МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет	Інформаційних технологій
Кафедра	Інформатики і прикладного програмного забезпечення
Спеціальність	Інженерія програмного забезпечення
Форма навчання	Денна

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА[1]

Бессонова Валентина Дмитровича
(прізвище, ім'я, по батькові здобувача)

на тему «Розробка програмного забезпечення моделювання інтер'єру помешкання»
(повна назва теми)

за матеріалами праць провідних спеціалістів з розробки ПЗ та проектування БД
(повна назва бази дослідження)

науковий керівник к.е.н., доцент _____ Лисенко В.С.
(наук. ступінь, вчене звання) *(підпис)* *(прізвище, ініціали)*

Робота допущена до захисту в ЕК
Протокол засідання кафедри
від 11.06.2025 № 12
Завідувач кафедри _____
(підпис)
д.т.н., професор Зеленський О.С.
Наук. ступень, вчене звання *Ініціали, прізвище*

Кривий Ріг – 2025

[1] Назва кваліфікаційної роботи відповідно до ОПП

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет	Інформаційних технологій
Кафедра	Інформатики і прикладного програмного забезпечення
Спеціальність	Інженерія програмного забезпечення
Форма навчання	Денна

«ЗАТВЕРДЖУЮ»
Завідувач кафедри _____ Зеленський О.С.
(підпис) (Прізвище, ініціали)
«11» червня 2025 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ[2]

1. Тема роботи «Розробка програмного забезпечення моделювання інтер'єру помешкання»

Керівник роботи к.е.н., доцент Лисенко В.С.
затверджені наказом закладу вищої освіти від «04» квітня 2025 р. № 222-ст

2. Строк подання здобувачем роботи до «09» червня 2025 р.

3. Зміст кваліфікаційної роботи, об'єкт, предмет та мета дослідження:

Розділ 1. Постановка задачі

Розділ 2. Розробка алгоритму розв'язання задачі

Розділ 3. Організація інформаційного забезпечення

Розділ 4. Розробка програмного забезпечення

Об'єкт дослідження: процес моделювання інтер'єру засобами комп'ютерної графіки

Предмет дослідження: методи та технології візуалізації тривимірних моделей приміщень з можливістю редагування елементів дизайну

Мета кваліфікаційної роботи: *розробка програмного забезпечення моделювання інтер'єру приміщень з можливістю редагування та збереження*

5. Дата видачі завдання «04» квітня 2025 р.

[2] Назва кваліфікаційної роботи відповідно до ОПП

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів МДР	Строк виконання етапів роботи	Відмітка керівника про виконання етапів (дата, підпис)
1	Підготовка розділу 1	04.04.2025-13.04.2025	
2	Підготовка розділу 2	14.04.2025-26.04.2025	
3.	Підготовка розділу 3	27.04.2025-15.05.2025	
4	Підготовка розділу 4	16.05.2025-08.06.2025	
5	Реєстрація завершеної кваліфікаційної роботи	09.06.2025	Реєстраційний №___ «09»червня 2025 р.
6	Отримання відгуку від наукового керівника	10.06.2025	
7	Подання кваліфікаційної роботи на перегляд завідувачу кафедри	11.06.2025	
8	Отримання зовнішньої рецензії	12.06.2025	
9	Попередній захист кваліфікаційної роботи на кафедрі	13.06.2025	
10	Підготовка до захисту в ЕК	16.06.2025-21.06.2025	

Завдання підготував науковий керівник _____ ___Лисенко В.С.___
(підпис) (прізвище та ініціали)

Завдання одержав _____ ___Бессонов В.Д.___
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

на кваліфікаційну бакалаврську роботу

«Розробка програмного забезпечення моделювання інтер’єру помешкання»

Бессонова Валентина Дмитровича

Кваліфікаційна бакалаврська робота на здобуття освітньо-кваліфікаційного рівня бакалавра зі спеціальності 121 «Інженерія програмного забезпечення» – Державний університет економіки і технологій – Кривий Ріг, 2025.

Проведено детальний аналіз характеристик задачі, визначено вхідні дані, необхідні для проектування інтер'єрів (параметри приміщень, елементи дизайну, матеріали), та вихідну інформацію, яку має надавати система. Обґрунтовано актуальність проблеми та визначено основні вимоги до програмного забезпечення.

Розроблено алгоритми розв'язання задачі, що забезпечують функціональність десктопного та веб додатків.

Розроблено ефективну структуру баз даних та інформаційних масивів, що забезпечує швидкодію системи та зручність доступу до елементів бібліотеки компонентів при моделюванні приміщень різного призначення та конфігурації.

Представлено архітектуру програмного забезпечення, описано принципи взаємодії користувача з інтерфейсом, реалізацію інструментів інтерактивного редагування, а також механізми візуалізації 3D-моделей у режимі реального часу.

Ключові слова: 3D-МОДЕЛЬ, АЛГОРИТМИ, ІНФОРМАЦІЙНІ МАСИВИ, БАЗА ДАНИХ, ІНТЕР’ЄР, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, C++, JAVASCRIPT.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СУБД	Систему управління базами даних
ПЗ	Програмне забезпечення
UI	Користувацький інтерфейс
MFC	Microsoft Foundation Class
JS	JavaScript

ЗМІСТ

ВСТУП

У сучасному світі, де технології стрімко розвиваються, а цифровізація охоплює все більше сфер життя, проектування інтер'єрів переходить у віртуальний простір. Моделювання та візуалізація інтер'єру приміщень за допомогою комп'ютерних технологій стає незамінним інструментом як для професійних дизайнерів, так і для звичайних користувачів, які бажають спланувати власний простір.

Актуальність даної роботи зумовлена зростаючим попитом на інтуїтивно зрозумілі інструменти для 3D-моделювання інтер'єрів. Традиційні підходи до проектування приміщень часто вимагають значних технічних знань, спеціалізованого програмного забезпечення та потужного обладнання, що обмежує коло потенційних користувачів. Ринок програмного забезпечення для 3D-моделювання розвивається швидкими темпами, але існує значний розрив між професійними інструментами, які вимагають серйозної підготовки, та спрощеними рішеннями з обмеженими можливостями.

Особливу важливість представляє можливість інтерактивного редагування елементів дизайну в режимі реального часу, що дозволяє прискорити процес проектування та робить його більш інтуїтивним. Розробка універсального крос-платформного рішення, яке поєднує доступність і широкий функціонал, є відповіддю на сучасні ринкові потреби та тенденції проектування інтер'єрів.

Метою дослідження є розробка та реалізація програмного комплексу для моделювання інтер'єру приміщень з можливістю інтерактивного редагування елементів у режимі реального часу, що поєднує доступність веб-інтерфейсу та розширену функціональність десктопного додатку.

Відповідно до мети було сформульовано наступні завдання дослідження: дослідити технології та методи реалізації інтерактивних 3D-моделей для віртуалізації інтер'єрів; розробити архітектуру програмного комплексу, що поєднує веб-інтерфейс та десктопний додаток; реалізувати

функціонал для інтерактивного редагування елементів дизайну в режимі реального часу; створити інтуїтивно зрозумілий UI, доступний для користувачів різного рівня підготовки; провести тестування та оцінку ефективності розробленого програмного рішення.

Об'єктом дослідження виступає процес моделювання та візуалізації інтер'єрів приміщень з використанням сучасних комп'ютерних технологій.

Предметом дослідження є методи та технології розробки крос-платформного програмного забезпечення для інтерактивного 3D-моделювання та редагування елементів інтер'єру в режимі реального часу.

Практична значущість даної роботи полягає у створенні доступного та функціонального інструментарію для проектування інтер'єрів, що дозволить користувачам з різним рівнем підготовки візуалізувати свої ідеї та приймати обґрунтовані рішення щодо дизайну приміщень. Розроблена система спрощує комунікацію між дизайнерами та клієнтами, дозволяє експериментувати з різними стилями та уникати помилок у плануванні простору, що в результаті економить час і ресурси.

Таким чином, ця робота є важливим кроком у напрямку розвитку доступних інструментів для 3D-моделювання інтер'єрів та підвищення загальної якості проєктування житлових приміщень.

РОЗДІЛ 1
ПОСТАНОВКА ЗАДАЧІ

1.1 Характеристика задачі

Постановка задачі є одним з найбільш важливих етапів при створенні програмного забезпечення. Від того, наскільки повно, точно і ясно визначені вимоги до розроблюваного програмного забезпечення, його функції та можливості, значною мірою залежить якість кінцевого продукту та відповідність потребам користувачів.

Успіх проєкту розробки програмного забезпечення значною мірою залежить від якості вихідних вимог. Дефекти у вимогах, виявлені на пізніх стадіях розробки, можуть коштувати в 100 разів дорожче, ніж ті, що виявлені на етапі аналізу [1, с.105].

При розробці програмного забезпечення для моделювання інтер'єру приміщень було поставлено задачу створити крос-платформне рішення, що складається з двох основних компонентів:

1. Веб-додаток, який дозволяє користувачам створювати 3D-моделі кімнат, додавати вікна, двері, застосовувати текстури та керувати освітленням.
2. Десктопна програма на основі MFC, яка розширює функціональність веб-додатка, додаючи можливості застосування шейдерів, імпорту 3D-об'єктів з файлів та збереження проектів у спеціалізованому форматі.

Кінцевий продукт має відповідати наступним основним вимогам:

1. Функціональність:
   - Створення базової 3D-моделі приміщення з вказанням розмірів
   - Додавання та редагування елементів інтер'єру (двері, вікна)
   - Застосування та налаштування текстур для різних поверхонь

- Керування параметрами освітлення
- Збереження та завантаження проектів
- У десктопній версії - підтримка шейдерів, імпорту 3D-моделей і текстур

2. Зручність використання:
   - Інтуїтивно зрозумілий інтерфейс
   - Мінімальний поріг входження для користувачів без спеціальної підготовки
   - Логічна організація інструментів та меню
3. Продуктивність та надійність:
   - Оптимізація роботи з 3D-графікою для забезпечення плавності відображення
   - Стабільна робота з моделями різної складності
   - Коректне збереження та відтворення даних проекту
4. Кросплатформність та доступність:
   - Веб-додаток має працювати в сучасних браузерах без встановлення додаткового ПЗ
   - Десктопна програма має бути сумісною з операційною системою Windows
   - Мінімальні вимоги до апаратного забезпечення для базової функціональності
5. Безпека та захист даних:
   - Захист користувацьких проектів від несанкціонованого доступу
   - Безпечне збереження даних у локальному сховищі

Особливу увагу в розробці програмного забезпечення приділено питанням візуалізації 3D-моделей, оскільки якість та реалістичність відображення інтер'єру є ключовими факторами при оцінці дизайнерських рішень. Також важливим аспектом є інтерактивність взаємодії користувача з

моделлю, що дозволяє в режимі реального часу бачити результати внесених змін.

Інтерактивна візуалізація в сучасних 3D-програмах повинна забезпечувати швидкий відгук на дії користувача, оскільки затримки понад 100 мс помітно погіршують користувацький досвід та знижують продуктивність роботи з системою. Особливо критичним є баланс між якістю візуалізації та продуктивністю, що потребує ретельної оптимізації рендерингу [2, с.427].

В результаті реалізації поставлених задач має бути отримано ПЗ, яке дозволить користувачам різного рівня підготовки створювати, редагувати та візуалізувати 3D-моделі інтер'єрів приміщень з можливістю детального налаштування елементів дизайну та освітлення.

1.2 Вхідна інформація

Розроблюване програмне забезпечення для моделювання інтер'єру приміщень працює з різними типами вхідних даних, які користувач надає системі в процесі роботи.

Правильне збирання та обробка користувацьких даних у 3D-моделюванні вимагає чіткої класифікації вхідних параметрів та розробки відповідних алгоритмів валідації. Система повинна забезпечувати перевірку на реалістичність введених величин, оскільки навіть незначні помилки в розмірах можуть призвести до суттєвих спотворень кінцевої моделі [3, с.54].

Для роботи з веб-додатком користувачу необхідно зареєструватися в системі, надавши ім'я, прізвище, номер телефону та пароль для авторизації. Ці дані зберігаються в базі даних та використовуються для ідентифікації користувача, а також для забезпечення доступу до збережених проектів через особистий кабінет.

При створенні або редагуванні моделі приміщення користувач має змогу змінювати розміри стін, включаючи довжину, висоту та ширину. Система

відображає початкову стандартну кімнату з типовими розмірами, яку можна модифікувати за потребами користувача. Для вікон і дверей можна задавати розміри (ширину та висоту) та позицію на стіні.

Користувач має можливість застосовувати різні текстури до елементів приміщення. Програма містить вбудовану колекцію текстур для стін і підлоги. Також передбачена можливість завантажити власні текстури з файлів у форматах JPEG та PNG.

В десктопній версії програми користувач може імпортувати 3D-моделі об'єктів для розміщення в приміщенні. Підтримуються формати файлів OBJ для геометрії об'єкта, MTL для матеріалів і JPEG або PNG для текстур. При розміщенні об'єктів можна регулювати їх позицію, масштаб та орієнтацію в просторі.

Для створення реалістичного відображення інтер'єру користувач може налаштовувати загальне освітлення, змінюючи його інтенсивність та напрямок.

Для забезпечення стабільності роботи програми встановлені обмеження на розміри приміщення, вікон та дверей. Мінімальні та максимальні значення для площі кімнати, висоти стелі, розмірів вікон і дверей є важливими параметрами, що забезпечують реалістичність моделі.

Всі вхідні дані перевіряються системою для забезпечення коректної роботи програми та уникнення помилок при візуалізації інтер'єру приміщення.

1.3 Вихідна інформація

Ефективні системи моделювання інтер'єру повинні забезпечувати збереження проектів у форматах, що підтримують як візуальну інформацію, так і структуровані метадані. Особливо важливим є баланс між рівнем деталізації зберігання даних та економією дискового простору, оскільки 3D-моделі можуть швидко збільшуватись у розмірі з додаванням нових елементів [4, с.142].

Основним результатом роботи користувача в системі є створена 3D-модель інтер'єру приміщення. У веб-додатку проекти зберігаються в базі даних під обліковим записом користувача з можливістю подальшого доступу через особистий кабінет. Це дозволяє користувачам повертатися до своїх проектів з будь-якого пристрою, що має доступ до інтернету. Десктопна програма зберігає проекти у форматі XML, що містить всю інформацію про модель приміщення, та дозволяє відкривати раніше збережені файли для продовження роботи.

Важливим вихідним продуктом є візуальне представлення спроектованого інтер'єру. Система генерує реалістичне тривимірне зображення приміщення з усіма доданими елементами, текстурами та освітленням. Користувач може взаємодіяти з цим зображенням, оглядаючи простір під різними кутами та оцінюючи дизайнерські рішення.

Система формує та зберігає технічні дані про спроектоване приміщення, включаючи:

- Розміри всіх елементів (стін, вікон, дверей)
- Перелік використаних матеріалів і текстур
- Налаштування освітлення
- Розташування об'єктів інтер'єру (у десктопній версії)

У десктопній версії програми вихідний XML-файл містить структуровану інформацію про всі елементи приміщення. Файл включає дані про:

- Розміри приміщення
- Параметри вікон і дверей (розміри, позиції)
- Застосовані текстури з посиланнями на ресурси
- Параметри розміщених 3D-об'єктів

У веб-версії системи в особистому кабінеті користувача відображаються:

- Список збережених проектів з назвою

- Дата створення/останньої модифікації проектів
- Посилання на сторінку з приміщенням для подальшого редагування

Вихідні дані програмного забезпечення зберігаються у форматах, що забезпечують цілісність інформації та можливість подальшого редагування. Це дозволяє користувачам створювати, зберігати та вдосконалювати свої проекти інтер'єрів з максимальною зручністю та ефективністю.

Висновки до розділу 1

У даному розділі було детально розглянуто постановку задачі створення програмного забезпечення для моделювання інтер'єру приміщень. Визначено, що якісна постановка задачі є критично важливим етапом розробки, оскільки дефекти у вимогах, виявлені на пізніх стадіях, можуть коштувати набагато дорожче порівняно з етапом аналізу.

Сформульовано комплекс основних вимог до програмного продукту, які охоплюють функціональність, зручність використання, продуктивність, кросплатформність та безпеку даних.

Особлива увага приділена візуалізації 3D-моделей та інтерактивності взаємодії користувача з системою, що має забезпечувати швидкий відгук на дії користувача (менше 100 мс) для оптимального досвіду роботи.

