Київський національний торговельно-економічний університет

Кафедра кібернетики та системного аналізу

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
на тему:

«Розробка програмного забезпечення дизайнерських рішень за допомогою VR технологій»

Студентки 2 курсу, 1м групи,

спеціальності
051 «Економіка»

спеціалізації
«Економічна кібернетика»

_____ Хорольської Карини Вікторівни

Науковий керівник
кандидат економічних наук, доцент _____ Мороз Ірина Олегівна

Гарант освітньої програми
доктор економічних наук, професор _____ Гамалій Володимир Федорович

Київ 2018

Київський національний торговельно-економічний університет

Факультет обліку, аудиту та інформаційних систем
Кафедра кібернетики та системного аналізу
Спеціальність 051 «Економіка»
Спеціалізація «Економічна кібернетика»

Затверджую
Зав. кафедри _____Роскладка А. А.
«05» листопада 2017р.

Завдання
на випускну кваліфікаційну роботу (проект) студентці

Хорольській Карині Вікторівні

1. Тема випускної кваліфікаційної роботи (проекту)
«Розробка програмного забезпечення дизайнерських рішень за допомогою VR технологій»
Затверджена наказом ректора від *«02» жовтня 2017 р. № 3035*
2. Строк здачі студентом закінченої роботи *15 листопада 2018 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: розробка прототипу програмного забезпечення за допомогою технологій віртуальної реальності для отримання максимально реального відтворення дизайнерських рішень.
Об'єкт дослідження: технології розробки дизайнерських рішень.
Предмет дослідження: програмне забезпечення з використанням технологій віртуальної реальності.
4. Перелік графічного матеріалу_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Мороз І. О.	05.11.2017 р.	05.11.2017 р.
2	Мороз І. О.	05.11.2017 р.	05.11.2017 р.
3	Мороз І. О..	05.11.2017 р.	05.11.2017 р.

6. Зміст випускної кваліфікаційної роботи (проекту) (перелік питань за кожним розділом)

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ВСТУП

РОЗДІЛ 1. ОГЛЯД ТА АНАЛІЗ ПРОСТОРУ ВІТУАЛЬНОЇ РЕАЛЬНОСТІ ТА ЙОГО СКЛАДОВИХ

1.1. Сутнісні характеристики віртуальної реальності в контексті дизайну

1.2. Складові віртуальної реальності та їх вплив на просторове сприйняття

Висновки до розділу 1

РОЗДІЛ 2. ВИКОРИСТАННЯ ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ В ПРОГРАМНОМУ ЗАБЕЗПЕЧЕННІ ДЛЯ ДИЗАЙНЕРІВ

2.1. Технологія створення програмного забезпечення для віртуальної реальності

2.2. Аналіз технологій віртуальної реальності для створення клієнтської частини програмного забезпечення

2.3 Аналіз існуючих програмних рішень для дизайнерів з використанням технологій віртуальної реальності

Висновки до розділу 2

РОЗДІЛ 3. МЕТОДОЛОГІЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Вибір схеми моделі життєвого циклу програмного забезпечення

3.2. Аналіз та визначення засад розробки прототипу програмного забезпечення з використанням технологій віртуальної реальності

3.3 Розробка проекту програмного забезпечення

3.4 Побудова та тестування створеного програмного забезпечення

Висновки до розділу 3

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

7. Календарний план виконання роботи

№ пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		за планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*01.10.2017*	*01.10.2017*
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*05.11.2017*	*05.11.2017*
3	*Вступ*	*01.04.2018*	
4	*Розділ 1.* ОГЛЯД ТА АНАЛІЗ ПРОСТОРУ ВІТУАЛЬНОЇ РЕАЛЬНОСТІ ТА ЙОГО СКЛАДОВИХ	*01.05.2018*	
5	*Розділ 2.* ВИКОРИСТАННЯ ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ В ПРОГРАМНОМУ ЗАБЕЗПЕЧЕННІ ДЛЯ ДИЗАЙНЕРІВ	*20.06.2018*	
6	*Підготовка статті у збірник наукових статей магістрів*	*15.09.2018*	
7	*Розділ 3.* МЕТОДОЛОГІЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	*01.10.2018*	
8	*Висновки*	*01.11.2018*	
9	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*15.11.2018*	
10	*Попередній захист випускної кваліфікаційної роботи*	*22.11.2018*	
11	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*25.11.2018*	
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*28.11.2018*	
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «05» листопада 2017 р.

9. Керівник випускної кваліфікаційної роботи (проекту)

Мороз І.О.

10. Гарант освітньої програми Гамалій В.Ф.

11. Завдання прийняв до виконання студент-дипломник

Хорольська К.В.

12. Відгук керівника випускної кваліфікаційної роботи (проекту)

____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____

Керівник випускної кваліфікаційної роботи (проекту)
__________________ 15.11.2018 р.

13. Висновок про випускну кваліфікаційну роботу (проект)

Випускна кваліфікаційна робота (проект) студента _____________________

може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми__________________________ Гамалій В. Ф.____

Завідувач кафедри _________________________________ Роскладка А.А.___

«_____»_________________2018 р.

АНОТАЦІЯ

Процес побудови інформаційної моделі являє собою революційний метод порівняно з традиційними методами управління та розробки до дизайн-проектів. Проте ця технологія, навіть незважаючи на те, що впродовж останнього часу значно збільшилось її використання в галузі дизайну, архітектури та будівництва, все ще створює перешкоди та обмеження в її широкому використанні через складність.

Віртуальна реальність - це технологія, що впродовж останніх років набула значного покращення у своїй технічній підтримці, що в свою чергу дозволило розповсюджувати шоломи віртуальної реальності за доступними цінами, а отже, це зробило їх більш поширеними. Потужність занурення, пов’язана з інтерактивністю програмного забезпечення віртуальної реальності, дає різний потенціал для індустрії дизайну завдяки здатності користувача візуалізувати простір в масштабі 1:1. Таким чином, дає змогу краще зрозуміти простір чи характер моделей, перш ніж вони будуть створені у реальному світі.

Було використано ігровий движок для розробки прототипу інтерактивної програми для дизайнерів з використанням технологій віртуальної реальності, в якій зацікавлені сторони проектів можуть взаємодіяти та оцінювати дизайн проекту за допомогою віртуальної реальності, в якому не потрібно будь-якого типу технічного формування (користувальницький інтерфейс це інтуїтивно зрозумілий) і де результат роботи в середовищі віртуальної реальності автоматично експортується з моделі віртуальної реальності до моделі BIM.

Ключові слова: віртуальна реальність, дизайн, прототип програмного забезпечення.

Anotation

The process of information model construction is a revolutionary method in comparison with traditional methods of design management and design development in projects. This technology, despite the fact that over the last period its usage in design,

architecture and construction has increased considerably, still there are some barriers and limitations in its widespread usage through complexity.

Virtual reality is a technology that over the past years has gained significant improvement in its technical support, which in turn has allowed the distribution of virtual reality helmets at affordable prices and, consequently, made them more widespread. The dive capacity associated with the interactive virtual reality software provides a brand-new potential for the design industry because of users' ability to visualize in a scale of 1: 1. This way, it allows better understanding of the space or of the models before they are created in the real world.

A game engine was used to develop a prototype of a program for designers using virtual reality technologies in which interested project parties can interact and evaluate project design with virtual reality and it does not require any kind of technical education (user interface is intuitive) and where the result of work in a virtual reality environment is automatically exported from the virtual reality model to the BIM model.

Keywords: virtual reality, design, software prototype.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

2D – 2-dimensional – двовимірний простір

3D – 3-dimensional – тривимірний простір

API – application programming interface – інтерфейс програмного додатку

BIM – building information model – інформаційна модель будівлі

BOOM – Binocular Omni-Orientational Monitor – зрівноважений стереоскопічний пристрій для перегляду, що забезпечує інтерактивне керування відображення з позиції точки оптичного зору в 3D середовищі у режимі реального часу, створеному комп'ютером або камерою

CAD – computer-aided design/drafting – технологія автоматизованого проектування

CRT – cathode ray tube – електронно-променева трубка

CSV – comma-separated values – файловий формат для представлення табличних даних, для відокремлення поля символом коми та переходу на новий рядок.

CAVE – computer-aided virtual environment – комп'ютерне середовище віртуального середовища

DVE – distributed virtual environment – розподілене віртуальне середовище

DVR – digital video recorder – цифровий відеореєстратор

FBX – filmbox – технологія та формат файлів, використовується для забезпечення сумісності програм тривимірної графіки

GUI – graphical user interface – графічний інтерфейс користувача

HMD – head-mounted display – шолом віртуальної реальності

IPD – integrated project delivery – інтегрована доставка проектів

IPS – in plane switching – технологія екрана для рідкокристалічних дисплеїв

LOD – levels of detail – рівні деталізації

OBJ – стандартний формат опису геометрії у 3D розробці

RP – rapid prototyping – швидке прототипування

SGI – Silicon Graphics Inc. – американська комп'ютерна компанія

VR – virtual reality – віртуальна реальність

ПЗ – програмне забезпечення

ВСТУП

Ефективний зв’язок між клієнтами та дизайнерами має вирішальне значення для практики дизайну. Дизайнери можуть володіти провідними навичками у своїй галузі, але все ще проблемою залишається питання комунікації з клієнтами; якщо процес спілкування з клієнтами не буде успішним та двосторонньо зрозумілим, - не може бути гарантії, що проект дизайну буде успішним. Клієнти повинні відчувати себе комфортно і відкрито у співпраці з дизайнерами [1]. Очікування, уподобання та зауваження повинні бути відкрито донесені до дизайнера. Без достатнього розуміння бажань та вимог клієнтів – співпраця не буде плідною і, як результат – збитковою як для дизайнерів так і для клієнтів, а бажаний результат так і не буде досягнутий.

На сьогодні, використання програмного забезпечення для автоматизованого проектування з метою побудови дизайнерських моделей є найпопулярнішим способом подання та оцінки дизайнерських пропозицій. Порівняно з дизайном виконаним за допомоги олівця та паперу, моделювання CAD дозволяє більш ефективно та точно передавати дизайнерські ідеї [2]. Кожен компонент моделі може бути побудований з дотриманням точного розміру та положення в 3D просторі. Крім того, CAD модель дозволяє швидше модифікувати конструкції з нуля, без необхідності повторно малювати креслення чи дизайн. Однак, для того, щоб клієнти могли переглядати та модифікувати CAD-модель, існують певні умови та вимоги як до комп’ютерного обладнання, так і до програмного забезпечення. Крім того, найбільш поширеними засобами, що використовують дизайнери, (такими як AutoCAD, Revit та MicroStation) є високопрофесійні пакети програмного забезпечення, які потребують значного вивчення для правильного застосування та використання. Перед початком процесу проектування неможливо та недоцільно проводити підготовку та навчання клієнтів вищезазначеним продуктам. А отже, клієнтам потрібна технічна підтримка дизайнерів для роботи з 3D-моделями. Традиційно дизайнери надають клієнтам лише візуалізацію; клієнтам рідко зустрічається можливість оглянути інтерактивні моделі 3D.

Через брак відповідних навичок (наприклад, комп'ютерного моделювання або навичок робити ескізи від руки) більшість клієнтів не можуть поділитися своїм дизайнерським уявленням, оскільки можуть використовувати лише письмовий або усний опис своєї ідеї. Хоча текстові описи є достатніми для визначення деяких специфікацій дизайну, наприклад вимог до матеріалу та бюджету, такий опис не може чітко визначити детальні технічні характеристики проекту, так як це виконує наглядна графічна презентація. У результаті дизайнери повинні витрачати додаткову енергію, створюючи візуалізацію запропонованої конструкторської моделі, щоб переконатись, що це те що бажає клієнт. Це може бути довгим і повторюваним процесом, який вимагає великої кількості зустрічей з метою успішного завершення дизайн-проекту. Такий неефективний підхід підштовхує до дослідницької пропозиції нового комунікаційного підходу, що базується на технологіях віртуальної реальності, що дозволяє візуалізувати варіанти дизайну за допомогою інтерактивної цифрової графіки.

Мета роботи: розробка прототипу програмного забезпечення за допомогою технологій віртуальної реальності для отримання максимально реального відтворення дизайнерських рішень. Дослідницький підхід полягає у інтерактивному використанні віртуальної реальності та автоматичному створенні базових моделей дизайну. Віртуальна реальність - це комп'ютерна технологія, що використовує технології 3D візуалізації для імітування фізичної присутності. Оскільки така система усуває як часові, так і географічні обмеження, програмне забезпечення з використанням технологій віртуальної реальності забезпечує прямий та безпосередній візуальний досвід для користувачів з персональними комп'ютерами та системами віртуальної реальності.

Створення програмного забезпечення на базі технології віртуальної реальності підвищить ефективність співпраці між клієнтами та дизайнерами, а також дозволить технології віртуальної реальності стати щоденно використовуваним інструментом для проектування та комунікації під час процесу дизайну. Як вже зазначалося, одна з найбільших перешкод для цього полягає в труднощах попередньої обробки та неможливість візуалізувати дизайн в режимі

реального часу. Тому основними цілями є розробка методів та алгоритмів, які дозволяють безпосередньо візуалізувати великі та складні моделі в режимі реального часу. Ця робота описує дизайн інтер'єру як приклад дослідження.

Об'єкт дослідження: технології розробки дизайнерських рішень.

Предмет дослідження: програмне забезпечення з використанням технології віртуальної реальності спрямоване на використання у галузі дизайну.

Задля досягнення мети даної роботи, нам необхідно виконати такі **завдання:**

- Висвітлити передумови та поточний стан візуалізації дизайну в реальному часі з використанням технологій віртуальної реальності. Проблема візуалізації в реальному часі неодноразово висвітлювалась в багатьох дослідженнях в розрізі теорії – не прикладного рішення. У поточній роботі міститься багато прикладів, в яких виражаються проблеми та передумови використання технології віртуальної реальності для цілей візуалізації дизайн проектів.
- Описати методики та існуючі бібліотеки для реалізації програмного забезпечення з використанням технологій віртуальної реальності. Існує ряд методів та алгоритмів, які можуть бути використані для автоматизації візуалізації в реальному часі. Вони мають як сильні так і слабкі сторони, а вибір сильно залежить від типу 3D-середовища, до якого він повинен застосовуватися. Наприклад, величезний, відкритий пейзаж, який спостерігається в симуляторі польотів, сильно відрізняється від детального середовища квартири.
- Визначити роль інтеграції технології віртуальної реальності у процес розробки дизайну. Для того, щоб технологія віртуальної реальності стала природною та інтегрованою частиною процесу проектування дизайну, потрібно подолати кілька бар'єрів. У цьому контексті технічна можливість візуалізувати великі та складні дизайни в режимі реального часу відображає лише одну з них. Доступність цих методів, а також зручність та інтерфейс стають другою важливою складовою. Наприклад, якщо трудомісткі процеси

- хоча автоматизовані - необхідні для реалізації сеансу візуалізації в віртуальній реальності - негативно вплинуть на інтеграцію. Аналогічним чином, якщо фактичний навігаційний інтерфейс виявиться занадто складним для не-експертів, буде важко підтримувати участь кінцевих користувачів.

- Розробити прототип програмного забезпечення з використанням технологій віртуальної реальності для побудови та візуалізації дизайн-рішень.

Після вирішення поставлених перед нами завдань, ми створимо новий світ віртуальної реальності. Віртуальне середовище стане новим для дизайнерів, і для архітекторів. Здатність «гуляти по дизайну» дає абсолютно інший рівень опрацювання та інші можливості для тестування.

РОЗДІЛ 1
ОГЛЯД ТА АНАЛІЗ ПРОСТОРУ ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ ТА ЙОГО СКЛАДОВИХ

1.1. Сутнісні характеристики віртуальної реальності в контексті дизайну

Терміни віртуальне середовище та VR (далі «віртуальна реальність») часто використовують як синоніми для опису комп'ютера, штучного «середовища» або «реальності», що представляється користувачеві [3].

Віртуальне середовище намагається викликати сильне відчуття реальності у користувача. Це досягається шляхом створення штучного введення користувачам візуальних, акустичних та гаптичних почуттів. Завдяки взаємодії деяких артикуляцій користувачів у реальному світі у віртуальному середовищі користувач може свідомо взаємодіяти із середовищем.

Як правило, інтерфейси до пристроїв прямої маніпуляції використовуються, але в даний час найвибагливішими науковими інтересами стали більш розвинені способи взаємодії, такі як розпізнавання мови та жестів. Генерування високоякісних візуальних зворотних зв'язків з віртуального середовища часто розглядається як найважливіший аспект у створенні високого ступеня занурення. Прагнення підвищити ступінь занурення призвело до розробки складних генераторів зображень та пристроїв відображення.

Починаючи з моноскопічних CRT-дисплеїв з низькою роздільною здатністю, що використовуються в ранніх польотних симуляторах та генераторах зображень, де вони здатні відображати лише кілька сотень полігонів за секунду, розробка просунулася до сьогоднішніх стереоскопічних систем відображення високої роздільної здатності, таких як CAVE та доступні графічні карти, які надають сто мільйонів полігонів у секунду.

Залежно від виду програми, програмне забезпечення віртуального оточення повинно керувати такими різноманітними дисплеями, як дисплеї на головному

екрані (HMD), перегляд HMD для розширеної реальності, активні стерео проекційні системи з скляними заглушками з екраном персонального комп'ютера, системи пасивної стерео проекції з використанням поляризації для розділення зображень або різноманітних багатоекранних проекційних систем.

Паралельно з розробкою нових пристроїв відображення, генераторів зображень та пристроїв введення, розробляються різні набори інструментів та рамки застосування. Вони забезпечують базову програмну інфраструктуру для розробки додатків віртуального середовища. Основною метою цих зусиль є максимізація повторного використання програмного забезпечення з метою мінімізації необхідних ресурсів розробки для розробки додатків. Розроблена для різних доменів додатків, єдиним загальним номінатором більшості наборів інструментів та структур є об'єктна модель на базі сценаріїв. Забезпечений API, підтримувані апаратні засоби та операційні системи, а також набір підтримуваних пристроїв відображення та введення значно відрізняються [4].

Більш детальна оцінка представницьких наборів інструментів і структур показує, що, хоча кожна система має - залежно від області застосування - перспективні та часом широко прийняті рішення для конкретних наборів завдань, жодна з них не має всіх необхідних властивостей, щоб слугувати основою для загального призначення DVE framework [5].

Спільний компроміс полягає між розробленим об'єктом та моделлю подій разом з API, а з іншого боку – продуктивність системи, що підкреслює, що API, дружній для розробників, як правило, нехтують підтримкою оптимізації продуктивності, тоді як системи, оптимізовані для продуктивності, часто надають нерозвинені API та об'єктні моделі. Ця теза стверджує, що розробка основи для розробки DVE-додатків повинна базуватися на добре відомих та добре зрозумілих концепціях дизайну від автономних наборів інструментів віртуального середовища та схем, щоб їх було прийнято розробниками та підвищити продуктивність. У той же час, максимальна продуктивність є основною вимогою для створення інтерактивних додатків, і не повинна скомпрометуватися дизайном API-інтерфейсів. Тому найуспішніші концепції дизайну обираються з існуючих

автономних структур віртуального середовища та об'єднуються в основний дизайн системи, наприклад як для системи Avocado.

Віртуальна реальність та занурення означають вигаданий наратив, що може дати представлення інакшого сприйняття реального світу, для огляду і створення нових структур та дизайнів. Віртуальне середовище дає змогу розробляти сценарії для моделювання можливостей, в результаті чого виникають нові можливості дизайну. Ці можливості є потенційно нескінченні, і залежать від комбінації чинників, таких як технологічна еволюція [6].

Віртуальна реальність розглядається як технологія та засіб масової інформації, а також поле для досліджень в області художнього створення та експериментів. Віртуальна реальність є не тільки інтерактивним для представлення ідей, а й динамічним інструментом для моделювання в 3D-просторі. Виразний, інтерактивний та багатоплановий підхід, досягнутий за допомогою технологій віртуальної реальності, призводить до інновацій у створенні та розвитку тривимірних середовищ. Віртуальна реальність також використовується для вивчення експериментальних шляхів для нових художніх підходів. Tilt Brush [18] - це відповідний приклад технології, що застосовується для розробки нових технологічних інструментів для художніх цілей. Іншим інструментом, що використовує віртуальну реальність для художніх цілей, є Infectious Ape [19].

Доповнюючи вищеописане, віртуальна реальність є полем для представлення ідей та понять, у тому числі і художнього підходу, в віртуальному середовищі. 3D-інтерактивне середовище в контексті віртуальної реальності - це поле для розробки і представлення інноваційних та творчих ідей, та концепцій з точки зору дизайну. Деякі експерименти в віртуальній реальності були виконані з використанням нейронних мереж, таких як Deepdream, для вивчення концепцій фрактального дизайну. Поява віртуальної реальності як середовища, призвела до появи нових підходів у новому дизайн-мистецтві, збільшуючи можливості творчого розвитку та дослідження, а отже – збільшуючи можливості взаємодії.

Систематичний огляд існуючої літератури дає змогу виділити та класифікувати функції, пов'язані з віртуальною реальністю, для створення дизайну, що покращують процес взаємодії між користувачами та інформацією в багатовимірних формах. Компоненти взаємодії та процесів занурення повинні підходити користувачеві, а також дизайнерам. Отже необхідне налаштування адаптації цих міждисциплінарних дослідницьких областей для розробки взаємодій та образів у віртуальному середовищі.

Розробка моделей та підходів до створення взаємодій та сутностей у віртуальному середовищі, що поєднує досвід користувачів, дизайнерів, ергономічні та пізнавальні чинники, та експресивну силу, а також технологічний розвиток, є сучасними проблемами для віртуального середовища, в якому розробляється дизайн.

Віртуальна реальність - це складне середовище багатовимірних підходів і фокусів, що описує реальність (навіть альтернативну реальність, яка має свої власні правила). Тому слід пам'ятати про деякі чинники, оскільки віртуальна реальність, як нестабільне середовище, має свої власні особливості спілкування та взаємодії. Ці функції мають когнітивні, технологічні, дизайнерські, артистичні та ергономічні підходи, і, є інтерактивним середовищем [7].

1.2. Складові віртуальної реальності та їх вплив на просторове сприйняття

Віртуальна реальність має чотири ключові елементи для створення повноти досвіду використання технології в реальному часі: віртуальний світ, занурення, інтерактивність та сенсорні зворотні зв'язки [20].

Щодо *віртуального світу,* то живопис, комп'ютерна графіка та комп'ютерна анімація вже вирішили питання привабливого представлення «віртуального» світу. Віртуальна реальність також використовує подібне моделювання, відображення та технологію освітлення для створення свого віртуального світу.

У дизайн-візуалізації в реальному часі геометричний метод схожий на комп'ютерне зображення та анімацію. Однак, щоб зменшити час рендеринга, щоб отримати плавний рух через віртуальний світ, геометрія максимально мінімізується. Складні геометричні моделі важко працюють в системах віртуальної реальності.

