структура сцени з мультимедіа представлена наступним чином:

```
Wreath_IvanaKupala (Node3D)
├── InteractionArea (Area3D)
├── CanvasLayer
│   ├── DescriptionLabel (Label)
│   └── BackgroundRect (ColorRect)
├── WreathSprite (Sprite3D)
└── VideoPlayer (VideoStreamPlayer)
```

Аналогічно, у Statue_Kobzar використано AudioStreamPlayer3D і Label3D для виводу назви об’єкта у просторі. Обидва приклади показують, що мультимедіа інтегрується органічно в загальну архітектуру кожного експоната.

Контекстна роль у навчальному застосунку

У проєкті мультимедійні елементи не є другорядною візуальною прикрасою. Вони виконують освітню функцію, посилюючи емоційне сприйняття теми. Звуки бандури чи пісні кобзаря дають змогу не лише прочитати опис, а й відчути дух епохи. Відео ж дозволяє побачити те, що неможливо повноцінно передати текстом: рух, ритм, ритуал.

Переваги реалізованого підходу

- Вибірковість - користувач сам вирішує, активувати звук або відео.
- Технічна простота - завдяки використанню вбудованих вузлів Godot (AudioStreamPlayer3D, VideoStreamPlayer).
- Розширюваність - можливість додати новий мультимедійний об’єкт шляхом налаштування лише змінних.
- Узгодженість із текстом - вказівки щодо натискання клавіш інтегруються в опис експонатів.

Зображення

Ілюстративні компоненти є ключовими у сценах типу book.tscn (рис.2.20). Тут використано TextureRect, до якого підключається текстура через експортовану змінну:

```
@export var illustration: Texture2D
```

Це дає змогу підставити будь-яке зображення без створення нової сцени. У головній сцені розробник може налаштувати зображення, заголовок та опис книги індивідуально:

```
@export var book_title = "Book Title"
@export var book_description = "..."
```

Відображення ілюстрації відбувається синхронно з іншими елементами інтерфейсу:

```
tween.tween_property(illustration_texture, "modulate:a", 1, 0.3)
```

При цьому, якщо зображення не задане, поле автоматично очищається:

```
if illustration:
      illustration_texture.texture = illustration.duplicate()
else:
      illustration_texture.texture = null
```

У режимі налагодження система перевіряє, чи не перекривається ілюстрація текстом:

```
if desc_rect.intersects(illus_rect):
      print("ПОМИЛКА: Опис перекриває ілюстрацію")
```

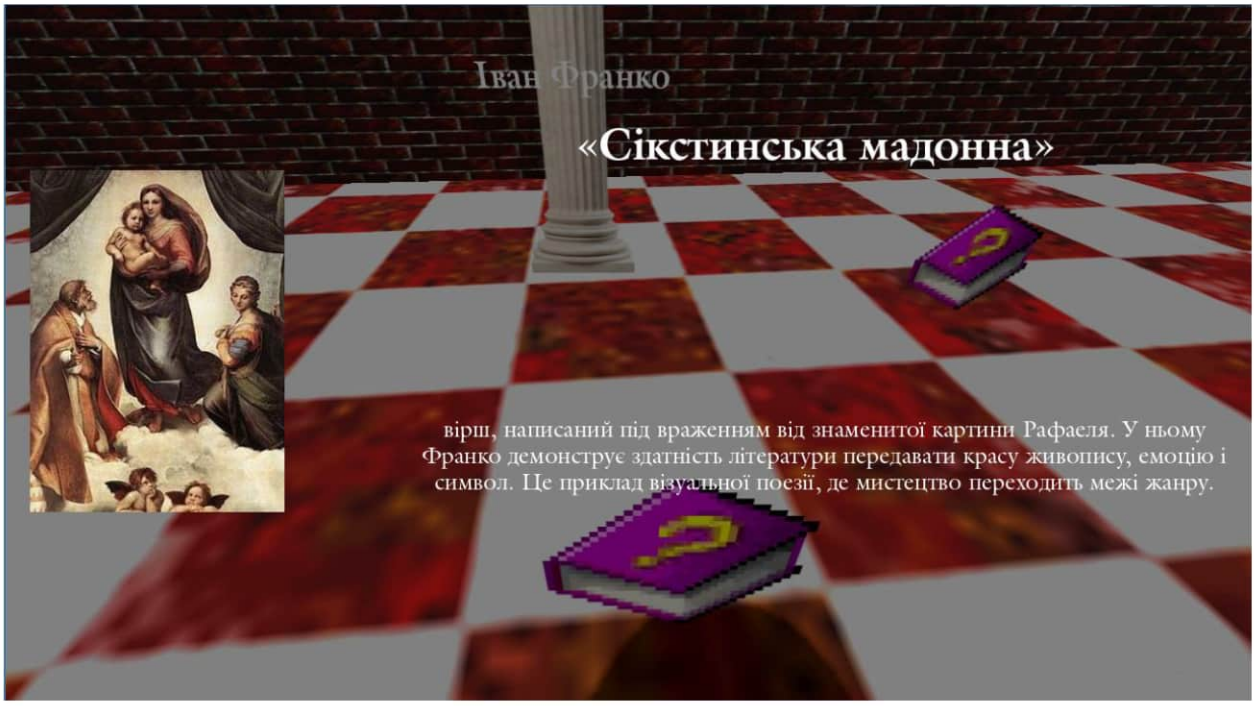


Рисунок. 2.20 - демонстрація реалізації зображеня в book.tscn

Повторне використання сцен

Завдяки використанню @export для параметрів, сцени типу book.tscn чи painting.tscn не потребують дублювання. Вони один раз налаштовуються як універсальні шаблони, а далі можуть використовуватись повторно в головній

сцені, змінюючи лише текстуру, текст чи аудіо/відео. Це суттєво прискорює розробку та спрощує підтримку проєкту.

Шрифти

У Godot є можливість використання власних шрифтів, що дозволяє підлаштувати візуальний стиль під тематику експозиції. Щоб додати свій шрифт, достатньо помістити відповідний .ttf або .otf файл у директорію Fonts/ проєкту. Надалі його можна підключити через редактор або програмно:

```
label.add_theme_font("font", load("res://Fonts/MyFont.ttf"))
```

Також у book.gd експортовано параметри для зміни розміру шрифтів безпосередньо в інспекторі:

```
@export var title_font_size = 45
@export var description_font_size = 24
```

Це забезпечує повну гнучкість у стилізації інтерфейсу без зміни коду.

2.3.4 Оптимізація для слабких пристроїв

Незважаючи на те, що цілеспрямоване тестування на широкому спектрі пристроїв у рамках проєкту не проводилося, сам підхід до розробки структури віртуального музею та обрані інструменти дозволяють зробити обґрунтоване припущення щодо високої продуктивності навіть на слабких пристроях - зокрема, ноутбуках початкового рівня, офісних ПК або освітніх нетбуках.

2.3.4.1. Полегшена графічна сцена

Проєкт реалізовано у форматі 2.5D, де простір побудований за допомогою 3D-вузлів (Node3D, MeshInstance3D, Sprite3D), але візуально він нагадує класичну 2D-гру. Такий підхід дозволяє:

- уникнути складних геометричних моделей;
- використовувати плоскі спрайти або прості сітки (CubeMesh, PlaneMesh);
- зменшити навантаження на відеокарту.

Більшість сцен побудовано зі статичних об’єктів без складної анімації або динамічної фізики, що також суттєво знижує обчислювальні витрати.

2.3.4.2. Відсутність важких постобробок

Проєкт не використовує візуальні ефекти типу Bloom, SSAO, Screen Space Reflections, глибину поля або тіньові карти з високою роздільністю. Освітлення сцен здійснюється в основному за допомогою DirectionalLight3D та OmniLight3D з базовими налаштуваннями.

Завдяки цьому досягається стабільний FPS навіть без налаштування графічних профілів. Інтерфейс не накладає додаткового навантаження, оскільки більшість візуальних елементів реалізовано через CanvasLayer, що не залучає обробку в просторі сцени.

2.3.4.3. Економне використання текстур і ресурсів

Зображення та відеофрагменти, які використовуються в експонатах, заздалегідь оптимізовано:

- текстури мають помірну роздільність, яка достатня для екранного сприйняття (зазвичай до 1024×1024);
- відеофайли підготовлено в стисненому форматі (.webm, .ogv);
- аудіофайли використовуються у форматах .ogg або .mp3, які не створюють додаткового навантаження на дискову підсистему.

Жоден експонат не завантажує зовнішні ресурси під час виконання, що дозволяє уникнути втрати продуктивності через дискові операції чи буферизацію.