Детально описано структуру вхідних даних, які система отримує від користувачів — від реєстраційної інформації до параметрів приміщення, текстур та імпортованих 3D-об'єктів. Визначено формати вихідних даних, включаючи збереження проектів (XML-формат для десктопної версії та база даних для веб-додатку), візуальне представлення та технічну інформацію про елементи інтер'єру.

РОЗДІЛ 2
РОЗРОБКА АЛГОРИТМУ РОЗВ’ЯЗАННЯ ЗАДАЧІ

2.1 Розробка алгоритму розв’язання задачі на мові програмування C++

Для виконання поставленої задачі моделювання інтер'єру приміщення необхідно розробити алгоритм, який забезпечить ефективну візуалізацію та редагування 3D-моделі. Десктопна програма, реалізована мовою програмування C++ з використанням бібліотеки MFC, дозволяє створювати та редагувати 3D-моделі приміщень з розширеними можливостями.

Розробка алгоритмів редагування 3D-моделей вимагає особливої уваги до оптимізації рендерингу та управління пам'яттю, оскільки неефективна реалізація може спричинити значні затримки при взаємодії з користувачем, особливо при роботі зі складними моделями на комп'ютерах середньої потужності [5, с.347].

Робота з програмою здійснюється на основі управління через пункти головного меню та панелі інструментів (рис. 2.1).

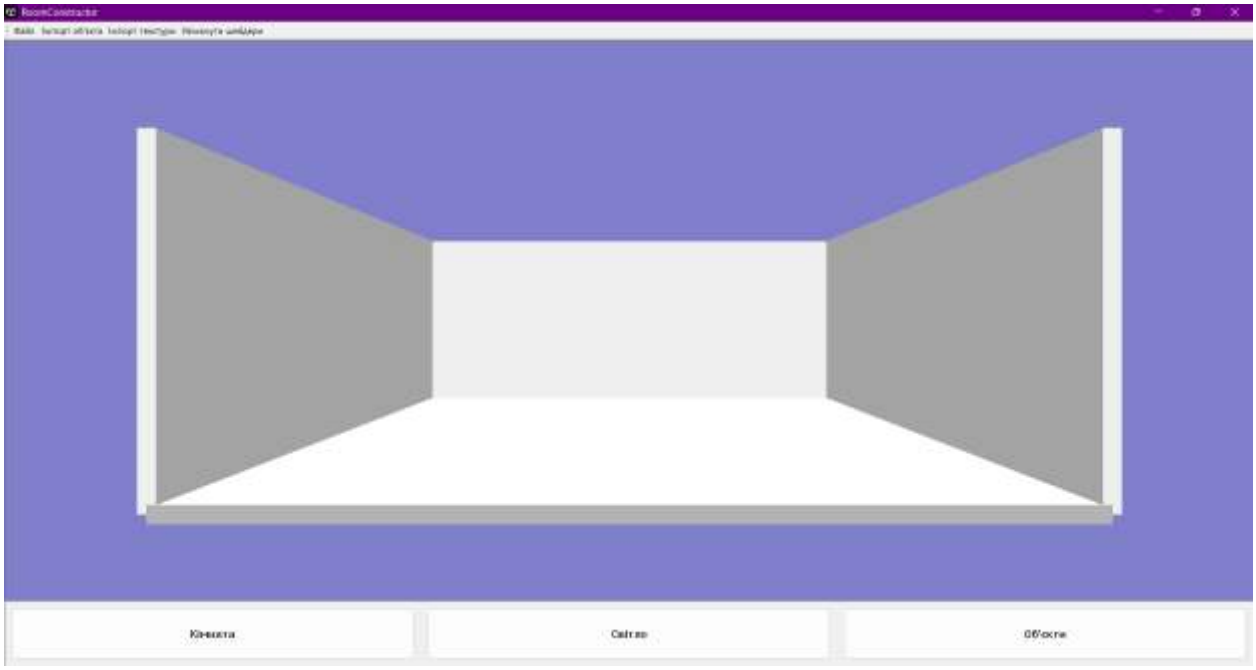

Рис. 2.1. Головне вікно програми

Окрім основної функціональності відображення кімнати, програма включає інструменти для управління освітленням, що реалізовано безпосередньо у меню (рис. 2.2).

Рис. 2.2. Меню управління освітленням

Для додавання нових елементів в кімнату використовується меню «Об’єкти» (рис. 2.3).

Рис. 2.3. Меню «Об’єкти»

Редагування параметрів приміщення реалізовано у меню «Кімната» (рис. 2.4).

Рис. 2.4. Меню «Кімната»

Основний алгоритм роботи програми складається з наступних кроків:

1. Ініціалізація програми та створення стандартної моделі кімнати.
2. Відображення базової 3D-моделі приміщення з типовими параметрами.
3. Надання користувачу можливості модифікувати параметри приміщення через інтерфейс.
4. Обробка введених даних та оновлення 3D-моделі в реальному часі.
5. Збереження проекту у форматі XML.

Повний алгоритм роботи програми представлено нижче (рис. 2.5).

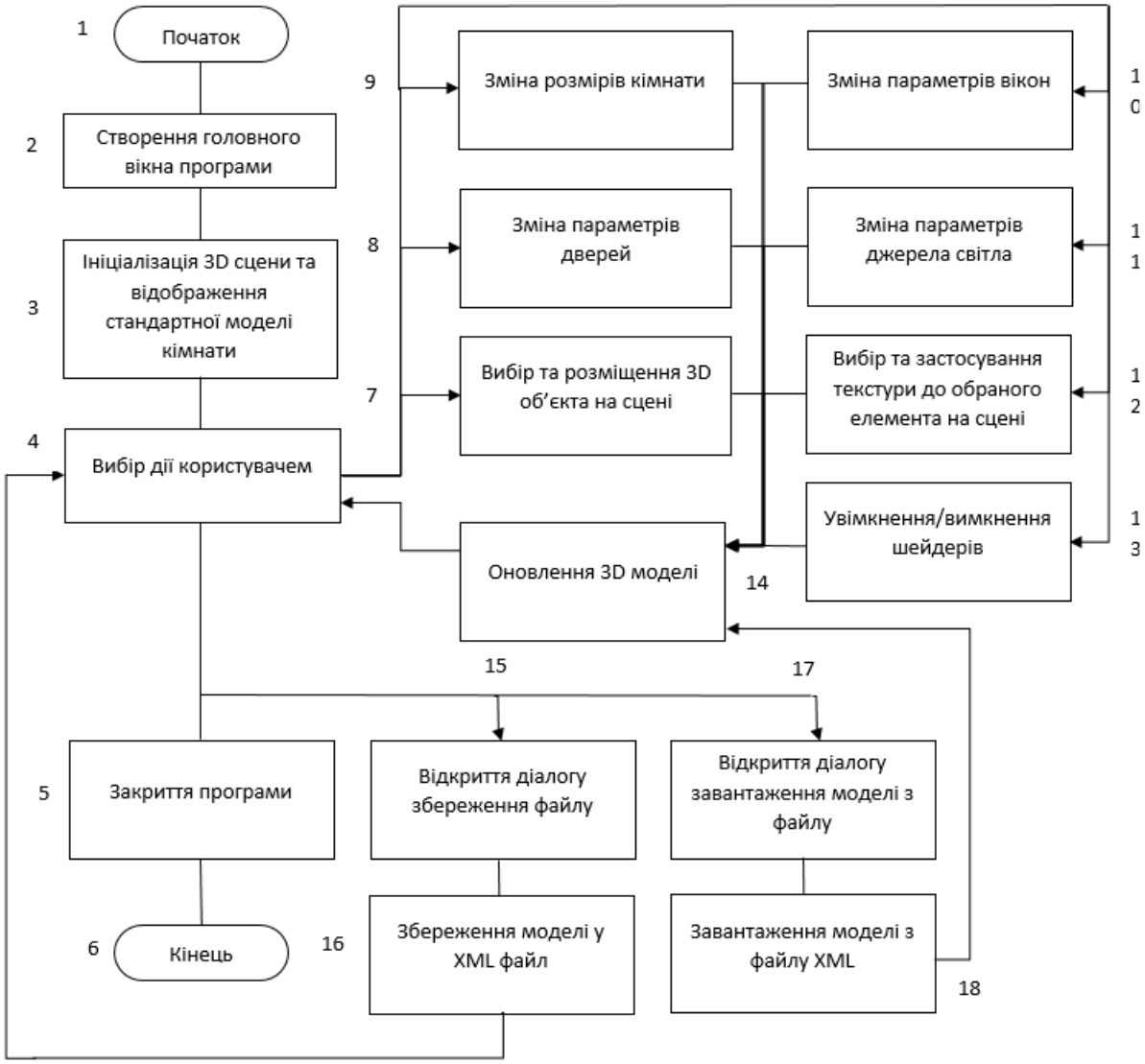

Рис. 2.5. Алгоритм роботи програми

2.2 Розробка алгоритму розв’язання задачі на мові програмування JS

Для виконання поставленої задачі моделювання інтер'єру приміщення у веб-середовищі необхідно розробити алгоритм, що забезпечить ефективну візуалізацію та редагування 3D-моделі через веб-інтерфейс. Веб-додаток, реалізований з використанням HTML5, CSS, JavaScript та бібліотек Node.js та Three.js, дозволяє створювати та редагувати 3D-моделі приміщень з базовими можливостями, доступними через веб-браузер.

Робота з конструктором здійснюється через інтуїтивно зрозумілий інтерфейс користувача, що складається зі сцени та єдиного меню керування у нижній частині сторінки (рис. 2.6).

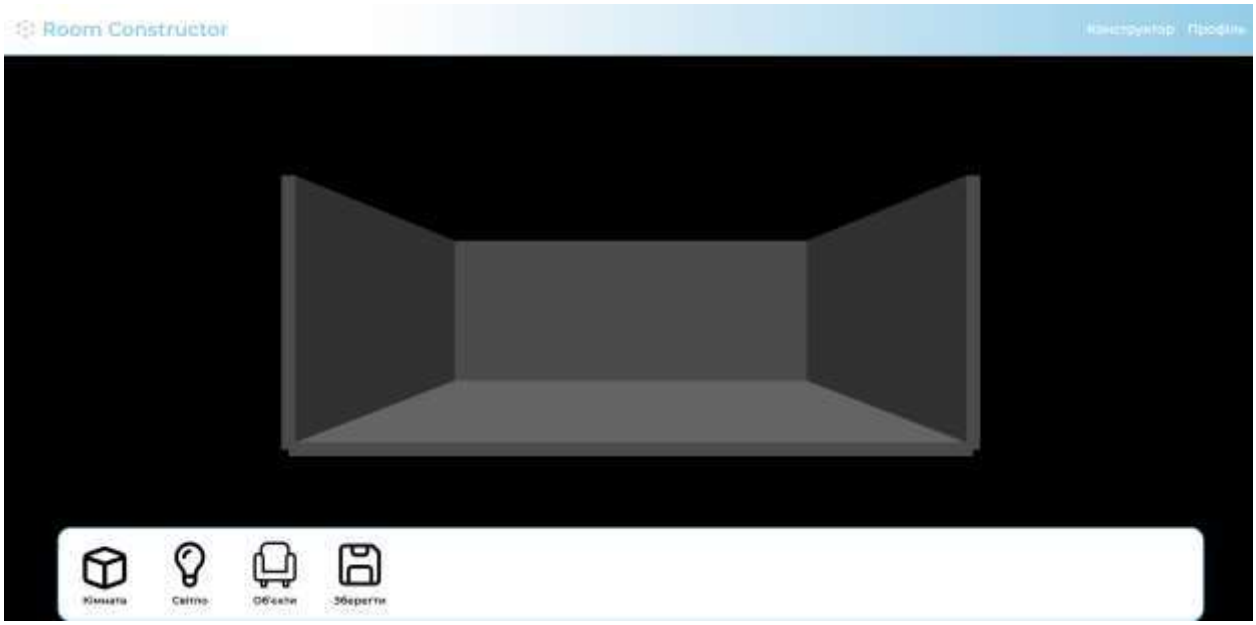

Рис. 2.6. Сторінка конструктора

Для зміни параметрів кімнати використовується спеціальна панель керування (рис. 2.7).

Рис. 2.7. Панель керування кімнатою

Вибір та застосування текстур до елементів інтер'єру відбувається через вбудовану колекцію текстур (рис. 2.8).

Рис. 2.8. Панель вибору текстур

У панелі керування джерелом світла можна налаштувати позицію, кут й інтенсивність освітлення приміщення (рис. 2.9).

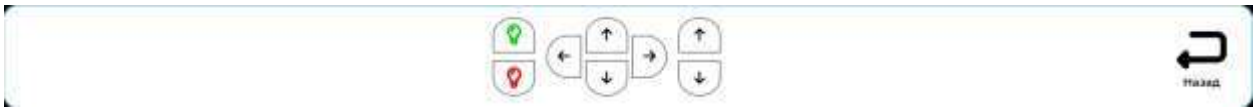

Рис. 2.9. Панель керування світлом

У веб-додатку також передбаченя можливість додавати на сцену вбудовані об’єкти (рис. 2.10).

Рис. 2.10. Панель вибору об’єктів

Збережені кімнати користувач має змогу переглянути в особистому кабінеті (рис. 2.11) і, за потреби, видалити або відредагувати її.

Рис. 2.11. Список збережених кімнат на сторінці особистого кабінету

Основний алгоритм роботи веб-додатку складається з наступних кроків:

1. Ініціалізація веб-додатку та створення контексту WebGL.
2. Відображення базової 3D-моделі приміщення з типовими параметрами.
3. Надання користувачу можливості модифікувати параметри приміщення через інтерфейс.
4. Обробка введених даних та оновлення 3D-моделі в реальному часі.
5. Авторизація користувача
6. Збереження проекту у базі даних з можливістю подальшого доступу через особистий кабінет.

Повний алгоритм роботи веб-додатку представлено нижче (рис. 2.12).

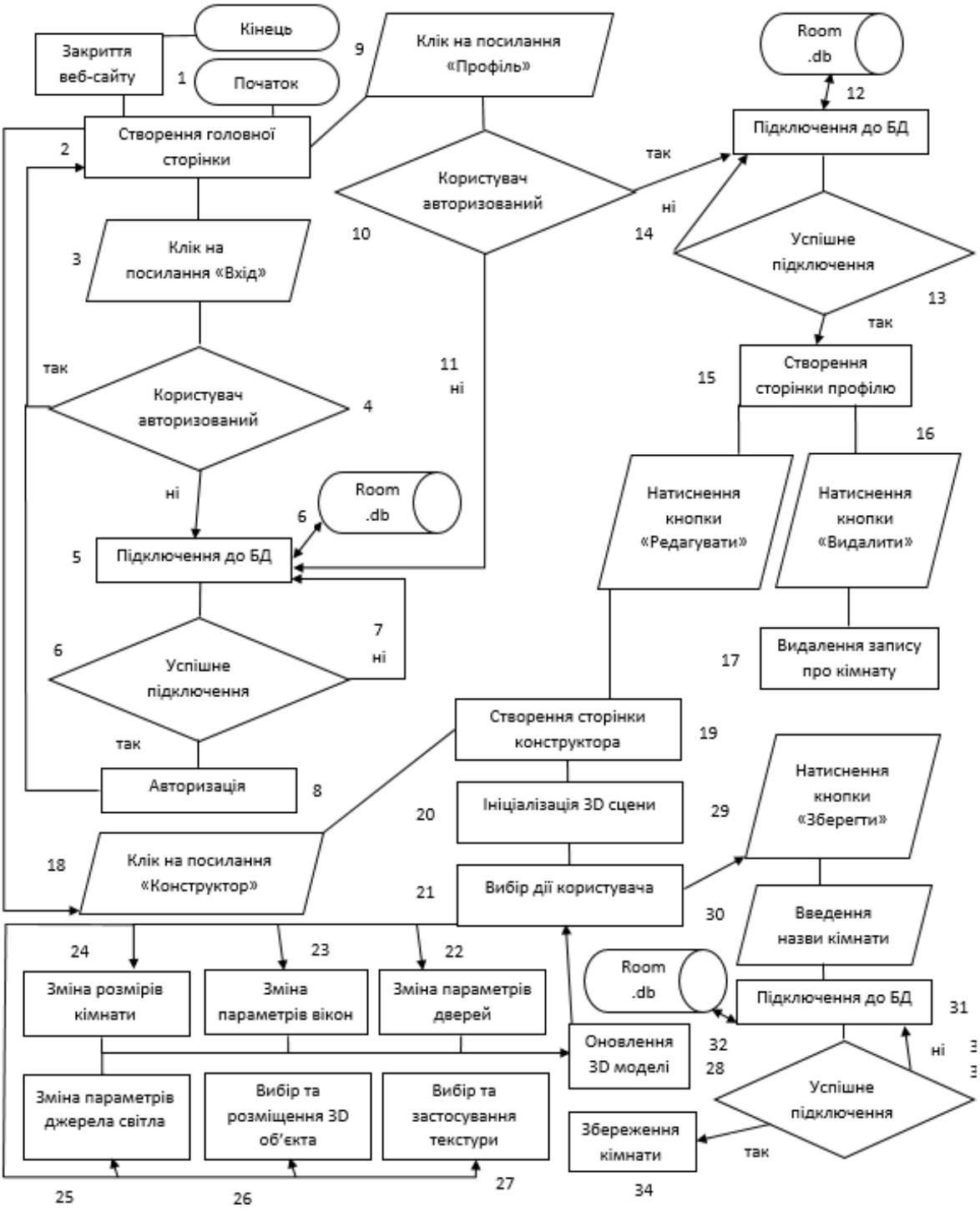

Рис. 2.12. Алгоритм роботи веб-додатку

Представлений алгоритм забезпечує послідовну обробку користувацьких дій та синхронне оновлення 3D-сцени, що гарантує стабільну роботу системи моделювання. Алгоритм передбачає ефективне управління пам'яттю через оптимізовані методи рендерингу та кешування текстур, що мінімізує навантаження на веб-браузер користувача. Модульна структура алгоритму дозволяє легко інтегрувати додаткові функції та розширювати

можливості системи без значного впливу на продуктивність основних компонентів. Запропонований підхід може бути адаптований для різних типів 3D-моделювання у веб-середовищі з урахуванням специфічних вимог конкретних застосувань.