На відміну від комп'ютерної графіки та анімації, для спрощення моделі та гарантії високої частоти кадрів у віртуальній реальності текстурне відображення виконується з використанням текстурних карт, для підвищення швидкості рендерингу в режимі реального часу. Текстурна карта може імітувати більшість ефектів загального матеріалу, як тінь і текстура.

На сьогоднішній день віртуальна реальність не підтримує сучасні освітлювальні технології, такі як радіовипромінювання та промінь трасування, які використовуються в комп'ютерній графіці та анімації. Фактурна карта часто використовується для імітації ефектів освітлення. Причиною є те, що дуже складно отримати плавне переміщення в віртуальній реальності з сучасними технологіями освітлення на сучасних комп'ютерних графічних системах.

Отже, в порівнянні з комп'ютерною графікою та анімацією, віртуальний світ, створений віртуальною реальністю, часто спрощується, щоб гарантувати плавне переміщення на комп'ютерних графічних системах. Однак ця техніка часто погіршує візуальне сприйняття віртуального світу.

Занурення. Німецький композитор та режисер театру Річард Вагнер у своєму есе 1849 р. «Мистецька творчість майбутнього» використовував термін Gesamtkunstwerk (у перекладі з нім. - «загальне творіння»), щоб проілюструвати його ідеал створення театру, який об'єднує всі форми мистецтва. Його намір полягав у створенні захоплюючого досвіду, коли глядач психічно пересаджується на сцену через мультисенсорну стимуляцію. Вагнер не єдиний, хто мав бажання максимально збільшити відчуття занурення, - протягом всієї історії мистецтва, художники намагалися реалізувати подібне бажання за допомогою різних форм мистецтва, включаючи картини, романи та фільми.

У сьогоденні, завдяки віртуальній реальності, існує можливість створювати світи, більш занурені, ніж коли-небудь було можливо раніше. Одна з найбільш відмінних характеристик віртуальної реальності полягає в тому, що вона пропонує повноцінний досвід учасника. У віртуальній реальності, користувач повністю оточений віртуальним середовищем, виключаючи себе з фізичної реальності. Ці сенсорні подразники, такі як 3D-стереоскопічна проекція, просторовий звук і зворотний зв'язок, дозволяють нам пересаджуватись в іншу реальність, що в кінцевому підсумку веде нас до повного залучення до неї.

У технічно-орієнтованих дослідженнях занурення часто розглядається як об'єктивний і технологічний аспект віртуальної реальності, а, як підрозділ занурення - присутність - це психологічний фактор і пізнавальний наслідок занурення. Слейтер і Вилбур (Slater & Wilbur) визначають занурення як «об'єктивний та кількісний опис того, що забезпечує будь-яка конкретна система», а присутність - «стан свідомості, (психологічне) відчуття буття у віртуальному середовищі».

З вище описаного - існує два типи занурення, психічне та фізичне занурення. Психічне занурення - це те, що досягається засобами візуалізації нерухомих зображень. Комп'ютерна анімація, з її здатністю створювати відчуття часу і руху, покращує рівень психічного занурення, але лише віртуальна реальність досягає фізичного занурення [8].

Психічне занурення означає «відчуття присутності» в середовищі [21]. Для деяких віртуальних світів, таких, як описано в романах, психічне занурення має вирішальне значення для успіху всього віртуального досвіду.

Для успішної дизайн-візуалізації важливо переконатися, що аудиторія отримує правильне відчуття присутності. Хоча зображення намальоване від руки, комп'ютерна графіка та анімація досягають певного рівня психічного занурення, віртуальна реальність піднімає його на більш високий рівень. Інтерфейс візуалізації в режимі реального часу пропонує аудиторії сприйняття особистості на сцені. Тим часом поняття часу і руху у віртуальній реальності не визначено, як у комп'ютерній анімації. Це дає більше шансів аудиторії діяти так, наче вони в

фізичному світі, зберігаючи більш точне відтворення ментального почуття візуалізованого простору.

Фізичне занурення здійснюється шляхом представлення користувачам віртуального світу на основі їх розташування та орієнтації та надання синтетичних стимулів для одного чи декількох їхніх почуттів у відповідь на їх положення та дії. Система віртуальної реальності представляє перспективні, незалежні зображення для кожного ока та синхронізований аудіосигнал до вух. Коли користувач рухається, у відповідь змінюються візуальні, слухові, хаптичні та інші якості, які становлять фізичне занурення. Сьогоднішній інтерфейс віртуальної реальності може досягти високої точності фізичного занурення. Горизонтальні дисплеї, BOOM, CAVE та проекційні системи можуть допомогти аудиторії віртуальної реальності досягти вражаючого фізичного занурення. Сьогодні гаптичні, нюхові та смакові занурення не є популярними в дизайн-візуалізації через технологічні та витратні проблеми. Проте в віртуальній реальності легко створювати тривимірне джерело звуку, звукові ефекти, такі як звук для відкриття дверей, часто використовуються в дизайн-візуалізації для створення фізичного занурення. Фактично, з іншими традиційними засобами візуалізації, такими як малюнок від руки, комп'ютерна графіка та комп'ютерна анімація, фізичного занурення немає. Віртуальна реальність є єдиним середовищем візуалізації, що дозволяє значно збільшити реалістичний ефект візуалізації, перетворюючи дизайн-візуалізацію в досвід повного фізичного занурення.

Як ключовий компонент дизайн-візуалізації, рівень занурення має вирішальне значення для вірності сприйняття ідеї дизайнера. У цій сфері віртуальна реальність може відображати стан речей набагато краще, ніж інші традиційні засоби візуалізації. Це може допомогти дизайнерам, клієнтам та іншим учасникам в дизайн проекті більш точно зрозуміти ідею дизайну та зробити більш обґрунтоване рішення.

Інтерактивність. Враховуючи, що занурення є відповіддю на можливу статичну форму подання, елемент інтерактивності вимагає динамічного

середовища. Це не просто здатність учасника здійснювати навігацію по навколишньому середовищу, але, головне, віртуальне середовище, що реагує на дії користувача. Використання датчиків системи віртуальної реальності для захоплення позиціонування користувачів та надання їм свободи переміщення може бути задовільним досвідом, але коли ці дії не викликають жодних реакцій у віртуальному світі, це обмежує ступінь занурення користувача.

Відповідно до словника Вебстера (1989), інтерактивність – це «взаємні дії або вплив. Таким чином, взаємодія з середовищем, створюваною комп'ютером, відбувається, коли відповідні дії комп'ютера реагують на вхідні дані користувача» [22]. А отже, використовуючи метафори для опису інтерактивності - це порівняння з театральною діяльністю. Користувачі VR подібні до театральних аудиторій, вони не тільки можуть дивитися п'єсу, але також можуть приєднатися до сцени, щоб брати участь, стати різними персонажами, змінювати дії за своїми словами та робити їх у ролях [23]. Ця взаємодія здійснюється через користувацький інтерфейс. Зв'язок між людьми та фізичним світом є своєрідним інтерактивним зв'язком. Люди пересуваються і діють у фізичному світі, і отримують відгуки від нього. Ця інтерактивність становить важливу частину переживання в фізичному світі. З розвитком комп'ютерних графічних систем все більше і більше дизайнерів і інженерів у комп'ютерній візуалізації стали намагатися включати інтерактивність в комп'ютерне представлення. Віртуальна реальність є плодом цих зусиль.

На відміну від попередніх джерел дизайн-візуалізації, інтерактивність є значною відмінністю віртуальної реальності. У досвіді віртуальної реальності аудиторія може пройти через простір, обертати точку зору, відкрити двері, переміщувати об'єкти, відтворювати відео, вимикати світло та багато іншого. Іншими словами, метою віртуальної реальності є створення віртуального світу, подібного до фізичного світу, або подібного до сюжета фільму «Матриця».

У віртуальному середовищі учасник може досягти відчуття занурення через взаємодію з віртуальним середовищем та об'єктами, що його займають. Марк

Мейн (Mark Mine) ілюстрував три основні методи, за якими більшість форм взаємодії можуть бути виконані в рамках віртуального середовища:

1. Прямий інтерфейс користувача - інтуїтивно зрозумілий та гнучкий метод взаємодії, в якому учасник взаємодіє з об'єктами так само, як і в реальному світі. Для безпосередніх користувацьких інтерфейсів надзвичайно важливо розробляти природні інтуїтивні відображення між діями учасника та відповідними діями у віртуальному світі;

2. Фізичний контроль – метод контролю віртуального середовища, котрий використовує апаратні кнопки реального світу, клавіатуру, джойстики, рульові колеса. Позитивний зворотний зв'язок наданий учаснику через контрольні пристрої, що підвищує почуття занурення та полегшує точний контроль. Проте фізичний метод не має високої гнучкості.

3. Віртуальний контроль - це метод, в якому взаємодіють з віртуальним світом через віртуальні об'єкти, що представляють собою комп'ютерні зображення. Оскільки елементи керування є віртуальними, він не має гаптичного зворотного зв'язку, і учасник має загальну складність у взаємодії з віртуальними об'єктами.

Варто зазначити, що інтерактивність віртуальної реальності включає в себе:

1. Маніпулювання - означає здатність «діяти руками або спеціально механічними засобами» [24]. У віртуальній реальності ми маємо більше можливостей маніпулювати віртуальним світом. Загалом, для виконання маніпуляцій є три способи: пряме управління користувачем, фізичний контроль та віртуальний контроль. Прямий контроль над користувачем використовує «жести інтерфейсу, що імітують взаємодію в реальному світі». Цей контроль працює лише тоді, коли учасник знаходиться у віртуальному світі. Фізичний контроль стосується пристроїв, імітуючих фізичний дотик користувача. Пристрої існують у фізичному світі, і учасник керує ними віртуальним світом. Багато комп'ютерних ігор використовують подібні маніпуляції. Наприклад, джойстик є операційним ресурсом у багатьох комп'ютерних іграх. Віртуальний контроль стосується пристроїв, які користувач може віртуально торкнутися. На відміну від фізичного

контролю, віртуальні контрольні пристрої існують лише у віртуальному світі. У більшості випадків віртуальний контроль - це моделювання фізичного контролю у віртуальному світі;

2. Навігація - це можливість пройти через віртуальну реальність. Вона складається з двох частин: пошуку шляху та переміщення. Пошук шляху - це методи, які допомагають аудиторії у віртуальній реальності визначити власну позицію в віртуальному світі та напрями куди можна піти далі. В дизайн-візуалізації в режимі реального часу імітуються двері, вікно, хол, коридор, сходи. Все відбувається як у фізичному світі. Подорож відноситься до здатності глядачів переміщатися у віртуальному світі за допомогою пристроїв. У віртуальній дизайн реальності, створений EON Studio47, ми можемо використовувати мишу, щоб керувати переміщенням, наприклад, ми можемо йти вперед і назад, змінювати висоту і повертати камеру.

3. Комунікація - відноситься до взаємодії з іншими агентами у віртуальній реальності. Це є популярним в комп'ютерних іграх, і багато ігор підтримують двостороннє або багатостороннє спілкування. В дизайні взаємодія віртуальної реальності з великою кількістю агентів ще не набула розвитку. Хоча це дає можливість дизайнерам розширювати віртуальне середовище проектування, є й недоліки – високі вимоги до апаратного забезпечення, програмного забезпечення та інтерфейсу візуалізації в режимі реального часу.

Ступінь інтерактивності залежить від різних чинників. Штауер (Steuer) вважає, що взаємодія залежить головним чином від трьох факторів:

- швидкості - визначається швидкістю відображення дій користувача в віртуальному світі. Швидша реакція пов'язана з більшою кількістю дій та змін у віртуальному світі, ця швидкість безпосередньо залежить від можливостей системи віртуальної реальності.

- діапазону – визначається як число можливих результатів для будь-якої дії користувача. Цей фактор можна порівняти з набором інструментів, в якому чим більше існує інструментів, тим більш працездатним є середовище;

- прогнозування відображення - означає здатність системи передбачати карту дій користувача від зміни навколишнього середовища.

Справедливим є ствердження, що користувач мусить мати змогу певною мірою передбачити результати своїх дій, інакше його дії є лише чистими рухами, а не намірами дії. Інтерактивність в більшості додатків віртуальної реальності - мусить бути максимально інтуїтивно зрозумілою та практичною, адаптованою до цілі самої програми.

Як вже зазначалось, інтерактивність є важливою частиною віртуальної реальності. Вона вважається одним з ключових компонентів вірності у віртуальній реальності і унікальності середовища. Завдяки постійному вдосконаленню комп'ютерного обладнання та програмного забезпечення, у віртуальному світі можна буде створити ще більше взаємодій з фізичним віртуальним світом, що призведе до ще більш реалістичного досвіду від використання віртуальної реальності.

Сенсорні зворотні зв'язки є ключовим елементом для того, щоб зробити досвід віртуальної реальності більш вдалим. На відміну від інших типів подачі інформації, системи віртуальної реальності надають користувачам можливість унікально впливати на події у віртуальному світі.

Краще, ніж інші традиційні візуалізаційні засоби передачі графічної інформації, віртуальна реальність також пропонує зворотній зв'язок [9]. В більшості випадків візуальний зворотній зв'язок є основним сенсорним зворотним зв'язком у віртуальній дизайн реальності. Однак аудіо також корисне для дизайнерів, щоб аудиторія могла отримати більш реалістичний віртуальний досвід. Належний звук можна імітувати, щоб відкрити двері або відтворити відео. Нюхові та смакові зворотні зв'язки потребують високошвидкісних комп'ютерів та розширений інтерфейс. Сьогодні вони рідко використовуються в дизайн-візуалізації, оскільки їх час та вартість часто не є практичними. Через поточний стан розвитку сенсорних зворотних зв'язків у дизайн-віртуальній візуалізації - вони не є акцентом цієї роботи. Незважаючи на це, в майбутньому, при постійному розвитку комп'ютерних технологій, подібний сенсорний зворотній

зв’язок буде застосовано до дизайн-візуалізації за розумною ціною та ефективною швидкістю та для підвищення якості віртуального досвіду.

Описуючи чотири ключові компоненти віртуальної реальності стає очевидним, що віртуальне середовище має неперевершені переваги перед іншими способами візуалізації. Всі ці переваги гарантують покращення точності дизайн-візуалізації.

З допомогою віртуальної реальності дизайнери отримують можливість розробляти дизайн у віртуальному середовищі. Давайте представимо процес дизайну. Працівник створює простір шляхом нагромадження елементів, таких як меблі, техніка та покриття в масштабі 1:1 у фізичному просторі. Відносини працівника до дизайн простору є реальними, що часто відбувається в двовимірних кресленнях та трьох-мірних уявленнях. Процес проектування дизайнерів був іншим до виникнення віртуальної реальності. Раніше дизайнери могли використовувати лише двовимірні та тривимірні засоби для візуалізації, щоб отримувати відгуки для дизайнерських ідей, а потім поступово їх втілювати. Фізична модель - це ще один підхід до дизайнерського спілкування.

Вдосконалення технології віртуальної реальності змінює все, пропонуючи дизайнерам віртуальний дизайн середовища [10]. В рамках цього захоплюючого дизайнерського середовища створення форми в космосі вперше стає можливим без будь-якого посередництва. Як у мага, жест дизайнера може піднімати стіни, розрізати отвори і регулювати нахил дахів. Підлога та сходи можуть бути додані та модифіковані відповідно до реакції та суджень, спровокованих перцептуальним впливом. Якщо дизайн базується на обсягах, можливо використовувати логічні операції додавання або віднімання, що дозволяє формувати віртуальний простір, подібний до створення скульптури шляхом формування та різання. Все це відбувається в 1:1 масштабі віртуального простору, що гарантує дизайнерам отримання найбільш реалістичного та точного сприйняття всього простору. Тепер дизайнерам не потрібні окремі двовимірні чи тривимірні зображення або моделі. Вони можуть створювати віртуальний простір у віртуальній реальності, так само як будівельник створює фізичний простір.

Висновки до розділу 1

Систематичний аналіз літератури привів нас до виділення та поділу функцій, що стосуються віртуальної реальності, з метою створення моделей, необхідних для покращення процесу взаємодії між користувачами та інформацією в віртуальному режимі [11]. Компоненти взаємодії та процеси занурення у віртуальну реальність повинні співвідноситись з досвідом користувача. Було виділено аспекти необхідні для налаштування та адаптації міждисциплінарних дослідницьких областей для розвитку технології віртуальної реальності.

Розробка моделей та підходів до розробки взаємодій віртуального простору у віртуальному середовищі, що поєднує досвід користувачів, ергономічні та пізнавальні чинники, експресивну силу, а також технологічний розвиток, є актуальними питаннями для створення програмного забезпечення з використанням технологій віртуальної реальності.

Метою аналізу було визначення аспектів для опису потреб та передумов створення запропонованого програмного забезпечення, та огляду його можливостей як у практичних так і в дослідницьких підходах.

РОЗДІЛ 2
ВИКОРИСТАННЯ ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ В ПРОГРАМНОМУ ЗАБЕЗПЕЧЕННІ ДЛЯ ДИЗАЙНЕРІВ

2.1. Технологія створення програмного забезпечення

У епоху глобалізації віртуального середовища VR технології почали відігравати важливу роль у сфері дизайну. Дизайнери продукують ідеї у своїй уяві, згодом створюють креслення, чи 3D моделі, що інтерпретуються на 2D екрані. Цього замало, галузь дизайну стоїть перед проблемою теорії (початкової ідеї та її 3D реалізації) та застосування, щоб подолати прогалини між вирішенням дизайнерської проблеми та її перетворення на матеріальний продукт [12]. Віртуальний прототип допомагає візуально зрозуміти розмір і форму продукту, а фізичні продукти можуть бути реалізовані за допомогою Rapid Prototyping (RP). Проте, продукт може вимагати додаткових модифікацій за відсутності зворотного зв’язку під час процесу віртуального прототипування, оскільки дизайнери можуть не сприймати такі характеристики, як текстури, еластичність, вага, глибина, просторовість тощо.

Покращення технологій тривимірної візуалізації та збільшення попиту на інноваційні методи в дизайнерській галузі змушують використовувати широкий спектр новітніх методик, що включають віртуальні середовища. Технологія віртуальної реальності забезпечує реалізм та інтерактивність. Віртуальна реальність разом з таким пристроєм як haptic, що інтегрований з віртуальним прототипом, може зменшити різницю між фантазією та реальністю віртуальної моделі, та забезпечити реальність, як сприйняття прототипованих продуктів і значно покращити кінцевий продукт. Віртуальну реальність можна успішно поєднувати з CAD, щоб забезпечити зворотний зв’язок, формування сприйняття простору та форми, що дозволяє подолати існуючий розрив, у певній мірі, між уявними / задуманими та реальними продуктами, реалізованими через RP.

Галузь дизайну затребувана «за вимогою клієнтів», а, оскільки, клієнти більше зосереджують увагу на ергономічних характеристиках (наприклад, комфорт, зовнішній вигляд, текстуру, легкість використання тощо), саме ці характеристики продукту зазвичай приймаються за базис на ранній стадії розробки продукту, що, як наслідок, впливає на процес виробництва та кінцевий результат. Таким чином, технологія віртуальної реальності та додаткової реальності є доцільною для прискорення та надання широкого вибору, в питаннях, що потребують прийняття рішень на ранній стадії процесу проектування, основуючись на характеристиках, бажаних клієнтом. Віртуальна реальність має величезний потенціал, який допомагає візуалізувати та зрозуміти складні поняття та теорії, привести до створення нового дизайну продукту.

Віртуальна реальність уже починає використовується для промислового дизайну, військової підготовки, автомобільного та аерокосмічного дизайну, медичних (хірургічних, стоматологічних, а також для лікування фобій та аутизму), технічного обслуговування, ремонту та розваг [13]. Віртуальна реальність також може забезпечувати відповідне середовище для галузі дизайну, що допоможе скоротити час розробки та витрати, а також поліпшити якість та зручність використання нових продуктів.

Технологія створення програмного забезпечення глобально розділяється на два питання, що потребують огляду.

Першим питанням є питання використовуваного устаткування. Це питання не є безпосереднім питанням розробки програми, але невід'ємною частиною процесу розробки.

Другим питанням є питання безпосередньої розробки програмного забезпечення для віртуального середовища.

Використовуване устаткування. Система віртуальної реальності складається з декількох компонентів, які повинні бути добре інтегрованими. Ці компоненти - це апаратне забезпечення системи, разом із програмним забезпеченням, що підтримує зв'язок дисплеїв з введенням дій користувача, розробка інтерфейсу користувача, що забезпечує зручний спосіб взаємодії,

нарешті і настільки ж важливим є віртуальний світ, що містить вміст, з яким користувач буде взаємодіяти [14].

Обладнання, яке використовується в системах віртуальної реальності, можна грубо класифікувати як пристрої відображення, з пристроями введення, які користувач свідомо ініціює, інші пристрої вводу, які відповідають за відстеження позиції користувача, разом із комп'ютером, який підтримує моделювання та візуалізацію віртуального світу.

Обчислювальний компонент обладнання віртуальної реальності відповідає за розрахунок всієї поведінки віртуального світу та надання його для представлення користувачам переважно візуальними або аудіопрограмами.

Візуальне відображення апаратного забезпечення найбільше впливає на загальний дизайн системи віртуальної реальності. Це вплив є наслідком того, що візуальна система є ключовим елементом більшості віртуальних реалій у галузі дизайну, інженерії та будівництва. В основному, це два види візуального дисплея: стаціонарні та портативні.

До портативних пристроїв відносяться:

1. Пристрої відстеження – пристрої що відображають та слідкують за орієнтацією та положенням. Головним чином пристрої відстеження використовують наступні способи стеження за орієнтацією та положенням:

- акустичний – спосіб що використовує принцип ехолокатора, - обчислення часу польоту звукової хвилі до об'єкту та назад, для визначення позиції об'єкта в просторі;
- електромагнітний – спосіб що використовує джерело низькочастотного електромагнітного поля та сенсорів (подібний до сучасних дисплеїв мобільних телефонів, тільки більшого розміру). Проблемою такого способу є не стабільна робота коли в середовищі присутні провідники, а також малий радіус дії;
- оптичний – найрозповсюдженіший спосіб відстеження, представлений спеціальною камерою, що відслідковує певний об'єкт та його рухи.

2. HMD (Head-Mounted Displays) або шоломи віртуальної реальності – пристрої, основною ціллю яких є надання користувачеві високого відчуття занурення. Принципом роботи пристрою є обман бінокулярної системи людського організму на ряду з слуховим сприйняттям. Як правило, HMD системи представляють собою:

- Шолом з двома дисплеями оптично настроєними для одного ока.
- Системи відстеження положення в просторі. Зазвичай це оптична система.

3. Інтерактивні рукавиці – пристрої для обрахунку положення кисті руки (та рук як похідної), а також положення пальців та відстеження рухів і жестів.

Засоби розробки програмного забезпечення для віртуального середовища. Перш ніж вирішити, яка система найкраще підходить для конкретної програми, необхідно визначити, які питання потрібно вирішити. Вони включають в себе знання того, які можливості потрібні для реалізації програми та як визначити, які системи відповідають цим вимогам. У цьому розділі порушується ряд питань, і в багатьох випадках присутнє те, що можна вважати найкращими відповідями. Звертаючи увагу на рішення проблем, стосовно використання різних архітектур - прослідковується, що кожне середовище «вирішує» проблему по-різному.