2.3.4.4. Відсутність складної фізики

Усі колізії в сценах представлені елементарними формами (BoxShape3D, SphereShape3D) та обробляються лише у вузлах типу Area3D, а не RigidBody3D чи CharacterBody3D. Це означає:

- відсутність симуляції фізики в реальному часі;
- мінімальне навантаження на CPU;
- просту логіку зіткнень, яка не потребує складних обчислень.

Таким чином, навіть пристрої без підтримки апаратного прискорення фізики можуть стабільно відображати сцени музею.

2.3.4.5. Архітектурна простота й модульність

Структура проєкту побудована так, що всі сцени легко розділяються та не завантажуються одночасно. Кожен зал чи експонат існує у вигляді окремої сцени (PackedScene), яка додається до головного простору лише за необхідності. Це знижує споживання оперативної пам'яті та покращує масштабованість.

Окрім того, UI-елементи не рендеряться постійно, а з'являються лише у момент взаємодії з експонатом, що додатково знижує навантаження на GPU.

2.3.4.6. Ефективність рушія Godot

Рушій Godot сам по собі є легким і добре оптимізованим для низькорівневих систем. У поєднанні з мінімалістичним підходом до графіки та взаємодії проєкт не залежить від великої кількості плагінів або сторонніх бібліотек.

Усі основні компоненти (анімація, звук, текст, UI) реалізовані засобами самого рушія, без інтеграції об'ємних фреймворків або зовнішніх викликів.

Зважаючи на все вищезазначене, можна стверджувати, що навіть без додаткового профілювання чи ручної оптимізації, проєкт є енергозберігаючим, легким для запуску та потенційно сумісним із широким спектром пристроїв - включно з нетбуками, офісними комп'ютерами або освітніми планшетами, які найчастіше використовуються у навчальних закладах.

2.4 Оцінка та перспективи використання

Розроблений застосунок «Інтерактивний навчальний музей» є прикладом поєднання сучасних цифрових технологій із завданнями шкільної освіти. Він поєднує в собі ознаки віртуального середовища, мультимедійної енциклопедії, ігрового простору та методичного інструменту.

Впровадження подібних систем у навчальний процес може суттєво змінити підхід до подання матеріалу - від пасивного сприйняття до активної

взаємодії. Завдяки можливості керування персонажем, самостійному вибору маршруту, інтерактивному ознайомленню з експонатами, учень перетворюється з пасивного спостерігача на дослідника. Це створює передумови для формування критичного мислення, зацікавлення історико-культурною спадщиною, розвитку навичок роботи з цифровим середовищем.

У цьому розділі розглянуто переваги використання рушія Godot у шкільному середовищі, потенційні обмеження, пов'язані з технічними й методичними аспектами, а також окреслено можливості масштабування та подальшого розвитку системи.

2.4.1 Потенційні обмеження

Попри численні переваги, використання Godot у шкільному середовищі та створення освітніх застосунків потребує врахування низки потенційних обмежень, які можуть впливати як на технічну реалізацію, так і на методичну ефективність використання віртуального музею.

Передусім, однією з умовних слабких сторін є відсутність повноцінної офіційної підтримки українською мовою в документації. Хоча сам інтерфейс редактора перекладено, для поглибленого освоєння рушія користувачеві доводиться звертатися до англомовних ресурсів. Це може ускладнити процес навчання для учнів, які лише починають працювати з ігровими рушіями.

Іншим чинником є неоднорідна підтримка мультимедійних форматів. Наприклад, для відтворення відео у Web-версії застосунку необхідне ручне кодування файлів у формат .webm із відповідними кодеками, інакше відеоплеєр може не працювати у браузері. Це вимагає від розробника додаткових технічних знань, що виходять за межі стандартної шкільної програми.

Також варто враховувати, що рушій Godot не має вбудованих засобів для збору аналітики, збереження результатів проходження, відстеження прогресу чи авторизації учнів. Для інтеграції таких можливостей потрібна зовнішня

серверна інфраструктура, що унеможливлює їх використання в більшості типових шкільних умов без додаткової технічної підтримки.

У методичному аспекті потенційним обмеженням є відсутність готових навчальних сценаріїв для педагогів, які б дозволяли інтегрувати віртуальний музей у вже існуючі програми без додаткового планування. Успішне впровадження вимагає від учителя цифрової грамотності, розуміння ігрової логіки, основ візуального дизайну - що наразі не є масовим явищем в освітньому середовищі.

Окремо варто згадати про відсутність підтримки мобільних сенсорних екранів у базовій реалізації керування. Для пристроїв типу планшетів потрібна розробка альтернативних методів взаємодії (віртуальні кнопки, сенсорні жести), які не передбачені у стандартній версії застосунку.

Загалом, хоча зазначені обмеження не є критичними, вони вимагають додаткового врахування у разі масштабного впровадження розробки в освітню практику.

2.4.2 Можливості масштабування і розвитку

Ідея інтерактивного навчального музею, реалізованого на рушії Godot із використанням форматів 2.5D, має значний потенціал для масштабування - як у змістовому, так і в технологічному сенсі. Водночас важливо, щоб розширення функціоналу не перетворювало проєкт на вузькоспеціалізоване середовище, а зберігало його доступність для широкого кола користувачів: учнів, студентів, учителів, аматорських гуртків.

Основні напрями розвитку можна класифікувати за кількома критеріями (таб. 2.2), при цьому збереження доступності залежить від дотримання принципу технологічної простоти та шаблонного підходу:

Таблиця.2.2 - Основні напрями масштабування та умови збереження доступності

Напрям	Потенційне розширення	Доступність зберігається якщо…
Тематичне	Музеї з біології, мистецтва, техніки	Нові сцени створюються копіюванням шаблонів
Форматне	Додавання аудіогідів, відеоекскурсій	Все редагується без коду (через інтерфейс рушія)
Технологічне	Інтеграція з QR, браузерна версія, WebXR	Використовуються лише безкоштовні інструменти
Педагогічне	Проєктна діяльність учнів зі створення експонатів	Шаблони мають просту логіку, яку можна пояснити
Локалізаційне	Створення музеїв про місцеву культуру	Досить лише замінити зображення і текст

У межах цифрової трансформації освіти, відповідно до ідеології НУШ, важливо дотримуватись принципу: *«мінімальний бар'єр входу - максимальний ефект»*. Саме тому рушій Godot, що не потребує комерційної ліцензії, є легким для освоєння та підтримує візуальні редактори, обрано як базову платформу.

Суть масштабування полягає не в ускладненні технічної частини, а в розширенні сценаріїв застосування. Зокрема, можливе використання:

1. у класах - як цифрових лабораторій;
2. у гуртках - як платформи для творчих проєктів;
3. у громадах - як архівів локальної історії;
4. у конкурсах - як основи для STEM-робіт.

Перспективними є й майбутні сценарії (рис. 2.21):

1. Шкільна виставка власних віртуальних музеїв, де кожен клас створює експонати на задану тему.
2. Всеукраїнський онлайн-конкурс «Віртуальний музей моєї громади».

3. Інтеграція з Moodle або Google Classroom для завантаження сцен як навчальних модулів.

4. Офлайн-версії для шкіл, які не мають постійного доступу до інтернету.

Таким чином, ідея створення інтерактивного музею виходить за межі одного застосунку - це гнучка концепція, яка може бути адаптована до будь-якого рівня освіти, технічної бази та цільової аудиторії. Її сила - у простоті впровадження, відкритості технологій та підтримці педагогічної співтворчості.

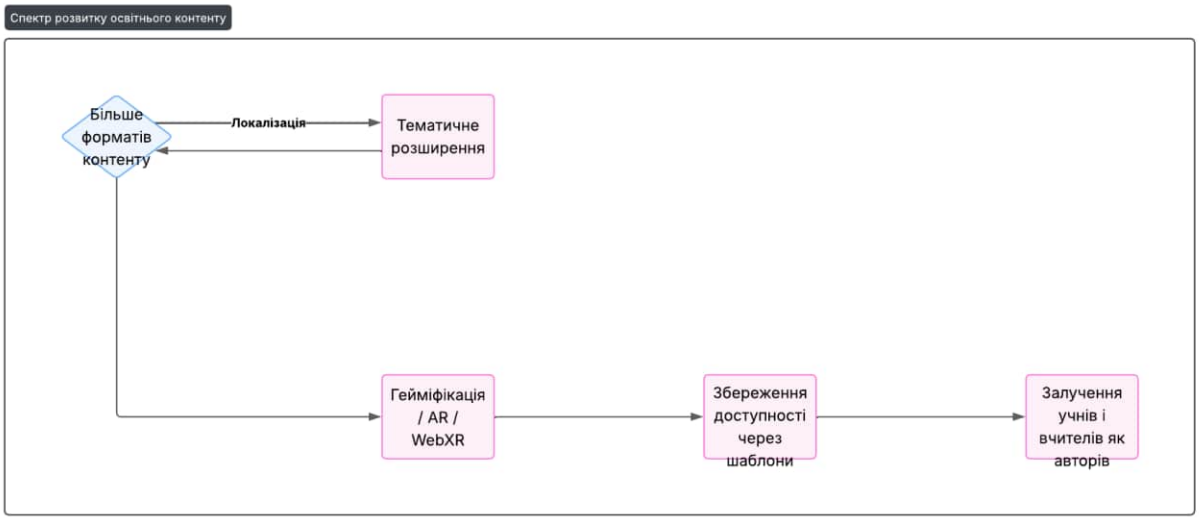


Рисунок. 2.21 – Спектр розвитку освітнього контенту в інтерактивному музеї

2.5 Висновки по другому розділу

У другому розділі було проаналізовано архітектурні та функціональні рішення, реалізовані у процесі створення інтерактивного навчального музею з використанням рушія Godot. Основна увага приділялася реалізації механізмів взаємодії, виводу навчального контенту, а також можливостям масштабування розробленої моделі.