Висновки до розділу 2

У даному розділі детально розглянуто алгоритмічні основи розробленого програмного забезпечення для моделювання інтер'єру приміщень у двох реалізаціях.

Для десктопної програми представлено інтерфейс користувача з системою меню та панелями інструментів, включаючи спеціалізовані функції управління освітленням. Окреслено основний алгоритм роботи програми, що включає п'ять ключових етапів: ініціалізацію з створенням стандартної моделі, відображення базової 3D-моделі, надання інтерфейсу для модифікації параметрів, обробку даних з оновленням моделі в реальному часі та збереження проекту у форматі XML.

Для веб-додатку описано інтерфейс з єдиним меню керування в нижній частині екрану та основну сцену візуалізації. Представлено алгоритм роботи веб-версії, що складається з шести етапів: ініціалізації з контекстом WebGL, відображення базової моделі, надання інтерфейсу модифікації, обробки даних з оновленням моделі, авторизації користувача та збереження проекту в базі даних.

Особливу увагу в розробці алгоритмів приділено оптимізації рендерингу та управлінню пам'яттю для забезпечення швидкої взаємодії користувача з програмою навіть при роботі зі складними моделями на комп'ютерах середньої потужності. Обидва рішення забезпечують інтуїтивно зрозумілий інтерфейс та ефективний робочий процес для користувачів з різним рівнем технічної підготовки.

РОЗДІЛ 3
ОРГАНІЗАЦІЯ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Загальна характеристика інформаційного забезпечення

Для веб-додатку основним сховищем даних є реляційна база даних, організована з використанням системи управління MySQL з механізмом InnoDB. InnoDB - це універсальний механізм сховища, який поєднує високу надійність та високу продуктивність [6].

В десктопному додатку дані зберігаються у вигляді XML-файлів на локальному комп'ютері користувача. XML-файл – це тип розширюваного файлу мови розмітки, який використовується для структурування даних для зберігання та транспортування. Він використовує ієрархічну структуру елементів та користувацьких тегів для зберігання даних, які потім можуть бути зчитані комп’ютерними програмами [7].

Обидва підходи забезпечують структуроване зберігання та швидкий доступ до інформації про 3D-моделі.

Головним завданням системи зберігання даних є гарантоване збереження інформації про моделі та забезпечення швидкого доступу до них. База даних веб-додатку складається з таблиць для зберігання інформації про користувачів, їхні проекти, параметри приміщень, застосовані текстури та інші елементи дизайну.

База даних для розробленого програмного продукту відповідає таким основним критеріям:

- Структурованість — дані організовані у взаємопов'язані таблиці з чітко визначеними полями;
- Наявність ключових полів та зв'язків між таблицями для забезпечення цілісності даних;
- Використання типу InnoDB для створення зв'язків між таблицями;
- Контроль валідності даних перед внесенням змін у базу даних;

- Захищеність як на клієнтській, так і на серверній частині;
- Розмежування прав доступу — звичайні користувачі мають доступ лише до власних проектів, а адміністратори можуть керувати каталогом текстур та користувачами.

У десктопній програмі файли XML зберігають інформацію про всі елементи моделі в структурованому вигляді. Кожен XML-файл містить дані про розміри кімнати, позиції та параметри вікон, дверей, застосовані текстури, налаштування освітлення та розміщені 3D-об'єкти.

Важливою складовою інформаційного забезпечення є також вбудована бібліотека текстур і 3D-об'єктів, яка надає користувачам набір готових елементів для моделювання інтер'єру. У десктопній версії програми додатково передбачена можливість завантаження користувацьких текстур та 3D-моделей у форматах OBJ/MTL.

Така організація інформаційного забезпечення дозволяє ефективно зберігати, обробляти та використовувати дані, необхідні для функціонування програмного забезпечення моделювання інтер'єру приміщень, забезпечуючи при цьому зручність роботи користувачів та цілісність інформації.

3.2 Структура баз даних та інформаційних масивів

У базі даних для зберігання інформації про користувачів системи моделювання інтер'єру використовується таблиця «user». Ця таблиця містить основні дані для авторизації та ідентифікації користувачів, що працюють з веб-додатком. Вона є ключовою складовою системи, оскільки забезпечує персоналізований доступ до проектів та їх збереження відповідно до облікових записів.

Правильно спроектована схема бази даних користувачів системи є основою успішної реалізації механізмів аутентифікації та авторизації, забезпечуючи належний рівень безпеки та персоналізації при роботі з додатками моделювання та візуалізації [8, с.324]. Лістинг коду бази даних

наведено в додатку А. Нижче наведено структуру таблиці з описом полів (табл. 3.1).

Таблиця 3.1

Опис складових частин об'єкта «user»

Назва атрибута	Тип поля	Довжина	Дублювання значень	Опис
u_id	INT, PK	11	Ні	Первинний ключ
login	VARCHAR	50	Ні	Логін користувача
password	VARCHAR	100	Так	Пароль користувача

Для зберігання інформації про спроектовані користувачами кімнати в системі використовується таблиця «saved_room». Ця таблиця є центральним елементом бази даних, в якій зберігаються всі створені моделі приміщень разом з їх параметрами. Кожен запис у таблиці представляє окрему 3D-модель кімнати з унікальними характеристиками та прив'язкою до конкретного користувача. Таблиця «saved_room» пов'язана з таблицею «user» через зовнішній ключ, що забезпечує зв'язок між користувачами та їхніми проектами. Нижче наведено структуру таблиці з описом полів (табл. 3.2).

Таблиця 3.2

Опис складових частин об'єкта «saved_room»

Назва атрибута	Тип поля	Довжина	Дублювання значень	Опис
r_id	INT, PK	11	Ні	Первинний ключ
user	INT, FK user(u_id)	11	Так	Зовнішній ключ на ключ користувача в таблиці «user»
wall_width	INT	11	Так	Ширина кімнати
wall_length	INT	11	Так	Довжина кімнати

Продовження табл. 3.2

wall_height	INT	11	Так	Висота кімнати
wall_thickness	INT	11	Так	Товщина стін
presented_walls	LONGTEXT	-	Так	JSON об’єкт, що включає інформацію про стан стін
objects	LONGTEXT	-	Так	JSON об’єкт, що включає інформацію про стан об’єктів
lights	LONGTEXT	-	Так	JSON об’єкт, що включає інформацію про стан світла
light_x	INT	11	Так	Позиція світла по вісі X
light_y	INT	11	Так	Позиція світла по вісі У
light_z	INT	11	Так	Позиція світла по вісі Z
light_intensity	INT	11	Так	Яскравість світла
room_name	VARCHAR	50	Так	Назва кімнати
datetime	DATETIME	-	Так	Дата та час створення або останнього редагування кімнати

Для зберігання та організації даних про кімнати в десктопній версії програми використовується XML-формат, який забезпечує структуроване зберігання всіх параметрів 3D-моделі. На відміну від веб-версії, де дані зберігаються в базі даних MySQL, десктопний додаток зберігає інформацію у вигляді XML-файлів на локальному комп'ютері користувача. Такий підхід дозволяє працювати з програмою без підключення до інтернету та забезпечує зручний обмін проектами між користувачами. XML-файл містить детальну інформацію про розміри кімнати, параметри стін, вікон, дверей, текстур та розміщених об'єктів.

Використання XML для зберігання структурованих даних у локальних додатках надає значні переваги в контексті портативності та інтеграції, особливо коли потрібен формат, який є самоописовим та легко трансформується без втрати семантичної цілісності даних [9, с.267]. Нижче наведено структуру XML-файлу з описом основних елементів (табл. 3.3).

Таблиця 3.3

Опис складових частин XML-файлу кімнати

Назва елементу	Тип даних	Батьківський елемент	Обов'язковість	Опис
Room	Контейнер	Кореневий	Так	Головний елемент, що містить всю інформацію про кімнату
WallWidth	INT	Room	Так	Ширина кімнати в умовних одиницях
WallLength	INT	Room	Так	Довжина кімнати в умовних одиницях
WallHeight	INT	Room	Так	Висота кімнати в умовних одиницях
WallThickness	INT	Room	Так	Товщина стін в умовних одиницях
LightX	FLOAT	Room	Так	Координата X джерела світла
LightY	FLOAT	Room	Так	Координата Y джерела світла

Продовження табл. 3.3

LightZ	FLOAT	Room	Так	Координата Z джерела світла
LightIntensity	FLOAT	Room	Так	Інтенсивність джерела світла
Walls	Контейнер	Room	Так	Контейнер для елементів стін
Wall	Контейнер	Walls	Так	Опис окремої стіни
Name	STRING	Wall	Так	Ідентифікатор стіни (back, front, left, right)
Presented	BOOLEAN	Wall	Так	Наявність стіни у візуалізації
HasWindow	BOOLEAN	Wall	Так	Наявність вікна на стіні
HasDoor	BOOLEAN	Wall	Так	Наявність дверей на стіні
WindowWidth	INT	Wall	Ні	Ширина вікна (якщо HasWindow=true)
WindowHeight	INT	Wall	Ні	Висота вікна (якщо HasWindow=true)
WindowOffsetY	INT	Wall	Ні	Зміщення вікна по висоті (якщо HasWindow=true)
DoorWidth	INT	Wall	Ні	Ширина дверей (якщо HasDoor=true)
DoorHeight	INT	Wall	Ні	Висота дверей (якщо HasDoor=true)
DoorOffsetX	INT	Wall	Ні	Зміщення дверей по горизонталі (якщо HasDoor=true)
TexturePath	STRING	Wall	Ні	Шлях до файлу текстури стіни
FloorTexture	STRING	Room	Так	Шлях до файлу текстури підлоги
Objects	Контейнер	Room	Ні	Контейнер для об'єктів інтер'єру
Object	Контейнер	Objects	Ні	Опис окремого об'єкта
Type	STRING	Object	Так	Тип об'єкта (вбудований або імпортований)
Name	STRING	Object	Так	Назва об'єкта

Продовження табл. 3.3

PosX	FLOAT	Object	Так	Позиція об'єкта по вісі X
PosY	FLOAT	Object	Так	Позиція об'єкта по вісі Y
PosZ	FLOAT	Object	Так	Позиція об'єкта по вісі Z
RotX	FLOAT	Object	Так	Обертання об'єкта навколо вісі X
RotY	FLOAT	Object	Так	Обертання об'єкта навколо вісі Y
RotZ	FLOAT	Object	Так	Обертання об'єкта навколо вісі Z
Scale	FLOAT	Object	Так	Масштаб об'єкта
Visible	BOOLEAN	Object	Так	Видимість об'єкта
ModelPath	STRING	Object	Ні	Шлях до файлу моделі (якщо Type=importedModel)

Таким чином, для зберігання даних у системі моделювання інтер'єру використовуються дві основні таблиці бази даних: «user» для управління обліковими записами користувачів та «saved_room» для збереження параметрів створених 3D-моделей приміщень з прив'язкою до конкретних користувачів через зовнішній ключ. У десктопній версії програми замість бази даних застосовується XML-формат для локального зберігання всіх параметрів проектів, що забезпечує автономну роботу без підключення до інтернету та зручний обмін файлами між користувачами.

Висновки до розділу 3

У даному розділі представлено різні підходи до організації зберігання даних, що забезпечують структуроване збереження інформації та швидкий доступ до 3D-моделей.

Для веб-додатку обрано реляційну базу даних MySQL з механізмом InnoDB, що забезпечує оптимальне поєднання надійності та продуктивності. База даних відповідає ключовим критеріям структурованості, наявності

зв'язків між таблицями, контролю валідності даних та розмежування прав доступу. Детально описано структуру основних таблиць системи: «user» для зберігання інформації про користувачів та «saved_room» для збереження параметрів спроектованих приміщень.

Для десктопної програми реалізовано зберігання даних у форматі XML на локальному комп'ютері користувача. Цей формат забезпечує ієрархічну структуру даних та зручний спосіб організації інформації про всі елементи моделі. XML-файли містять вичерпні дані про параметри приміщення, включаючи розміри, елементи інтер'єру, текстури та налаштування освітлення.

Важливою частиною інформаційного забезпечення є вбудована бібліотека текстур і 3D-об'єктів, що надає користувачам набір готових елементів для моделювання. У десктопній версії додатково передбачено можливість завантаження користувацьких текстур та 3D-моделей у поширених форматах.

Обрані підходи до організації інформаційного забезпечення дозволяють ефективно зберігати та обробляти дані, забезпечуючи зручність роботи та цілісність інформації, що є критично важливим для програмного забезпечення.

РОЗДІЛ 4
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗАДАЧІ

4.1 Опис головного модулю програми (головного вікна) на мові С++

Головний модуль програми реалізовано з використанням бібліотеки MFC, що забезпечує об'єктно-орієнтований підхід до створення графічного інтерфейсу користувача для Windows-додатків. Основне вікно програми представляє собою інтерактивне середовище для моделювання та конструювання 3D-кімнат з можливістю додавання різноманітних елементів інтер'єру, налаштування освітлення та застосування шейдерів.

Розробка з використанням MFC надає розробникам потужний фреймворк для створення високопродуктивних додатків з розвиненим графічним інтерфейсом, де кожен елемент управління є об’єктом з власною поведінкою та властивостями, що суттєво спрощує побудову складних візуальних систем взаємодії з користувачем [10, с.437].

При запуску програми користувач бачить головне вікно з областю відображення 3D-сцени (рис. 4.1), що займає центральну частину екрану, та панеллю інструментів, розташованою у верхній частині вікна. Нижня частина інтерфейсу містить вкладки для управління різними аспектами моделювання: «Кімната», «Світло» та «Об'єкти». Така структура інтерфейсу забезпечує інтуїтивне та зручне керування усіма функціями програми.

Ефективний дизайн інтерфейсу спеціалізованого програмного забезпечення повинен забезпечувати збалансований розподіл уваги користувача між робочою областю та елементами керування, де основний вміст займає домінуючу частину екрану, а інструменти взаємодії згруповані за функціональними категоріями для полегшення навігації та взаємодії [11, с.283].