Існує три основні вимоги для розробки програмного забезпечення з використанням віртуального середовища:

Продуктивність - як зазначалось вище, - існує три основні вимоги до системи розробки віртуальної реальності, одна з них це – продуктивність. Для ефективного «занурення» у середовище віртуальної реальності потрібна висока частота кадрів (100 Гц) і низька латентність. Погана продуктивність - це не просто незручність для кінцевого користувача; це може викликати неприємні побічні ефекти, включаючи дезорієнтацію та хворобу руху. Тому система віртуальної реальності повинна мати можливість використовувати усі наявні ресурси системи, такі як процесор та спеціальне графічне устаткування. Сама система повинна мати

якнайменше «перевантажень», дозволяючи апаратурі працювати з максимальною ефективністю.

Гнучкість - середовище розробки повинно вміти адаптуватися до багатьох конфігурацій обладнання та програмного забезпечення. Якщо середовище не може адаптуватися до нових конфігурацій, додатки будуть обмежені в можливостях. Середовище розробки повинно бути вбране з огляду на те що розробник не повинен переписувати додатки для кожної нової конфігурації. Крім того, архітектура самої системи не повинна обмежувати застосування додатків, які можуть бути розроблені разом з нею.

Простота використання - система розробки повинна бути легкою в налаштуванні, вивченні та використанні. Інтерфейси прикладного програмування (API) та / або мови, які використовуються для створення додатків, повинні бути чітко сплановані та приховувати якомога більшу частину складної внутрішньої системи додатку.

Ідеальне середовище розробки мусить задовольняти кожну з наведених вище вимог. Насправді вони часто конфліктують. Простий у використанні інтерфейс системи може обмежувати параметри розробника, приносячи в жертву гнучкість. Дуже гнучку систему може бути важко оптимізувати для продуктивності через кількість варіантів, представлених розробнику. Кожна з систем, що буде представлятись, вибрана з розрахунку на баланс усіх вимог.

До можливостей навколишнього середовища належать:

Крос-платформ розробка. Що трапиться, якщо у додатку є двадцять потенційних клієнтів, але десять з них використовують робочі станції UNIX, а інші віддають перевагу системам Windows? Якщо багатий інструментарій доступний на двох або більше платформах, тому важливо враховувати поточну та майбутню можливість адаптації програми. Добре розроблені інструменти повинні приховати специфічні для платформи деталі налаштування, для того щоб адаптація програми між різними платформами з тим самим набором інструментів потребувала мінімальну кількість змін до самої програми. Для інструментів, які використовують власні формати даних та мови сценаріїв, часто трапляється, що

ніяких змін взагалі не потрібно. З іншого боку, інструменти, де розробник пише власний код на такій мові, як C ++, взагалі не помічають проблем з переходом між різними платформами. Вдало вибрана мова програмування повинна полегшити подальше використання додатку.

Підтримка устаткування для відтворення віртуальної реальності - для використання якого устаткування розроблена програма? Чи достатньо для використання просто джойстика або magic wand, чи рукавиці для керування? Чи зможе продукт використовуватися з проекційними системами, чи HMD, або з ними обома сторонами? Великі інвестиції вкладаються в низку апаратних засобів, тому важливо переконатися, що програма зможе підтримувати більшу їх частину. Можна також працювати навпаки - спочатку спроектуйте програму, а потім придбайте обладнання, яке найкраще підходить для неї. На додаток до першочергових потреб, слід розглянути майбутні плани - чи буде найближчим часом доступна рукавичка, magic wand або якийсь інакший вихідний пристрій? Якщо так, то, можливо, вони повинні бути додані до списку апаратних пристроїв, які повинні підтримуватись системою.

Апаратна абстракція - підтримка необхідного обладнання є обов'язковою, але настільки ж важливим є те, наскільки добре інструментарій абстрагує деталі апаратних інтерфейсів. Чи мають пристрої того самого типу однаковий інтерфейс, або існують конкретні API для кожного з них? Це відбувається при заміні обладнання. Наприклад: якщо програма використовує систему відстеження A, але нова, краща система відстеження - B стає доступною, чи буде потрібно змінити програму для її використання? Програма мусить мати можливість підтримувати нове устаткування за допомогою зміни сценарію або файлу конфігурації, не вимагаючи повторного кодування. Добре продумана апаратна абстракція дуже важлива. Незважаючи на те, що менш загальний інтерфейс може краще використовувати всі переваги унікальних функцій певного пристрою, загальний інтерфейс робить програму більш гнучкою та простішою для оновлення, підтримки та розуміння. Хоча важливі зміни, такі як заміна джойстика на рукавицю, можуть потребувати переосмислення інтерфейсу користувача, менші

зміни, такі як переключення однієї системи відстеження на іншу або зміну між моделями HMD, не повинні вимагати змін у самому додатку віртуальної реальності.

Локальний розподіл навантаження - додатки покликані підвищити продуктивність, розділяючи навантаження між декількома комп'ютерами в тій самій мережі. Наприклад, для створення двох зображень, які використовують HMD, можуть бути використані дві робочі станції. Деякі набори інструментів, що розроблюються в ході даної наукової роботи, мають вбудовану підтримку для розподілу навантаження, а деякі - ні. Розподіл навантаження на локальні станції в умовах роботи системи як одного цілого - може бути не тривіальною задачею, тому що алгоритми можуть бути як цілком прозорими, так і можуть вимагати спеціального розгляду низки фактів, яка то інформація що повинна бути спільною і коли, як її передавати, як її збирати. Розподіл навантаження одного додатку на декілька робочих станцій має ряд переваг, починаючи від збільшення частоти кадрів програми, так і збільшення кількості пристроїв вводу або каналів відображення, доступних для програми, або використання додаткових комп'ютерів для моделювання та інших обчислювальних завдань. У розробці нашого ПЗ однозначно вибрано використання системи локального розподілу навантаження та обрахункових потужностей.

Розподіл середовища – з релізом програми актуальним стане питання про підключення до системи не тільки з декількох машин в одному місці. Використовуючи технологію розподіленого середовища ідея розширюється до підключення машин - і користувачів - на віддалених сайтах у мережі. Кінцевий продукт з такою властивістю відкриває можливість залучити людей до співпраці у віртуальному світі, або замовникам демонструвати роботу клієнту безпосередньо у віртуальному світі без необхідності зустрічі «на місці». Одним із прикладів такої системи є компонент dVISE, розроблений для того, щоб дозволити кільком користувачам зустрітися разом для вивчення та огляду дизайну продукту. Деякі інструментарії пропонують можливості такого роду мереж, хоча рівень підтримки змінюється. Основні проблеми включають контроль взаємодії з кількома

користувачами та затримку змінної мережі. Наскільки корисною ця функція є, насправді, залежить від програми та середовища. Для централізованої організації це може бути зайвим. Для багатонаціональної корпорації це може бути цінним інструментом, що зменшує витрати на подорожі та зв'язок.

Швидке прототипування - оскільки більшість розробників матимуть доступ лише до одного або двох пристроїв підтримки віртуального середовища, важливо мати можливість запускати додаток та взаємодіяти з системою, не використовуючи все спеціальне обладнання. Інакше розробники будуть витрачати багато часу, чекаючи на свою чергу на обладнання. Щоб дозволити розробникам швидко прототипувати додатки, система розробки повинна включати в себе середовище, яке імітує функціональність фактичного обладнання для відтворення віртуального середовища. Це зазвичай передбачає малювання дисплея у вікні монітора та використання клавіатури та миші, щоб імітувати відстеження голови та інші пристрої введення. Незважаючи на те, що підтримка такого роду низькоапаратного встановлення є досить універсальною, велика кількість симуляції вхідних даних та близькість, з якою вона моделює фактичне обладнання, коливається. Тренажери для складних пристроїв вводу, наприклад, рукавичок або костюмів для відстеження тіла, можуть не підтримувати повний спектр входів фактичних пристроїв. Для швидкого розвитку важливі ще кілька факторів, включаючи рівень підтримки відстеження поведінки об'єктів та наявність простих мов сценаріїв для керування віртуальним світом.

Гнучкість під час виконання - важливо мати можливість змінювати апаратні конфігурації на льоту без перезапуску самої програми. Іноді пристрої виходять з ладу, їх потрібно перезавантажувати або замінювати, а іноді трапляється так, що початкова конфігурація не те, що очікує користувач. Деякі аспекти конфігурації можуть вимагати довгих випробувань, наприклад, пошук оптимального розподілу процесів та ресурсів на декількох машинах. Іноді зручно швидко переключатися між двома пристроями, можливо, між двома системами відстеження, щоб побачити, що працює краще. Програми VR зазвичай забирають багато часу для запуску: процеси повинні бути створені та синхронізовані, ініціалізовані апаратні

засоби, завантажені бази даних тощо. Перезапуск програми кожного разу, коли потрібні невеликі зміни або потрібно перезапустити пристрій, може зайняти багато часу. Все, що потрібно, це підтримка зі сторони системи: API для заміни пристроїв під час виконання, а також абстракція, яка примушує програму помітити, що певні налаштування змінюються. Варто зазначити, що більшість нехтує розробкою такої системи, та як результат витрачають більше чашу на етапі тестування та відладки продукту.

Передумови розробки інтерфейсів, інструментів та використання мов [17]:

- Високорівневі та низькорівневі інтерфейси.

Кожне із середовищ розробки віртуальної реальності, що описується, надає розробнику інший інтерфейс для створення додатків. Деякі з них забезпечують дуже високий рівень перегляду, де додатки можуть бути створені за допомогою власних мов сценаріїв та графічних інструментів, і сама система несе більшу частину відповідальності за розрахунки, геометрію та взаємодію. Інші плавають трохи вище рівня апаратного забезпечення, використовуючи відомі графічні API та мови програмування, щоб забезпечити максимальну продуктивність та гнучкість. Часто інструменти вищого рівня дозволяють швидше розвивати роботу з більш низькою кривою навчання. Інша сторона аргументу: «Якщо ви хочете, щоб щось будо зроблено правильно, робіть це самостійно». Чим більше одна з цих систем готова все робити сама, тим більше імовірність того, що вона зробить щось небажане, або зробить це в спосіб, який не є оптимальним для конкретної програми. Тому виникає постійна потреба у використанні середовищ низького рівня, а також їх більш привабливих аналогів. Ключ, звичайно, полягає у знанні того, що правильно для рішення кожної із задач.

- Графічні інтерфейси

Середовище розробки віртуальної реальності радикально відрізняється від стандартних середовищ обробки графіки. Деякі середовища підтримують завантаження моделей, створених в самих різних зовнішніх редакторах та пакунках автоматизованого проектування (CAD), а інші надають розробникам певний API для створення моделі. Деякі з них дозволяють користувачеві

маніпулювати моделями на рівні полігону, тоді як інші дозволяють завантажувати та переглядати моделі. Деякі підтримують архітектури графів сцени, інші вимагають, щоб всі графічні процедури, такі як візуалізація та виявлення зіткнень, були написані вручну. Так як програма є інструментом для дизайнерів, розробникам та дизайнерам буде необхідно доступ до рівня полігону, щоб мати змогу редагувати моделі повністю. На даному етапі важливо повернутися до питання віртуального середовища, якщо для програми потрібний рівень графічного інтерфейсу, який не підтримує існуюче віртуальне середовище, отримання потрібного ефекту може бути дуже болісним або навіть неможливим. Важливо також розглянути, який інтерфейс API підтримує бібліотека. Чи є він власним користувальницьким API для визначення графічних елементів або чи використовує він загальний API, такий як OpenGL або Direct3D. Популярні графічні інтерфейси мають кілька переваг. Там, як правило, є велика кількість документації з різних джерел.

- Взаємодія

Як середовище обробляє деталі взаємодії користувача та програми? У деяких середовищах розробник створює оброблювачі подій для вирішення змін у середовищі. Подія може бути будь-чим, наприклад: «Користувач схопив цей об’єкт» або «Ці два об’єкти, що просто зіткнулися» або «Доступні нові дані трекеру голови». Крім того, розробнику, можливо, доведеться написати код, який буде проводити опитування поточного стану пристроїв введення і вирішувати які слід ініціювати взаємодії у віртуальному світі. Як правило, «високорівневі» середовища з власними мовами моделювання або використання CAD, або подібних джерел для об’єктів матимуть більший діапазон можливостей взаємодії. Такі середовища, як правило, мають свої власні процедури виявлення зіткнень, і навіть можуть мати вбудовану підтримку фізики об’єктів (так що випадаючі елементи падають та відбиваються від віртуальної підлоги тощо). Середовища, які дають розробнику можливість створювати власну графіку в OpenGL [15] або іншому API, часто дають їй величезну відповідальність за обробку взаємодій між користувачами та об’єктами.

- API та мови програмування

Інша річ, яку слід враховувати при виборі середовища розробки VR, полягає в тому, як розробники дійсно створюватимуть додатки. Чи будуть вони використовувати користувальницькі інтерфейси, користатися спеціалізованими мовами сценаріїв чи якоюсь існуючою мовою, такою як C ++, Java або Scheme? Користувацькі мови часто мають спеціальні функції або структури, що робить їх особливо привабливими для виконання тих чи інших завдань. Ці мови можуть мати довгострокові переваги, і вони можуть допомогти швидкому прототипуванню, але можуть бути короткочасні та в перспективі стати тягарем. Врешті-решт, кожна нова мова має свою (можливо, круту) криву навчання та свої власні вимоги до навчання. Крім того, деякі розробники не люблять вивчати нові мови і, швидше за все, дотримуються того, що вони добре знають. Власні мови також мають недолік, що вимагає перезапису будь-якого коду, який вже написано для інших проектів. Якщо уже існує симулятор освітлення, і є тисячі рядків коду в FORTRAN [16], які контролюють моделювання відбиття світла, варто переконатися, що нове середовище розробки дозволить використовувати поточний код. Якщо середовище використовує існуючу мову, крива навчання звичайно не настільки складна. Проте можуть бути інші проблеми, коли компілятор стає частиною середовища розробки віртуальної реальності. Наприклад, коди C та C ++ можуть мати проблеми переносимості між платформами. За допомогою власних мов сценаріїв ці проблеми зміщуються від розробника додатків і до розробників середовища віртуальної реальності.

Інші фактори:

- Можливість масштабування

Бібліотека програмного забезпечення віртуальної реальності повинна бути легко поширюваною на нові пристрої, нові платформи та нові віртуальні-інтерфейси. Галузь віртуальної реальності постійно змінюється. Бібліотека програмного забезпечення віртуальної реальності повинна вміти легко адаптуватися до змін, щоб не стати застарілим непотребом.

- Мінімальні обмеження

Хоча простота це цінність, програмне забезпечення не повинно обмежувати просунутих користувачів від того, що їм потрібно. Середовище програми не повинно мати надмірно обмежувальної структури, а також не повинно стати непроникною перешкодою між розробником та комп'ютерною системою - має бути спосіб вийти за межі середовища та безпосередньо отримувати доступ до операційної системи або апаратного забезпечення, коли це необхідно. В ідеалі, не повинно бути ніяких обмежень, які кваліфікований користувач не може обійти.

- Моніторинг продуктивності

Програмна система віртуальної реальності повинна мати можливість збирати дані про продуктивність. Ця інформація необхідна для оптимізації конфігурацій програмного та апаратного забезпечення, а також для пошуку вузьких місць. З огляду на зібрані дані про ефективність, користувач може змінити систему або змінити програму, щоб максимально підвищити продуктивність.

- Джерела комерційного та дослідницького характеру

Деякі системи, що описуються в даній науковій роботі, є комерційними продуктами, тоді як інші вийшли з дослідницьких або відкритих середовищ. Обидва джерела мають свої переваги. Більшість дослідницьких систем доступні без будь-яких витрат, що має певну фінансову привабливість. У багатьох випадках вихідний код дослідницьких або відкритих середовищ також доступний, дозволяючи розробникам повністю контролювати систему. З іншого боку, комерційні системи можуть бути більш зручними для вирішення певних задач. Вони, як правило, стабільніші та портативні, і їх було випробувано в більшій кількості різних середовищ. Комерційні системи, як правило, краще документовані.

2.2. Аналіз технологій віртуальної реальності для створення клієнтської частини програмного забезпечення

Iris Performer - це високопродуктивний графічний API для машин Silicon Graphics Inc. (SGI). Він орієнтований на ринок візуального моделювання в режимі

реального часу, але може бути використаний для створення дуже високоефективних додатків. Якщо потрібно максимальну перевагу графічної продуктивності з машини SGI, то Iris Performer, безумовно, варто розглянути як рендеринг або як основу для власного рішення віртуальної реальності.

Перш за все, слід зазначити, що Iris не призначений для створення середовища розробки віртуального світу. Iris націлений на створення високоефективних програм візуального моделювання. Iris - графічна бібліотека на базі C / C ++, розроблена передовим графічним підрозділом SGI. Таким чином, Iris розроблений, щоб максимізувати продуктивність графічних додатків на високотехнологічних машинах SGI. Це дозволяє розробникам SGI досягти максимальної продуктивності, практично не втручаючись з боку користувача для багатьох простих програм. Через здатність Performer досягти пікової продуктивності на машинах SGI, вона зазвичай використовується в якості основи для користувацьких бібліотек віртуальної реальності. Наприклад, він використовується в Avocado, Lightning та CAVE.

Iris - API на основі графічного сценарію. Графік сцени містить повне уявлення про всі об’єкти у віртуальному світі. Це означає, що всі геометричні дані всередині сцени побудовані з об’єктів вузла. Iris надає можливість використовувати широкий спектр об’єктів вузлів, які можуть бути використані для створення програми. Підключення цих вузлів до спрямованого ациклічного графа, структури даних, що відбиває вузли та їх відносини один з одним, формує граф сцени. Iris має основні вузли сцени-графи, такі як вузли перетворення, але також підтримує більш складні вузли. Наприклад: вузли для підтримки рівня деталізації (LOD), анімації та морфінгу.

Щоб імпортувати геометричні дані до програми, Iris надає велику кількість завантажувачів баз даних, щоб користувачі могли завантажити свої дані. Iris забезпечує завантаження понад тридцять форматів бази даних. Завантажувачі динамічно залучаються, якщо потрібно перетворити вхідний формат файлу у структуру графічного інтерфейсу Iris. Після імпорту даних та побудови

внутрішньої сцени, розробники можуть маніпулювати всіма аспектами геометричних даних за допомогою API Iris.

Iris надає розробникам повний контроль над сценарними графами та геометрією, що містяться в них. Можна маніпулювати даними до рівня вершин і багатокутників. Виконавець також дозволяє додавати функції зворотного виклику користувача до графа сцени. Це можна використовувати для того, щоб створювати спеціальні рендерингові процедури, написані в OpenGL, у графі сцени, щоб створити спеціальні ефекти візуалізації. Комбінація вузлів геометрії та функцій зворотного виклику дозволяє розробникам створювати будь-який тип графічних ефектів, які є необхідними. Iris не накладає ніяких графічних обмежень розробникам.

Iris має в своєму розпорядженні високопродуктивний рендеринговий процесор, розроблений для отримання максимальної продуктивності графіки для всієї лінії SGI-графічної архітектури. Iris використовує кілька стратегій для оптимізації продуктивності візуалізації. Перш за все, більшість візуалізаційних циклів спеціально налаштовані, щоб оптимально відправити графічні команди на апаратне забезпечення рендеринга. Ці процедури налаштовуються вручну для забезпечення максимальної пропускної спроможності. Крім того, системи управління центральним процесором відстежують поточний стан рендеринга, щоб звести до мінімуму кількість дорогих змін, необхідних для графічного обладнання.

Основною перевагою Iris для користувачів віртуальної реальності - це його здатність автоматично виконувати багатопроцесорну обробку. Iris використовує конвеєрну багатопроцесорну модель для виконання додатків. Основний рендеринговий канал складається з етапу завантаження, етапу аналізу та етапу рендерингу. Стадія завантаження оновлює графік сцени та зазвичай виконує будь-який користувацький код. Шаблон визначає, які частини сцени видно. Потім етап аналізу надає лише геометрію, яка пройшла через стадію завантаження. Програми можуть мати кілька рендерингових каналів, спрямованих на декілька контурів апаратного забезпечення графіки. Iris автоматично виконує всі етапи рендеринга

каналів у багатопроцесорній манері, хоча є можливість ручного налаштування, коли користувач може дати Iris різного ступеня налаштування при виборі способу розподілу системних ресурсів. Користувач може призначити Iris використовувати загальний метод розподілу процесу або взяти безпосередній контроль над розподілом ресурсів.

Окрім багатопроцесорного рендерингу, Iris також забезпечує додаткові асинхронні етапи. Він забезпечує етап перетину, який може бути використаний для виявлення зіткнень, обчислювальної стадії, яка може бути використана для загальних обчислень, та етап бази даних (dbase) для обробки пейджингової бази даних. Всі ці функції є прозорі для користувача, оскільки Iris внутрішньо обробляє такі питання, як синхронізація, виключення даних та узгодженість.

Iris надає розробникам інструменти для створення додатків у реальному часі. Iris базується на системі REACT SGI, яка дозволяє розробникам точно налаштовувати пріоритети процесу та планування процесорів для критичних до ємкості часу додатків. Iris використовує цей тонкий контроль над плануванням процесора та пріоритетом процесу, щоб дозволити створювати додатки в реальному часі. Користувацькі додатки, написані в «Iris», також можуть скористатися перевагами «REACT» для забезпечення того, щоб користувацькі процеси отримували ресурси, необхідні для забезпечення продуктивності в реальному часі.

Iris має можливість підтримувати постійну частоту кадрів, тоді як вміст сюжету та складність змінюються. Iris підтримує відповідні частоти кадрів, видаливши частини сцени, які не видно. Iris також використовує вузли LOD в графічному сценарії, щоб вибрати різну складність моделей для візуалізації. Це дозволяє відображати менш складні версії об’єкта, який буде відображатися, коли глядач буде знаходитись на значній відстані від об’єкта. Ці методи зменшують кількість геометрії, яку потрібно надіслати на графічне обладнання. Інший інструмент, який може використовувати Iris - це динамічна роздільна здатність відео (DVR). DVR - це функція деяких розширених графічних архітектур SGI, що дозволяє системі динамічно змінювати розмір області візуалізації у буфері кадрів.

Ця область потім масштабується відповідно до графічного вікна, яке бачить користувач. Використовуючи DVR, додатки Performer можуть зменшити їх вимоги до ресурсів.

Iris має підходи до керування вікнами, які дозволяють розробникам використовувати передові технології візуалізації обладнання SGI. Iris дозволяє використовувати кілька графічних конвеєрів, декілька вікон на конвеєр, кілька каналів відображення в одному вікні та динамічну роздільну здатність відео. Ці функції дозволяють програмам використовувати всі можливості базового апаратного забезпечення. Ці функції є ключовими можливостями, необхідними при роботі з системами віртуальної реальності, такими як CAVE.