Однією з ключових інженерних і педагогічних ідей є використання формату 2.5D, що дозволяє зберегти баланс між візуальною виразністю, технічною простотою та продуктивністю на застарілому обладнанні,

характерному для багатьох навчальних закладів. Застосування цього підходу в контексті НУШ відкриває доступ до цифрових інструментів навіть у сільських школах, де відсутні сучасні ПК.

Було детально розглянуто реалізацію зон взаємодії (Area3D) і механізмів тригерів, які дозволяють учням ініціювати отримання навчальної інформації через просту та інтуїтивну модель: наближення до об’єкта > підказка > натискання клавіші > виведення контенту. Такий підхід відповідає принципам активного пізнання, що лежать в основі концепції діяльнісного навчання.

Крім того, було доведено, що структура експонатів є повністю модульною: додавання нових сцен не потребує зміни основного коду, а редагування контенту (тексту, зображень, відео) здійснюється через візуальний редактор. Це створює умови для педагогічної творчості, коли вчитель або учень може самостійно створити новий музейний об’єкт.

Окремо розглянуто перспективи масштабування та розвитку проєкту як ідеї: від локальних віртуальних музеїв до конкурсних чи міжпредметних платформ. Усі варіанти розвитку передбачають збереження головного принципу -доступність реалізації без спеціальних знань у програмуванні.

Таким чином, розроблений застосунок демонструє не лише технічну працездатність, а й відповідає сучасним освітнім викликам, включно з цифровою трансформацією, гнучкістю навчального середовища та потребою у локалізованому, персоналізованому контенті.

ВИСНОВКИ

У межах кваліфікаційної роботи було спроєктовано, реалізовано та протестовано інтерактивний навчальний застосунок у форматі віртуального музею, побудованого на рушії Godot із використанням підходу 2.5D. Метою роботи було створення доступного, масштабованого й педагогічно доцільного цифрового середовища, яке відповідає сучасним викликам освіти в умовах реалізації концепції Нової української школи.

У теоретичній частині дослідження було проаналізовано особливості використання ігрових технологій в освіті, обґрунтовано вибір рушія Godot як відкритої, безкоштовної та технологічно гнучкої платформи для створення навчального контенту. Було розглянуто поняття цифрового музею як інструменту діяльнісного навчання, з акцентом на інтерактивність, мультимедійність та інклюзивність.

У практичній частині реалізовано основну сцену з підтримкою руху гравця, колізій, камер і базового інтерфейсу, створено серію експонатів (сцен), кожен з яких містить інформаційні блоки, інтерактивні зони та мультимедійні елементи. Впроваджено механізм взаємодії через Area3D: при вході гравця з’являється підказка, після натискання клавіші E виводиться навчальний контент.

Окремо було розроблено архітектуру, що забезпечує можливості масштабування - як технічного (додавання нових сцен без зміни коду), так і педагогічного (використання вчителями й учнями як шаблону для власного контенту).

У роботі запропоновано методику, яка дозволяє створювати нові експонати без знань програмування - шляхом копіювання і редагування шаблонних сцен. Це відкриває перспективу широкого використання проєкту у шкільній практиці, проєктній діяльності, конкурсах, міжшкільних ініціативах і краєзнавчих дослідженнях.

Розроблений застосунок поєднує технічну простоту з функціональністю, підтримує мультимедіа, адаптується до будь-якої навчальної тематики й сприяє формуванню цифрової компетентності учнів та педагогів.

Таким чином, поставлену мету кваліфікаційної роботи досягнуто. Створений застосунок є практично придатним інструментом, що має потенціал для широкого впровадження в освітню практику, а закладені в ньому принципи можуть слугувати основою для подальших досліджень і вдосконалень у сфері цифрових освітніх середовищ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. **Міністерство освіти і науки України.** Нова українська школа: концептуальні засади реформування середньої освіти [Електронний ресурс]. – Режим доступу: https://mon.gov.ua/tag/nova-ukrainska-shkola?&tag=nova-ukrainska-shkola

2. **Постанова КМУ №87 від 21.03.2018.** Про затвердження Державного стандарту початкової освіти [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/87-2018-%D0%BF#Text

3. **Єрмоленко А.Б.** Інтерактивні технології навчання [Електронний ресурс]. – Режим доступу: https://lib.iitta.gov.ua/id/eprint/731177/1/Інтерактивні%20технології%20навчання_ЕНК.pdf

4. **Osvita.ua.** Інтерактивні технології в особистісно зорієнтованій освіті
[Електронний ресурс]. – Режим доступу: https://osvita.ua/school/method/technol/982/

5. **Дія.Освіта.** Інтерактивне навчання: інструменти та технології [Електронний ресурс]. – Режим доступу: https://osvita.diia.gov.ua/courses/interactive-learning

6. **CSTA.** Beginner Unity Projects for Your Game Development Classroom
[Електронний ресурс]. – Режим доступу: https://csteachers.org/beginner-unity-projects-for-your-game-development-classroom/

7. **На Урок** – платформа з інтерактивними тестами, рейтингами та сертифікатами [Електронний ресурс]. – Режим доступу: https://naurok.com.ua/

8. **МійКлас** – адаптивні завдання з автоматичною перевіркою, підказками та нагородами [Електронний ресурс]. – Режим доступу: https://www.miyklas.com.ua/p

9. **Edugames** – ігрові вправи з математики, мови, логіки для учнів початкової школи [Електронний ресурс]. – Режим доступу: https://edugames.rozumniki.com/

10. **Learning.ua** – освітній онлайн-ресурс з інтерактивними тренажерами для учнів [Електронний ресурс]. – Режим доступу: https://learning.ua/

11. **GIOS** – навчальні вправи з елементами гейміфікації для шкільного курсу [Електронний ресурс]. – Режим доступу: https://edtech.net.ua/gios

12. **MCVUK.** Putting Unreal Engine in the classroom [Електронний ресурс]. – Режим доступу: https://mcvuk.com/development-news/putting-unreal-engine-in-the-classroom/

13. **Itch.io.** Top Educational Games Made with Godot [Електронний ресурс]. – Режим доступу: https://itch.io/games/made-with-godot/tag-educational

14. **GDQuest.** Learn GDScript From Zero (курс із програмування в Godot)
[Електронний ресурс]. – Режим доступу: https://gdquest.github.io/learn-gdscript/

15. **Офіційна документація Godot Engine** [Електронний ресурс]. – Режим доступу: https://docs.godotengine.org/en/stable/

16. **Chris Bradfield.** *Godot Engine Game Development Projects: Build five cross-platform 2D and 3D games with Godot 3.0* (2018)

17. **Daniel Buckley.** *Godot Game Development For Beginners* (2024)

18. **Godot Engine –** офіційна спільнота користувачів і розробників [Електронний ресурс]. – Режим доступу: https://godotengine.org/community/

19. **HEROES Simulation** – система моделювання дій екстрених служб, створена на Unreal Engine [Електронний ресурс]. – Режим доступу: https://arxiv.org/html/2309.14508v2

20. **Super Practica** – освітній гейміфікований проєкт для вивчення математики в початковій школі [Електронний ресурс]. – Режим доступу: https://superpractica.org/

21. **Digital Logic Sim 2** – симулятор для вивчення цифрової логіки та побудови логічних схем [Електронний ресурс]. – Режим доступу: https://github.com/UkrainianBanderasCat/Digital-Logic-Sim2

22. **Bioblox 2.5D Educational Project.** (Проєкт у сфері біохімії; шукати на Google Scholar або в Godot Showcase)