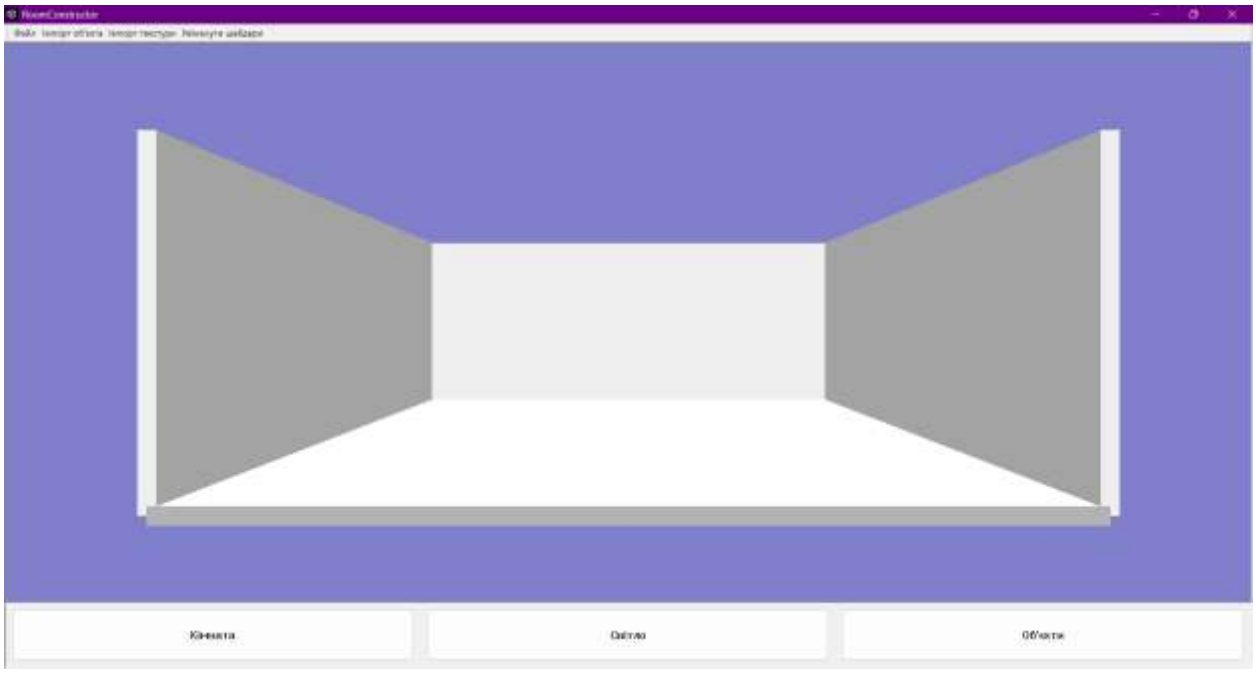

Рис. 4.1. Головне вікно програми

При активації вкладки «Кімната» користувачеві відкривається набір інструментів для управління базовими елементами конструйованого приміщення (рис. 4.2). Інтерфейс цієї вкладки розділений на п'ять основних блоків: «Стіни», «Вікна», «Двері», «Текстури» та навігаційну кнопку «Назад». Кожен з цих елементів представлений у вигляді окремої кнопки в нижній частині головного вікна програми.

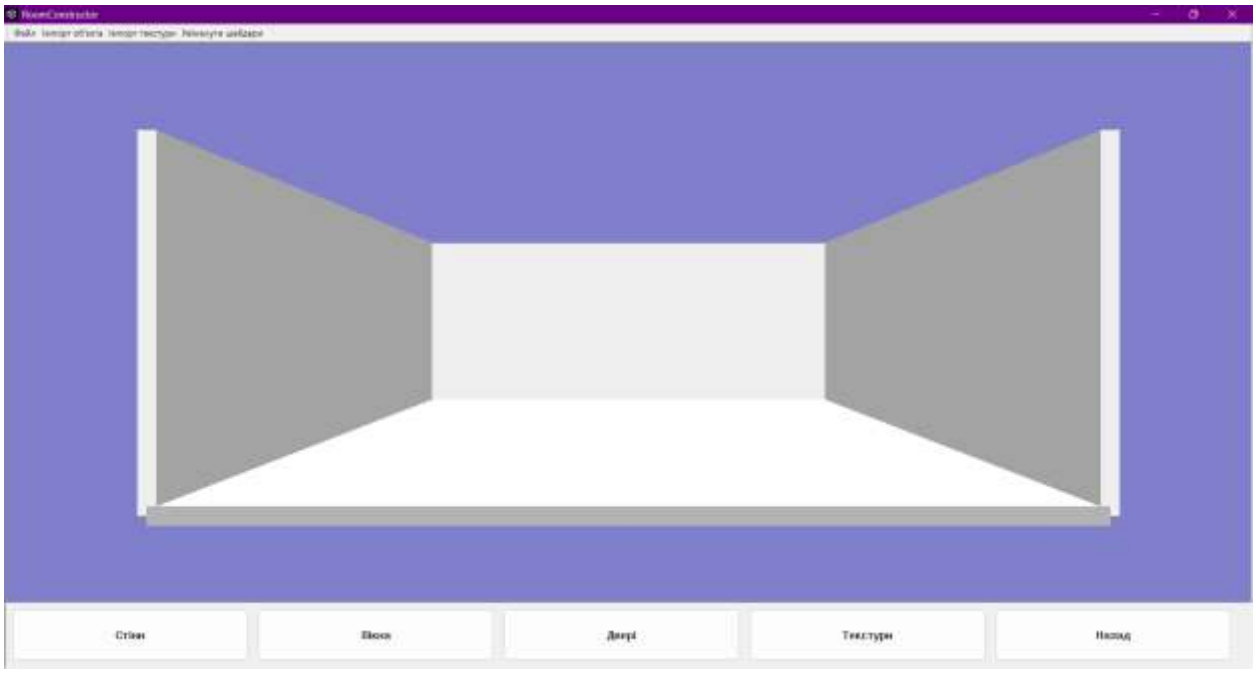

Рис. 4.2. Вкладка «Кімната» у головному вікні програми

При натисканні кнопки «Стіни» з розділу «Кімната» користувачеві відкривається підменю з функціональними елементами для маніпуляції стінами віртуального приміщення (рис. 4.3). Ця підсекція інтерфейсу містить п'ять основних кнопок: «Додати», «Видалити», «Ширина/Довжина», «Висота» та «Назад».

Ієрархічна організація елементів керування в інтерфейсах програм 3D-моделювання має вирішальне значення для зменшення когнітивного навантаження користувача, особливо коли функціональність програми включає широкий спектр операцій з просторовими об'єктами. Логічне групування команд за категоріями та забезпечення чіткої навігаційної структури дозволяють користувачам швидше опанувати складне програмне забезпечення [12, с.78].

Функціональні можливості цього підменю забезпечують всебічний контроль над геометрією стін:

- Кнопка «Додати» дозволяє створювати нові стіни в приміщенні, розширюючи його конфігурацію;
- «Видалити» призначена для видалення вибраних стін з моделі;
- «Ширина/Довжина» відкриває вкладку з кнопками для налаштування горизонтальних розмірів стін;
- «Висота» дає можливість регулювати вертикальний параметр стін кімнати;
- Кнопка «Назад» повертає користувача до основного меню вкладки «Кімната».

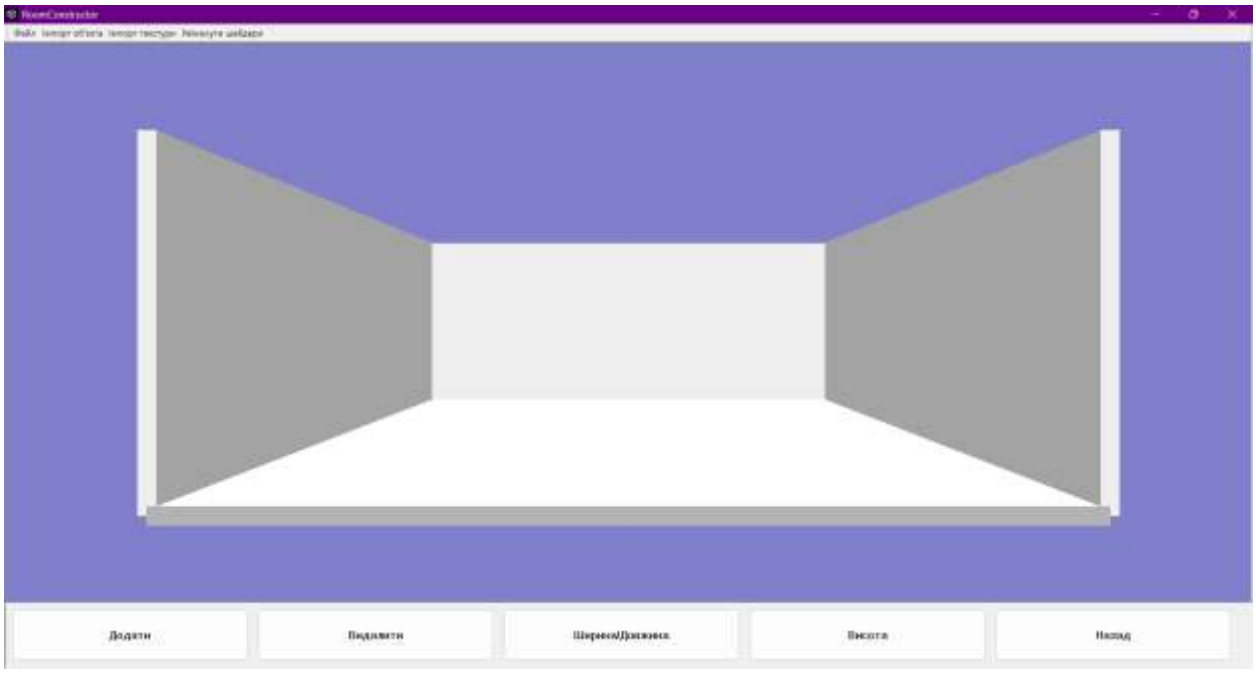

Рис. 4.3. Вкладка «Стіни» у головному вікні програми

При переході до вкладки «Вікна» через головне меню «Кімната», користувачеві відкривається інтерфейс управління віконними отворами в моделі приміщення (рис. 4.4). Панель інструментів містить чотири функціональні кнопки: «Додати», «Видалити», «Параметри» та «Назад».

Функціональність вкладки «Вікна» забезпечує повний контроль над віконними елементами приміщення:

- Кнопка «Додати» активує режим вставки нового вікна, дозволяючи користувачеві вказати його положення на одній зі стін кімнати;
- «Видалити» забезпечує функцію видалення вибраного віконного елемента з моделі;
- «Параметри» відкриває вкладку кнопками для налаштування розміру і позиції вікон по вісі У;
- Кнопка «Назад» повертає користувача до основного меню вкладки «Кімната».

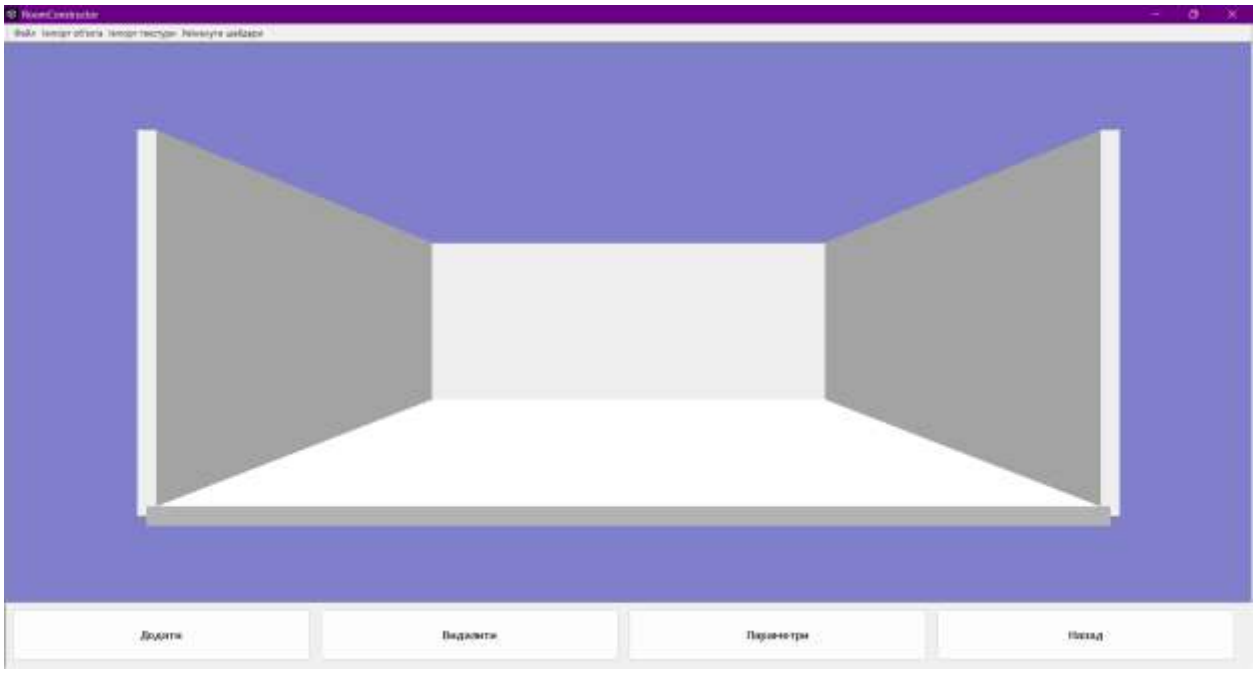

Рис. 4.4. Вкладка «Вікна» у головному вікні програми

Вкладка «Двері» надає інтерфейс для управління дверними прорізами в моделі приміщення (рис. 4.5), повторюючи структуру та функціональність вкладки «Вікна» для забезпечення інтуїтивної одноманітності інтерфейсу користувача. Нижня панель містить чотири функціональні кнопки: «Додати», «Видалити», «Параметри» та «Назад».

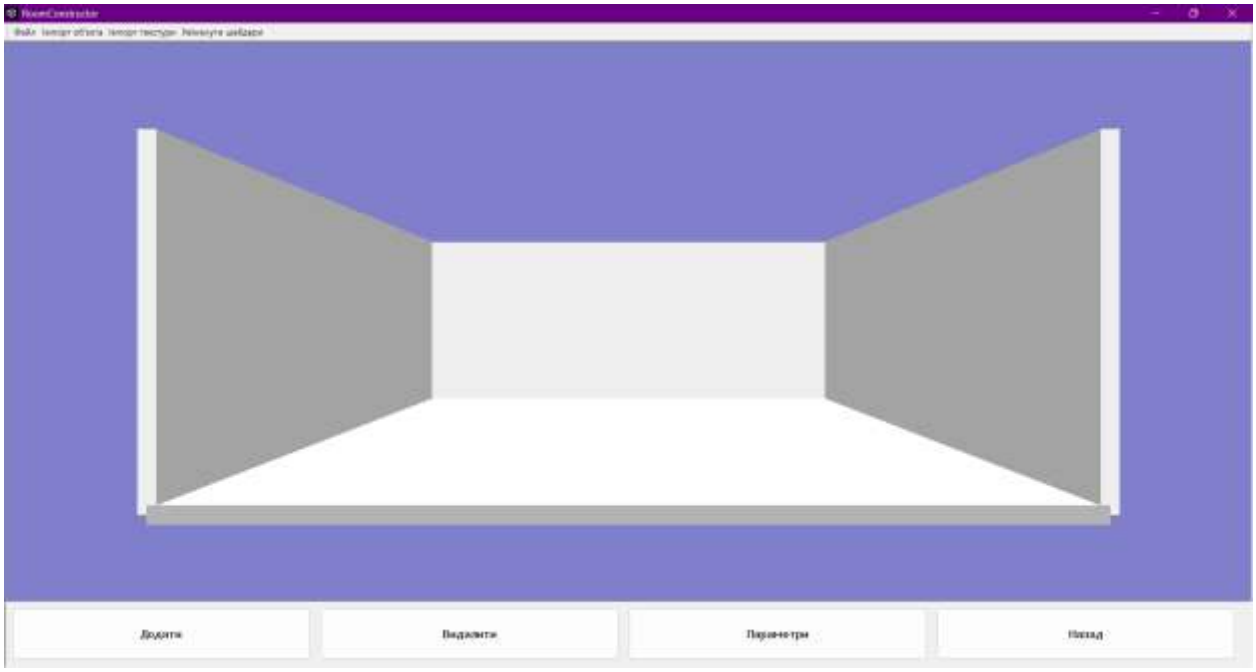

Рис. 4.5. Вкладка «Двері» у головному вікні програми

Вкладка «Текстури» є останньою складовою розділу «Кімната» та забезпечує функціональність для управління візуальним оформленням поверхонь у віртуальному приміщенні (рис. 4.6). Інтерфейс цієї вкладки містить три основні кнопки: «Додати», «Видалити» та «Назад».

Інтеграція системи управління текстурами в програмному забезпеченні для 3D-моделювання є критичним аспектом, що впливає на реалістичність кінцевої візуалізації та задоволення користувачів. Ефективний інтерфейс для текстурування має забезпечувати не лише простий доступ до бібліотеки матеріалів, але й інтуїтивно зрозумілий процес їх застосування до конкретних елементів моделі [13, с. 347].

Натискаючи кнопку «Додати», користувач потрапляє на вкладку з кнопками для вибору поверхні, для якої він хоче застосувати текстуру (рис. 4.7).