Iris включає в себе можливість збирати статистику по всіх частинах програми. Потім ці дані можуть бути використані для пошуку вузьких місць застосунку та настроювання програми для максимальної продуктивності. Наприклад, можна точно визначити точний час всіх етапів каналу, щоб визначити, який розділ програми займає найбільше часу. Iris також відстежує багато статистики малюнків, які можуть допомогти розробникам налаштовувати програми. Iris відстежує такі параметри, як кількість змін графічного стану, кількість перетворень, кількість відтінених трикутників, розмір трикутників та інші.

Припустимо, наприклад, що програма має частоту кадрів 24 Гц, але цільова частота кадрів 48 Гц. Вважається, що повинна бути можливість підтримувати більш високу частоту кадрів, але користувач не знає, що призводить до сповільнення. Переглядаючи статистику Iris, можна визначити, яка стадія трубопроводу викликає сповільнення. Після визначення того, яка стадія повільна, можна дізнатись, чи це код користувача або внутрішній код виконавця, який займає додатковий час. Статистичні дані виконавців дозволяють розробникам швидко переміщатись на область відхиленої програми. Через обмеження в реальному часі застосунків віртуальної реальності такі можливості, як підтримка максимальної частоти кадрів для додатків віртуальної реальності, необхідні.

Значним мінусом є те що Iris Performer доступний для використання лише на машинах SGI, а також той факт що Iris Performer не має прямої підтримки для пристроїв введення, крім миші та клавіатури. Також не підтримує невізуальний сенсорний вихід, як звук.

Alice - швидка система прототипування для створення інтерактивних комп'ютерних графічних додатків. Alice розроблена як інструмент, який дозволяє людям, які не мають технічної підготовки, створювати додатки для віртуальної реальності.

Система Alice призначена для швидкого створення прототипів інтерактивних графічних додатків. Розробка програмного забезпечення VR зазвичай складається з багатьох запитань «Що, якщо». «Що робити, якщо ми масштабуємо модель?», «Що робити, якщо ми обертаємось швидше?», «Що робити, якщо ми переходимо через середовище, використовуючи цей новий шлях?» - це всі приклади питань, які зазвичай потребують перекодування та перекомпіляції. Швидкі системи прототипування, такі як Alice, дозволяють швидко оцінювати «що, якщо». Вносячи незначні зміни в сценарій, можна випробувати багато ідей та варіантів протягом дуже короткого часу. Швидке створення прототипів може значно скоротити час розробки додатків віртуальної реальності.

Окрім швидкого прототипування, Alice призначена надавати користувачам, які не є технічними працівниками, можливості писати програми з підтримкою віртуальної реальності. Це означає, що інтерфейс та мова розробки повинні бути простими та легко вчитися, а також бути такими, що легко зрозуміти. З огляду на це, розробники Alice використовують модифікований Python як мову для написання сценаріїв Alice. Python - це високорівнева, інтерпретована, об'єктно-орієнтована мова.

```
...
WorldCamera.PointAt(Walls[i], EachFrame)
GoWallsBuildAnim = DoInOrder {
        self.T.RBA.Move(Forward, Duration = 1, Concurently)
```

```
        self.B.RBA.Move(Forward, Duration = 1, Concurently)
        self.SetHeight(50, Duration = 1)
}
for wall in Walls :
        wall.Top.LeftTopAnch.SetPosition2D(0,0)
        wall.Bot.LeftTopAnch.SetPosition2D(0,0)
        wall.Top.RightBotAnch.SetPosition2D(50,0)
        wall.Bot.RighBotAnch.SetPosition2D(50,0)
        wall.GoWallsBuildAnim()
...
```

Як видно з прикладу сценаріїв, мова скриптів дуже читабельна. Просто, дивлячись на сценарій, можна зрозуміти, що він робить. Поєднуючи легко зрозумілу мову сценаріїв з простим графічним середовищем розробки користувальницького інтерфейсу (GUI), новачки можуть легко писати сценарії, які будуть працювати в Alice.

Alice організовує світ як ієрархічну колекцію об’єктів. Відносини родич / дитина в ієрархії можуть бути легко змінені під час виконання. Alice надає розробникам можливість легко переключатися між системами координат об’єкта. Будь-який об’єкт можна позиціонувати на основі локальної системи координат іншого об’єкта. Можливість перемикати координатні системи дає розробнику програми велику гнучкість, оскільки об’єктні перетворення можуть бути задані відносно будь-яких інших об’єктів у ієрархії.

Щоб підтримувати високу частоту кадрів, система Alice відокремлює симуляцію від рендеринга. Alice відокремлює процес обчислення програми від процесу рендеринга програми. Перший процес обчислює стан симуляції, а другий процес зберігає геометричні дані і перетворює їх у поточну сцену. Розділення дозволяє виконувати процеси рендеринга якомога швидше, оскільки не потрібно чекати завершення розрахунків моделювання. Слід зазначити, що поділ обробки цілком прозорий для програміста. Програміст пише послідовний додаток з одним потоком, а система Alice сама піклується про деталі багатопроцесорної обробки.

Переваги Alice:

- *Швидке створення прототипів:* Alice розроблена з урахуванням можливості швидкого створення прототипів. Це дозволяє зробити прототипування легким і потужним. Інтерпретована скриптова мова дозволяє легко випробувати багато сценаріїв в короткий час. Навколишнє середовище розробки є чітким і робить розробку додатків дуже простою і лаконічною.
- *Легка у вивчені:* Alice призначена для нетехнічних користувачів. Через це продукт дуже простий в навчанні та використанні. Мова сценаріїв (Python) проста, але потужна. Середовище розробки графічного інтерфейсу є чітким і простим у використанні.

Недоліки Alice:

- *Додаткові пристрої віртуальної реальності:* для створення додатків віртуальної реальності з маніпуляторами потрібна додаткова система, що буде давати змогу використовувати пристрої для віртуальної реальності. Рідна система Alice для використання віртуальної реальності з маніпуляторами не є дуже гнучкою.
- *Обмеження щодо застосування:* Аліса розміщує обмеження на типи додатків віртуальної реальності, які можуть бути розроблені. Alice не дуже підходить для програм, які оброблюють велику кількість даних, і вимагають оновлення геометрії кожного кадру. Alice не дуже підходить для будь-яких програм, які потребують повного контролю над геометрією на рівні полігону та вершин.

dVISE та його компоненти забезпечують високий рівень середовища для розробки додатків віртуальної реальності. Він передбачає імпорт CAD-даних для створення віртуальних представлень для проектування та оцінки.

dVISE робить акцент на моделювання інтерактивного продукту (IPS), і це є ключовим впливом на внутрішню API та базове середовище виконання dVS. Метою IPS є створення віртуального представлення деякого продукту -

автомобіля, двигуна або будь-якого іншого виду машини чи об’єкта та дати можливість користувачеві взаємодіяти з ним.

dVISE представляє творця віртуального середовища з набором дуже високорівневих інтерфейсів. Зокрема, він використовує просту мову сценаріїв, щоб захистити розробника від необхідності виконувати фактичне програмування для створення об’єктів (званих «збірок») та визначення їх взаємодії.

Геометрія агрегатів найчастіше створюється за допомогою інструментів CAD. Це дозволяє розробнику використовувати будь-який інструмент, який він обирає, або імпортувати дані безпосередньо з CAD файлу іншої програми.

Система dVISE може імпортувати та перекладати багато форматів файлів таких програм як: Autodesk dxf, VRML, Inventor, MultiGen flt, obj objection Wavefront, STEP і власний формат Pro / Engineer. Інструменти перетворення також дозволяють розробнику додавати геометрії інформації про кольори, текстуру та освітлення.

Визначення об’єктів та взаємодії зберігаються в текстових файлах ASCII, написаних на простій скриптовій мові на основі подій. Визначення збірки може включати в себе посилання на його опис геометрії, інформацію про його положення та орієнтацію, визначення дитячих зборів та інструкції щодо обробки подій. Система dVS включає в себе модулі для обробки фізики об’єктів та виявлення зіткнень, і вони можуть, наприклад, надіслати об’єкт «подія» або «дотик». Ось невеликий приклад для вимикача світла [8]:

```
…
Assembly (Name=LightSwitch) {
Visual {
        Geometry { «LightSwitches/Switcher» }
}
Orientation { -10, 0, 0}
Event {
        Create {
                dvAssign («%state», «Off»);
```

```
}
Touch {
        dvCallElse («seq (%state, Off) «, *, On, *, Off);
    dvAssemblyEvent (light, toggle);
    }
    On {
    dvAssemblyOrientation (., 50, 0, 0);
        dvAssign («%state», «On»);
}
    Off {
    dvAssemblyOrientation (., -50, 0, 0);
    dvAssign («%state», «Off»);
    }
}
}
…
```

Цей компактний приклад коду визначає перемикач, його геометрія зберігається в «LightSwitches/Switcher» і з початковим обертанням вниз для відображення положення вимкнено. Потім оголошуються чотири функції обробки подій. «Створення» та «Дотик» - це стандартні події, створені під час виконання DVS, а «On» та «Off» - це спеціальні події, створені для цього об’єкта. Коли цей перемикач створюється, викликається обробник Create event, встановлюючи локальну змінну state на «Off». Експлуатація сенсорів викликається, коли користувач торкається перемикача, і виконує дві функції. По-перше, збірка звертається до своїх методів «On» або «Off» (залежно від значення стану змінної). Потім він посилає подію «перемикання» на збірку під назвою «light» (яка, імовірно, має обробник подій, визначений для користувацької події «toggle», точно так само, як для «On» та «Off»). Нарешті, обробники подій для «Увімкнути» та «Вимкнути» змінюють орієнтацію агрегату таким чином, щоб перемикач був перевернутий, а потім оновлює значення стану.

Перемикач світла - це лише простий приклад, і не демонструє повну потужність мови сценаріїв dVISE. Кілька особливостей, що представляють особливий інтерес, є:

- Можливість використовувати будь-яке визначення асоціації як шаблон для створення нових, злегка різних агрегатів.
- Можливість визначення обмежень на рух об’єктів.
- Можливість робити анімацію (за допомогою події «Tick»), а також можливість визначати основні кадри для анімації.

Переваги dVISE:

- *Інтерфейс сценаріїв високого рівня*: високорівневий інтерфейс dVISE може скоротити час розробки, а також захистить розробників від багатьох болісних деталей базової системи. Оскільки скрипти та імпортована геометрична інформація об’єкта зберігаються в незалежних від системи файлах, для перенесення програми на інші підтримувані платформи не потрібно перезапису чи перекомпіляції.
- *Моделювання продуктів*: можливість dVISE імпортувати дані з CAD-програм дуже потужна та гнучка. Особливий інтерес представляє здатність використовувати мережеві системи, що дозволяють групам працювати разом у змодельованому світі та здійснювати спільні огляди дизайну.
- *Портативність:* dVISE доступний на особливо великій кількості різних платформ.
- *Масштабованість графіки*. Однією з унікальних характеристик середовища виконання dVS є те, що на різних платформах може бути встановлений різний рівень потужності рендеринга. Наприклад, високотехнологічна робоча станція може використовувати текстуроване, зображене за технологією Phong-shaded, аналізоване відтворення, тоді як ПК може просто відображати згладжування багатокутників.

Недоліками dVISE є:

- *Користувальницька мова:* мова сценаріїв dVISE, хоча вона досить проста і добре підходить для своїх цілей, є ще однією мовою з іншим набором команд і правил синтаксису для розробника.
- *Вузька спеціалізованість:* завдяки сильному акцентуванню dVISE на симуляції продукту, він може бути неприйнятним для дуже різних завдань, таких як ті, для яких завантаження фіксованих моделей не є корисним.

Avocado - це програмне забезпечення віртуальної реальності, розроблене GMD. Основна мета бібліотеки полягає в інтеграції широкого спектру пристроїв віртуальної реальності, що використовуються в GMD. Система також призначена для створення прототипів для швидкого розроблення та тестування додатків. Крім того, Avocado підтримує розробку розподілених програм.

Структура сценарію Avocado базується на виконавці Iris. В результаті використання виконавця, авокадо може працювати тільки на машинах SGI. Щоб повністю відобразити віртуальний світ, Avocado має поширювати структуру графіків виконавця. Виконавець визначає лише візуальні характеристики навколишнього середовища. Avocado розширює графічні вузли виконавця для створення об’єктів графіка авангарду сцени. Ці нові об’єкти мають додаткові можливості. Не кожен вузол Avocado повинен базуватися на відповідному вузлі Iris. Наприклад, виконавець не має підтримки звуку, Avocado розширює графік сцени, щоб дозволити звукові вузли.

Avocado використовує об’єктно-орієнтовану структуру діаграм сцени для представлення віртуального світу. Графік сцени є спрямованим ациклічним графіком, структурою даних, що відображає вузли та їх відносини один з одним. Все в світі представлено як об’єкти вузлів, стан яких маніпулюється. Представлення є повним представленням, тобто всі можливі сенсорні виводи представлені в одному графі сцені. Це важливо, оскільки це означає, що представлені не лише візуальні аспекти навколишнього середовища, а й слухові

та тактильні. Для того, щоб представити оточення користувачеві, кожен сенсорний канал має окремий візуалізатор.

Кожен об’єкт Avocado інкапсулює внутрішній стан у полях. Avocado визначає єдиний загальний інтерфейс для доступу до даних поля, так що всіма об’єктами можна маніпулювати однаковим способом. Це дозволяє реалізувати динамічний інтерфейс скриптів та завантаження нових об’єктів під час виконання програми.

Поля Avocado можуть бути пов’язані між собою, створюючи мережі передачі даних, тобто якщо поле A підключено з поля B, поле A отримує значення поля B кожного разу, коли змінюється поле B. Можливість взаємоз’єднання полів може усувати більшу частину навантаження на програму віртуальної реальності. Мережа передавання даних дозволяє вузлам мати свої атрибути «пов’язані» з атрибутами інших вузлів та об’єктів у системі. Ця здатність дозволяє Avocado дуже легко визначати складні моделі поведінки за допомогою пов’язаних мереж об’єктів.

Крім вузлів, Avocado надає ще два види об’єктів: датчики та послуги. Датчики містять дані поля, але не похідні від класів виконавця. Датчики використовуються для імпорту та експорту даних між Avocado та іншою частиною системи. Вони невидимі жодному сенсорному каналу, тому не є частиною графіки сцени. Датчики можуть бути використані для таких об’єктів, як вікна дисплея, дані пристрою тощо. Avocado також надає сервісні об’єкти. Сервісні об’єкти забезпечують API для системних функцій. Вони можуть бути використані для реалізації речей, таких як драйвери пристроїв. Об’єкти сенсора можуть використовувати об’єкти обслуговування пристроїв, щоб отримувати дані пристрою. Дані цього пристрою зберігаються в полях об’єкта датчика, де до них можуть посилатися вузли в графі сцені.

Додаток Avocado можна розглядати як сукупність груп вузлів, які інкапсулюють певну поведінку. Групи вузлів можна розглядати як інструменти, які розробники додатків мають у своєму розпорядженні. На додаток до груп вузлів, Avocado може бути розширено зовсім новими вузлами, датчиками та

службами для створення нових інструментів. Деякими прикладами інструментів, розроблених для Avocado, є вузли вибуху, вузли текстури відео, вузли збірки, вузли для перетягування та служби перетину. Легко створювати нові групи вузлів для створення нових інструментів.

Всі відповідні частини системи Avocado відображуються на мові сценаріїв Scheme. Мова сценаріїв також виключає необхідність перекомпілювати додаток, коли потрібні зміни. Це значно полегшує процес розробки, дозволяючи швидке створення прототипів.

Avocado підтримує розподілене середовище, прозоро розподіляючи всі вузли графіки сцени. Це робиться шляхом обміну даними між вузлами та між різними клієнтськими програмами, які переглядають спільне середовище. Створення та видалення об’єктів також прозоро розподіллено між усіма браузерами. Це дозволяє розробляти додатки, коли багато користувачів можуть здійснювати навігацію через єдине спільне віртуальне середовище. У багатьох бібліотеках VR, написання таких програм може бути важким, якщо не неможливо. Але через те, як Avocado використовує єдиний графічний сценарій для зберігання всього, бібліотека робить розподіл середовищ відносно простим. Взаємодія з користувачем може бути оброблена, підтримуючи вузли графіка сцени, які відповідають кожному користувачеві в середовищі.

Переваги Avocado:

- *Сценарії:* мова сценаріїв дозволяє швидке прототипування додатків.
- *Поля:* мережа передачі даних дозволяє створювати дуже потужні програми. Можливість мати датчики як частину мережі потоку даних значно спрощує обробку вводу
- *Розширюваність:* вузол, сенсор і об’єкти сервісу дуже простягаються. Оскільки кожен об’єкт має єдиний API, після створення нового класу дуже легко почати використовувати його в системі.

Недоліки Avocado:

- *Відсутня підтримка багатьох платформ*: адже авокадо заснований на Iris Performer, він працює тільки на платформах SGI
- *Scheme:* вибір Scheme як скриптової мови може бути перешкодою, оскільки Scheme не є добре відомим мовою. Крім того, функціональні мови можуть бути складними у вивчені.

2.3 Аналіз існуючих програмних рішень для дизайнерів з використанням VR технологій

Історично склалося так, що дизайнери виготовляли фізичні моделі та робили декілька двовимірних зображень, щоб передати свої ідеї клієнтам. Однак цей спосіб не має можливості передавати масштаб та давати почуття присутності. Дизайнери та клієнти повинні залучати уяву і знання дизайну, щоб мати змогу зрозуміти дизайнерську ідею оформленого простору. Використовуючи технологію віртуальної реальності в галузі дизайну, дизайнери можуть підняти візуалізацію дизайну на новий рівень. З розвитком апаратного та програмного забезпечення віртуальної реальності, багато компаній вже почати розробляти програмне забезпечення для дизайн індустрії, з метою створення іммерсних рішень для презентації дизайну проекту. Серед таких програмних забезпечень можна виділити наступні:

- WorldViz - піонер у галузі віртуальної реальності. WorldViz протягом десятиліття революціонізував взаємодію людей з віртуальною реальністю. Програмне забезпечення, що називається Vizard VR, дозволяє користувачам створювати модель віртуальної реальності зі складним симуляційним алгоритмом. WorldViz також пропонує складне рішення як з апаратним забезпеченням, так і з програмним забезпеченням, щоб відповідати вимогам дизайнерів для вивчення та взаємодії з віртуальною реальністю.

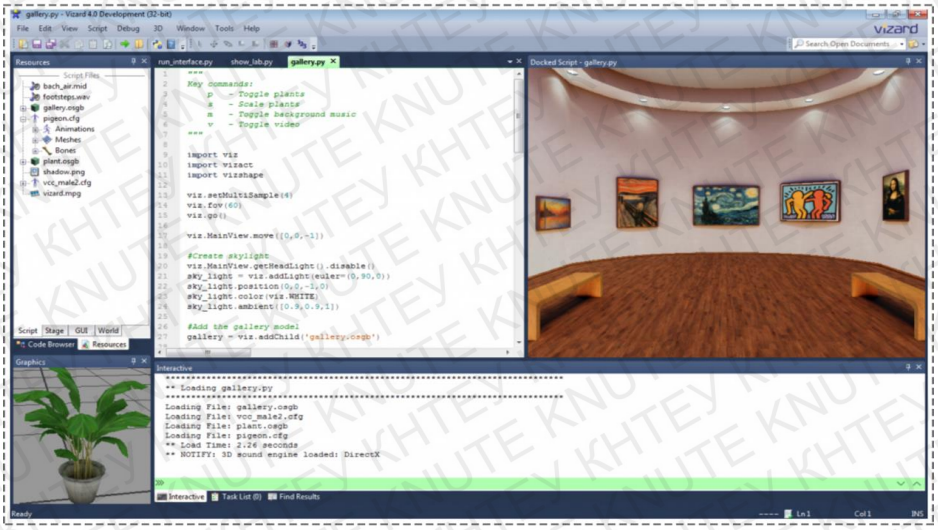

Рис. 2.1 Інтерфейс Vizard VR [46].

- IrisVR - в умовах представлення дизайнерських проектів, IrisVR дозволяє дизайнерам реалізувати поглиблену віртуальну візуалізацію проекту дизайну у віртуальній реальності на комп'ютері одним натиском миші. Команда IrisVR розробила програмне забезпечення для дизайнерів та архітекторів, що має змогу перетворювати 3D-моделі в потрібні файли для відтворення у віртуальній реальності. Для розробки програмного забезпечення на платформі IrisVR потрібен Oculus HMD.

Рис. 2.2 Процес редагування середовища в віртуальній реальності IrisVR [47].

- ArqVR - також забезпечує рішення відтворення віртуальної реальності, спрямоване на полегшення роботи дизайнерів чи архітекторів. Команда ArqVR пропонують до використання власну платформу для моделювання віртуальної реальності, що в свою чергу дозволяє використовувати (та вимагає використовувати) стороні гаджети, такі як Oculus HMD, ручні контролери, системи відстеження руху, спеціалізовані робочі станції. ArqVR допомагає створювати середовище віртуальної реальності на основі наданої моделі дизайну, а дизайнери можуть реалізовувати свої проекти на платформі ArqVR. ArqVR пропонує різні типи якості моделювання дизайну: від більш низької якості простору, задля швидкого прототипування проекту дизайну, до гіперреалістичного середовища з розвиненою системою освітлення, картографічними та персональними навігаційними об’єктами.

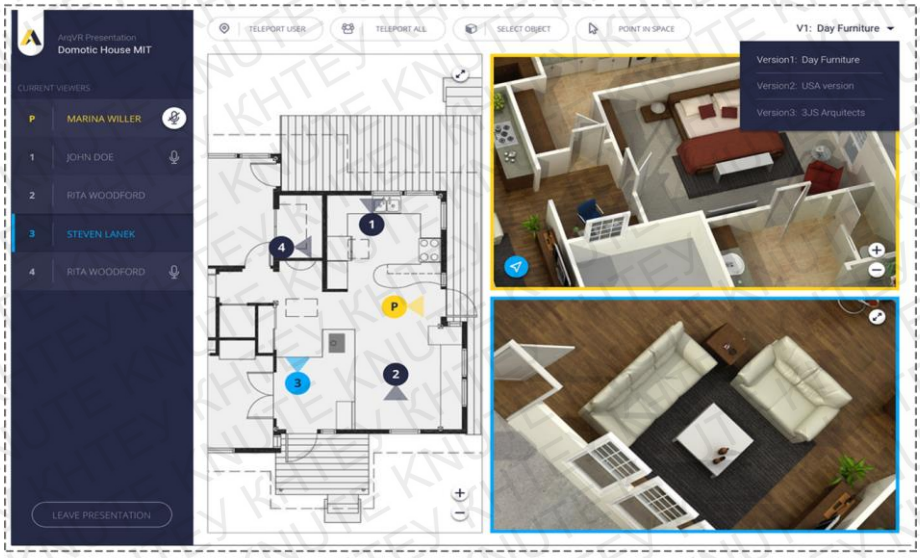

Рис. 2.3 Інтерфейс ArqVR [48].