ДОДАТОК А.
ПРОГРАМНИЙ КОД

А.1 Програмний код main.gd

```
extends Node3D

@onready var pause_layer = $PauseLayer
@onready var pause_background = $PauseLayer/PauseBackground
@onready var continue_button = $PauseLayer/ContinueButton
@onready var exit_button = $PauseLayer/ExitButton

var is_paused = false

func _ready():
	if not pause_layer:
		print("Помилка: PauseLayer не знайдено.")
	else:
		pause_layer.visible = false
		pause_background.visible = false
		continue_button.visible = false
		exit_button.visible = false
		if continue_button and exit_button:
			var err_continue = continue_button.connect("pressed", Callable(self, "_on_continue_pressed"))
			var err_exit = exit_button.connect("pressed", Callable(self, "_on_exit_pressed"))
			if err_continue != OK:
				print("Помилка підключення ContinueButton: ", err_continue)
			if err_exit != OK:
				print("Помилка підключення ExitButton: ", err_exit)
			else:
				print("Сигнали підключено: ContinueButton і ExitButton.")
				continue_button.process_mode = Node.PROCESS_MODE_ALWAYS
```

```
                        exit_button.process_mode                        =
Node.PROCESS_MODE_ALWAYS
                else:
                        print("Помилка: Одна або обидві кнопки не знайдені.
ContinueButton: ", continue_button, " ExitButton: ", exit_button)
            # Приховуємо курсор на старті
            Input.set_mouse_mode(Input.MOUSE_MODE_CAPTURED)

      func _input(event):
            if event.is_action_pressed("ui_cancel"): # Натиснення Esc
                  is_paused = !is_paused
                  get_tree().paused = is_paused
                  if pause_layer and pause_background and continue_button and
exit_button:
                        pause_layer.visible = is_paused
                        pause_background.visible = is_paused
                        continue_button.visible = is_paused
                        exit_button.visible = is_paused
                        if is_paused:
                              # Показуємо курсор під час паузи
                              Input.set_mouse_mode(Input.MOUSE_MODE_VISIBLE)
                              print("Пауза    увімкнена.    Кнопки    видимі:
Continue=", continue_button.visible, " Exit=", exit_button.visible)
                        else:
                              # Приховуємо курсор при виході з паузи
                              Input.set_mouse_mode(Input.MOUSE_MODE_CAPTURED)
                              print("Пауза вимкнена.")
                  else:
                        print("Помилка: Один із вузлів паузи не знайдено.")

      func _on_continue_pressed():
            is_paused = false
            get_tree().paused = false
            if  pause_layer  and  pause_background  and  continue_button  and
exit_button:
                  pause_layer.visible = false
                  pause_background.visible = false
                  continue_button.visible = false
```

```
                exit_button.visible = false
                # Приховуємо курсор
                Input.set_mouse_mode(Input.MOUSE_MODE_CAPTURED)
                print("Гру продовжено.")
        else:
                print("Помилка: Один із вузлів паузи не знайдено при продовженні.")

    func _on_exit_pressed():
        get_tree().quit()
        print("Вихід із гри.")
```

А.2 Програмний код main_menu.gd

```
    extends Node2D

    @onready var start_button = $CanvasLayer/StartButton
    @onready var exit_button = $CanvasLayer/ExitButton
    @onready var instruction_layer = $CanvasLayer/InstructionLayer
    @onready var instruction_background = $CanvasLayer/InstructionLayer/InstructionBackground
    @onready var instruction_text = $CanvasLayer/InstructionLayer/InstructionText

    func _ready():
        start_button.pressed.connect(_on_start_pressed)
        exit_button.pressed.connect(_on_exit_pressed)
        instruction_layer.visible = false
        instruction_background.visible = false
        instruction_text.visible = false

    func _on_start_pressed():
        instruction_layer.visible = true
        instruction_background.visible = true
        instruction_text.visible = true
        instruction_text.text = "Вітаю вас у макеті проекту інтерактивного музею! Тут ви побачите унікальні експонати, що відображають історію та культуру України. Досліджуйте, взаємодійте з об’єктами та дізнавайтеся більше про
```

```
минуле.\n\nКерування:\nW - вперед\nS - назад\nA - ліво\nD - право\nE - взаємодіяти"

	func _on_exit_pressed():
		get_tree().quit()

	func _input(event):
		if instruction_layer.visible and event.is_action_pressed("interact"): # Натиснення E
			_change_to_main_scene()

	func _change_to_main_scene():
		var main_scene_resource = load("res://Scenes/main(node_3d).tscn")
		if main_scene_resource == null:
			print("Помилка: файл main_node_3d.tscn не завантажено. Перевір шлях: res://Scenes/main_node_3d.tscn")
			return
		print("Файл main(node_3D).tscn знайдено.")
		get_tree().change_scene_to_packed(main_scene_resource) # Прямий перехід без instantiate
```

А.3 Програмний код student.gd

```
	extends CharacterBody3D

	@export var speed := 5.0 # Базова швидкість руху гравця
	@export var sprint_multiplier := 1.5 # Множник швидкості під час біги (Shift)
	@export var mouse_sensitivity := 0.002 # Чутливість миші
	@export var acceleration := 10.0 # Прискорення
	@export var friction := 8.0 # Гальмування

	var camera_rot_x := 0.0 # Обертання камери вгору/вниз
	@onready var camera = $Camera3D # Посилання на камеру

	func _ready():
		Input.set_mouse_mode(Input.MOUSE_MODE_CAPTURED)
```

```
func _input(event):
	if event is InputEventMouseMotion:
		# Обертання гравця по горизонталі (вліво/вправо)
		rotate_y(-event.relative.x * mouse_sensitivity)

		# Обертання камери по вертикалі (вгору/вниз)
		camera_rot_x = clamp(
			camera_rot_x - event.relative.y * mouse_sensitivity,
			deg_to_rad(-85), deg_to_rad(85)
		)
		camera.rotation.x = camera_rot_x

	# Вихід з режиму захоплення миші (Esc)
	if event.is_action_pressed("ui_cancel"):
		Input.set_mouse_mode(Input.MOUSE_MODE_VISIBLE)

func _physics_process(delta):
	var input_dir = Vector3.ZERO

	# Ввід напрямку руху
	if Input.is_action_pressed("move_forward"):
		input_dir -= transform.basis.z
	if Input.is_action_pressed("move_back"):
		input_dir += transform.basis.z
	if Input.is_action_pressed("move_left"):
		input_dir -= transform.basis.x
	if Input.is_action_pressed("move_right"):
		input_dir += transform.basis.x

	input_dir.y = 0
	input_dir = input_dir.normalized()

	# Визначення цільової швидкості (з урахуванням бігу)
	var current_speed = speed
	if Input.is_action_pressed("sprint"):
		current_speed *= sprint_multiplier

	var target_velocity = input_dir * current_speed
```

```
	# Плавне прискорення
	velocity = velocity.lerp(target_velocity, acceleration * delta)

	# Гальмування при відсутності руху
	if input_dir == Vector3.ZERO:
		velocity = velocity.lerp(Vector3.ZERO, friction * delta)

	# Рух
	move_and_slide()
```

А.4 Програмний код exhibit.gd

```
extends Node3D

@onready var interaction_area = $InteractionArea
@onready var title_label = $CanvasLayer/TitleLabel
@onready var description_label = $CanvasLayer/DescriptionLabel
@onready var background_rect = $CanvasLayer/BackgroundRect
@onready var exhibit_sprite = $ExhibitSprite

var player_inside = false # Гравець у зоні
@export var birth_death_years = "" # Роки, наприклад "1814-1861"
@export var exhibit_text = "Exhibit Name" # Короткий текст (назва
експоната)
@export var description_text = "111" # Об'ємна інформація про експонат
@export var sprite_texture: Texture2D: # Текстура для ExhibitSprite
	set(value):
		sprite_texture = value
		if exhibit_sprite and is_inside_tree():
			if sprite_texture:
				exhibit_sprite.texture =
sprite_texture.duplicate()
			else:
				exhibit_sprite.texture = null
@export var title_font_size = 48
@export var description_font_size = 24

func _ready():
```

```
            # Встановлюємо текстуру та робимо її локальною
            if sprite_texture:
                  exhibit_sprite.texture = sprite_texture.duplicate()

            # Встановлюємо розмір шрифту
            title_label.add_theme_font_size_override("font_size",
title_font_size)
            description_label.add_theme_font_size_override("font_size",
description_font_size)

            # Підключаємо сигнали Area3D
            interaction_area.body_entered.connect(_on_body_entered)
            interaction_area.body_exited.connect(_on_body_exited)
            # Приховуємо текст і фон
            title_label.modulate.a = 0
            description_label.modulate.a = 0
            background_rect.color.a = 0
            # Встановлюємо короткий текст для ExhibitLabel
            $ExhibitLabel.text = exhibit_text

      func _on_body_entered(body: Node) -> void:
            if body.is_in_group("player"):
                  player_inside = true
                  $ExhibitLabel.text = "Press E"

      func _on_body_exited(body: Node) -> void:
            if body.is_in_group("player"):
                  player_inside = false
                  # Приховуємо текст і фон
                  var tween = create_tween()
                  tween.tween_property(title_label, "modulate:a", 0, 0.3)
                  tween.tween_property(description_label, "modulate:a", 0,
0.3)
                  tween.tween_property(background_rect, "color:a", 0, 0.3)
                  $ExhibitLabel.text = exhibit_text

      func _input(event: InputEvent) -> void:
```

```
            if player_inside and event.is_action_pressed("interact"): #
Клавіша E
                  if description_label.modulate.a > 0:
                        # Приховуємо текст і фон
                        var tween = create_tween()
                        tween.tween_property(title_label, "modulate:a", 0,
0.3)
                        tween.tween_property(description_label, "modulate:a",
0, 0.3)
                        tween.tween_property(background_rect, "color:a", 0,
0.3)
                  else:
                        # Спочатку затемнюємо екран, потім показуємо текст
                        var tween = create_tween()
                        tween.tween_property(background_rect, "color:a", 0.5,
0.3)
                        tween.tween_callback(func():
                              var title_text = exhibit_text
                              if birth_death_years != "":
                                    title_text += " (" + birth_death_years +
")"
                              title_label.text = title_text
                              description_label.text = description_text
                        ).set_delay(0.2)
                        tween.tween_property(title_label, "modulate:a", 1,
0.3)
                        tween.tween_property(description_label, "modulate:a",
1, 0.3)
```