Обравши поверхню, буде представлено вкладку з кнопками для вибору текстури. Ця вкладка включає вбудовані та імпортовані користувачем текстури (рис. 4.8).

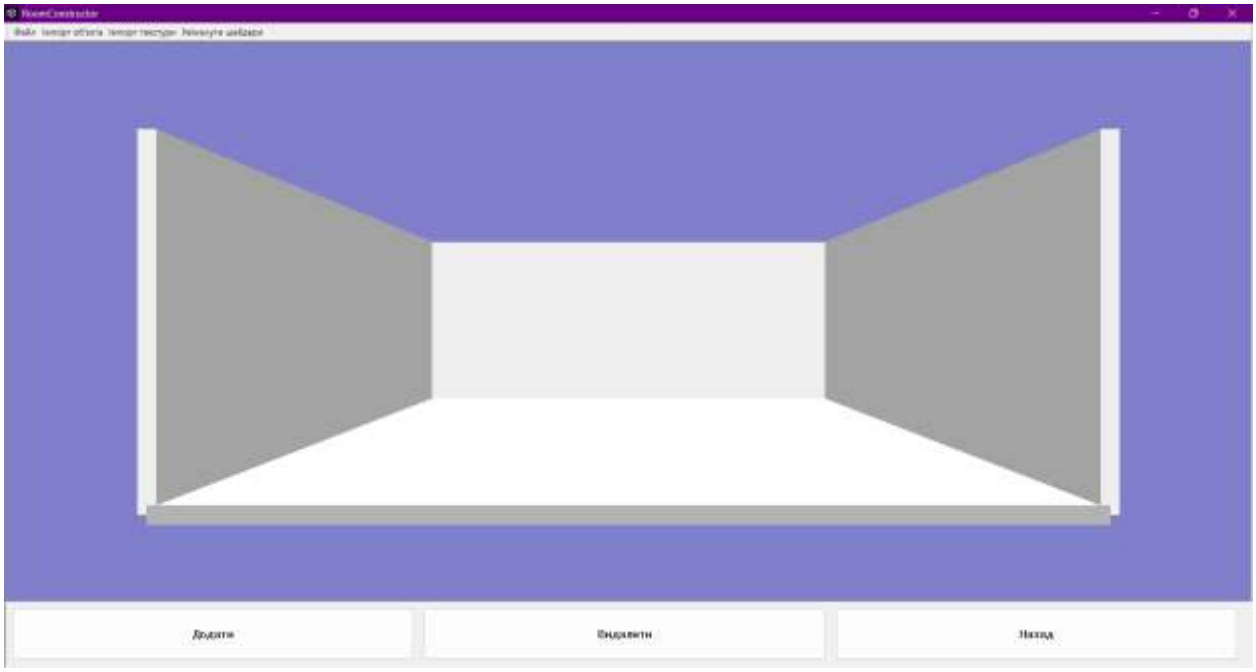

Рис. 4.6. Вкладка «Текстури» у головному вікні програми

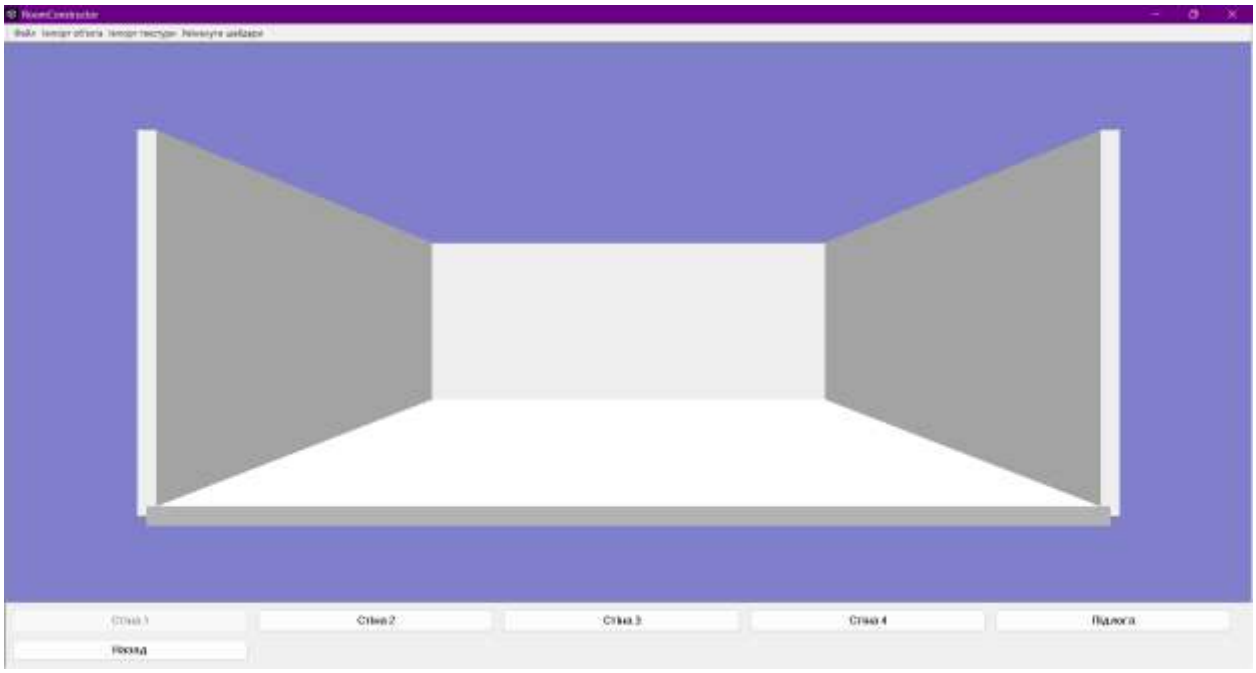

Рис. 4.7. Вкладка вибору поверхні для застосування текстури

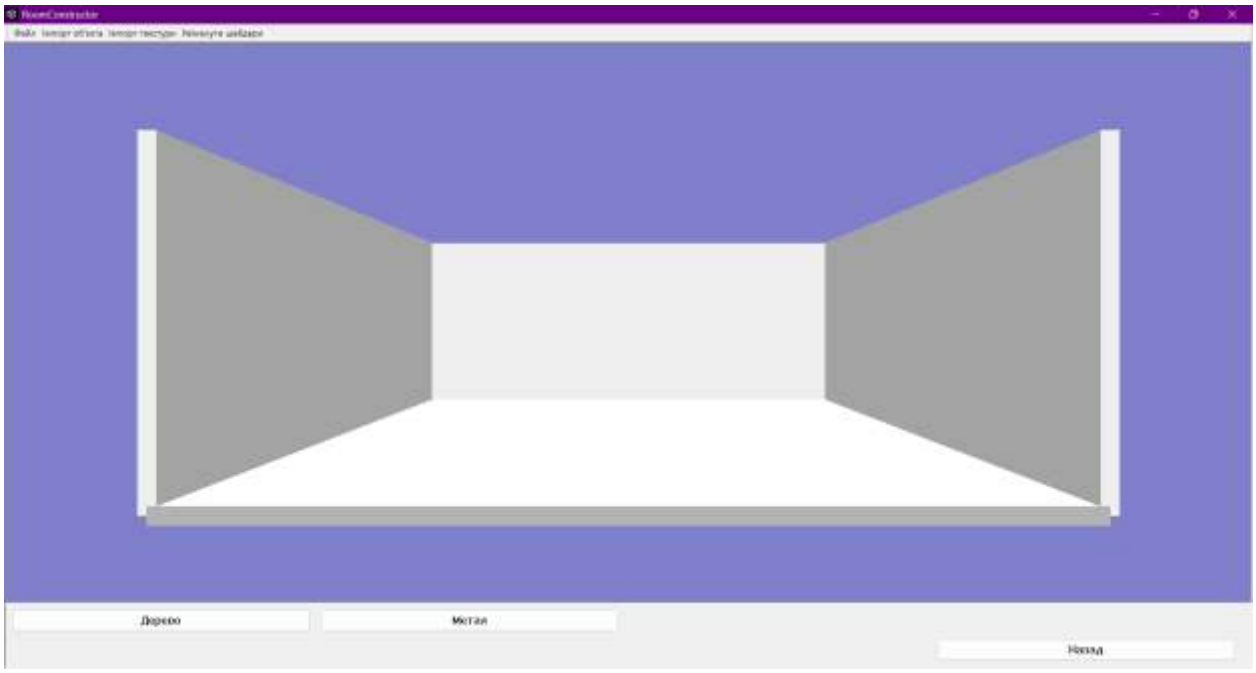

Рис. 4.8. Вкладка вибору текстури

Вкладка «Світло» з початкового меню програми дозволяє користувачу налаштовувати параметри освітлення сцени (рис. 4.9).

Управління освітленням є одним із найважливіших аспектів 3D-візуалізації, оскільки саме світло створює атмосферу, підкреслює деталі та формує сприйняття простору. Візуальне представлення джерел світла у вигляді інтерактивних елементів значно покращує розуміння користувачем

просторових параметрів освітлення та спрощує процес створення реалістичних сцен [14, с.129].

Джерело світла наочно візуалізується у вигляді червоної лінії з прицілом у верхній частині, що дозволяє користувачу легко ідентифікувати положення та напрямок світла у просторі. Це полегшує процес налаштування освітлення для досягнення бажаного візуального ефекту.

У нижній частині інтерфейсу розташовані елементи керування для налаштування параметрів світла, включаючи поля для зміни:

- Позиції X (два поля)
- Позиції Y (два поля)
- Позиції Z (два поля)
- Інтенсивності світла (два поля)

Кнопка «Назад» дозволяє повернутися до попереднього стану програми або вкладки.

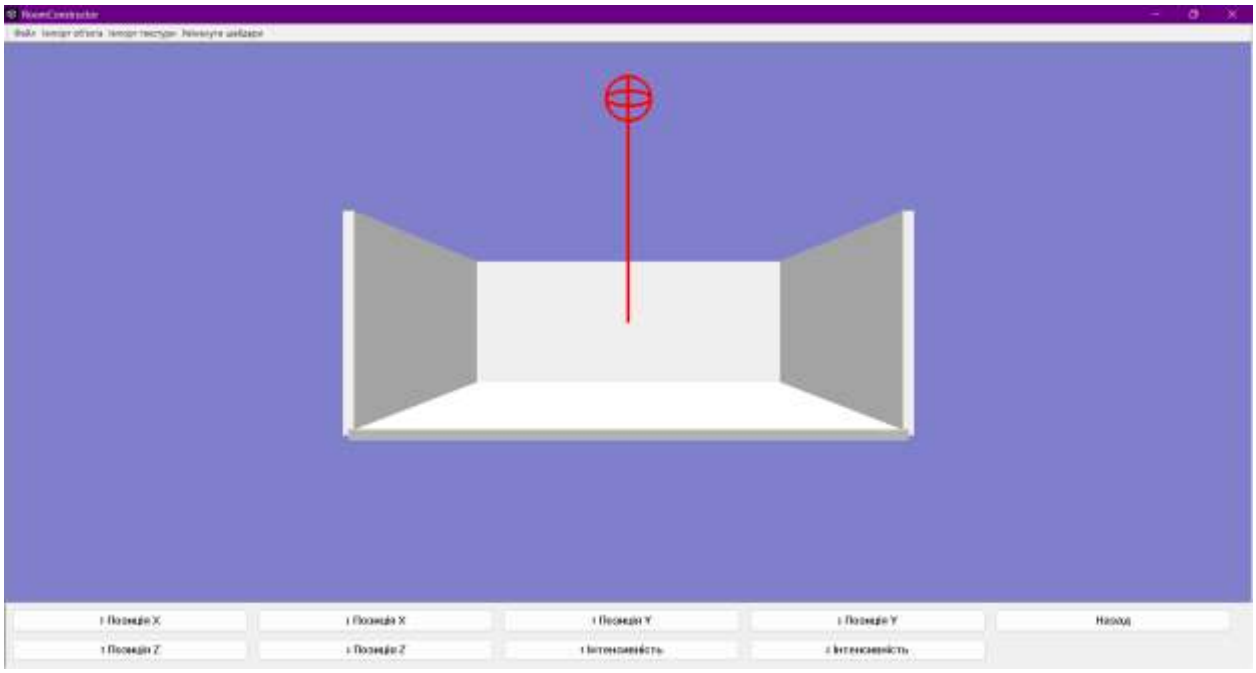

Рис. 4.9. Вкладка «Світло» у головному вікні програми

Відкривши вкладку «Об’єкти», користувач має змогу побачити меню взаємодії з об’єктами на сцені (рис. 4.10). Панель інструментів містить чотири функціональні кнопки:

- Кнопка «Додати» відкриває меню для вибору доступних об’єктів: вбудованих і імпортованих (рис. 4.11);
- «Видалити» забезпечує функцію видалення обраного об’єкта;
- «Параметри» відкриває вкладку кнопками для налаштування розміру і позиції об’єкта на сцені (рис. 4.12);
- Кнопка «Назад» повертає користувача до основного меню.