- The Third Fate - у порівнянні з попередніми програмами, які в основному перетворюють цифрові моделі в реалістичне середовище, The Third Fate записує данні (360 градусів) реального середовища, що результує в відтворенні реальних місць, що практично можуть бути модифіковані у віртуальному середовищі та представлені

користувачам. Проте, на етапі модифікації середовища – якість відображень суттєво погіршується.

Рис. 2.4 Інтерфейс The Third Fate [49].

Висновки до розділу 2

Аналізуючи всі аспекти використання технології віртуальної реальності в програмному забезпеченні для дизайнерів виникла потреба детального систематичного огляду різних частин технології.

Першою частиною був аналіз устаткування необхідного для відображення та використання систем віртуальної реальності.

Другою частиною було визначення основних передумов для створення програмного забезпечення з використанням віртуальної реальності.

Третьою частино був аналіз можливих засобів які дають можливість розроблювати запропоноване програмне забезпечення. Були виділені слабкі та сильні стороно існуючих бібліотек.

Четвертою частиною був аналіз існуючих програмних рішень. Огляд існуючих програмних рішень був необхідний для підтвердження теорії запропонованого програмного забезпечення, а також, для визначення додаткових передумов створення віртуального середовища, з огляду на еволюцію в розрізі технічного прогресу.

РОЗДІЛ 3
МЕТОДОЛОГІЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Вибір схеми моделі життєвого циклу програмного забезпечення

Життєвий цикл розвитку програмного забезпечення, як продукту, починається з його створення і закінчується припиненням підтримки. Впродовж цього процесу програмне забезпечення проходить різні етапи. Коен (Cohen) визначає «вимоги, аналіз, дизайн, конструкцію (або кодування), тестування (перевірку), установку, експлуатацію, технічне обслуговування та припинення підтримки» як ключові компоненти процесу розробки. Висновком є те що, загалом, життєвий цикл складається з п’яти фаз: дослідження, вимоги користувачів, аналіз, дизайн, реалізація та випуск.

Загалом, життєвий цикл сприймається як часовий проміжок, який охоплює розробку нової системи до можливого припинення підтримки. Це процес, що починається з появи ідеї, проходить етап реалізації та закінчується припиненням, роблячи пріоритетними питання життєздатність та юзабіліті на проміжних етапах.

Схеми життєвого циклу будуються з конкретною метою, для конкретної організації. Через цю специфіку, кожен процес розробки системи вимагає стандартизованих методик для моніторингу прогресу розвитку на всіх етапах життєвого циклу.

Загалом можна визначити десять основних схем життєвого циклу програмного забезпечення:

Модель водоспаду – була вперше представлена Ройсом (Royce) в 1970 році, в контексті проектування програмного забезпечення для розробки космічних апаратів, і є одним з найпопулярніших методів оцінки еволюції продукту чи системи. По суті, це поетапний етап розробки життєвого циклу розробки системи, послідовного опису життєвого циклу продукту, який охоплює 5 різних стадій (рис. 3.1): аналіз, проектування програми / дизайн, кодування, тестування, реліз.

Ключовими положеннями моделі водопаду є: по перше - «розробка дизайну». Спочатку потрібно дозволити дизайнерам бути частиною процесу через

їх безпосередній вклад в розуміння наявних ресурсів для відображення та обмежень. Друга – «документація дизайну». Детальна документація процесу розробки є найважливішою, не лише для полегшення управління процесом, але й для полегшення оцінки ефективності, що в результаті робить процес виправлення помилок більш ефективним. Третій – «робити все двічі», посилаючись на те, що остаточна версія продукту, насправді. - друга версія, що пройшла всі етапи, і в якій було легше точно визначити сильні та слабкі сторони, щоб виправити помилки. Четверте - це «планування, контроль та моніторинг тестування». Тестування є фундаментальним етапом. Важливо залучати фахівців, які не брали участі на більш ранніх етапах процесу. Важливо також перевірити кожен аспект проекту незалежно від того, наскільки це актуально. Нарешті, п’яте, керівництво – «залучити замовника». Отримані відгуки замовника враховані під час процесу розробки значно підвищить потенціал для загального прийняття остаточної версії програмного забезпечення.

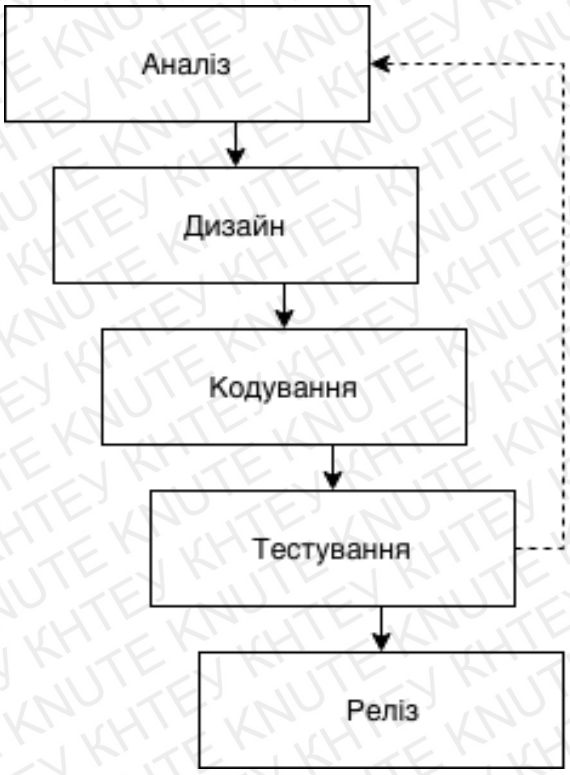

Рис. 3.1 Схематичне представлення моделі водоспаду. [авторська розробка]

Інкрементна модель - це особлива еволюція моделі водоспаду, яка намагається вирішувати її більш виражені недоліки, тобто повільність циклу. Що також спрямовано на визначення більш гнучкого процесу, який вимагає менш широкого планування [25].

Згідно з цим підходом, замість поділу моделі на статичні, ізольовані кроки, проектується цілий процес, протестований і реалізований на одну частку за раз, послідовними етапами, так що з кожним етапом (або збільшенням) стає можливим отримати відгук від клієнта. Цей відгук надасть цінну допомогу в наступному кроці і так далі. З кожним кроком продукт випробовується та покращується, відповідно до цілей та очікувань клієнта, що полегшує його можливий успіх [26].

Рис. 3.2 Схематичне представлення інкрементної моделі. [авторська розробка]

За допомогою цієї моделі процес розробки програмного забезпечення здійснюється за рахунок інкрементального випуску версій програмного забезпечення. Момент запуску продукту є першим інкрементом, що готовий до споживача, має мінімальний набір функціоналу. Тоді, відповідно до реакції клієнтів на програмне забезпечення, робляться нові інкрементальні версії покликані поліпшити продукт. Ланцюжок інкрементальних версій результує у кінцевому продукті.

Спіральна модель - датується кінцем 1980-х років, коли вона була представлена Баррі Бемом, і запропонувала те, що інші моделі не враховували - аналіз ризиків. По суті спіральна модель намагається об’єднати ключові аспекти деяких інших відомих моделей (а саме модель водоспад, та інкрементальна

модель), намагаючись зібрати найбільш відповідні риси з кожного, оскільки проекти можуть бути більш-менш адаптований до конкретних моделей [26].

Згідно з цією моделлю, процес розробки системи складається з ряду циклів або ітерацій. Кожен цикл починається з визначення цілей і вимог поточного етапу, а також аналізу альтернатив і обмежень. Цей процес покликаний висвітлювати сфери невизначеності (ризику), які будуть враховуватися під час наступного етапу. Цей процес передбачає постійне вдосконалення прототипу, оскільки ризики зменшуються (тоді як інші можуть виникнути). Після того, як прототип стає досить надійним і ризик зменшиться до прийнятного рівня, наступний етап розвивається відповідно до основного підходу водоспаду, через послідовність етапів: концепцію, вимоги, дизайн та впровадження. Після завершення цього циклу починається черговий цикл.

Бьєм (Boehm) стверджує, що кожен цикл або ітерація процесу незмінно відображатиме шість конкретних характеристик, які він назвав «інваріантами» [10].

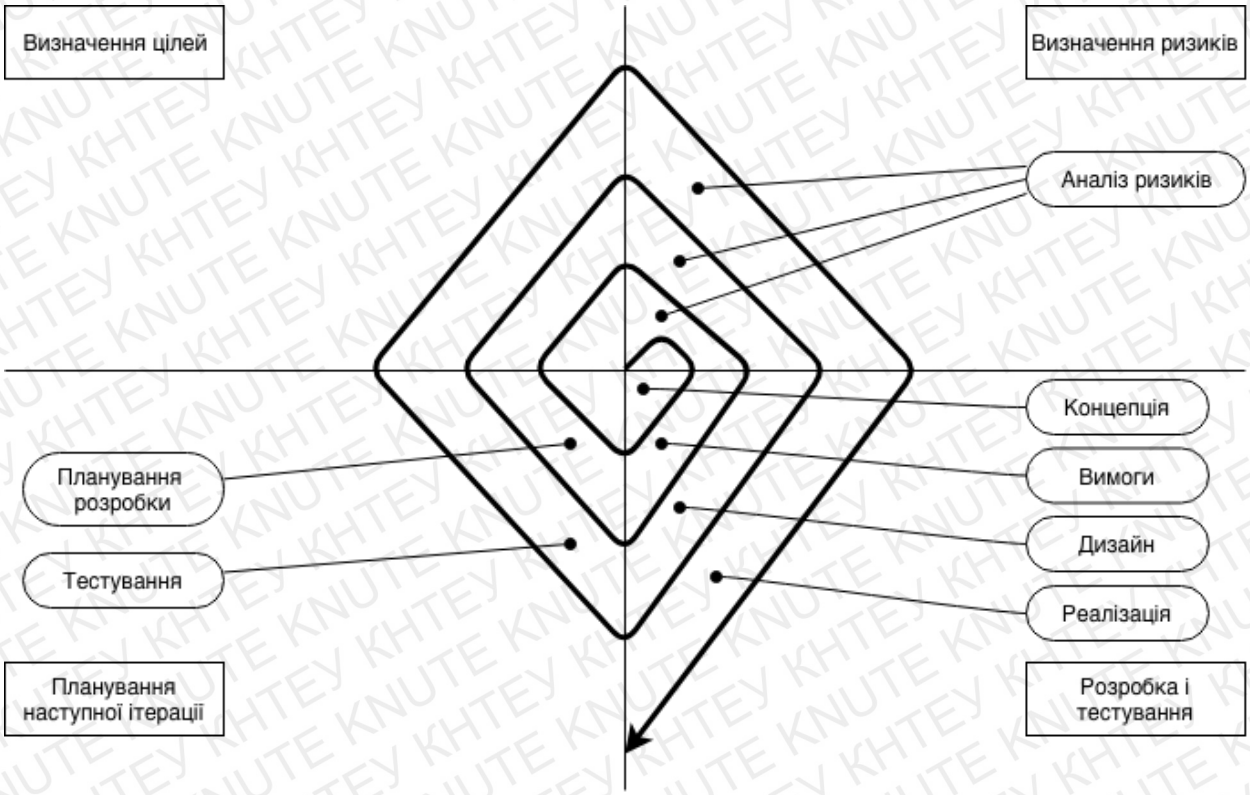

Рис. 3.3 Схематичне представлення спіральної моделі. [авторська розробка]

Інваріант 1 - це одночасне визначення ключових факторів, таких як ідея, вимоги, план, дизайн та кодування. Стверджується, що визначення цих факторів у послідовній структурі може стримувати проект надмірно жорсткими упередженнями.

Інваріант 2 полягає в тому, що кожен цикл відповідає чотирьом стратегічним принципам, які відповідають чотирьом квадратикам моделі: визначають цілі, оцінюють ризики, розробляють та тестують, а також планують наступну ітерацію. Порушення руху у відповідності до цієї базової стратегії може негативно вплинути на весь процес.

Інваріант 3 полягає в тому, що рівень зусиль визначається міркуваннями щодо ризику. Розумні часові рамки повинні бути встановлені для кожного проекту відповідно до оцінок ризику, щоб визначити «скільки достатньо деталей» у кожному завданні.

Інваріант 4 полягає в тому, що ступінь деталізації обумовлена міркуваннями щодо ризику. Так само, як інваріант 3, тут важливо визначити «скільки достатньо деталей» на кожному етапі процесу.

Інваріант 5 стосується використання основних етапів опорної точки, які Бьєм називає «Цілями життєвого циклу, архітектурою життєвого циклу та початковим операційним потенціалом» [27]. На кожному з опорних точок зацікавлені сторони переглянуть основні фактори етапу.

Інваріант 6 стверджує, що, крім аспектів кодування, процес розробки повинен також зосереджуватися на самому життєвому циклі. Це означає, що завжди слід враховувати довготривалі проблеми.

Модель – V – була вперше представлена в кінці 1980-х років Паулом Руком як варіація моделі водоспаду, яка намагалася підкреслити існуючу зв’язок між кожним етапом процесу розробки та його відповідним етапом випробувань. Зосереджуючись на цих відносинах, він гарантує, що належні вимірювання якості та тестування постійно відбуваються протягом усього життєвого циклу.

Таким чином, представлена модель полягає в тому, що кожен крок реалізується шляхом використання детальної документації з попереднього кроку.

За допомогою цієї документації продукт перевіряється та затверджується на кожному етапі процесу, перш ніж він може перейти до наступного етапу [28]. Завдяки постійному тестуванню та відповідній документації можна підвищити загальну ефективність процесу, насамперед тому, що можливі проблеми можуть бути виявлені та вирішені на ранніх етапах [29].

Рис. 3.4 Схематичне представлення моделі V. [авторська розробка]

V-Model базується з схожих положень з класичними моделями водоспадів. Проект йде від аналізу вимог і специфікацій, до архітектурного та детального дизайну, та нарешті до кодування. Однак замість того, щоб продовжувати цю низхідну, існує паралельна структура, яка рухається вгору від етапу кодування, даючи моделі його чітко виражену V форму. Схема що рухається вгору описує кожен етап тестування, який слідує за кодуванням, починаючи з юніт-тестами і закінчуючи прийомним тестуванням, останній етап до остаточного випуску.

Модель швидкої розробки додатків – вперше запропонована в 1970-х роках, модель швидкої розробки додатків була розроблена та формалізована Джеймсом Мартіном на початку 1990-х років. Модель зумовлена думкою, що існуючі моделі життєвого циклу занадто жорсткі, і не дозволяє швидко розвивати проект; отже, існує потреба в структурі, яка може забезпечити швидку розробку, зберігаючи при цьому високі стандарти якості. Джеймс Мартін ґрунтується на принципі, що

поетапні структуровані життєві цикли неминуче тягнуть за собою затримки та помилки, що викликає необхідність альтернативної методології

Модель швидкої розробки являє собою набір інструментів і керівних принципів, що полегшують короткочасну розробку, в межах заздалегідь визначеного терміну або «timebox». Деякі з цих інструментів та керівних принципів включають методи планування, моделювання даних та процесів, створення коду, тестування та налагодження.

Важливо зазначити, що як розробники, так і клієнти беруть участь у всіх етапах. Команди, як правило, невеликі, висококваліфіковані та високодисципліновані. Вони повинні гнучко адаптуватися до можливих змін вимог та відгуків від клієнтів. Тим не менш, важливо досягти належного балансу між гнучкістю та структурною стабільністю. Базові моделі дизайну продукту все ще є невід'ємними, але не як жорсткі покрокові посібники.

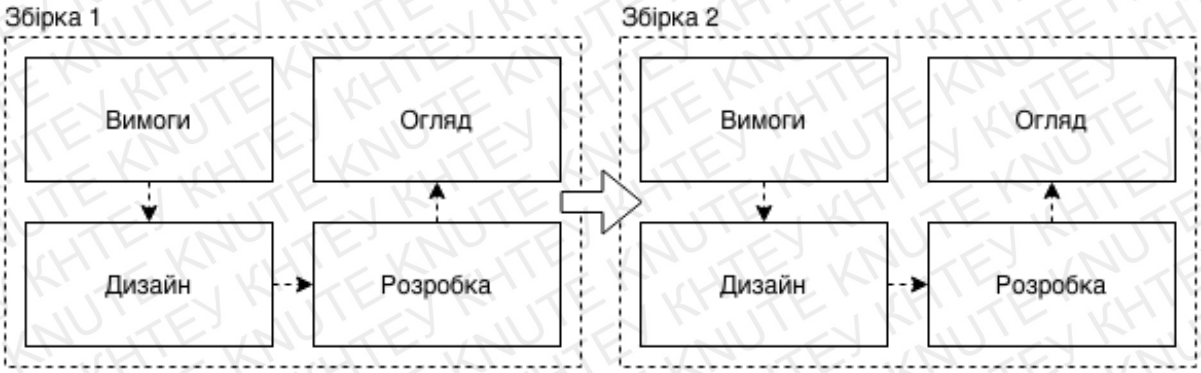

Рис. 3.5 Схематичне представлення моделі швидкої розробки. [авторська розробка]

Модель Agile - з популяризацією водоспадних моделей виник альтернативний підхід, який намагається протидіяти їх жорсткості та відсутності гнучкості. Вже згадувався вище такі приклади як модель швидкої розробки додатків. У 2001 році маніфестом для гнучкого розробки програмного забезпечення було представлено, в рамках нової спроби об'єднати найкращі риси інших моделей в одну структуру. З тих пір все більш популярними стали гнучкі методи розвитку.

Існує 12 принципів, які керують моделями гнучкого розвитку, описані в маніфесті Agile. Ці принципи можна підсумувати наступним чином: [30]

- задоволення потреб клієнтів є найвищим пріоритетом;
- зміна вимог вітається, це більше не перешкода;
- програмне забезпечення постачається регулярно в послідовних випусках;
- мотиваційні особи є ключовими для успішних проектів;
- особисті бесіди (не дистанційні) має першорядне значення для успішної співпраці;
- робоче програмне забезпечення - це показник прогресу проекту;
- слід заохочувати сталий розвиток;
- наголос на технічній та дизайнерській якості;
- простота повинна бути сприятливою;
- самоорганізовані команди є найкращою формою розробки проектів;
- необхідно проводити регулярні дискусії щодо вдосконалення команди.

Якщо врахувати різницю в термінах або описі етапів, можна визначити загальний шлях, який буде виконуватися в процесі гнучкого розвитку, описаний у чотири етапи.

Першим кроком є вибір і схвалення проекту. На цьому етапі команда, що складається з розробників, менеджерів та клієнтів, встановлює обсяг, цілі та вимоги до продукту. Існує також ретельний аналіз різних альтернатив для досягнення поставлених цілей, а також оцінка ризиків для кожної ідеї.

Другий крок - ініціювання проекту. Після створення узгодженого проекту з відповідними цілями та масштабами, створюється робоча група з відповідним середовищем та інструментами, а також робочою архітектурою, на якій буде заснована система. Це теж обговорюється серед усіх зацікавлених сторін. На цьому етапі також є адекватним встановлення робочих часових рамок і графіків.

Третій крок - це побудова ітерацій, причому кожна ітерація складається як з планування, так і з розробки. Розробники випускають робоче програмне забезпечення з послідовним інкрементом, яке буде відповідати еволюції вимог,

визначених різними зацікавленими сторонами. Тому тісна співпраця є фундаментальним аспектом цього процесу, як найефективніший спосіб забезпечення якості та чіткого визначення пріоритетів проекту. Широке тестування кожної ітерації також є найважливішим.

Четвертий і останній крок - вихід продукту. Цей етап охоплює два етапи: по-перше, завершується остаточне тестування всієї системи, а також будь-які необхідні остаточні переробки та документації. Далі продукт випускається, після чого користувачеві надаються тренінги для максимального оперативного інтеграції. Робоча група може підтримувати проект, щоб забезпечити поліпшення якості продукту, а також підтримки користувачів.

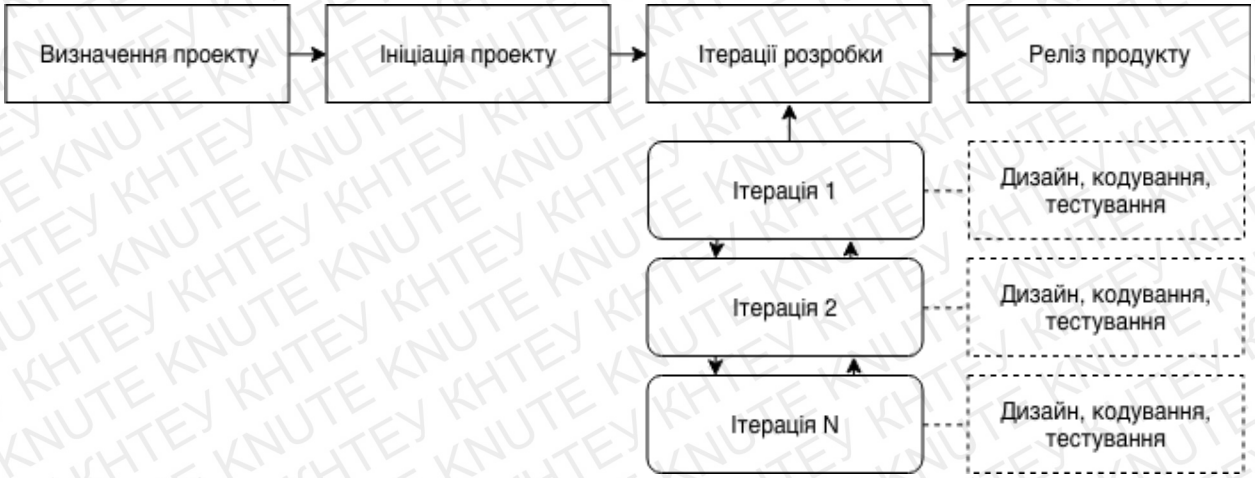

Рис. 3.6 Схематичне представлення моделі Agile. [авторська розробка]

Модель прототипів є ітераційною структурою, яка є центром багатьох більш гнучких підходів до розробки програмного забезпечення, починаючи з початку 1980-х років, що призвело до того, що в деяких дослідженнях це описується як конкретна модель сама по собі. У 1997 році Карр і Вернер відзначають, що модель прототипування, виявилися більш динамічно та більш чутливою до потреб клієнтів, а також менш ризикованою та більш ефективною.

Модель прототипів базується на ідеї створення цілісної або частини системи у пілотній версії, яка називається прототипом. Це можна розглядати як процес, або той, який є частиною більшого життєвого або центральним його елементом, який самостійно визначає життєві цикли. Мета полягає в кінцевому підсумку

розробки в різних версіях і послідовно удосконалювати ці версії до кінцевого продукту. Акцент робиться на створенні програмного забезпечення, при цьому менша увага до документації. Це також орієнтований на користувача підхід, оскільки відгуки користувачів є основою для розробки наступних прототипів і, врешті-решт, кінцевого продукту [31].