A.5 Програмний код book.gd

```
extends Node3D

@onready var interaction_area = $InteractionArea
@onready var title_label = $CanvasLayer/TitleLabel
@onready var description_label = $CanvasLayer/DescriptionLabel
@onready var background_rect = $CanvasLayer/BackgroundRect
@onready var illustration_texture = $CanvasLayer/IllustrationTexture
```

```
	var player_inside = false # Гравець у зоні
	@export var book_title = "Book Title" # Назва книги
	@export var book_description = "This is a placeholder text for the book!" # Опис книги
	@export var illustration: Texture2D: # Ілюстрація
		set(value):
			illustration = value
			if illustration_texture and is_inside_tree():
				if illustration:
					illustration_texture.texture = illustration.duplicate()
				else:
					illustration_texture.texture = null
				if Engine.is_editor_hint():

	illustration_texture.notify_property_list_changed()
					notify_property_list_changed()
	@export var title_font_size = 45 # Розмір шрифту для TitleLabel
	@export var description_font_size = 24 # Розмір шрифту для DescriptionLabel

	func _ready():
		# Встановлюємо ілюстрацію
		if illustration:
			illustration_texture.texture = illustration.duplicate()

		# Встановлюємо розміри шрифтів
		title_label.add_theme_font_size_override("font_size", title_font_size)
		description_label.add_theme_font_size_override("font_size", description_font_size)

		# Підключаємо сигнали Area3D
		interaction_area.body_entered.connect(_on_body_entered)
		interaction_area.body_exited.connect(_on_body_exited)
		# Приховуємо UI
		title_label.modulate.a = 0
```

```
            description_label.modulate.a = 0
            background_rect.color.a = 0
            illustration_texture.modulate.a = 0

            # Дебаг позицій UI
            if OS.is_debug_build():
                  print("Книга:", name, "TitleLabel Rect:",
title_label.get_rect())
                  print("Книга:", name, "DescriptionLabel Rect:",
description_label.get_rect())
                  print("Книга:", name, "IllustrationTexture Rect:",
illustration_texture.get_rect())

      func _on_body_entered(body: Node) -> void:
            if body.is_in_group("player"):
                  player_inside = true

      func _on_body_exited(body: Node) -> void:
            if body.is_in_group("player"):
                  player_inside = false
                  # Приховуємо UI
                  var tween = create_tween()
                  tween.tween_property(background_rect, "color:a", 0, 0.3)
                  tween.tween_property(title_label, "modulate:a", 0, 0.3)
                  tween.tween_property(description_label, "modulate:a", 0,
0.3)
                  tween.tween_property(illustration_texture, "modulate:a", 0,
0.3)

      func _input(event: InputEvent) -> void:
            if player_inside and event.is_action_pressed("interact"): #
Клавіша E
                  if title_label.modulate.a > 0:
                        # Приховуємо UI
                        var tween = create_tween()
                        tween.tween_property(background_rect, "color:a", 0,
0.3)
```

```
                        tween.tween_property(title_label, "modulate:a", 0,
0.3)
                        tween.tween_property(description_label, "modulate:a",
0, 0.3)
                        tween.tween_property(illustration_texture,
"modulate:a", 0, 0.3)
                  else:
                        # Показуємо UI
                        var tween = create_tween()
                        tween.tween_property(background_rect, "color:a", 0.5,
0.3)
                        tween.tween_callback(func():
                              title_label.text = book_title
                              description_label.text = book_description
                              if OS.is_debug_build():
                                    var title_rect = title_label.get_rect()
                                    var desc_rect =
description_label.get_rect()
                                    var illus_rect =
illustration_texture.get_rect()
                                    print("Книга:", name, "Заголовок:",
book_title, "Rect:", title_rect)
                                    print("Книга:", name, "Опис:",
book_description.length(), "символів, Rect:", desc_rect)
                                    print("Книга:", name, "Ілюстрація Rect:",
illus_rect)
                                    if desc_rect.intersects(illus_rect):
                                          print("ПОМИЛКА: Опис перекриває
ілюстрацію для", name, "! Кінець опису X:", desc_rect.position.x +
desc_rect.size.x, "Початок ілюстрації X:", illus_rect.position.x)
                                    if desc_rect.position.y +
desc_rect.size.y > illus_rect.position.y + illus_rect.size.y:
                                          print("ПОМИЛКА: Опис виходить за
межі ілюстрації вниз для", name, "! Кінець опису Y:", desc_rect.position.y +
desc_rect.size.y, "Кінець ілюстрації Y:", illus_rect.position.y +
illus_rect.size.y)
                        ).set_delay(0.2)
```

```
                        tween.tween_property(title_label, "modulate:a", 1,
0.3)
                        tween.tween_property(description_label, "modulate:a",
1, 0.3)
                        tween.tween_property(illustration_texture,
"modulate:a", 1, 0.3)
```

А.6 Програмний код ivan_fedorov.gd

```
extends Node3D

@onready var interaction_area = $InteractionArea
@onready var statue_sprite = $StatueSprite
@onready var exhibit_label_3d = $ExhibitLabel3D
@onready var canvas_layer = $CanvasLayer
@onready var background_rect = $CanvasLayer/BackgroundRect
@onready var description_label = $CanvasLayer/DescriptionLabel
@onready var video_player = $VideoPlayer
@onready var video_instruction_label =
$CanvasLayer/VideoInstructionLabel

var player_inside = false
@export var statue_texture: Texture2D:
      set(value):
            statue_texture = value
            if statue_sprite and is_inside_tree():
                  if statue_texture:
                        statue_sprite.texture =
statue_texture.duplicate()
                  else:
                        statue_sprite.texture = null
@export_multiline var description_text = ""
@export var description_font_size = 24
@export var video_stream: VideoStream:
      set(value):
            video_stream = value
            if video_player and is_inside_tree():
                  video_player.stream = video_stream
```

```
	@export var video_instruction_text = "Щоб подивитися приклад друкарського станка, натисніть клавішу п"

	func _ready():
		if statue_texture:
			statue_sprite.texture = statue_texture.duplicate()
		if video_stream:
			video_player.stream = video_stream

		exhibit_label_3d.text = "Друкар Іван Федоров"
		exhibit_label_3d.pixel_size = 0.03
		exhibit_label_3d.billboard = BaseMaterial3D.BILLBOARD_ENABLED
		exhibit_label_3d.render_priority = 1

		description_label.add_theme_font_size_override("font_size", description_font_size)
		video_instruction_label.add_theme_font_size_override("font_size", 16)

		interaction_area.body_entered.connect(_on_body_entered)
		interaction_area.body_exited.connect(_on_body_exited)

		background_rect.color.a = 0
		description_label.modulate.a = 0
		video_player.visible = false
		video_player.size = Vector2(640, 360) # Відновлюємо розмір, який ти вибрав
		video_instruction_label.modulate.a = 0

		if OS.is_debug_build():
			print("Statue:", name, "Ready.")

	func _on_body_entered(body: Node) -> void:
		if body.is_in_group("player"):
			player_inside = true
			exhibit_label_3d.text = "натисни E"
			if OS.is_debug_build():
```

```
                        print("Statue:", name, "Player entered.
player_inside:", player_inside)

      func _on_body_exited(body: Node) -> void:
            if body.is_in_group("player"):
                  player_inside = false
                  exhibit_label_3d.text = "Друкар Іван Федоров"
                  var tween = create_tween()
                  tween.tween_property(background_rect, "color:a", 0, 0.3)
                  tween.tween_property(description_label, "modulate:a", 0,
0.3)
                  tween.tween_property(video_instruction_label, "modulate:a",
0, 0.3)
                  if video_player and video_player.is_playing():
                        video_player.stop()
                  video_player.visible = false
                  if OS.is_debug_build():
                        print("Statue:", name, "Player exited.
player_inside:", player_inside)

      func _input(event: InputEvent) -> void:
            if player_inside and event.is_action_pressed("interact"): #
Натиснення E
                  if OS.is_debug_build():
                        print("Statue:", name, "Interact pressed. Current
modulate.a:", description_label.modulate.a)
                  if description_label.modulate.a > 0:
                        var tween = create_tween()
                        tween.tween_property(background_rect, "color:a", 0,
0.3)
                        tween.tween_property(description_label, "modulate:a",
0, 0.3)
                        tween.tween_property(video_instruction_label,
"modulate:a", 0, 0.3)
                        if video_player and video_player.is_playing():
                              video_player.stop()
                        video_player.visible = false
                        if OS.is_debug_build():
```

```
                        print("Statue:", name, "Hiding text and video.")
                else:
                        var tween = create_tween()
                        tween.tween_property(background_rect, "color:a", 0.5,
0.3)
                        tween.tween_callback(func():
                                description_label.text = description_text
                                video_instruction_label.text =
video_instruction_text
                                if OS.is_debug_build():
                                        print("Statue:", name, "Showing text:",
description_text)
                        ).set_delay(0.2)
                        tween.tween_property(description_label, "modulate:a",
1, 0.3)
                        tween.tween_property(video_instruction_label,
"modulate:a", 1, 0.3)
                        video_player.visible = true

        if player_inside and event.is_action_pressed("video_play"): #
Натиснення G
                if video_player:
                        if not video_player.is_playing():
                                video_player.play()
                                if OS.is_debug_build():
                                        print("Statue:", name, "Video started.")
                        else:
                                video_player.stop()
                                if OS.is_debug_build():
                                        print("Statue:", name, "Video stopped.")
```