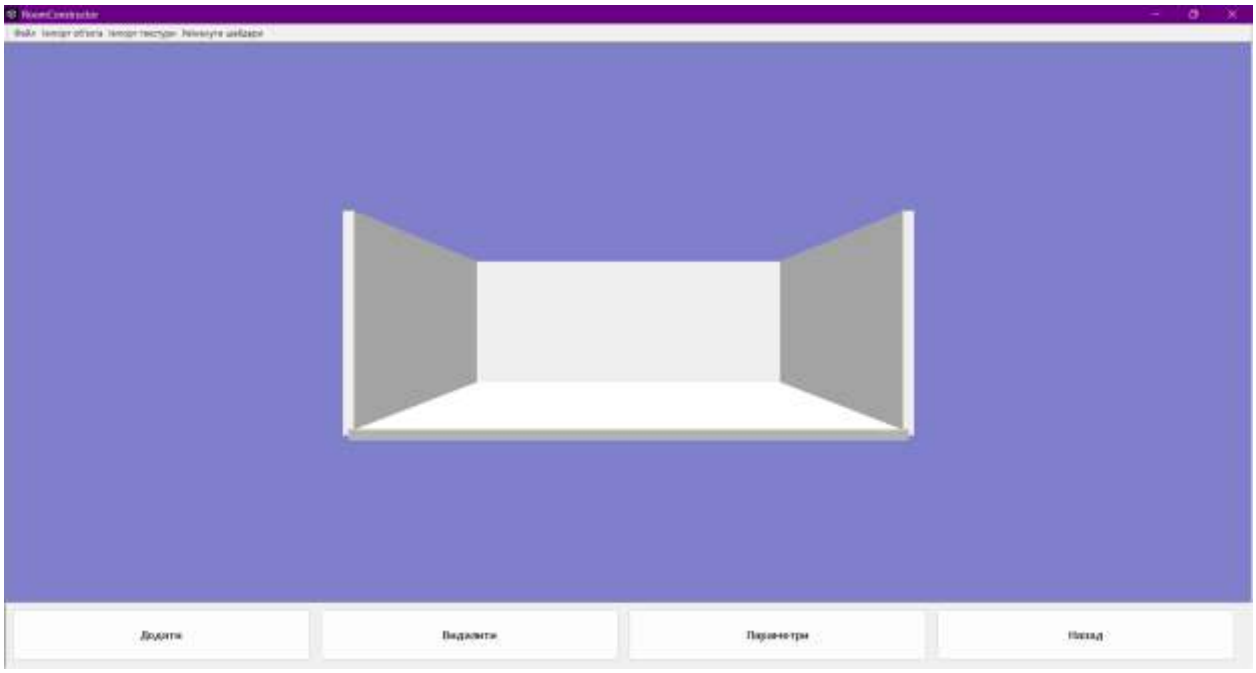

Рис. 4.10. Вкладка «Об’єкти» у головному вікні програми

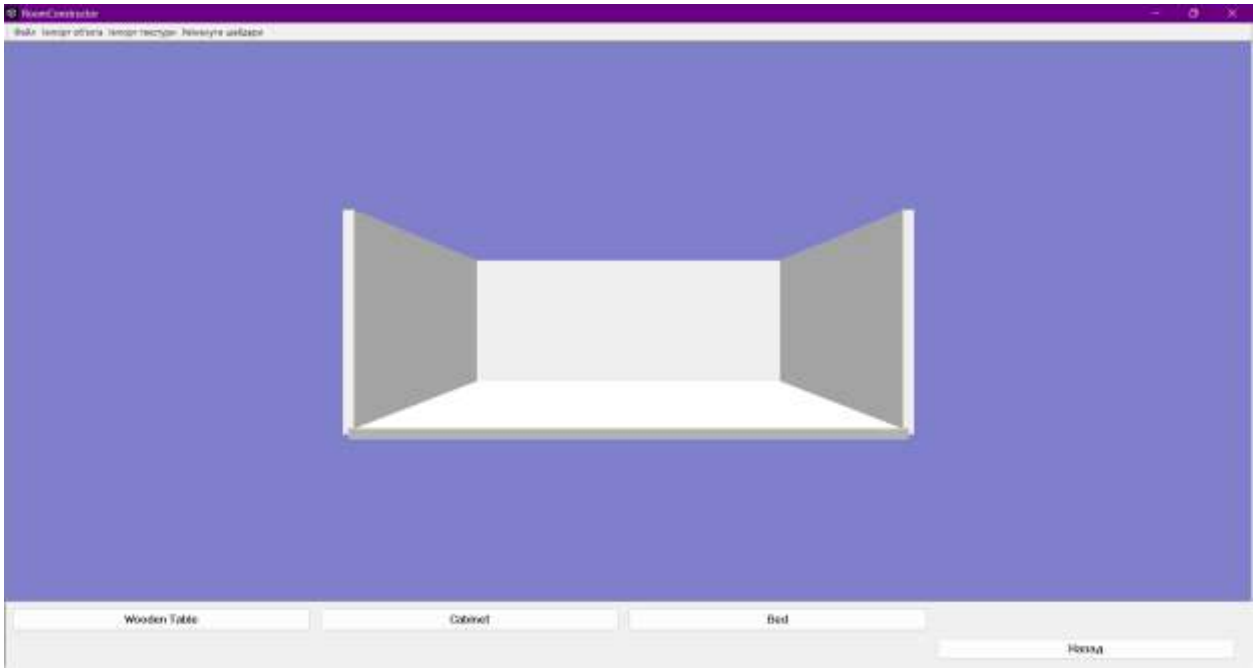

Рис. 4.11. Вкладка вибору об’єкта

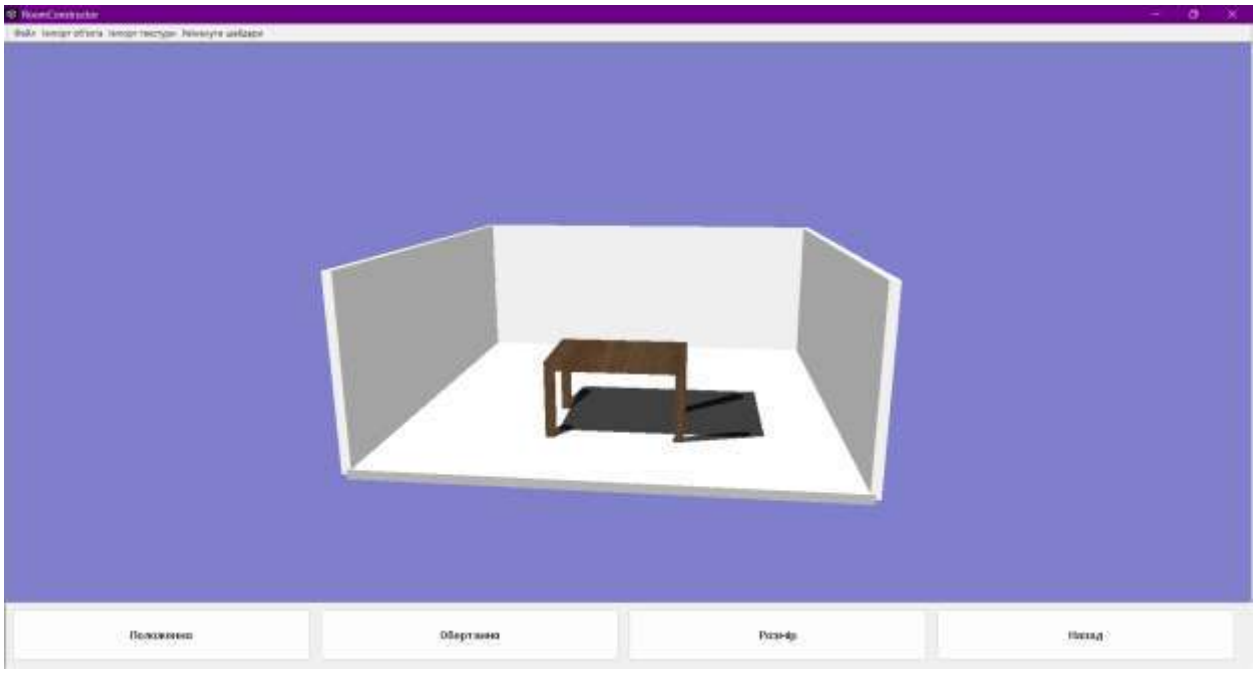

Рис. 4.12. Вкладка редагування об’єкта

Використавши всі вищевказані можливості, користувач має змогу повністю облаштувати кімнату за своїм смаком (рис. 4.13). Увімкнувши шейдери, представлення 3D-моделі вигладяє наближено до реальності (рис. 4.14). Лістинг коду файлу заголовків класу Room, який відмальовує кімнату наведено у додатку Б.

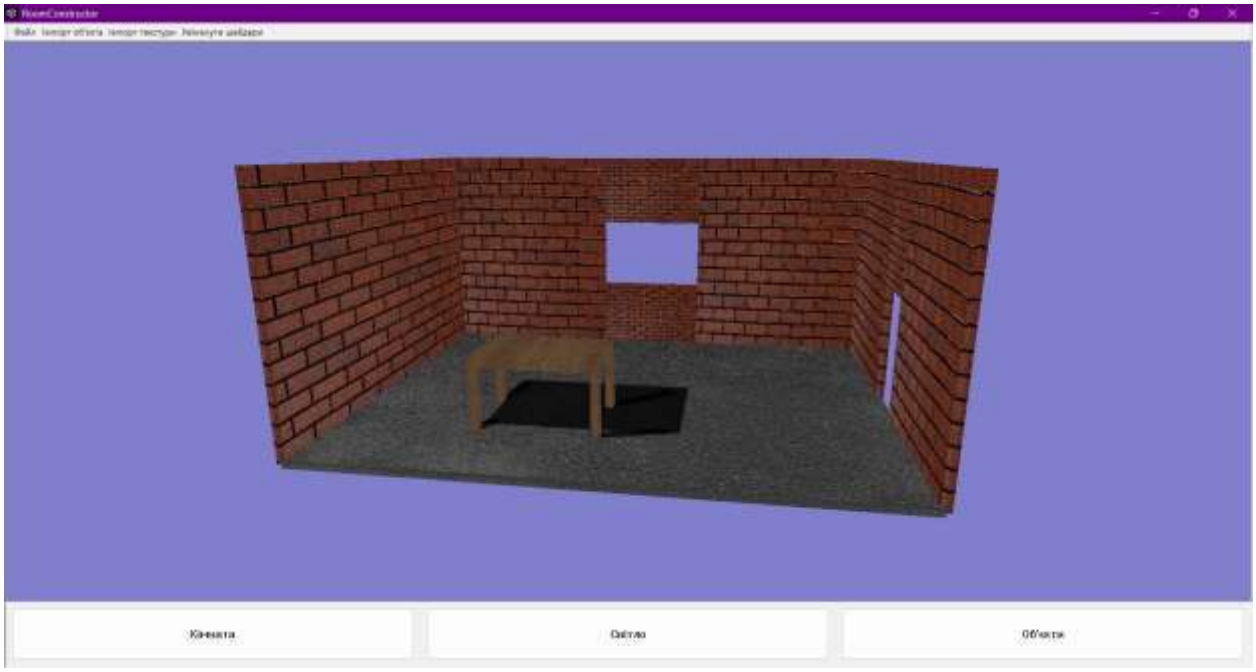

Рис. 4.13. Приклад готової кімнати

Рис. 4.14. Приклад готової кімнати з увімкненими шейдерами

На панелі інструментів знаходиться решта елементів керування (рис. 4.15): «Файл», «Імпорт об’єкта», «Імпорт текстури», «Увімкнути/вимкнути шейдери».

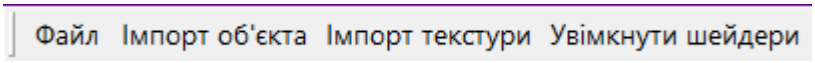

Рис. 4.15. Панель інструментів

Пункт «Файл» містить в собі два елемента: «Відкрити» та «Зберегти». Ці елементи створюють діалогове вікно, завдяки яким користувач може відкрити або зберегти кімнату в XML-файл.

Пункт «Імпорт об’єкта» створює діалогове вікно для вибору OBJ-файлу з подальшим імпортом об’єкта у програму. Алгоритм сканує директорію обраного файлу та шукає пов’язані з об’єктом файли: MTL і JPG/PNG.

Пункт «Імпорт текстури» створює діалогове вікно для вибору JPG або PNG файлів й імпортує їх у програму у якості текстур.

По натисненню на пункт «Увімкнути шейдери» програма застосовує шейдери до сцени та змінює стан кнопки на «Вимкнути шейдери». Повторне натиснення вимикає їх і повертає попередній стан кнопки.

4.2 Опис головного модулю програми (головного вікна) на мові JS

Головна сторінка веб-додатку (рис. 4.16) представляє собою інтуїтивно зрозумілий інтерфейс для 3D-моделювання приміщень. Верхня частина сторінки містить навігаційну панель з логотипом програми та основними розділами: «Вхід», «Конструктор» та «Профіль».

Центральна частина сторінки розділена на три інформаційні блоки, кожен з яких демонструє ключові функціональні можливості програми.

У нижній частині сторінки розміщена навігаційна панель з посиланнями на розділи: «Головна сторінка», «Конструктор», «Профіль», «Вхід», «Десктоп застосунок» та «FAQ», а також іконки соціальних мереж для зручності поширення інформації про додаток.

Загальний дизайн інтерфейсу виконаний у світлих тонах з акцентами на елементах керування, що створює комфортне середовище для творчої роботи з дизайном інтер'єру.

Ефективний користувацький інтерфейс веб-додатків для 3D-моделювання повинен бути одночасно інтуїтивним та інформативним, дозволяючи користувачам швидко зрозуміти функціональні можливості системи без тривалого навчання. Розміщення елементів керування та навігації має відповідати логічним очікуванням користувачів і дотримуватись принципів web-usability, які передбачають легку доступність основних функцій з будь-якої точки інтерфейсу [15, с.143].

Рис. 4.16. Головна сторінка веб-додатку

Сторінка входу (рис. 4.17) представляє собою мінімалістичний та функціональний інтерфейс авторизації користувача.

Форми авторизації є критичним елементом користувацького досвіду, і дослідження показують, що кожне додаткове поле в таких формах збільшує показник відмови від завершення процесу на 10%. Мінімалістичний підхід до дизайну форм входу не лише покращує візуальну естетику інтерфейсу, але й значно підвищує коефіцієнт конверсії, дозволяючи користувачам зосередитись на основній меті — швидкому доступі до функціоналу системи [16, с.89].

Центральна частина сторінки містить форму авторизації з двома основними полями введення: «Пошта» та «Пароль». Під полями введення розташована кнопка «Увійти», яка запускає процес автентифікації

користувача. Нижче розташоване посилання на сторінку реєстрації, якщо користувач ще не має облікового запису.

Рис. 4.17. Сторінка входу

Сторінка профілю користувача (рис. 4.18) представляє собою персональний кабінет із збереженими проєктами та особистими даними користувача.

Інтерфейс персонального кабінету користувача повинен забезпечувати баланс між інформаційною насиченістю та візуальною чистотою, що особливо важливо для систем 3D-моделювання, де користувачі повертаються до своїх проєктів через певний час. Організація збережених робіт у вигляді візуальних карток із мініатюрами проєктів значно покращує швидкість розпізнавання потрібних елементів і зменшує когнітивне навантаження на користувача під час пошуку та відновлення роботи над раніше створеними моделями [17, с.84].

У верхній секції сторінки відображається електронна адреса поточного користувача та посилання «Вийти», що дозволяє вийти з профілю.

Основний контент сторінки організований під заголовком «Збережені кімнати», де представлений список проєктів користувача у вигляді карток. Кожна картка містить:

1. Назву кімнати
2. Дату збереження або останнього редагування

3. Інтерактивні елементи управління:
   - Посилання на редагування кімнати (олівець)
   - Кнопка видалення запису про кімнату (кошик)

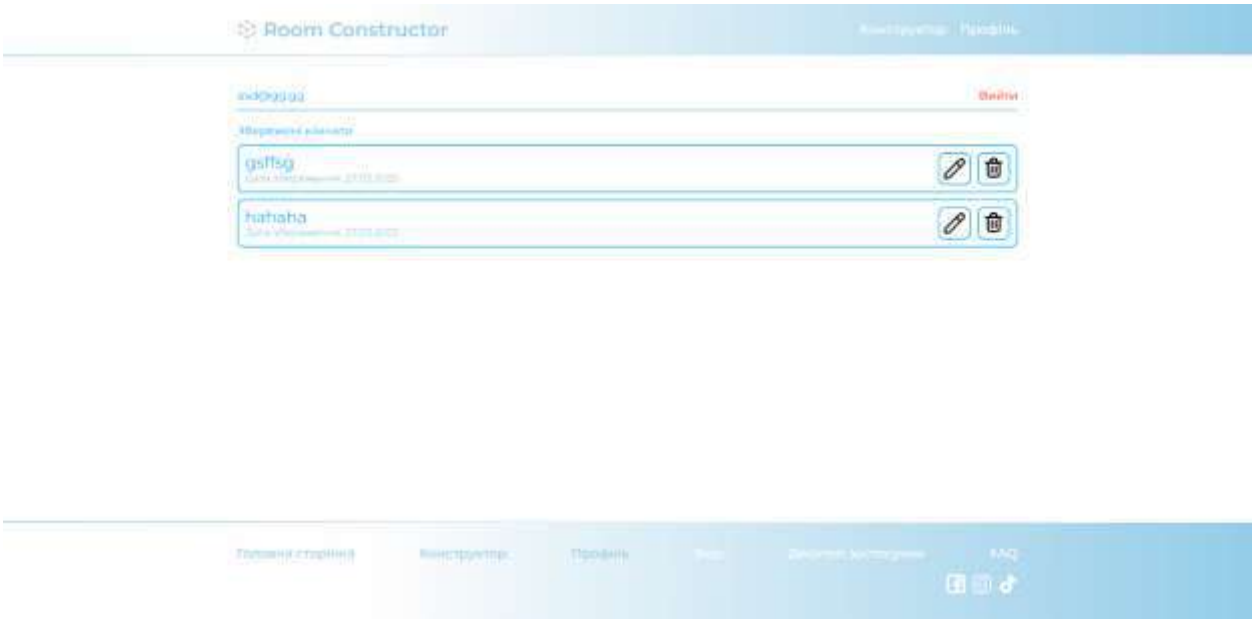

Рис. 4.18. Сторінка профілю

Сторінка конструктора (рис. 4.19) представляє собою ключовий робочий простір веб-додатку, де відбувається безпосереднє моделювання та редагування 3D-кімнати. Лістинг коду конструктора класу кімнати наведено у додатку В.

Центральна частина сторінки представлена основним робочим полем із 3D-візуалізацією віртуальної кімнати. На початковому етапі користувач бачить пусту кімнату з базовими стінами та підлогою в нейтральних темно-сірих відтінках на чорному фоні. Такий контрастний фон підкреслює геометрію 3D-моделі та допомагає користувачу краще сприймати форму об'єкта в просторі. Кімната відображається у вигляді тривимірного пустого простору з чітко окресленими стінами та підлогою, створюючи базу для подальшого дизайну.

У 3D-редакторах контраст між робочим простором моделі та оточуючим інтерфейсом має критичне значення для ефективності сприйняття просторових співвідношень. Дослідження в галузі когнітивної ергономіки показують, що темні нейтральні фони суттєво підвищують точність оцінки

глибини та об'єму при роботі з тривимірними моделями, особливо для новачків, які ще не розвинули навички просторової орієнтації в цифровому середовищі моделювання [18, с.172].

Особливу увагу варто звернути на нижню панель інструментів, яка є ключовим елементом інтерфейсу користувача для взаємодії з 3D-моделлю. Панель майже не відрізнається від тієї, що була попередньо розглянута у десктоп-додатку. Вона має чистий білий фон та містить чотири основні функціональні кнопки з інтуїтивно зрозумілими іконками:

1. «Кімната» (іконка з кубом) — для редагування основних параметрів кімнати, таких як стіни, вікна, двері та текстури (рис. 4.20).
2. «Світло» (іконка з лампочкою) — для налаштування освітлення, що дозволяє змінювати яскравість і положення джерела світла (рис. 4.21).
3. «Об'єкти» (іконка з кріслом) — для додавання, розміщення та налаштування меблів та декоративних елементів у віртуальній кімнаті (рис. 4.22).
4. «Зберегти» (іконка з дискетою) — для збереження кімнати в базі даних з подальшим відображенням у профілі користувача (рис. 4.23).