Рис. 3.7 Схематичне представлення моделі прототипів. [авторська розробка]

Існують різні типи моделі прототипів, відповідно до конкретних потреб проекту. Вони можуть бути підсумовані в трьох категоріях:

- дослідницький підхід зосереджений на тому, що вимоги ретельно досліджуються з кожною ітерацією. Під цією категорією мається увазі швидке створення прототипів, що є, по суті, способом доставки швидких випусків продукту з кожною ітерацією, вивчення потреб і вимог з кожною версією та відповідно вдосконалення наступної версії. Потреби оцінюються як продукт, який використовується та випробовується. З іншого боку, спіральна модель, що згадувалась вище, є ще однією формою дослідницького прототипу, де прототипи застосовуються на наступних етапах процесу розробки, кожен з яких слідує за водопадом;
- експериментальний підхід передбачає, що спочатку пропонують рішення потреби користувача, а потім оцінюють їх за допомогою

експериментального шляху. Застосування програм моделювання та скелетного програмування (доставки лише найважливіших функцій системи, щоб користувач міг отримати загальне уявлення про те, що буде остаточним продуктом) підпадає під цю категорію, але є багато інших прикладів, оскільки це є найбільш поширеною формою прототипів;

- еволюційний підхід по суті описує розвиток у послідовних варіантах і найближчий до ітераційних моделей, тому що його основна мета полягає в тому, щоб врахувати можливі зміни вимог та потреб. Прототип використовується принципово, щоб забезпечити легкий контакт з продуктом, щоб визначити сприйняті потреби. Кожен прототип є не більш ніж версією продукту, і кожна версія служить прототипом для наступного.

Модель «Зірковий життєвий цикл» - була запропонована Гартоном та Хішем наприкінці 1980-х років, як результат широкого спостереження за розробниками в реальному часі. Це окремий варіант інженерії зручності використання, орієнтований на користувача набір правил розробки програмного забезпечення, і, таким чином, відкидає жорсткий, поступовий характер водоспадних моделей.

Найбільш раціоналізаторська передумова полягає в тому, що кожен етап розвитку не обов’язково займає фіксовану позицію в рамках фіксованого процесу. Натомість передбачається, що існує ряд важливих етапів розвитку, але їх можна обробляти в різній послідовності та в різні терміни, відповідно до конкретних потреб проекту, з можливістю повернення до попередніх етапів або повністю пропустити певний етап, якщо він виявиться невідповідним до нових змін. Таким чином, наприклад, розробник може почати роботу, експериментуючи з різними варіантами дизайну, і в цьому процесі вивчати більш конкретні вимоги проекту, та як результат повернутись до попереднього етапу для його зміни.

Основним правилом є те, що кожен етап повинен супроводжуватися обширною оцінкою. Всі етапи взаємопов’язані, і в процесі розробки можна

перейти до будь-якого з них у будь-якій точці. Подібним чином, кожна виконана дія, незалежно від її порядку, повинна ретельно аналізуватися. Це включає в себе тестування та збір даних щодо цієї конкретної діяльності, за допомогою таких методів, як опитування користувачів або спостереження за їх використанням у робочому контексті.

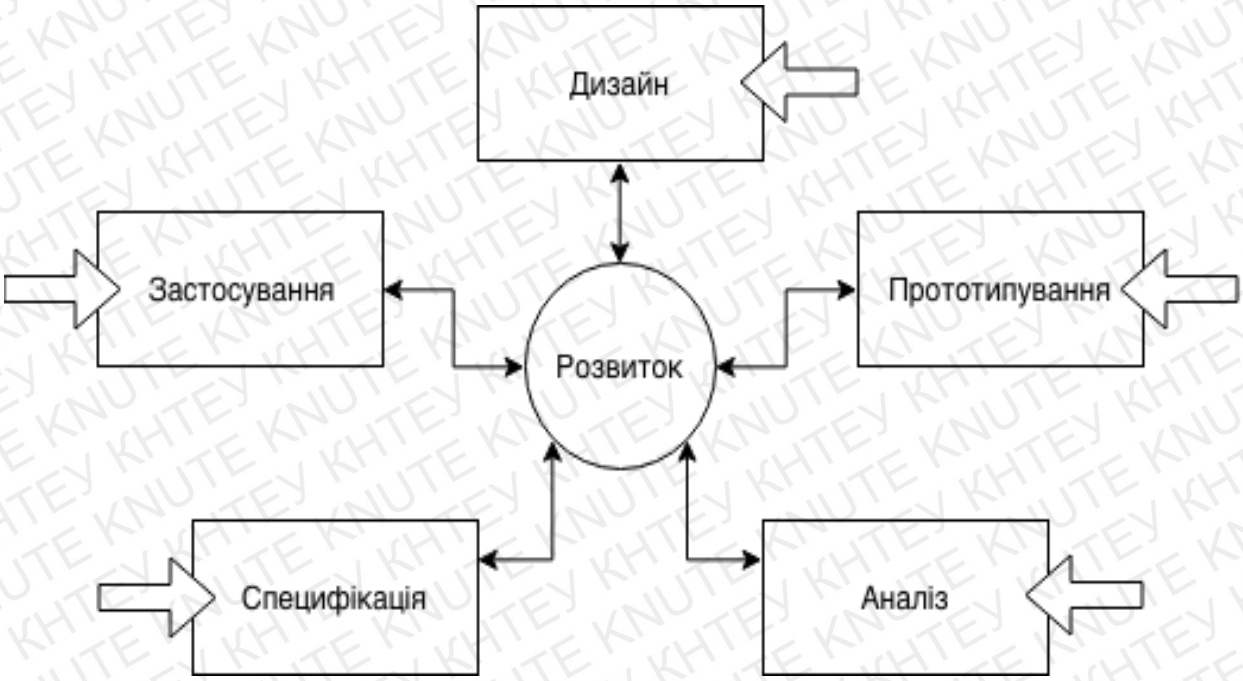

Рис. 3.8 Схематичне представлення моделі «Зірковий життєвий цикл». [авторська розробка]

Гібридна модель – виникла як результат підвищення конкурентоспроможності та вимог до систем, які є безпечними та надійними. Вони також повинні бути адаптованими та гнучкими до змін, які можуть статися в будь-який момент. Це призводить до ускладнень процесу розробки. Гібридна модель стала відповіддю на ці питання і є поєднанням різних моделей розробки системи.

Кожена із моделей має свої особливі характеристики, які можуть бути як вигідними, так і згубними залежно від типу вимог та характеристик проекту. Як тільки гіпотезується проект, - модель вибирається відповідно до цілей проекту. Але якщо конкретні характеристики проекту не відповідають одній конкретній

моделі, можна поєднувати засади з більш ніж однієї моделі. Ця комбінація, перш за все, полягає в тому, щоб використовувати сильні сторони моделі та зменшити її слабкі сторони, використавши сильні сторони іншої моделі [32].

Прикладом комбінованих моделей життєвого циклу є випадок процесу розробки, який керується спіральною моделлю, але пізніше цей процес вимагає зміни вимог. Для того, щоб врахувати цю потребу, може бути залучена модель agile.

Мунассар і Говардхан [33] намагалися сформулювати базові засади гібридної моделі. Вони підібрали важливі риси з таких моделей, як водоспад, ітераційна, V-образна, спіральна. Вона складається із семи етапів, які взаємопов'язані один з одним: планування, вимоги (на якому проводиться аналіз ризиків), проектування, впровадження (включаючи тестування), розвиток інтеграції, впровадження та технічне обслуговування. Хоча цей процес викладено в тій же моделі водоспаду, відносини між різними етапами є гнучким і різноспрямованими, що враховує можливі зміни вимог та необхідність перегляду конструктивних особливостей після тестування. Автори стверджують, що такий підхід дозволить поєднати найкращі характеристики кожної моделі. Модель ітеративна, але все ще включає аналіз ризиків, а тестовий підхід дозволяє забезпечити високу юзабіліті.

Для вибору моделі було проаналізовано всі вищезазначені види існуючих та загальнопоширених моделей та виділено ряд фактів пов'язаних з кожною із них, що в подальшому вплинуло на остаточний вибір моделі. Результатами аналізу стало:

Модель водоспаду – є ідеалізованою і значно спрощеної концепцією. Це не дуже гнучко, але ця модель як і раніше популярний у формі концептуальної основи для інших структур або моделей. Перевага даної моделі полягає в тому, що вона визначає загальноприйняті позитивні засади розробки програмного забезпечення, такі як чітке і точне планування на ранніх етапах проекту, детальна документація всього процесу та наявність надійних дизайнерських концепцій перед початком кодування.

Реальність процесу розробки часто може бути набагато більш дезорганізованим, ніж це передбачає модель водоспаду. Саме тому вона не може бути прийнята в нашому випадку.

Інкрементна модель - принципово вказує на прогресивний процес розробки шляхом поступового додавання більшої кількості функцій до завершення роботи системи. Це більш гнучкий метод, тому що це дозволяє включити потреби, які, можливо, не були очевидні на початку проекту, і полегшує зміни, що постають з наступною оцінкою різних вимог.

Однак недоліком цього підходу є те, що кошторис розробки значно збільшується за рахунок випуску кількох версій продукту. А також, якщо інкрементальна версія продукту, що виникла через нову знайдений проблему може мати проблеми сумісності з попередніми версіями продукту, що також є великим недоліком.

Ітеративна модель не може бути використана в розрізі проекту де є не велика команда та обмежений бюджет.

Спіральна модель - має значні переваги над вище описаними моделями. Акцент на аналізі ризиків є значним поліпшенням і робить його ідеальною моделлю для великих. Недоліком моделі є те що це не дуже ефективно в невеликих проектах; процес оцінки ризику може збільшити витрати системи настільки, що навіть створення системи, незалежно від ризиків, може бути більш економічно вигідним. Оцінка ризику також є процедурою, що потребує дуже специфічної компетенції, що ще більше сприятиме різкому зростанню витрат.

З міркувань недоцільності залучення значної частини бюджету задля оцінки ризиків – ця модель не може бути вибрана для реалізації поточного проекту.

Модель V - вирішує помилки незабаром після їх ідентифікації, що знижує ціну їх виправлення, це є найбільшою перевагою використання цієї моделі, зокрема, якщо порівнювати з класичною моделлю водоспадів. Також, оскільки тестування фракціонується протягом всього процесу, всі сторони розробки несуть відповідальність за розробку. Це також означає, що методи тестування є

адекватними для кожного етапу. Крім того, той факт, що тести виконуються з самого початку процесу, тільки підвищують ефективність моделі.

З іншого боку, так само, як модель водоспаду, ця модель дуже жорстка і погано підходить для гнучкої адаптації, особливо тому, що будь-яка зміна вимог призведе до того, що всі наявні документи та тестування більше не можуть вважатись справедливими для поточної розробки.

Оскільки модель вимагає значних фінансових та робочих ресурсів - явно оптимізована для великих проектів у великих організаціях, і як результат – не підходить для поточного проекту.

Модель швидкої розробки додатків. Переваги даної моделі описані вище є очевидними, завдяки концентрації що базується на швидкій розробці та ефективному спілкуванні між розробником та клієнтом. Проте існує ще ряд питань, які піднімаються з використанням цієї моделі. Одним з найбільш очевидних недоліків є те, що модель приділяє значну увагу мінімальному плануванню та моделюванню на початку проекту, переміщуючи увагу на життєвий процес побудови системи. Ще однією важливою проблемою є те, що у більш швидких циклах розробки якість тестування стає менш пріоритетним фактором, що результує у гіршій якості в цілому.

Модель прототипів може бути частково використана в поточному проекті.

Модель Agile. Перевагою моделі є те, що вона дуже гнучка. Іноді її поєднують з іншими існуючими моделями. Вона має здатність підтримувати вимоги які постійно змінюються, в умовах жорстких термінів.

Ця модель часто обирається за високу ступінь задоволеності та зручності зі сторони користувачів. Agile моделі клієнт-орієнтовані і пропагують «короткі ітерації та невеликі релізи», щоб отримати відгук про те, що було досягнуто в кожному релізі. Завдяки отриманим відгукам може бути внесено покращення, що матиме позитивні наслідки для якості кінцевого продукту.

Модель Agile не може бути використана в поточному проекту незалежно від інших моделей. Проте частково можуть бути залучені різні засади agile.

Модель прототипів - слабка при аналізі та дизайні. Оскільки вимоги до продукту розвиваються разом з продуктом, що розроблюється у послідовних версіях – необхідний високий контроль над витратами та ресурсами. Похибки такого контролю можуть суттєво збільшити фінансові витрати проекту.

Таким чином, ми можемо зробити висновок, що прототип ідеально підходить для більших, клієнт-орієнтованих проектів. Часткові засади моделі прототипів можуть бути використані в поточному проекті.

Модель «Зірковий життєвий цикл» – ця модель була зразу відкинута, по причині високого рівня ризику хаотизації процесу розробки.

Гібридна модель – у поданні Мунассара і Говардхана є дуже потужним інструментом. Саме ця модель підходить для реалізації поточного проекту на початкових його етапах.

Розмір проекту є одним із найвпливовіших факторів. Як правило - чим більше проект, тим жорсткіша модель, тому що велика команда, що складається з багатьох розробників, ускладнює оперативну взаємодію. Проте у великого проекту також може бути і не велика команда. Гнучкий розвиток, де важливе спілкування, - це підхід, який є більш вигідним для малих груп, які працюють ближче. Завжди слід враховувати ресурси. Проекти, що передбачають складну динаміку та вимагають використання специфічних експертиз та технологій, легше реалізувати з моделями строгого планування, такими як водоспад, проте ці вимоги справедливі не для всіх етапів розвитку проекта.

3.2 Аналіз та визначення засад розробки прототипу програмного забезпечення з використанням технологій віртуальної реальності

Як вже згадувалося, віртуальна реальність має потенціал використання як спільного інструменту через його ефективне та інтуїтивне маніпулювання. З огляду на це, для цієї роботи було розроблено прототип програми, який дозволяє зацікавленим сторонам проекту (клієнтам, дизайнерам, інженерам, підрядчикам тощо) візуалізувати та взаємодіяти з проектною моделлю, використовуючи устаткування віртуальної реальності.

Для розробки прототипу програмного забезпечення раннього етапу було використано три незалежні інструменти (для підтримки моделі прототипів): Unity, Dynamo та Revit.

Основна концепція проілюстрована на рисунку 3.9, та показує, що зворотній зв'язок створюється в додатку віртуальної реальності, а потім знову повертається в модель Revit, що надає можливість робити зміни в моделі BIM. Створення віртуальної моделі виступає як віртуальний прототип, який потенційно замінює потребу фізичних моделей. Одне визначення завдання поточного прототипу програми - це програма розгляду та модифікації дизайну у середовищі віртуальної реальності, та основними функціями якої є:

- вибір матеріалів та постачальників вибраного обладнання;
- виявлення дефектів дизайну та модифікація дизайну;
- автоматично передавати інформацію, створену в моделі віртуальної реальності, до BIM модель.

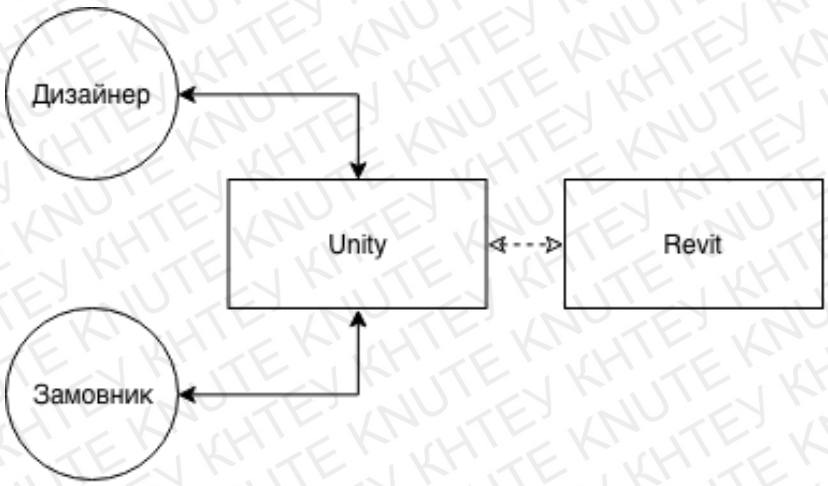

Рис. 3.9 Схематичне представлення концепції використання прототипу ПЗ. [авторська розробка]

Оскільки прототип програмного забезпечення використовується для оцінки та підтвердження дизайнерських рішень щодо дизайн-проекту, його використання призначено до етапу реалізації дизайн-проекту, раніше, ніж будь-яка з розглянутих частин дизайну вже була реалізована, це покликано дати змогу робити зміни в дизайн-проекті без додаткових фінансових затрат. Перевірка

дизайн-проекту проводиться таким чином, щоб задіяна команда могла оцінити, чи задовольняються вимоги дизайн-проекту, і вирішити, чи необхідно змінити елементи, з технічних, естетичних чи особистих причин.

Програмне забезпечення може бути використано як підтримка підходу до інтегрованої доставки проектів (IPD) через його аспект співпраці. Цей метод сприяє співпраці зацікавлених сторін проекту на попередніх етапах проекту з метою максимального підвищення ефективності проекту [34]. Як показує рисунок 3.10, спроможність впливати на вартість проекту значно вища на попередніх етапах завдяки здатності змінювати дизайн без необхідності фактичної переробки в реальному світі [35].

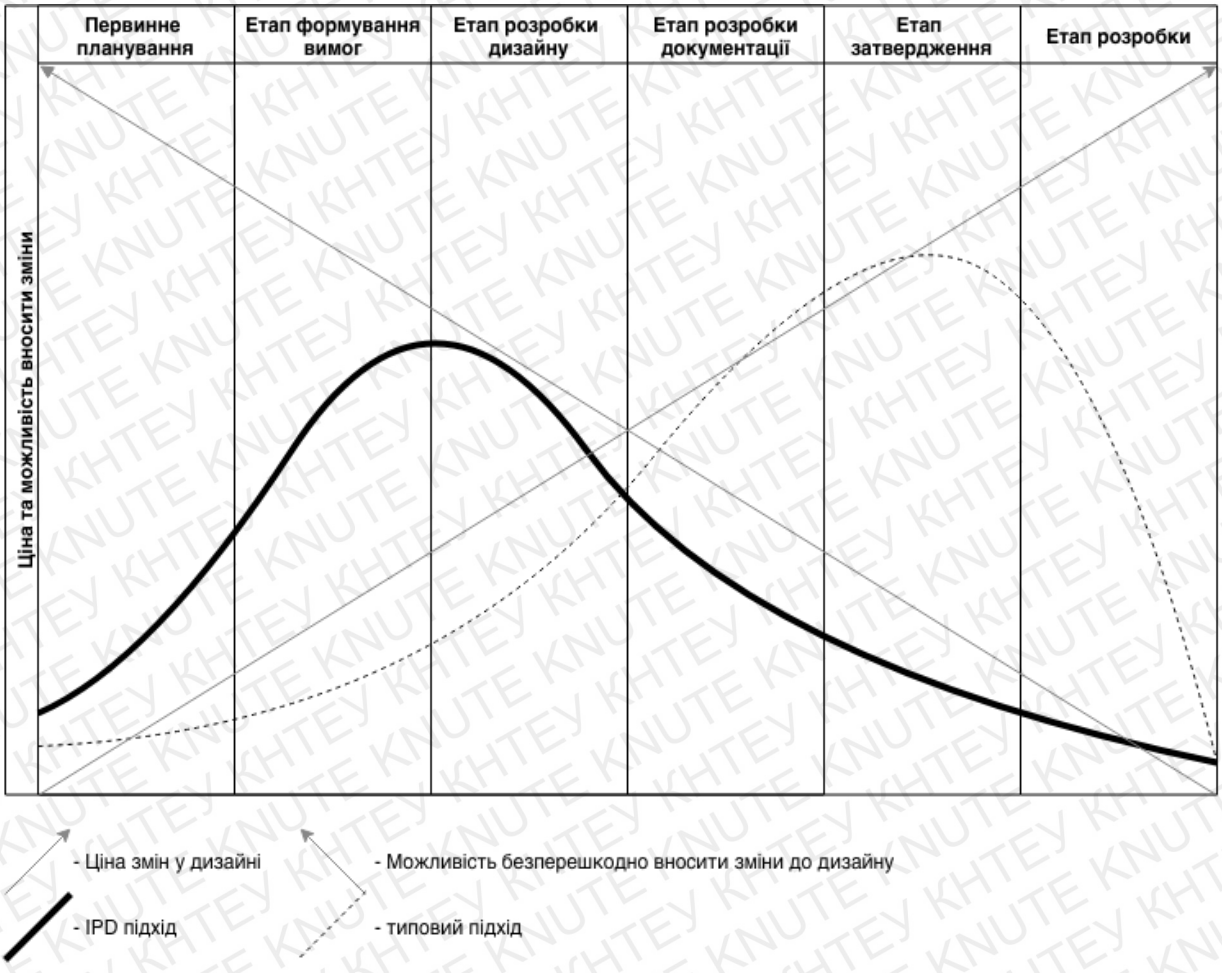

Рис. 3.10 IPD підхід впродовж всіх етапів проекту. [авторська розробка]

Варто зазначити, що прототип програмного забезпечення з використанням технологій віртуальної реальності є виконуваним файлом, який можна запускати без необхідності встановлювати будь-яке додаткове програмне забезпечення, що

полегшує розповсюдження. Використання програмного забезпечення може бути індивідуальним або груповим. Використання у режимі кооперації з зацікавленими сторонами результує у кращих результатах, тому що дизайн або інші аспекти можуть бути обговорені та вирішені негайно, відфільтровуючи можливі сумніви. Ще однією перевагою прототипу програми є те, що користувальницький інтерфейс є інтуїтивно зрозумілим, тому він не вимагає будь-яких попередніх технічних знань для використання, що полегшує інтеграцію з усіма учасниками проекту.

Одним з релевантних аспектів, який слід враховувати, це область дизайн-проекту, яку буде проаналізовано за допомогою програмного забезпечення. Якщо первинний намір полягає у оцінці вибору матеріалів та обладнання постачальника, залежно від типу конструкції – є не доцільним аналізувати всю будівлю. Достатню вибрати певну зону, а не всю будівлю, що значно зменшить повторюваність процесу. Наприклад, якщо проект дизайну стосується студентського гуртожитку, - зазвичай у більшості кімнат студентів однакові матеріали та обладнання. Використовуючи одну кімнату як модель, вибір дизайну можна оцінити набагато швидше, ніж проводячи ті самі дії з усією будівлею.

На попередньому розробці етапі визначаються технології що будуть використовуватись для реалізації первинного прототипу програмного забезпечення.

Всього для розробки прототипу було використано три програмні продукти: Revit, Unity і Dynamo. Перша програма - несе відповідальність за моделювання BIM, що містить всю інформацію про дизайн. Друга - відповідає за створення користувальницького інтерфейсу, елементів інтерактивності та для користування технологій віртуальної реальності. Третя – використовується для імпортування інформації створеної в додатку Unity, до моделі Revit.