A.7 Програмний код kobzar.gd

```
extends Node3D

@onready var interaction_area = $InteractionArea
@onready var description_label = $CanvasLayer/DescriptionLabel
@onready var background_rect = $CanvasLayer/BackgroundRect
```

```
	@onready var statue_sprite = $StatueSprite
	@onready var exhibit_label_3d = $ExhibitLabel3D
	@onready var audio_player = $AudioStreamPlayer3D

	var player_inside = false
	@export var description_text = "Кобзарі — хранителі української душі. Вони мандрували селами, співаючи дум і пісень, що розповідали про героїв, любов і свободу. Бандура, їхній вірний супутник, стала символом національної культури. Кобзарські пісні надихали людей зберігати свою ідентичність навіть у найтемніші часи.\nНатисни клавішу 3 щоб послухати"
	@export var exhibit_name = "Кобзар"
	@export var sprite_texture: Texture2D:
		set(value):
			sprite_texture = value
			if statue_sprite and is_inside_tree():
				if sprite_texture:
					statue_sprite.texture = sprite_texture.duplicate()
				else:
					statue_sprite.texture = null
	@export var audio_stream: AudioStream:
		set(value):
			audio_stream = value
			if audio_player and is_inside_tree():
				audio_player.stream = audio_stream
	@export var description_font_size = 24
	@export var label3d_font_size = 24

	func _ready():
		if sprite_texture:
			statue_sprite.texture = sprite_texture.duplicate()
		if audio_stream:
			audio_player.stream = audio_stream

		description_label.add_theme_font_size_override("font_size", description_font_size)
		if exhibit_label_3d:
			exhibit_label_3d.font_size = label3d_font_size
```

```
            interaction_area.body_entered.connect(_on_body_entered)
            interaction_area.body_exited.connect(_on_body_exited)

            description_label.modulate.a = 0
            background_rect.color.a = 0
            if exhibit_label_3d:
                  exhibit_label_3d.text = exhibit_name
                  exhibit_label_3d.pixel_size = 0.03 # Зменшено для кращої
адаптації
                  exhibit_label_3d.billboard =
BaseMaterial3D.BILLBOARD_ENABLED
                  exhibit_label_3d.render_priority = 1

            if OS.is_debug_build():
                  print("Kobzar:", name, "Ready. ExhibitLabel3D Text:",
exhibit_label_3d.text if exhibit_label_3d else "Null", "Position:",
exhibit_label_3d.global_transform.origin if exhibit_label_3d else
Vector3.ZERO)

      func _on_body_entered(body: Node) -> void:
            if body.is_in_group("player"):
                  player_inside = true
                  if exhibit_label_3d:
                        exhibit_label_3d.text = "Натисни E"
                  if OS.is_debug_build():
                        print("Kobzar:", name, "Player entered.
player_inside:", player_inside)

      func _on_body_exited(body: Node) -> void:
            if body.is_in_group("player"):
                  player_inside = false
                  if exhibit_label_3d:
                        exhibit_label_3d.text = exhibit_name
                  var tween = create_tween()
                  tween.tween_property(description_label, "modulate:a", 0,
0.3)
                  tween.tween_property(background_rect, "color:a", 0, 0.3)
```

```
                if audio_player and audio_player.playing:
                        audio_player.stop()
                if OS.is_debug_build():
                        print("Kobzar:", name, "Player exited.
player_inside:", player_inside)

        func _input(event: InputEvent) -> void:
                if player_inside and event.is_action_pressed("interact"):
                if OS.is_debug_build():
                        print("Kobzar:", name, "Interact pressed. Current
modulate.a:", description_label.modulate.a)
                if description_label.modulate.a > 0:
                        var tween = create_tween()
                        tween.tween_property(description_label, "modulate:a",
0, 0.3)
                        tween.tween_property(background_rect, "color:a", 0,
0.3)
                        if audio_player and audio_player.playing:
                                audio_player.stop()
                        if OS.is_debug_build():
                                print("Kobzar:", name, "Hiding text.")
                else:
                        var tween = create_tween()
                        tween.tween_property(background_rect, "color:a", 0.5,
0.3)
                        tween.tween_callback(func():
                                description_label.text = description_text
                                if OS.is_debug_build():
                                        print("Kobzar:", name, "Showing text:",
description_text)
                        ).set_delay(0.2)
                        tween.tween_property(description_label, "modulate:a",
1, 0.3)

                if player_inside and event.is_action_pressed("play_music"):
                if audio_player:
                        if not audio_player.playing:
                                audio_player.play()
```

```
                        if OS.is_debug_build():
                              print("Kobzar:", name, "Music started.")
                  else:
                        audio_player.stop()
                        if OS.is_debug_build():
                              print("Kobzar:", name, "Music stopped.")
```

А.8 Програмний код painting.gd

```
extends Node3D

@onready var interaction_area = $InteractionArea
@onready var description_label = $CanvasLayer/DescriptionLabel
@onready var background_rect = $CanvasLayer/BackgroundRect
@onready var painting_sprite = $PaintingSprite

@export var description_text: String = "This is a placeholder text for
the painting!" # Текст опису
@export var painting_texture: Texture2D # Текстура для картини

var player_inside: bool = false # Гравець у зоні

func _ready():
      # Підключаємо сигнали Area3D
      interaction_area.body_entered.connect(_on_body_entered)
      interaction_area.body_exited.connect(_on_body_exited)
      # Приховуємо текст і фон
      description_label.modulate.a = 0
      background_rect.color.a = 0
      # Встановлюємо текстуру для спрайта, якщо вона задана
      if painting_texture:
            painting_sprite.texture = painting_texture
      # Встановлюємо текст опису
      description_label.text = description_text

func _on_body_entered(body: Node) -> void:
      if body.is_in_group("player"):
            player_inside = true
```

```
	func _on_body_exited(body: Node) -> void:
		if body.is_in_group("player"):
			player_inside = false
			# Приховуємо текст і фон
			var tween = create_tween()
			tween.tween_property(description_label, "modulate:a", 0,
0.3)
			tween.tween_property(background_rect, "color:a", 0, 0.3)

	func _input(event: InputEvent) -> void:
		if player_inside and event.is_action_pressed("interact"): #
Клавіша E
			if description_label.modulate.a > 0:
				# Приховуємо текст і фон
				var tween = create_tween()
				tween.tween_property(description_label, "modulate:a",
0, 0.3)
				tween.tween_property(background_rect, "color:a", 0,
0.3)
			else:
				# Показуємо фон і текст
				var tween = create_tween()
				tween.tween_property(background_rect, "color:a", 0.5,
0.3)
				tween.tween_property(description_label, "modulate:a",
1, 0.3)
```