Ця панель інструментів має мінімалістичний дизайн, що дозволяє користувачу інтуїтивно орієнтуватися у функціоналі програми та швидко отримувати доступ до необхідних інструментів моделювання. Кожна іконка доповнена текстовою підказкою для кращого розуміння призначення інструменту.

Загальний дизайн інтерфейсу створює ефект занурення в процес 3D-моделювання, концентруючи увагу користувача на віртуальному просторі та пропонуючи зручний доступ до основних інструментів редагування.

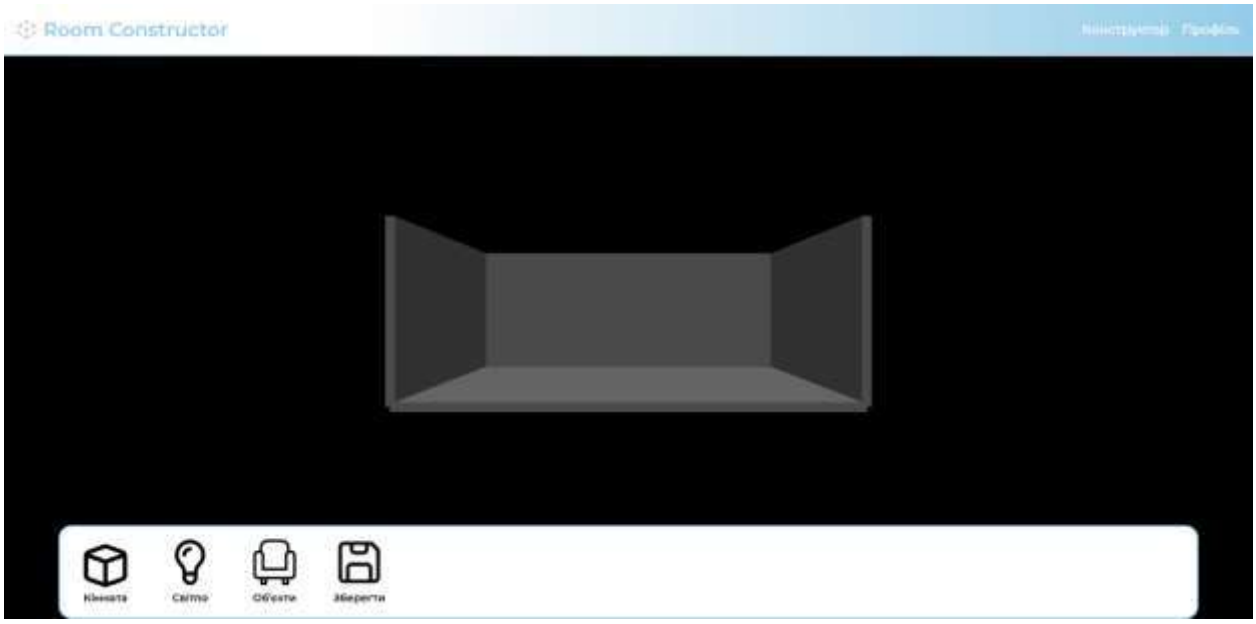

Рис. 4.19. Сторінка конструктора

Рис. 4.20. Вкладка «Кімната»

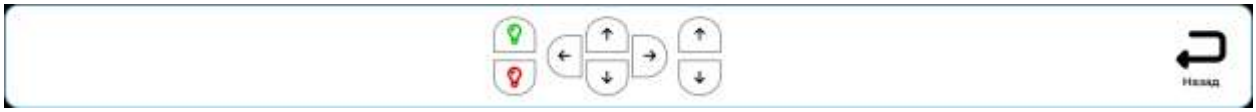

Рис. 4.21. Вкладка «Світло»

Рис. 4.22. Вкладка «Об’єкти»

Рис. 4.23. Вкладка «Зберегти»

На відміну від десктоп-додатка веб-додаток не має можливості приймати текстури й об’єкти з файлів. Користувачеві дозволено використовувати лише вбудовані файли текстур (рис. 4.24) й об’єктів (рис. 4.25).

Рис. 4.24. Вкладка вибору текстури

Рис. 4.25. Вкладка вибору об’єкта

Отже, веб-додаток включає три основні сторінки: мінімалістичну сторінку входу з формою авторизації, персональний кабінет користувача зі збереженими проєктами у вигляді візуальних карток, та робочу сторінку конструктора з 3D-візуалізацією на темному фоні. Інтерфейс конструктора містить нижню панель з чотирма основними інструментами для редагування параметрів кімнати, налаштування освітлення, додавання об'єктів та збереження проєктів.

Висновки до розділу 4

Отже, у цьому розділі особлива увага була приділена інтуїтивності інтерфейсу, логічній організації інструментів та зручності для користувачів різного рівня підготовки.

Десктопна версія програми забезпечує об'єктно-орієнтований підхід. Інтерфейс програми представлено у вигляді інтерактивного середовища з центральною областю для відображення 3D-сцени та функціональними панелями інструментів. Детально описано всі ключові складові інтерфейсу, кожен з яких містить відповідні елементи керування для редагування приміщення. Окремо розглянуто головне меню програми з функціями роботи з файлами, імпорту об'єктів і текстур та керування шейдерами, що дозволяє досягти реалістичного відображення 3D-моделі. Також було детально описано кожну сторінку веб-додатку.

Програмна реалізація повністю відповідає функціональним вимогам, визначеним на етапі постановки задачі, та забезпечує користувачам зручний інструментарій для створення, редагування та візуалізації 3D-моделей інтер'єрів приміщень з можливістю детального налаштування елементів дизайну та освітлення.

ВИСНОВКИ

У процесі виконання дипломної роботи було розроблено комплексне програмне рішення, що складається з двох взаємодоповнюючих компонентів: веб-додатку та десктопного застосунку. Основною метою проєкту було створення доступних та функціональних інструментів для тривимірного моделювання інтер'єрів приміщень, які поєднували б у собі інтуїтивно зрозумілий інтерфейс та потужні можливості для творчого проєктування.

Ключовою особливістю розробленої системи є взаємодоповнюваність обох компонентів. Веб-додаток забезпечує доступність та мобільність, дозволяючи користувачам працювати з проєктами з будь-якого пристрою, що має доступ до інтернету. Десктопний застосунок, у свою чергу, надає розширені можливості для професійного моделювання з використанням потужності локального обладнання.

Технічна реалізація проєкту включала:

- Розробку веб-додатку з використанням сучасних веб-технологій;
- Створення серверної інфраструктури для збереження проєктів користувачів;
- Програмування десктопного застосунку на C++ з використанням бібліотеки MFC;
- Інтеграцію систем рендерингу 3D-графіки в обох компонентах;
- Розробку механізмів збереження/завантаження даних.

Тестування розробленого програмного забезпечення продемонструвало його стабільність та відповідність поставленим вимогам. Обидва компоненти системи показали високу продуктивність при роботі з моделями приміщень різної складності.

Практична цінність розробленого програмного забезпечення полягає у можливості його використання як професійними дизайнерами інтер'єрів, так і звичайними користувачами. Веб-додаток забезпечує швидкий доступ до

основних функцій моделювання, а десктопний застосунок надає розширені можливості для детальної проробки дизайнерських рішень.

Перспективи подальшого розвитку проєкту включають:

- Розширення бібліотеки доступних об'єктів інтер'єру та текстур в обох компонентах;
- Вдосконалення механізмів синхронізації даних між веб та десктопною версіями;
- Розробку додаткових шейдерів та ефектів освітлення для підвищення фотореалістичності в десктопному застосунку;
- Оптимізацію алгоритмів рендерингу для роботи з особливо складними моделями приміщень;
- Впровадження інструментів колективної роботи над проєктами.

Таким чином, розроблена система успішно вирішує поставлені завдання та створює потужну платформу для тривимірного моделювання інтер'єрів, яка відповідає потребам різних категорій користувачів і має значний потенціал для подальшого розвитку та вдосконалення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sommerville I. Software Engineering / Sommerville I. - Pearson Education Limited, 2016. – 816 с.

2. Akenine-Möller T., Haines E., Hoffman N. Real-Time Rendering / Akenine-Möller T., Haines E., Hoffman N. - A K Peters/CRC Press, 2018. – 1198 с.

3. Dunn N. Digital Fabrication in Architecture / Dunn N. - Laurence King Publishing, 2012. – 192 с.

4. Eastman C., Teicholz P., Sacks R., Liston K. BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors / Eastman C., Teicholz P., Sacks R., Liston K. - Wiley, 2018. – 688 с.

5. Pharr M., Jakob W., Humphreys G. Physically Based Rendering: From Theory to Implementation / Pharr M., Jakob W., Humphreys G. - Morgan Kaufmann, 2016. – 1266 с.

6. MySQL 8.4 Reference Manual [Електронний ресурс]. – Режим доступу: https://dev.mysql.com/doc/refman/8.4/en/innodb-introduction.html

7. What is an XML file? [Електронний ресурс]. – Режим доступу: https://chimpkey.com/2023/02/28/what-is-an-xml-file-how-it-can-help-you-structure-data-for-storage-and-transport

8. Martin Kleppmann. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems / Martin Kleppmann. - O'Reilly Media, 2017. – 614 с.

9. Elliotte Rusty Harold, W. Scott Means. XML in a Nutshell / Elliotte Rusty Harold, W. Scott Means. - O'Reilly Media, 2004. – 712 с.

10. Jeff Prosise. Programming Windows with MFC / Jeff Prosise. - Microsoft Press, 1999. – 1168 с.

11. Alan Cooper, Robert Reimann, David Cronin. About Face: The Essentials of Interaction Design / Alan Cooper, Robert Reimann, David Cronin. - Wiley, 2014. – 720 с.

12. Jakob Nielsen, Raluca Budiu. Mobile Usability / Jakob Nielsen, Raluca Budiu. - New Riders, 2012. – 216 с.

13. Jennifer Tidwell, Charles Brewer, Aynne Valencia. Designing Interfaces: Patterns for Effective Interaction Design / Jennifer Tidwell, Charles Brewer, Aynne Valencia. - O'Reilly Media, 2020. – 599с.

14. Jeremy Birn. Digital Lighting and Rendering / Jeremy Birn. - New Riders, 2013. – 464 с.

15. Jakob Nielsen. Designing Web Usability: The Practice of Simplicity / Jakob Nielsen. - New Riders, 2000. – 432 с.

16. Steve Krug. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability / Steve Krug. - New Riders, 2014. – 216 с.

17. Jesse James Garrett. The Elements of User Experience: User-Centered Design for the Web and Beyond / Jesse James Garrett. - New Riders, 2010. – 192 с.

18. Patric J. Lynch, Sarah Horton. Web Style Guide: Foundations of User Experience Design / Patric J. Lynch, Sarah Horton. - Yale University Press, 2016. – 387 с.

ДОДАТКИ

Додаток А

Лістинг вихідного коду на мові SQL для створення схеми бази даних у середовищі СУБД MySQL

```
CREATE DATABASE room
  CHARACTER SET utf8mb4
  COLLATE utf8mb4_general_ci;


USE room;


/* Tables */
CREATE TABLE saved_room (
  r_id             int UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY
KEY,
  `user`           int UNSIGNED NOT NULL,
  wall_width       int UNSIGNED NOT NULL,
  wall_length      int UNSIGNED NOT NULL,
  wall_height      int UNSIGNED NOT NULL,
  wall_thickness   int UNSIGNED NOT NULL,
  presented_walls  longtext NOT NULL,
  objects          longtext NOT NULL,
  lights           longtext NOT NULL,
  light_x          int UNSIGNED NOT NULL,
  light_y          int UNSIGNED NOT NULL,
  light_z          int UNSIGNED NOT NULL,
  light_intensity  int UNSIGNED NOT NULL,
  room_name        varchar(50) NOT NULL,
  `datetime`       datetime NOT NULL DEFAULT CURRENT_TIMESTAMP
ON UPDATE CURRENT_TIMESTAMP,
  environment      varchar(20) NOT NULL,
  PRIMARY KEY (r_id)
) ENGINE = InnoDB;
```

```
CREATE TABLE `user` (
  login       varchar(50) NOT NULL,
  `password`  varchar(100) NOT NULL,
  u_id        int UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
  PRIMARY KEY (u_id)
) ENGINE = InnoDB;

/* Foreign Keys */
ALTER TABLE saved_room
  ADD CONSTRAINT `user`
  FOREIGN KEY (`user`)
    REFERENCES `user`(u_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE;
```

Додаток Б

Лістинг вихідного коду Header-файлу класу кімнати на мові C++

```
#pragma once

#include <windows.h>
#include «Shader.h»
#include <GL/gl.h>
#include <GL/glu.h>
#include <map>
#include <vector>
#include <string>
#include <algorithm>
#include <assimp/Importer.hpp>
#include <assimp/scene.h>
#include <assimp/postprocess.h>
#include <iostream>
#include <fstream>
#include <windows.h>
#include <gdiplus.h>
#pragma comment(lib, «gdiplus.lib»)
#define GL_CLAMP_TO_EDGE 0x812F

struct WallProperties {
    bool presented = true;
    bool hasWindow = false;
    bool hasDoor = false;
    bool hasTexture = false;
    bool isEditing = false;
    float windowWidth = 200.0f;
    float windowHeight = 200.0f;
    float windowOffsetY = 0.0f;
```

```
    float doorHeight = 200.0f;
    float doorWidth = 100.0f;
    float doorOffsetX = 0.0f;
    std::string texturePath = ««;
};

template <class T> void SafeRelease(T*& p)
{
    if (p) {
        p->Release();
        p = NULL;
    }
}

struct ModelVertex {
    float x, y, z;
    float nx, ny, nz;
    float u, v;
};


struct ModelMesh {
    std::vector<ModelVertex> vertices;
    std::vector<unsigned int> indices;
    GLuint textureID;
    bool hasTexture;
    float materialColor[4];
    bool hasMaterialColor;
    GLuint normalMapID = 0;
};
```

Продовження додатку Б

```
struct SimpleModel {
    std::vector<ModelMesh> meshes;
    bool isLoaded;

    SimpleModel() :
        meshes({}),
        isLoaded(false) {}
};

struct FloorProperties {
    std::string texturePath = "";
};

struct RoomObject {
    std::string type;
    std::string name;
    float posX, posY, posZ;
    float rotX, rotY, rotZ;
    float scale;
    bool visible;
    SimpleModel* model;

    RoomObject() :
        type(""), name(""),
        posX(0.0f), posY(0.0f), posZ(0.0f),
        rotX(0.0f), rotY(0.0f), rotZ(0.0f),
        scale(1.0f), visible(true), model(nullptr) {}

    RoomObject(const std::string& t, const std::string& n,
        float x, float y, float z,
        float rx, float ry, float rz,
```

```
        float s, bool v = true) :
        type(t), name(n),
        posX(x), posY(y), posZ(z),
        rotX(rx), rotY(ry), rotZ(rz),
        scale(s), visible(v), model(nullptr) {}

    ~RoomObject() {

    }
};

class Room {
public:
    Room();
    ~Room();

    struct TextureInfo {
        std::string displayName;
        std::string path;
    };

    void Initialize();



    void Render();



    void SetDimensions(float width, float length, float height,
float thickness);
```

```
public:
    bool IsWallPresented(const std::string& wallName) const {
        auto it = m_walls.find(wallName);
        return (it != m_walls.end()) ? it->second.presented :
false;
    }

    bool HasWindow(const std::string& wallName) const {
        auto it = m_walls.find(wallName);
        return (it != m_walls.end()) ? it->second.hasWindow :
false;
    }

    bool HasDoor(const std::string& wallName) const {
        auto it = m_walls.find(wallName);
        return (it != m_walls.end()) ? it->second.hasDoor :
false;
    }

    bool HasTexture(const std::string& wallName) const {
        auto it = m_walls.find(wallName);
        return (it != m_walls.end()) ? !it-
>second.texturePath.empty() : false;
    }

    void SetWallEditing(const std::string& wallName, bool
editing) {
        auto it = m_walls.find(wallName);
        if (it != m_walls.end()) {
            it->second.isEditing = editing;
        }
    }
```

```
    void ResetWallsEditing() {
        for (auto& pair : m_walls) {
            pair.second.isEditing = false;
        }
    }
    void GetImportedTextures(std::vector<TextureInfo>& textures)
const;
    bool ImportTextureWithCopy(const std::string& originalPath,
const std::string& textureName);
    void ScanTexturesDirectory();
    std::vector<std::string> GetAvailableTextures() const;
    std::vector<TextureInfo> m_availableTextures;
    std::vector<TextureInfo> m_standardTextures;
    std::vector<TextureInfo> m_importedTextures;
    void ShowLightHelper();
    void HideHelpers();

    void AddWall(const std::string& wallName) {
        if (m_walls.find(wallName) != m_walls.end()) {
            m_walls[wallName].presented = true;
            CreateWalls();
        }
    }

    bool ImportModelWithCopy(const std::string& originalPath,
const std::string& modelName);
    std::string CleanFileName(const std::string& name);
    void CopyTexturesFromMtl(const std::string& mtlPath, const
std::string& sourceDir, const std::string& targetDir);

    float GetWallHeight() { return m_wallHeight; }
    bool CheckAndLoadMTL(const std::string& objFilePath);
```