Autodesk Revit - це програмне забезпечення, розроблене для BIM, яке підтримує професійні дизайн-проекти та інфраструктуру, включаючи функції для дизайн проектування, конструкторського проектування, механічного електропостачання, сантехніки та будівництва. Програмне забезпечення містить

багато інструментів, які допомагають користувачеві планувати, проектувати, конструювати та керувати процесом інтелектуальної моделі [36]. Однією з можливостей програмного забезпечення є спільне використання. Це дозволяє декільком користувачам одночасно отримати доступ до спільної центральної моделі, в якій кожен користувач може синхронізувати зміни, змінюючи всі моделі.

Autodesk Dynamo - редактор графічного алгоритму програмного забезпечення з відкритим вихідним кодом, який розширює моделювання дизайну даними та логікою [37]. Програмне забезпечення являє собою інструмент візуального програмування, який призначений для доступу, незалежно користувач володіє досвідом програмного забезпечення чи ні. Це дозволяє користувачам визначати елементи логіки, візуальної поведінки сценаріїв та сценаріїв з використанням текстових мов програмування [38]. Ця програма може бути запущена у окремому режимі «Пісочниця» або як плагін для іншого програмного забезпечення Autodesk, такого як Revit. Якщо Dynamo запускається як плагін для Revit, Dynamo може використовуватися для різних цілей, таких як автоматичне керування інформацією, створення графічних уявлень, модифікація моделі BIM тощо.

Будучи графічним редактором алгоритмів, Dynamo представляє користувачеві інтуїтивно зрозумілий спосіб для параметричного проектування та управління інформацією про дизайн без необхідності знати мову програмування. Кодування відбувається за допомогою «вузлів», а не типових кодових ліній. Кожен вузол виконує операцію, яка варіюється від простої математичної операції до запиту складної геометрії. Ці вузли розміщуються через інтерфейс перетягування, і вони з'єднані між собою поєднувачами, створюючи послідовний логічний робочий процес. Поєднувачі транспортують інформацію через вхідні та вихідні порти, підключені до вузлів. З лівої сторони вузла знаходяться вхідні порти (отримує інформацію), а порти, які знаходяться в правій стороні, є вихідними портами (інформація про передачу).

Іншою функціональною одиницею вузлів є можливість писати код як на DynamoScript (DS), так і на мови програмування Python через вузли, звані «code blocks». Ці блоки дозволяють користувачеві програмувати функціональні можливості, недоступні для бібліотеки вузлів, що дозволяє розробляти більш складні робочі процеси. На рисунку 3.11 можна побачити динаміку коду Динамо. По-перше, послідовність вузла отримує три числа, що приводить до переліку порядкових номерів. Після цього «кодовий блок», що містить простий операційний сценарій, який помножує входи один на одного, отримує цю послідовність чисел.

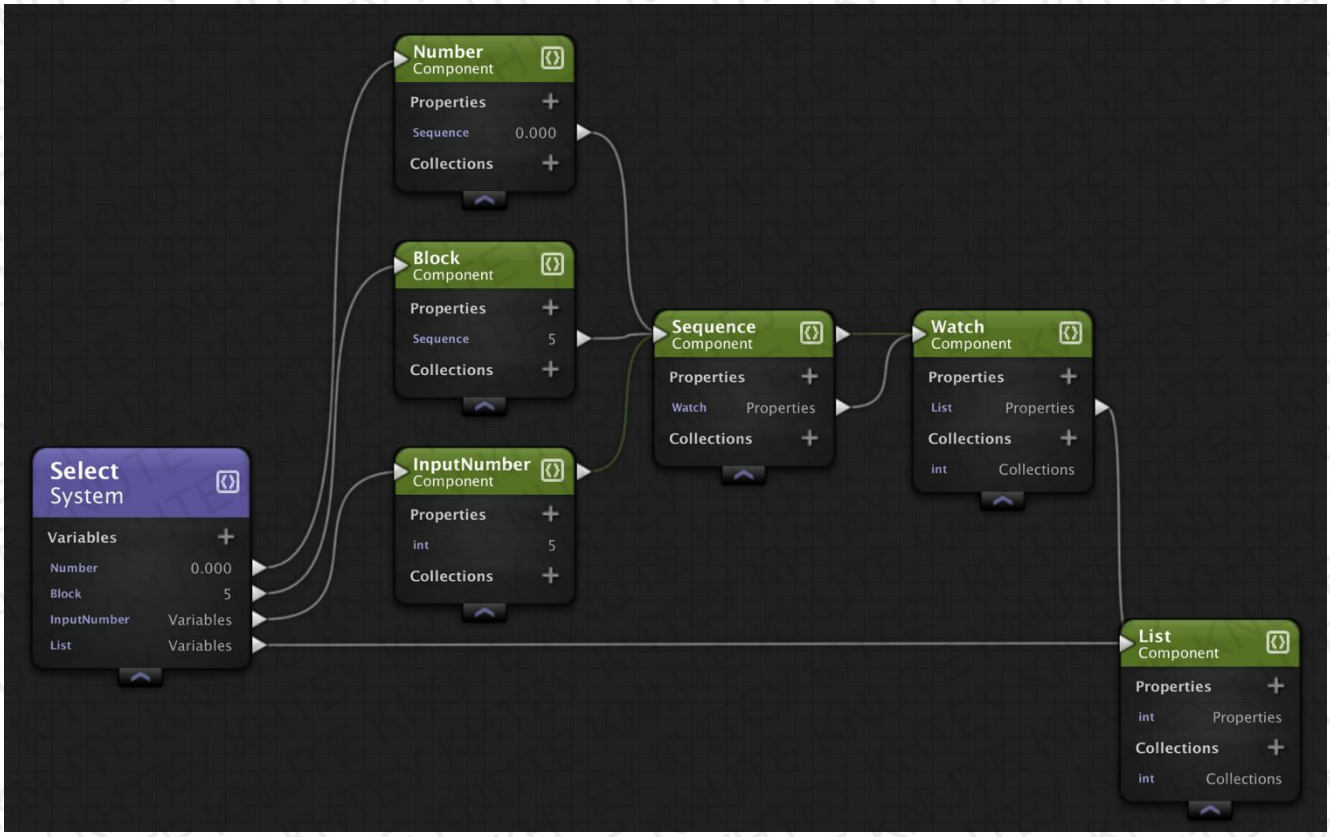

Рис. 3.11 Приклад DynamoScript. [авторська розробка]

Unity це багатофункціональний ігровий движок, який використовується для розробки як тривимірних, так і двовимірних моделей та відеоігор для комп'ютерів, мобільних телефонів та відео-ігрових консолей. Ця програма підтримує функції фізики, 2D або 3D-графіку та сценарії за допомогою мови програмування C #.

В даний час це програмне забезпечення доступне за чотирма різними ліцензіями; особистий, плюс, про та підприємство. Кожна ліцензія містить свої відмінності в рівні підтримки від компанії-розробника. Особиста ліцензія є найбільш базовою і в даний час вона безкоштовна, якщо компанія чи особа мають дохідну потужність в розмірі 100 000 доларів США на місяць. Всі інші варіанти ліцензії мають щомісячну комісію, яка починається від 35 доларів США. Дана програма на сьогодні є найпопулярнішим ігровим движком на ринку [39]. Основними причинами цього є здатність створювати контент для декількох платформ, пов'язаних із зручним інтерфейсом та великим посібником, який допомагає новим користувачам освоїти програмне забезпечення.

Незважаючи на те, що Unity не є програмним забезпеченням, призначеним для використання в промисловості, вона пропонує ряд функцій, які були враховані для вибору як програмне забезпечення для розробки програмного забезпечення з використанням технологій віртуальної реальності, такі як:

- підтримка розробки віртуальної реальності;
- масштабний сучасний продукт з великою підтримкою;
- перевірена як ігровий движок, який добре зберігає геометрію Revit за допомогою програмної сумісності [40];
- керується мовою програмування C #, що є сучасною та розповсюдженою мовою;

Для маніпуляції та відтворення віртуального середовища було обрано HTC Vive. HTC Vive - це гарнітура з віртуальною реальністю, яка була запущена в квітні 2016 р. У партнерстві між HTC і Valve, американським розробником відеоігор та компанією розповсюдження цифрових копій відеоігор. Основними компонентами системи є гарнітура Vive, контролери Vive та базові станції Vive. Всі складові можна побачити разом на рисунку 3.12.

Рис. 3.12 Комплект HTC Vive [50].

Одним з найважливіших елементів віртуальної реальності є система відстеження, яка відповідає за відчуття чуттєвого відгуку користувача. У системі Vive базова станція несе відповідальність за відстеження гарнітури та контролерів. Станції випромінюють 60 інфрачервоних імпульсів за секунду, які відстежуються навушниками та контролером з точністю до міліметра.

Для взаємодії з віртуальним світом є бездротові контролери Vive, що дають відчуття занурення. Контролер містить кілька способів введення, таких як кнопки зчеплення, сенсорна панель та тригер. Для того, щоб бути виявленим базовою станцією, кільце контролера має в цілому 24 інфрачервоних датчика.

Пристрій для гарнітури містить два екрана, кожен з яких має роздільну здатність дисплея 1080x1200 пікселів, поле зору 110 градусів і частоту оновлення 90 Гц. З міркувань безпеки вона включає в себе передню камеру, яка дозволяє користувачеві спостерігати за оточенням без необхідності зняття гарнітури (двічі натиснувши кнопку меню). Ще однією особливістю є можливість виявлення будь-яких статичних або рухомих об’єктів в приміщенні, попереджаючи користувача,

виводячи стіну у віртуальний світ, щоб користувач не стикався з будь-якими реальними перешкодами.

Для розробки прототипу програмного забезпечення у цій роботі, HTC Vive був обраний завдяки тому, що він є однією з найбільш розвинених систем віртуальної реальності.

3.3 Розробка проекту програмного забезпечення

Першим питанням розробки програмного забезпечення, в розрізі даної роботи – питання роботи з моделями для проекту. Побудова моделей проекту вимагає значного часу та зусиль, тому бажано використовувати найбільш ефективний спосіб підвищення продуктивності праці. Програмне забезпечення BIM, таке як Revit, дозволяє користувачеві моделювати параметризовану систему за допомогою багатьох функцій, що допомагає досягти бажаного дизайну. Наприклад, щоб спроектувати просту стінку, необхідно натиснути на відповідну кнопку та вибрати початкову точку і кінцеву точку, в якій буде автоматично створена стіна. Оскільки більша частина інформації елемента параметризована, після її створення легко змінювати такі функції, як висота, товщина та інші параметри. Revit також інтегрує багато попередньо визначених аспектів, які допомагають моделювати дизайн в рамках деяких правил конструктивності. Наприклад, стеля прикріплюється, перш за все, до стін нижче, або вставка вікна може бути зроблена лише в тому випадку, якщо вона вставлена в стіну.

Програмне забезпечення Unity, що використовується в цій роботі, має цілий набір інструментів, який дозволяє розробнику організувати свій проект за допомогою стилю інтерфейсів та компонентів, який представляє практичний та настроюваний користувальницький інтерфейс. Щоб мати можливість створити стіну з порожнім проміжком у середині, призначеному для вікна, найпростіший спосіб зробити це - створити чотири куби 3D-ігрових об’єктів (два для сторони, один бути на вершині і один, щоб бути внизу), змінити розмір і змінити положення кожного.

Через різницю зусиль у моделюванні помітно, що будівельні елементи, такі як стіни, вікна, стелі, перила, сходи тощо, більш ефективно створювати, використовуючи Revit завдяки своїм заздалегідь визначеним компонентам CAD, оптимізованим для будівництва та дизайну.

Багато дизайнерських компаній визначили найбільш сприятливе для себе програмне забезпечення BIM, для розробки своїх проектів. Більшість цих проектів BIM розробляються в 3D-моделях. Наприклад, через Revit можна експортувати такі 3D-файли, як FBX (FilmBoX) та OBJ (об'єктний файл), які підтримуються для імпорту Unity як ігрові об'єкти. Ці файли є взаємосумісні при збереженні геометрії моделі. Імпортуючи файл FBX до Unity, можна заощадити багато часу моделювання, особливо, якщо у бажаному проекті віртуальної реальності є багато моделей для об'єктів.

Аналізуючи процес імпорту нових елементів у середовище Unity, можна зробити висновок, що в основному є три шляхи:

- Завантаження об'єктів з магазину об'єктів - існує великий діапазон можливостей об'єктів. Від ультра-детальних об'єктів до дуже простих об'єктів.
- Імпортувати FBX або інші файли 3D-файлів до проекту. Існує широкий вибір 3D-об'єктів в Інтернеті, однак потрібно враховувати, що вони мусять бути оптимізовані для віртуальної реальності.
- Створювати об'єкт за допомогою інструментів Unity - якщо користувач проекту не має досвіду, цей параметр залишається лише для створення простих геометричних об'єктів.

Незважаючи на те, що імпорт об'єктів BIM з файлами FBX представляє заощадження часу, слід враховувати, що часто файли Revit не оптимізовані для використання у проекті віртуальної реальності. Елементи Revit, як це видно на рисунку 3.13, містять складну ієрархію з категоріями, сім'ями, типом та екземплярами. Оскільки всередині Revit ці об'єкти зазвичай параметризуються за допомогою різних параметрів, це призводить до великої кількості геометричної інформації.

Наприклад, один екземпляр вікна - в проектах BIM існує сімейство вікон розроблено таким чином, щоб бути максимально налаштовуваним настільки, наскільки це можливо, з метою змінення кількох параметрів (таких як висота, ширина, каркасний матеріал, шарнір, корпус, затвор та багато інших) для створення різних типів вікон без необхідності створювати нову сім'ю для кожного нового проекту. Навіть незважаючи на те, що це корисна функція для Revit, розглядаючи в контексті віртуальної реальності, де основною метою є просто представлення, це призводить до накопичення великої кількості непотрібної інформації що експортуються до Unity, що в подальшому не використовуються.

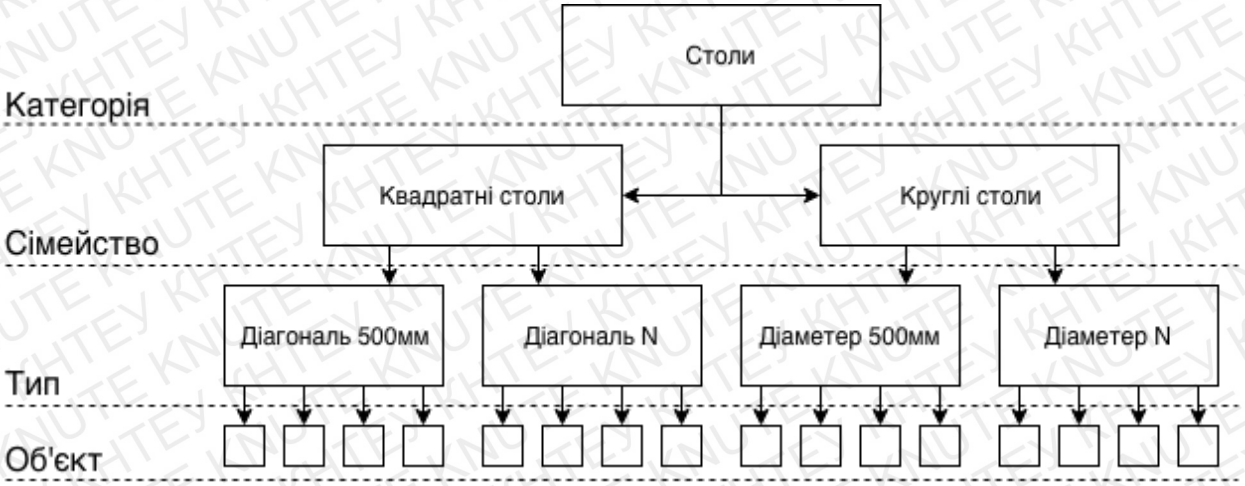

Рис. 3.13 Приклад ієрархії категорій в Revit. [авторська розробка]

Ще однією проблемою імпорту файлів FBX є втрата інформації про об'єкт. Файл FBX, створений за допомогою Revit, містить лише інформацію про геометрію. Така інформація, як текстура, матеріал та кольори, втрачається.

Як було обговорено раніше в цій роботі, досвід віртуальної реальності вимагає відмінної графічної продуктивності для сприйняття користувачем. Загальна ефективність роботи програмного забезпечення безпосередньо впливає на кількість полігонів, які потрібно відобразити, особливо якщо обчислювальні потужності апаратного забезпечення є обмеженими. Отже, якщо проект BIM містить значну кількість об'єктів з великою кількістю складних елементів, коли ці елементи імпортуються в Unity, це може призвести до зниження рейтингу в

продуктивності і результувати у меншій частоті кадрів в секунду, чого достатньою для того, щоб призвести до значного погіршення досвіду віртуальної реальності.

Через проблему обрахунку великої кількості полігонів, що експортуються безпосередньо через Revit, і втрат важливої інформації, одне рішення полягає у використанні посередницького програмного забезпечення, такого як Autodesk Maya або Autodesk 3DS Max, які можуть бути використані для оптимізації процесу експорту. Однак обидва приклади програмного забезпечення також мають високу вартість, наприклад, Autodesk 3DS Max, в даний час, підписка на 1 рік коштує приблизно 2000 євро на один комп'ютер [41]. До того ж використовуючи посередницьке програмне забезпечення необхідно додати ще один додатковий крок перед тим, як вставити моделі в ігрові движки, що результує у збільшені часу на підготовку.

Аналізуючи ситуацію з моделювання у проекті, - існує три основних аспекти, які необхідно розглянути при моделюванні, що суттєво змінюються в залежності від специфікації проекту: складність елементів, вимоги до специфікації естетики та планована площа. На закінчення, в даний час не існує ідеального рішення, яке є найбільш продуктивним для кожного окремого проекту, кожен проект має свої специфікації та контекст, який може сильно вплинути на кінцевий результат.

Другим питанням є питання користувацького інтерфейсу та інтерактивності. Інтерфейс користувача - це місце, де люди взаємодіють з програмою. Метою розробки інтерфейсу користувача є максимально ефективний та зручний (зручний для користувача) спосіб працювати з програмою для досягнення бажаних результатів. Незручний користувальницький інтерфейс несприятливо впливає на сприйняття користувачами програми [42].

У проектах, які не використовують віртуальну реальність, інформація для користувача зазвичай подається поверх екрана. Проте в віртуальній реальності, відображати інформацію, наклавши її на візуальний дисплей, не є правильним, оскільки люди не можуть зосередитись на об'єктах, що знаходяться занадто

близько до очей. Внаслідок цього, для проектування користувальницького інтерфейсу у віртуальній реальності потрібен інший підхід [43].

Для розробки інтерактивного графічного інтерфейсу, маючи доступне для цього обладнання, існували, три різні варіанти введення, які слід враховувати: фізичні контролери, розпізнавання сітки та розпізнавання мовлення.

Перший - фізичні контролери пропонують функціональність для вводу даних користувача; контролер HTC Vive має ряд різних варіантів вводу, таких як трекпад, кнопка меню, системна кнопка, тригер та кнопка спуску, як це показано на рисунку 3.14. Це надає розробнику велику кількість функціональних можливостей, які можуть бути інтегровані для взаємодії з віртуальним світом. Для запропонованої програми фізичні контролери використовуються для більшості взаємодій. Фізичний контролер дозволяє користувачеві просто взаємодіяти з об’єктами або елементами графічного інтерфейсу, натискаючи кнопку тригера або спуску.

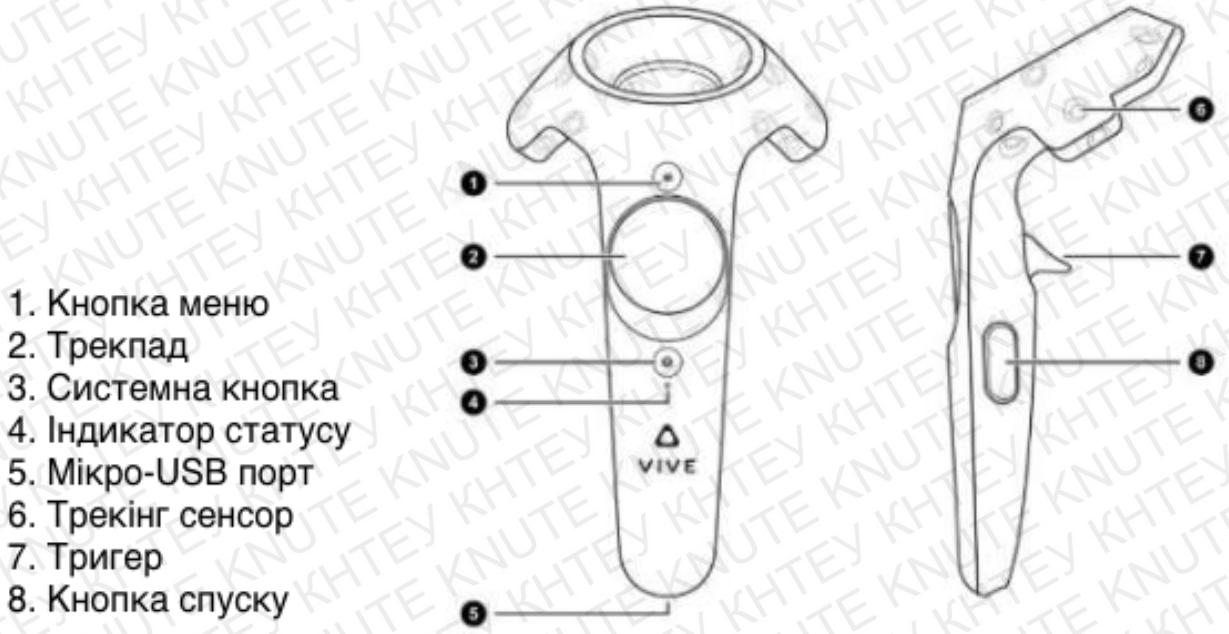

Рис. 3.14 Схема контролеру HTC Vive. [авторська розробка]

Другий - контролер сітки (розташований в шоломі) містить невелике коло в центрі поля зору користувача. Завдяки чому є можливість користуватися поглядом, щоб взаємодіяти із оточенням, це природний та інтуїтивно зрозумілий спосіб взаємодії з середовищем. Щоб мати змогу «активізувати» взаємодію, існує

кілька альтернатив, таких як натискання фізичної кнопки на контролері, або взаємодія погляд і «блокування» (Центруйте погляд на об’єкті та замираєте на декілька секунд). Цей тип взаємодії в основному використовується недорогими пристроями віртуальної реальності. У запропонованому програмному забезпеченні цей алгоритм може бути потенційно використаний, наприклад, для масштабування на об’єкті або відкритті панелі з інформацією. Однак, оскільки HTC Vive має два фізичних контролера, цей тип введення було проігноровано.

Третій - розпізнавання мови - це природний і практичний тип вводу, який дозволяє користувачеві щось сказати, і комп’ютер відповідає на його запит. Технологія, яка дозволяє це введення, розвинулася завдяки розробці віртуальних помічників, таких як Microsoft Cortana та Google Assistant [44].

Щоб бути в змозі взаємодіяти з віртуальним світом, було створено радіальне меню, показане на рисунку 3.15, яке активується, торкнувшись сенсорної панелі контролера Vive, це дозволяє користувачеві легко отримати доступ до інструментів, розроблених для програми, таких як вибір об’єктів. коментування, лінійка, взаємодія та відкриття панелі звітів. Користувач повинен лише провести пальцем до бажаному варіанту.