A.9 Програмний код painting_ivana_kupala.gd

```
	extends Node3D

	@onready var interaction_area = $InteractionArea
	@onready var description_label = $CanvasLayer/DescriptionLabel
	@onready var background_rect = $CanvasLayer/BackgroundRect
	@onready var painting_sprite = $PaintingSprite

	var player_inside = false
```

```
	@export var description_text = "Івана Купала — одне з найяскравіших свят українців, що відзначається в ніч на 7 липня. Це час магії, коли молодь стрибає через вогнища, плете вінки й пускає їх на воду, шукаючи долю. Купальські пісні та обряди зберегли давні традиції, пов’язані з природою та вірою в її силу."
	@export var painting_texture: Texture2D:
		set(value):
			painting_texture = value
			if painting_sprite and is_inside_tree():
				if painting_texture:
					painting_sprite.texture = painting_texture.duplicate()
				else:
					painting_sprite.texture = null
	@export var description_font_size = 24

	func _ready():
		if painting_texture:
			painting_sprite.texture = painting_texture.duplicate()

		description_label.add_theme_font_size_override("font_size", description_font_size)

		interaction_area.body_entered.connect(_on_body_entered)
		interaction_area.body_exited.connect(_on_body_exited)

		description_label.modulate.a = 0
		background_rect.color.a = 0

		if OS.is_debug_build():
			print("Painting:", name, "Ready.")

	func _on_body_entered(body: Node) -> void:
		if body.is_in_group("player"):
			player_inside = true
			if OS.is_debug_build():
				print("Painting:", name, "Player entered. player_inside:", player_inside)
```

```
	func _on_body_exited(body: Node) -> void:
		if body.is_in_group("player"):
			player_inside = false
			var tween = create_tween()
			tween.tween_property(description_label, "modulate:a", 0,
0.3)
			tween.tween_property(background_rect, "color:a", 0, 0.3)
			if OS.is_debug_build():
				print("Painting:", name, "Player exited.
player_inside:", player_inside)

	func _input(event: InputEvent) -> void:
		if player_inside and event.is_action_pressed("interact"): #
Натиснення E
			if OS.is_debug_build():
				print("Painting:", name, "Interact pressed. Current
modulate.a:", description_label.modulate.a)
			if description_label.modulate.a > 0:
				var tween = create_tween()
				tween.tween_property(description_label, "modulate:a",
0, 0.3)
				tween.tween_property(background_rect, "color:a", 0,
0.3)
				if OS.is_debug_build():
					print("Painting:", name, "Hiding text.")
			else:
				var tween = create_tween()
				tween.tween_property(background_rect, "color:a", 0.5,
0.3)
				tween.tween_callback(func():
					description_label.text = description_text
					if OS.is_debug_build():
						print("Painting:", name, "Showing text:",
description_text)
				).set_delay(0.2)
				tween.tween_property(description_label, "modulate:a",
1, 0.3)
```

А.10 Програмний код painting_ivana_kupala.gd

```
extends Node3D

@onready var interaction_area = $InteractionArea
@onready var description_label = $CanvasLayer/DescriptionLabel
@onready var background_rect = $CanvasLayer/BackgroundRect

var player_inside = false # Гравець у зоні
@export var plaque_text = "Hall Name" # Короткий текст (назва зали)
@export var description_text = "This is a placeholder text for the plaque!" # Детальна інформація про залу

func _ready():
	# Підключаємо сигнали Area3D
	interaction_area.body_entered.connect(_on_body_entered)
	interaction_area.body_exited.connect(_on_body_exited)
	# Приховуємо текст і фон
	description_label.modulate.a = 0
	background_rect.color.a = 0
	# Встановлюємо короткий текст
	$PlaqueLabel.text = plaque_text

func _on_body_entered(body: Node) -> void:
	if body.is_in_group("player"):
		player_inside = true
		$PlaqueLabel.text = "Натисни E"

func _on_body_exited(body: Node) -> void:
	if body.is_in_group("player"):
		player_inside = false
		# Приховуємо текст і фон
		var tween = create_tween()
		tween.tween_property(description_label, "modulate:a", 0, 0.3)
		tween.tween_property(background_rect, "color:a", 0, 0.3)
		$PlaqueLabel.text = plaque_text
```

```
	func _input(event: InputEvent) -> void:
		if player_inside and event.is_action_pressed("interact"): # 
Клавіша E
			if description_label.modulate.a > 0:
				# Приховуємо текст і фон
				var tween = create_tween()
				tween.tween_property(description_label, "modulate:a", 
0, 0.3)
				tween.tween_property(background_rect, "color:a", 0, 
0.3)
			else:
				# Спочатку затемнюємо фон, потім показуємо текст
				var tween = create_tween()
				tween.tween_property(background_rect, "color:a", 0.5, 
0.3)
				tween.tween_callback(func(): description_label.text = 
description_text).set_delay(0.2)
				tween.tween_property(description_label, "modulate:a", 
1, 0.3)
```

A.11 Програмний код statue.gd

```
	extends Node3D

	@onready var interaction_area = $InteractionArea
	@onready var description_label = $CanvasLayer/DescriptionLabel
	@onready var background_rect = $CanvasLayer/BackgroundRect
	@onready var statue_sprite = $StatueSprite
	@onready var exhibit_label_3d = $ExhibitLabel3D

	var player_inside = false
	@export var description_text = "This is a placeholder text for the 
statue!"
	@export var exhibit_name = "Майстриня-вишивальниця"
	@export var sprite_texture: Texture2D:
		set(value):
			sprite_texture = value
```

```
                if statue_sprite and is_inside_tree():
                        if sprite_texture:
                                statue_sprite.texture =
sprite_texture.duplicate()
                        else:
                                statue_sprite.texture = null
        @export var description_font_size = 24
        @export var label3d_font_size = 16

        func _ready():
                if sprite_texture:
                        statue_sprite.texture = sprite_texture.duplicate()

                description_label.add_theme_font_size_override("font_size",
description_font_size)
                if exhibit_label_3d:
                        exhibit_label_3d.font_size = label3d_font_size

                interaction_area.body_entered.connect(_on_body_entered)
                interaction_area.body_exited.connect(_on_body_exited)

                description_label.modulate.a = 0
                background_rect.color.a = 0
                if exhibit_label_3d:
                        exhibit_label_3d.text = exhibit_name
                        exhibit_label_3d.pixel_size = 0.05
                        exhibit_label_3d.billboard =
BaseMaterial3D.BILLBOARD_ENABLED
                        exhibit_label_3d.position.z = 2.0 # Збільшено відступ
                        exhibit_label_3d.render_priority = 1 # Пріоритет поверх
спрайта

                if OS.is_debug_build():
                        print("Statue:", name, "Ready. ExhibitLabel3D Text:",
exhibit_label_3d.text if exhibit_label_3d else "Null", "Font Size:",
label3d_font_size, "Position:", exhibit_label_3d.position if exhibit_label_3d
else Vector3.ZERO)
```

```
	func _on_body_entered(body: Node) -> void:
		if body.is_in_group("player"):
			player_inside = true
			if exhibit_label_3d:
				exhibit_label_3d.text = "Натисни E"
			if OS.is_debug_build():
				print("Statue:", name, "Player entered.
player_inside:", player_inside)

	func _on_body_exited(body: Node) -> void:
		if body.is_in_group("player"):
			player_inside = false
			if exhibit_label_3d:
				exhibit_label_3d.text = exhibit_name
			var tween = create_tween()
			tween.tween_property(description_label, "modulate:a", 0,
0.3)
			tween.tween_property(background_rect, "color:a", 0, 0.3)
			if OS.is_debug_build():
				print("Statue:", name, "Player exited.
player_inside:", player_inside)

	func _input(event: InputEvent) -> void:
		if player_inside and event.is_action_pressed("interact"):
			if OS.is_debug_build():
				print("Statue:", name, "Interact pressed. Current
modulate.a:", description_label.modulate.a)
			if description_label.modulate.a > 0:
				var tween = create_tween()
				tween.tween_property(description_label, "modulate:a",
0, 0.3)
				tween.tween_property(background_rect, "color:a", 0,
0.3)
				if OS.is_debug_build():
					print("Statue:", name, "Hiding text.")
			else:
				var tween = create_tween()
```

```
                        tween.tween_property(background_rect, "color:a", 0.5,
0.3)
                        tween.tween_callback(func():
                              description_label.text = description_text
                              if OS.is_debug_build():
                                    print("Statue:", name, "Showing text:",
description_text)
                        ).set_delay(0.2)
                        tween.tween_property(description_label, "modulate:a",
1, 0.3)
```