Продовження додатку Б

```
    void RemoveWall(const std::string& wallName);
    void AddWindow(const std::string& wallName);
    void AddDoor(const std::string& wallName);
    void RemoveWindow(const std::string& wallName);
    void RemoveDoor(const std::string& wallName);
    void UpdateWallWidth(float newWidthDelta);
    void UpdateWallLength(float newLengthDelta);
    void UpdateWallHeight(float newHeightDelta);
    GLuint CreateDefaultTexture();
    void UpdateWindowWidth(const std::string& wallName, float
delta);
    void UpdateWindowHeight(const std::string& wallName, float
delta);
    void UpdateWindowOffsetY(const std::string& wallName, float
delta);


    void UpdateDoorWidth(const std::string& wallName, float
delta);
    void UpdateDoorHeight(const std::string& wallName, float
delta);
    void UpdateDoorOffsetX(const std::string& wallName, float
delta);
    GLuint LoadTexture(const char* path);
    void RenderWoodenTable();
    void UpdateLightX(float delta);
    void UpdateLightY(float delta);
    void UpdateLightZ(float delta);
    void UpdateLightIntensity(float delta);
    void CreateWalls();
    void DrawCubeForObject(float size);
    bool LoadSimpleModel(const std::string& path);
    void RenderSimpleModel();
    void SetModelPosition(float x, float y, float z);
```

Продовження додатку Б

```
    void SetModelScale(float scale);

    void SetModelRotation(float angle, float x, float y, float
z);

    bool LoadModelForObject(const std::string& path,
SimpleModel* model);

    void AdjustModelScaleAndPosition();

    void AdjustObjectModelScaleAndPosition(RoomObject& obj);


    void RenderObjectModel(const SimpleModel& model);

    void AddObject(const std::string& type, const std::string&
name,

    float posX, float posY, float posZ,

    float rotX, float rotY, float rotZ, float scale);

    void RemoveObject(int index);

    void UpdateObjectPosition(int index, float x, float y, float
z);

    void UpdateObjectRotation(int index, float rx, float ry,
float rz);

    void UpdateObjectScale(int index, float scale);

    void SetObjectVisibility(int index, bool visible);

    void RenderCabinet();

    void RenderBed();


    const std::vector<RoomObject>& GetObjects() const { return
m_objects; }

    size_t GetObjectCount() const { return m_objects.size(); }

    const RoomObject& GetObject(int index) const { return
m_objects[index]; }



    void RenderObjects();

    void RenderObjectByType(const RoomObject& obj);

    bool IsModelLoaded() const {
```

Продовження додатку Б

```
  return m_simpleModel.isLoaded;
}


void RotateModelX(float angle) {
    m_modelRotAngle = angle;
    m_modelRotX = 1.0f;
    m_modelRotY = 0.0f;
    m_modelRotZ = 0.0f;
}

void RotateModelY(float angle) {
    m_modelRotAngle = angle;
    m_modelRotX = 0.0f;
    m_modelRotY = 1.0f;
    m_modelRotZ = 0.0f;
}

void RotateModelZ(float angle) {
    m_modelRotAngle = angle;
    m_modelRotX = 0.0f;
    m_modelRotY = 0.0f;
    m_modelRotZ = 1.0f;
}

void MoveModel(float deltaX, float deltaY, float deltaZ) {
    m_modelPosX += deltaX;
    m_modelPosY += deltaY;
    m_modelPosZ += deltaZ;
}
```

Продовження додатку Б

```
     void ScaleModel(float deltaScale) {
     m_modelScale += deltaScale;
        if (m_modelScale < 0.1f) {
            m_modelScale = 0.1f;
        }
    }

    bool AddImportedModel(const std::string& modelPath, const
std::string& modelName)
    {
        std::string uniqueName = GenerateUniqueName(modelName);

        RoomObject newObject(«importedModel», uniqueName,
            0.0f,
            -GetWallHeight() / 2.0f,
            0.0f,
            0.0f, 0.0f, 0.0f,
            1.0f);

        newObject.model = new SimpleModel();

        if (LoadModelForObject(modelPath, newObject.model)) {
            m_importedModelPaths[uniqueName] = modelPath;

            AdjustObjectModelScaleAndPosition(newObject);

            m_objects.push_back(newObject);

            TRACE(«Added imported model: %s\n»,
uniqueName.c_str());
            return true;
```

```
     }
          else {
             delete newObject.model;
             TRACE(«Failed to load imported model\n»);
             return false;
         }
    }

    std::string GenerateUniqueName(const std::string& baseName)
    {
        std::string uniqueName = baseName;
        int counter = 1;

        bool nameExists = std::any_of(m_objects.begin(),
m_objects.end(),
            [&uniqueName](const RoomObject& obj) { return
obj.name == uniqueName; });

        while (nameExists) {
            uniqueName = baseName + « (« +
std::to_string(counter++) + «)»;
            nameExists = std::any_of(m_objects.begin(),
m_objects.end(),
                [&uniqueName](const RoomObject& obj) { return
obj.name == uniqueName; });
        }

        return uniqueName;
    }
    void ResetModelTransforms() {
        m_modelPosX = 0.0f;
        m_modelPosY = 0.0f;
```

Продовження додатку Б

```
        m_modelPosZ = 0.0f;
        m_modelScale = 1.0f;
        m_modelRotAngle = 0.0f;
        m_modelRotX = 0.0f;
        m_modelRotY = 1.0f;
        m_modelRotZ = 0.0f;
    }
    void SetWireframeMode(bool enable) {
        m_wireframeMode = enable;
    }

    float GetModelRotationAngle() const {
        return m_modelRotAngle;
    }
    GLuint LoadTextureFromFile(const char* path);

    void ApplyWallTexture(const std::string& wallName, const
std::string& texturePath);

    void RemoveWallTexture(const std::string& wallName);

    void ApplyFloorTexture(const std::string& texturePath);

    void RemoveFloorTexture();

    bool HasTexture(const std::string& wallName);
    bool HasFloorTexture() const { return m_floorTextureID != 0;
}
    void SetShadersEnabled(bool enabled);
    bool GetShadersEnabled() const;
```

Продовження додатку Б

```
     float GetWallWidth() { return m_wallWidth; }

     std::string Room::GetWallTexturePath(const std::string&
wallName) const {
        auto it = m_walls.find(wallName);
        if (it != m_walls.end()) {
            return it->second.texturePath;
        }
        return «»;
    }

    std::string Room::GetFloorTexturePath() const {
        return m_floor.texturePath;
    }

    std::string Room::GetModelPath(const std::string& modelName)
const {
        auto it = m_importedModelPaths.find(modelName);
        if (it != m_importedModelPaths.end()) {
            return it->second;
        }
        return «»;
    }

    float m_lightX;
    float m_lightY;
    float m_lightZ;
    float m_lightIntensity;
    float m_wallWidth;
    float m_wallLength;
    float m_wallHeight;
    float m_wallThickness;
```

```
    std::map<std::string, WallProperties> m_walls;
    FloorProperties m_floor;
    std::map<std::string, std::string> m_importedModelPaths;
    std::vector<RoomObject> m_objects;
private:


    bool m_showLightHelper = false;
    bool m_wireframeMode = false;
    GLfloat m_materialAmbient[4];
    GLfloat m_materialDiffuse[4];
    GLfloat m_materialSpecular[4];
    GLfloat m_materialShininess;
    SimpleModel m_simpleModel;
    float m_modelPosX = 0.0f, m_modelPosY = 0.0f, m_modelPosZ =
0.0f;
    float m_modelScale = 1.0f;
    float m_modelRotAngle = 0.0f;
    float m_modelRotX = 0.0f, m_modelRotY = 1.0f, m_modelRotZ =
0.0f;




    std::map<std::string, GLuint> m_wallTextureIDs;
    GLuint m_floorTextureID;
    Shader* m_pShader;
    bool m_bShadersEnabled;
    bool m_bShadersInitialized;

    bool InitializeShaders();
    void CleanupShaders();
```

Продовження додатку Б

```
    void SetupShaderUniforms();
    void CreateLeftWall();
    void CreateRightWall();
    void CreateBackWall();
    void CreateFrontWall();
    void CreateFloor();
    void SetupLighting();
    void shadowProjectMatrix(GLfloat* m, GLfloat* groundplane,
GLfloat* lightpos);
    void DrawCube(float size);
};
```

Додаток В

Лістинг вихідного коду конструктора класу кімнати на мові JavaScript

```
class Room {
    constructor(scene, {
        wallWidth = 1000,
        wallLength = 1000,
        wallHeight = 400,
        wallThickness = 20,
        material = new THREE.MeshStandardMaterial({ color:
0x999999, side: THREE.DoubleSide }),
        presentedWalls = {},
        obj = {},
        lights = [],
        lightX = 0,
        lightY = 500,
        lightZ = 200,
        lightIntensity = 1
    } = {}) {
        this.scene = scene;
        this.wallWidth = wallWidth;
        this.wallLength = wallLength;
        this.wallHeight = wallHeight;
        this.wallThickness = wallThickness;
        this.material = material;
        this.walls = [];
        this.presentedWalls = {
            front: {
                presented: false,
                hasWindow: false,
                hasDoor: false,
                isEditing: false,
                windowWidth: 200,
```

Продовження Додатку В

```
        windowHeight: 200,
        windowOffsetY: 0,
        doorHeight: 200,
        doorWidth: 100,
        doorOffsetX: 0,
        texturePath: '',
        ...presentedWalls.front
    },
    back: {
        presented: true,
        hasWindow: false,
        hasDoor: false,
        isEditing: false,
        windowWidth: 200,
        windowHeight: 200,
        windowOffsetY: 0,
        doorHeight: 200,
        doorWidth: 100,
        doorOffsetX: 0,
        texturePath: '',
        ...presentedWalls.back
    },
    right: {
        presented: true,
        hasWindow: false,
        hasDoor: false,
        isEditing: false,
        windowWidth: 200,
        windowHeight: 200,
        windowOffsetY: 0,
        doorHeight: 200,
```

Продовження Додатку В

```
                doorWidth: 100,
                doorOffsetX: 0,
                texturePath: '',
                ...presentedWalls.right
            },
            left: {
                presented: true,
                hasWindow: false,
                hasDoor: false,
                isEditing: false,
                windowWidth: 200,
                windowHeight: 200,
                windowOffsetY: 0,
                doorHeight: 200,
                doorWidth: 100,
                doorOffsetX: 0,
                texturePath: '',
                ...presentedWalls.left
            },
            floor: {
                texturePath: '',
                ...presentedWalls.floor
            }
        };
        this.objects = {
            tableAndChairs: {
                name: 'Стіл і стільці',
                pathMTL:
'../object/tableAndChairs/tableAndChairs.mtl',
                pathOBJ:
'../object/tableAndChairs/tableAndChairs.obj',
```

```
                scale: 2.5,
                positionX: 0,
                positionY: -190,
                positionZ: 0,
                angle: 0,
                loadedObject: null,
            },
            woodenTable: {
                name: 'Дерев\'яний стілець',
                pathMTL:
'../object/woodenTable/woodenTable.mtl',
                pathOBJ:
'../object/woodenTable/woodenTable.obj',
                pathTex:
'../object/woodenTable/woodenTable.jpg',
                scale: 1,
                positionX: 0,
                positionY: -150,
                positionZ: 0,
                angle: 0,
                loadedObject: null,
            },
            tvStand: {
                name: 'Тумба під телевізор',
                pathMTL: '../object/tvStand/table.mtl',
                pathOBJ: '../object/tvStand/table.obj',
                pathTex: '../object/tvStand/wood-018_larch-
european-2_d.png',
                scale: 10,
                positionX: 300,
                positionY: -190,
                positionZ: 0,
```

Продовження Додатку В

```
            angle: 135,
            loadedObject: null,
        },
        chair: {
            name: 'Крісло',
            pathMTL: '../object/chair/Office_chair.mtl',
            pathOBJ: '../object/chair/Office chair.obj',
            pathTex:
'../object/chair/textiles_1_20090323_1963410504.png',
            scale: 0.2,
            positionX: 0,
            positionY: -190,
            positionZ: 0,
            angle: 0,
            loadedObject: null,
        },
        drawer: {
            name: 'Тумба',
            pathMTL: '../object/drawer/Free model
Drawer(Final) .mtl',
            pathOBJ: '../object/drawer/Free model
Drawer(Final) .obj',
            pathTex: '../object/drawer/Drawer
Texture(Final).png',
            scale: 40,
            positionX: 0,
            positionY: -190,
            positionZ: 0,
            angle: 0,
            loadedObject: null,
        },
        bed: {
```

```
            name: 'Ліжко',
            pathMTL: '../object/bed/bed.mtl',
            pathOBJ: '../object/bed/bed.obj',
            pathTex: '../object/bed/bed_d.png',
            scale: 110,
            positionX: 0,
            positionY: -190,
            positionZ: 0,
            angle: 0,
            loadedObject: null,
        },
        tv: {
            name: 'Телевізор',
            pathMTL: '../object/tv/MI SMART TV.mtl',
            pathOBJ: '../object/tv/MI SMART TV.obj',
            pathTex: '',
            scale: 100,
            positionX: 0,
            positionY: -190,
            positionZ: 0,
            angle: 0,
            loadedObject: null,
        }
    };
    this.objectsToDB = {
        tableAndChairs: {
            scale: 2.5,
            positionX: 0,
            positionY: -190,
            positionZ: 0,
            angle: 0,
```

```
        isPresented: false,
        ...obj.tableAndChairs
    },
    woodenTable: {
        scale: 1,
        positionX: 0,
        positionY: -150,
        positionZ: 0,
        angle: 0,
        isPresented: false,
        ...obj.woodenTable
    },
    tvStand: {
        scale: 10,
        positionX: 300,
        positionY: -190,
        positionZ: 0,
        angle: 135,
        isPresented: false,
        ...obj.tvStand
    },
    chair: {
        scale: 0.2,
        positionX: 0,
        positionY: -190,
        positionZ: 0,
        angle: 0,
        isPresented: false,
        ...obj.chair
    },
    drawer: {
```

```
            scale: 40,
            positionX: 0,
            positionY: -190,
            positionZ: 0,
            angle: 0,
            isPresented: false,
            ...obj.drawer
        },
        bed: {
            scale: 110,
            positionX: 0,
            positionY: -190,
            positionZ: 0,
            angle: 0,
            isPresented: false,
            ...obj.bed
        },
        tv: {
            scale: 100,
            positionX: 0,
            positionY: -190,
            positionZ: 0,
            angle: 0,
            isPresented: false,
            ...obj.tv
        }
    };
    this.textures = {
        wood1: {
            name: 'Дерево 1',
            path: '/src/img/texture/wood1.jpg'
```

```
        },
        wood2: {
            name: 'Дерево 2',
            path: '/src/img/texture/wood2.jpg'
        },
        brick: {
            name: 'Цегла',
            path: '/src/img/texture/brick.jpg'
        },
        wallpaper1: {
            name: 'Шпалери 1',
            path: '/src/img/texture/wallpaper1.jpg'
        },
        wallpaper2: {
            name: 'Шпалери 2',
            path: '/src/img/texture/wallpaper2.jpg'
        },
        tile: {
            name: 'Плитка',
            path: '/src/img/texture/tile.jpg'
        }
    }
    this.lights = lights;
    this.directionalLight = null;
    this.lightX = lightX;
    this.lightY = lightY;
    this.lightZ = lightZ;
    this.lightIntensity = lightIntensity;
    this.textureCache = {};
    this.createInitialRoom();
}
```