A.12 Програмний код vyshyvanka.gd

```
extends Node3D

      @onready var interaction_area = $InteractionArea
      @onready var title_label = $CanvasLayer/TitleLabel
      @onready var description_label = $CanvasLayer/DescriptionLabel
      @onready var background_rect = $CanvasLayer/BackgroundRect
      @onready var illustration_texture = $CanvasLayer/IllustrationTexture
      @onready var vyshyvanka_sprite = $VyshyvankaSprite

      var player_inside = false # Гравець у зоні
      @export var title_text = "Vyshyvanka Title"
      @export var description_text = "Це текст-заглушка для вишиванки!" #
Текст опису
      @export var illustration: Texture2D:
            set(value):
                  illustration = value
                  if illustration_texture and is_inside_tree():
                        if illustration:
                              illustration_texture.texture =
illustration.duplicate()
                        else:
                              illustration_texture.texture = null
      @export var sprite_texture: Texture2D:
            set(value):
                  sprite_texture = value
```

```
                if vyshyvanka_sprite and is_inside_tree():
                        if sprite_texture:
                                vyshyvanka_sprite.texture =
sprite_texture.duplicate()
                        else:
                                vyshyvanka_sprite.texture = null
        @export var title_font_size = 48
        @export var description_font_size = 24

        func _ready():
                if illustration:
                        illustration_texture.texture = illustration.duplicate()
                if sprite_texture:
                        vyshyvanka_sprite.texture = sprite_texture.duplicate()

                # Встановлюємо розміри шрифтів
                title_label.add_theme_font_size_override("font_size",
title_font_size)
                description_label.add_theme_font_size_override("font_size",
description_font_size)

                # Ініціалізуємо текст для стабільного відображення
                title_label.text = title_text
                description_label.text = description_text

                interaction_area.body_entered.connect(_on_body_entered)
                interaction_area.body_exited.connect(_on_body_exited)

                # Приховуємо інтерфейс
                title_label.modulate.a = 0
                description_label.modulate.a = 0
                background_rect.color.a = 0
                illustration_texture.modulate.a = 0

                if OS.is_debug_build():
                        print("Vyshyvanka:", name, "Ready. TitleLabel Rect:",
title_label.get_rect())
```

```
                print("Vyshyvanka:", name, "DescriptionLabel Rect:",
description_label.get_rect())
                print("Vyshyvanka:", name, "IllustrationTexture Rect:",
illustration_texture.get_rect())

      func _on_body_entered(body: Node) -> void:
            if body.is_in_group("player"):
                player_inside = true
                if OS.is_debug_build():
                      print("Vyshyvanka:", name, "Гравець увійшов.
player_inside:", player_inside)

      func _on_body_exited(body: Node) -> void:
            if body.is_in_group("player"):
                player_inside = false
                var tween = create_tween()
                tween.tween_property(title_label, "modulate:a", 0, 0.3)
                tween.tween_property(description_label, "modulate:a", 0,
0.3)
                tween.tween_property(background_rect, "color:a", 0, 0.3)
                tween.tween_property(illustration_texture, "modulate:a", 0,
0.3)
                if OS.is_debug_build():
                      print("Vyshyvanka:", name, "Гравець вийшов.
player_inside:", player_inside)

      func _input(event: InputEvent) -> void:
            if player_inside and event.is_action_pressed("interact"):
                if OS.is_debug_build():
                      print("Vyshyvanka:", name, "Натиснуто взаємодію.
Поточна прозорість:", description_label.modulate.a)
                if description_label.modulate.a > 0:
                      var tween = create_tween()
                      tween.tween_property(title_label, "modulate:a", 0,
0.3)
                      tween.tween_property(description_label, "modulate:a",
0, 0.3)
```

```
                        tween.tween_property(background_rect, "color:a", 0,
0.3)
                        tween.tween_property(illustration_texture,
"modulate:a", 0, 0.3)
                        if OS.is_debug_build():
                              print("Vyshyvanka:", name, "Приховуємо текст.")
                  else:
                        var tween = create_tween()
                        tween.tween_property(background_rect, "color:a", 0.5,
0.3)
                        tween.tween_callback(func():
                              # Встановлюємо текст та перевіряємо розміри
                              title_label.text = title_text
                              description_label.text = description_text
                              if OS.is_debug_build():
                                    var desc_rect =
description_label.get_rect()
                                    var illus_rect =
illustration_texture.get_rect()
                                    print("Vyshyvanka:", name, "Показуємо
текст. Заголовок:", title_text)
                                    print("Vyshyvanka:", name, "Прямокутник
опису:", desc_rect)
                                    if desc_rect.intersects(illus_rect):
                                          print("ПОМИЛКА: Опис перекриває
ілюстрацію для", name, "!")
                        ).set_delay(0.2)
                        tween.tween_property(title_label, "modulate:a", 1,
0.3)
                        tween.tween_property(description_label, "modulate:a",
1, 0.3)
                        tween.tween_property(illustration_texture,
"modulate:a", 1, 0.3)
```

А.13 Програмний код wreath_ivana_kupala.gd

```
extends Node3D
```

```
	@onready var interaction_area = $InteractionArea
	@onready var description_label = $CanvasLayer/DescriptionLabel
	@onready var background_rect = $CanvasLayer/BackgroundRect
	@onready var video_player = $VideoPlayer
	@onready var wreath_sprite = $WreathSprite

	var player_inside = false
	@export var title_text = "Реконструкція свята Івана Купала"
	@export var wreath_texture: Texture2D:
		set(value):
			wreath_texture = value
			if wreath_sprite and is_inside_tree():
				if wreath_texture:
					wreath_sprite.texture =
wreath_texture.duplicate()
				else:
					wreath_sprite.texture = null
	@export var video_stream: VideoStream:
		set(value):
			video_stream = value
			if video_player and is_inside_tree():
				video_player.stream = video_stream
	@export var description_font_size = 24

	func _ready():
		if wreath_texture:
			wreath_sprite.texture = wreath_texture.duplicate()
		if video_stream:
			video_player.stream = video_stream

		description_label.add_theme_font_size_override("font_size",
description_font_size)

		# Налаштування розміру відео
		if video_player:
			video_player.size = Vector2(200, 112) # Фіксований розмір

		interaction_area.body_entered.connect(_on_body_entered)
```

```
            interaction_area.body_exited.connect(_on_body_exited)

            description_label.modulate.a = 0
            background_rect.color.a = 0
            video_player.visible = false

            if OS.is_debug_build():
                  print("Wreath:", name, "Ready.")

      func _on_body_entered(body: Node) -> void:
            if body.is_in_group("player"):
                  player_inside = true
                  if OS.is_debug_build():
                        print("Wreath:", name, "Player entered.
player_inside:", player_inside)

      func _on_body_exited(body: Node) -> void:
            if body.is_in_group("player"):
                  player_inside = false
                  var tween = create_tween()
                  tween.tween_property(description_label, "modulate:a", 0,
0.3)
                  tween.tween_property(background_rect, "color:a", 0, 0.3)
                  if video_player and video_player.is_playing():
                        video_player.stop()
                  video_player.visible = false
                  if OS.is_debug_build():
                        print("Wreath:", name, "Player exited.
player_inside:", player_inside)

      func _input(event: InputEvent) -> void:
            if player_inside and event.is_action_pressed("interact"): #
Натиснення на вінок (E)
                  if OS.is_debug_build():
                        print("Wreath:", name, "Interact pressed. Current
modulate.a:", description_label.modulate.a)
                  if description_label.modulate.a > 0:
                        var tween = create_tween()
```

```
                        tween.tween_property(description_label, "modulate:a",
0, 0.3)
                        tween.tween_property(background_rect, "color:a", 0,
0.3)
                        if video_player and video_player.is_playing():
                              video_player.stop()
                        video_player.visible = false
                        if OS.is_debug_build():
                              print("Wreath:", name, "Hiding text and video.")
                  else:
                        var tween = create_tween()
                        tween.tween_property(background_rect, "color:a", 0.5,
0.3)
                        tween.tween_callback(func():
                              description_label.text = title_text + "\nНатисни
клавішу П щоб подивитися"
                              if OS.is_debug_build():
                                    print("Wreath:", name, "Showing title:",
title_text)
                        ).set_delay(0.2)
                        tween.tween_property(description_label, "modulate:a",
1, 0.3)
                        video_player.visible = true

            if player_inside and event.is_action_pressed("video_play"): #
Натиснення г (G) для відео
                  if video_player:
                        if not video_player.is_playing():
                              video_player.play()
                              if OS.is_debug_build():
                                    print("Wreath:", name, "Video started.")
                        else:
                              video_player.stop()
                              if OS.is_debug_build():
                                    print("Wreath:", name, "Video stopped.")
```