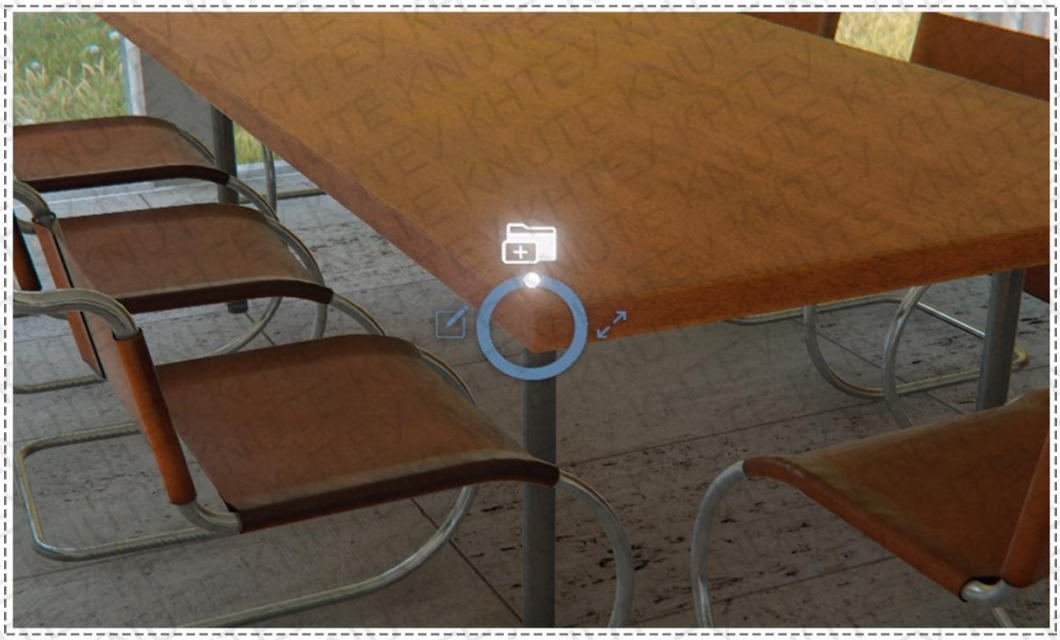

Рис. 3.15 Радіальне меню. [авторська розробка]

Відзначимо, вирішальне значення для користувальницького інтерфейсу програми віртуальної реальності, полягає в тому, що інформація не може бути накладена на дисплей користувача через брак в людських очах, здатних фокусуватися на близьких об’єктах. Внаслідок цього, панелі графічного інтерфейсу з інформацією з якою потрібно взаємодіяти, відображаються у просторі на зручній відстані, як показує рисунок 3.16, бажано відображати інформацію щонайменше в 0,5 метрах від користувача.

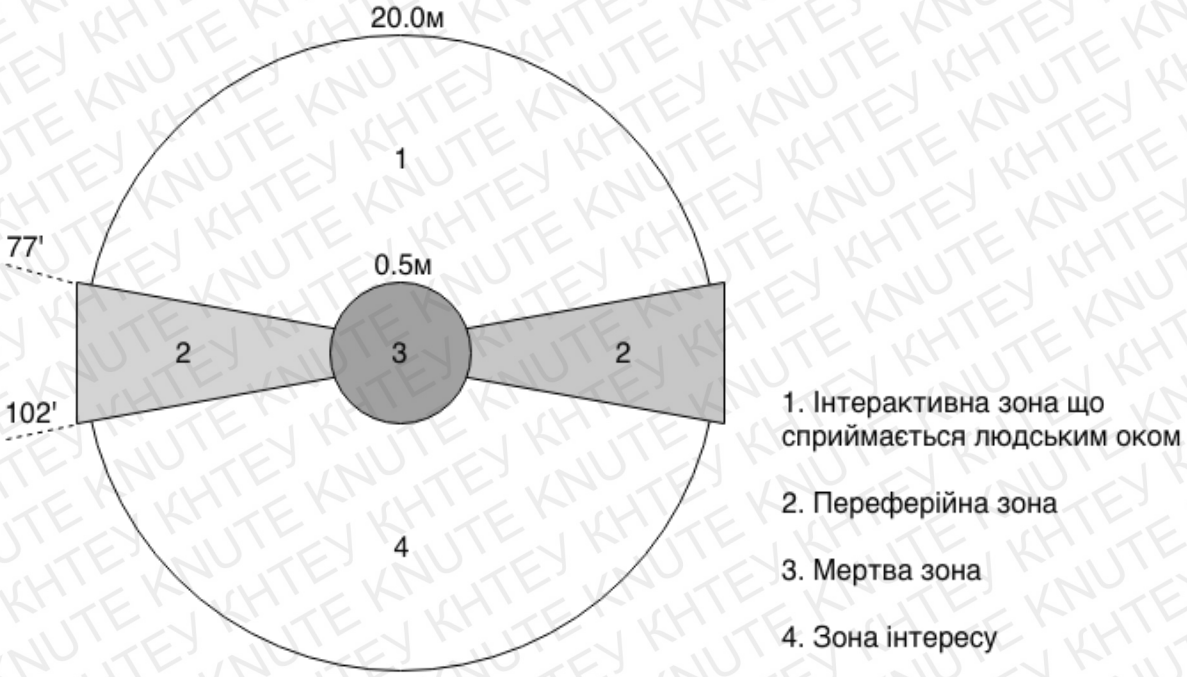

Рис. 3.16 Схематичне зображення зон, що сприймаються людським оком. [авторська розробка]

Критичним елементом віртуальної реальності є те, що учасник не відчуває простору. Як описано раніше, погіршення відчуття простору обумовлене, головним чином, сенсорним конфліктом між сприйняттям самого руху та вестибулярної системи, а також недосягненням комфортної частоти кадрів [45].

Щоб запобігти погіршенню почуття простору, важливо, щоб віртуальний світ правильно відреагував на рух користувача. Наприклад, якщо учасник рухає головою вправо, аватар з віртуального світу також рухає головою вправо з найменшою затримкою.

Датчики відстеження від апаратного забезпечення HTC Vive дозволяють отримувати надійну відповідь на рухи користувача, оскільки вони можуть відстежувати рух користувача та відтворити його у віртуальному світі. Однак в більшості випадків в реальному середовищі він менше, ніж у віртуальному середовищі. Через це, якщо користувач постійно ходить в реальному світі, в кінцевому підсумку він зіткнеться з такими перешкодами, як стіна або стіл і може завдати шкоди собі або пошкодити об’єкт.

У запропонованому програмному забезпечені необхідно, щоб користувач міг рухатися по різних кімнатах, та міг давати оцінку простору. Щоб мати можливість рухатись у віртуальному світі, одним із варіантів є використання трекпаду з контролера Vive. Щоб використовувати його як джойстик, - натиснути на напрямок і перейти у цьому напрямку, однак цей тип руху все ж погіршує відчуття простору, тому що користувач залишається нерухомим, коли його аватар рухається. Іншим рішенням було б не дозволяти користувачеві переміщати свій аватар у віртуальний світ. Створивши спеціальні гарячі точки, які користувач може вибирати і спостерігати навколишнє середовище з них. Однак це є великим обмеженням, тому що такий варіант не дає користувачеві можливість перевіряти деталі та взаємодіяти з ними.

Варіант, розроблений для запропонованої програми, здійснюється через телепортацію. Користувач вказує контролером на потрібне місце, і, натиснувши кнопку тригера, він телепортує до потрібного місця, як це видно на малюнку рис.3.17. Телепорт дає учаснику можливість вільно перевіряти незначні деталі та взаємодіяти з ними, в той же час він значно зменшує побічні ефекти від втрати відчуття простору, оскільки користувач не має чуттєвого конфлікту між тим, що він бачить, і що його тіло відчуває.

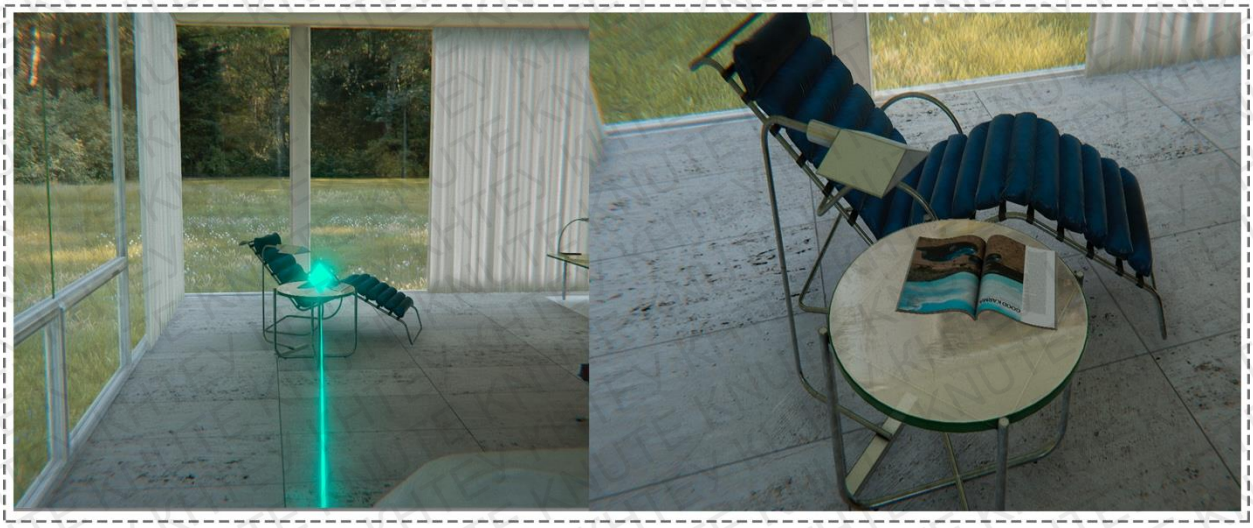

Рис. 3.17 Приклад переміщення у просторі шляхом телепортації. [авторська розробка]

Іншим аспектом було пересування між приміщеннями. Користувачеві не дозволяється проходити крізь стіни або двері, щоб зберегти імнергетичний чинник, оскільки наміром застосування програми є моделювання реальної фізики. Для відкриття дверей користувач повинен захопити дверну ручку за допомогою кнопки рукоятки, як і в реальному житті, і імітувати рух дверцят. Це інтуїтивно зрозумілий спосіб відкриття дверей, що не вимагає жодної форми пояснення.

Інтерактивні кнопки відносяться до релевантних елементів, які необхідно враховувати у побудові інтерфейсів та взаємодії з фіксованими елементам. Фіксовані елементи, такі як настил, плитка, сантехніка, освітлення, кухонні меблі та інші елементи - це все рішення, що впливають на використання простору в цілому, крім того, щоб внесення зміни до них, після нього вже побудований, вимагає значних витрат і зусиль. Тому дуже важливо, щоб вибір кожного елемента був перевіреним, і у випадку виникнення проблеми, створити відгуки про те, чому об’єкт не підходить. Інтерактивні кнопки можуть допомогти користувачам.

Оскільки цифрова модель працює як віртуальний прототип, елементи також можуть бути проаналізовані естетично. Наприклад, керамічна черепиця, розроблена на кухні, може бути оцінена, на приклад: чи розміри та матеріал

мають передбачуваний вигляд, інакше може бути створений коментар про те, що їх потрібно змінити.

Створення інтерактивних кнопок було розроблено через базу даних, яка містить всю інформацію щодо обладнання, це є корисним, оскільки це покращує організацію інформації і заощаджує час на введення інформації, будучи більш ефективним. Як можна побачити на рисунку 3.18, за допомогою спеціального списку скриптів можна додати та керувати інформацією щодо обладнання, такими як ім'я, локалізація, постачальник, матеріал та розміри. Пов'язаний із ним об'єкт несе відповідальність за управління об'єктом, який користувач побачить через панельну камеру. У цьому прикладі, показаному на рисунку, це пов'язано з настільною лампою.

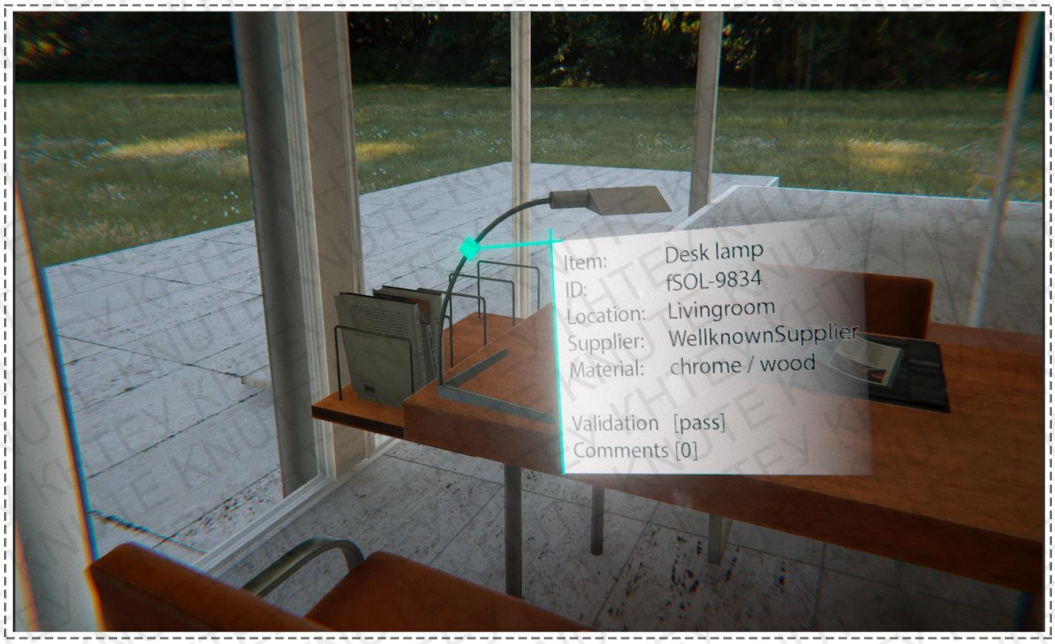

Рис. 3.18 Приклад інтерактивного меню в стадії його розробки (відсутні кнопки). [авторська розробка]

Кожного разу, коли користувач натискає інтерактивну кнопку, з'являється панель з інформацією, а вміст цієї панелі керує один контролер, пов'язаний з логікою натискання кнопки. Як видно на рисунку 3.18, контролер програми має сценарій інтерактивної панелі, яка містить одну ідентифікацію кореспондента, натискаючи на інтерактивну кнопку, вона змінює активну ідентифікацію інформаційної панелі та оновлює інформацію та камеру.

Підтверджуючи або не підтверджуючи елементи дизайну, користувач активує одну з анімацій, яка змінює колір кнопки. Якщо підтверджено, кнопка перетвориться на зелений. Аналогічно, якщо не підтверджено, кнопка перетворюється на червону. Це допомагає учаснику відстежувати, з якими елементами він вже взаємодіяв.

Якщо обладнання або елементи дизайну не підходять користувачеві, то релевантно зареєструвати причину. Проте, проблема занурення в віртуальну реальність полягає в тому, що фізичні контролери не оптимізовані для ефективного набирання тексту. Одне рішення, розроблене для цієї програми, використовувало один із самих інтуїтивних способів спілкування, просто розмовляючи (Додаток Б). Програма використовує ефективні служби розпізнавання мовлення Windows, що в даний час оптимізована для хмарних обчислень, які можуть точно розпізнати те, про що говорить користувач. На панелі коментарів, виходячи з рисунку 3.19, бачимо, що для початку диктування користувачеві потрібно натиснути кнопку запису та розпочати розмову, система прослуховує і записує, те що вона зрозуміла. Якщо користувач задоволений наданим текстом, він може продовжити та зберегти коментар, інакше він може повторити процедуру, натиснувши ще раз на кнопку запису.

Рис. 3.19 Приклад запису коментаря. [авторська розробка]

Можливість вимірювати відстані у віртуальному середовищі є корисною можливістю для перевірки розміру речей всередині віртуального світу. Це може бути використане для перевірки, наприклад, розташування світлових вимикачів, перевірки висоти стільниці, мінімальної відстані до небезпечних об’єктів, тощо.

Це досягається тим, що учасник вибирає через радіальне меню значок лінійки, а потім вказує на регулятор Vive і двічі натискає кнопку тригера, в результаті чого запускаються два промені, що зберігають його положення до місця, на якій покажчик вказує. Програма малює між цими двома точками та обчислює відстань між ними (Додаток А), дозволяючи користувачеві перевіряти відстані, як це зображено на рисунку 3.20.

Рис. 3.20 Приклад використання інструменту «лінійка». [авторська розробка]

Функція коментування, що вже описувалась вище, є інструментом, який дозволяє користувачеві створювати коментарі в проекті. Це корисна функція для зазначення проблем, пов’язаних з проектом, таких як дефекти, особисті уподобання тощо. Користувач повинен вибрати з радіального меню інструмент коментарів та вказати позицію чи предмет, на яку потрібно розмістити коментар, і натиснути кнопку тригера контролера Vive. Для того, щоб вставити коментар, також використовується розпізнавання голосу з служб розпізнавання мовлення

Windows. Коментар зберігається в базі даних, після чого його можна пізніше експортувати до панелі звітів за допомогою кнопки експорту.

Одним з важливих аспектів, який слід враховувати, є здатність користувача отримувати доступ до інформації про всі елементи, щоб перевірити та підтвердити прийняте рішення під час занурення в віртуальне середовище. Щоб мати доступ до цієї панелі, учасник повинен вибрати з радіального меню піктограму папки (рис. 3.16). Ця панель відображає інформацію про всі фіксовані елементи, підготовлені для перегляду проекту (рис. 3.21), і автоматично оновлюється відповідно до змін, внесених під час роботи в віртуальному середовищі, шляхом доступу до бази даних елементів.

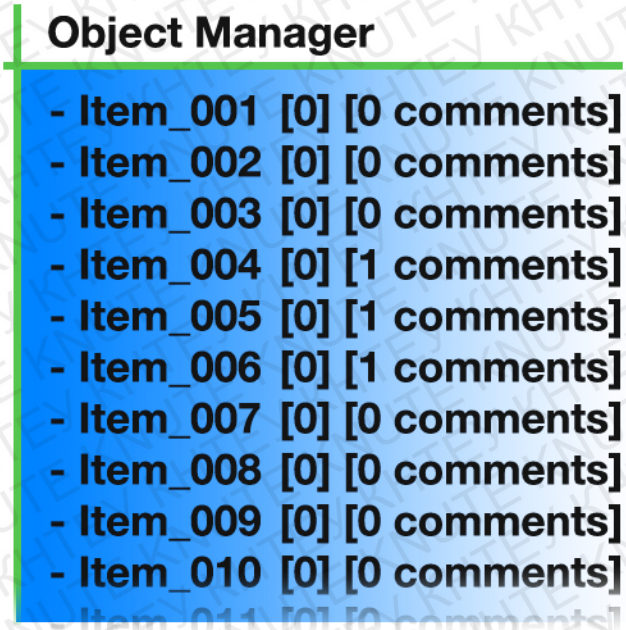

Рисунок 3.21 Приклад вигляду елементів у Object Manager. [авторська розробка]

На панелі звітів також міститься кнопка вилучення, яка відповідає за експорт бази даних всіх елементів і коментарів, створених для імпорту в модель проекту BIM.

Оскільки в проекті використовується два різних типи програмного забезпечення - Unity для віртуальної реальності та Univision для програмного забезпечення BIM Revit - необхідно створити проміжний лог-файл (заснований на .txt), який буде використовуватися обома з них для встановлення потоку

інформація, як це показано на рис.3.22. Файл журналу створюється Unity, а потім імпортується в Revit за допомогою плагіна Dynamo. Натискаючи кнопку експорту на панелі звітів, програма віртуальної реальності записує файл журналу, який має просту структуру з комами (CSV). Створення файлу журналу є корисним, оскільки полегшує потік інформації, і створюється автоматично, учасники дизайнерського огляду не зобов'язані писати щось на папері або заповнити лист Excel.

ID	Valid	Name	Local	Supplier	Material	Metrics	Comments	PosX	PosY	PosZ
0	TRUE	Item_001	Room 1	Wellknown Su...	Wood/Chrome	2490x600	NULL	-20.0575	95.342	-25.44
2	TRUE	Item_002	Room 4	Wellknown Su...	Iron	500x75	NULL	-35.0161	95.342	-155.32
3	TRUE	Item_003	Room 4	Wellknown Su...	Iron	500x75	1:"Here Iron material look ok, but not in the li...	-35.0283	95.342	-165.32
4	FALSE	Item_003	Room 2	Wellknown Su...	Iron	500x75	1:"Dont like the materials in this room"/2:"Ma...	-57.0455	95.342	-165.32

Рисунок 3.22 Приклад вихідного файлу CSV. [авторська розробка]

У файлі журналу є дві різні групи інформації: фіксовані елементи дизайну та нові коментарі. Перше стосується фіксованих елементів дизайну, підготовлених для огляду в проекті, та містить такі дані, як ідентифікатор, пов'язана інформація та координати. Друга група формується з нових коментарів, і має дані ідентифікатора, коментар, пов'язаний з ним об'єкт та його координати.

Щоб мати можливість представляти інформацію в Revit, було створено два нових сімейства Revit, які розміщуються в координатах, отриманих з файлу журналу, крім восьми параметрів Revit, які використовуються як власники інформації, отриманої від програми віртуальної реальності. Ці елементи мають подібну структура, яка використовується для позначення кнопок у додатку віртуальної реальності, щоб створити графічну ідентичність елементів.

Подібно до логіки зміни кольору при застосуванні всередині додатку віртуальної реальності. У Revit було створено графічний фільтр, який змінює колір сімей відповідно до статусу. Це призначено для роботи як трекера проблем для зацікавлених сторін проекту, що працюють у файлі Revit. Червоний - звичайно використовується як колір, який вказує на проблему, через те, що в фільтрі колір елемента змінюється на червоний, коли параметр UNT_Valid рівний як false, так само якщо елемент підходить, колір змінюється на зелений. Тому,

використовуючи різні кольори, користувачі можуть відстежувати, чи проблеми вже були вирішені.

Значення, розділене комами, має прямолінійну структуру, що дозволяє коду Dynamo імпортувати данні без необхідності програмування великого коду, для імпортування потрібно лише три вузли. Кожен рядок файлу блокноту відповідає одному списку в Dynamo, і це полегшує управління інформацією всередині плагіна.

Логіка, створена для імпорту інформації з кодом «Dynamo», проста, і її можна коротко пояснити. Код має чотири основні групи вузлів. Перша відповідає за читання файлу CSV, створеного Unity, і перетворення значень у списки, які будуть використовуватися в «Dynamo». Після цього існує 3 групи вузлів, які керують значеннями, отриманими з файлу CSV, видаляючи нульові значення та розділяючи списки на два (один для фіксованих елементів дизайну і один для нових коментарів). Третя група несе відповідальність за створення елементів Revit у моделі BIM і розміщує її відповідно до отриманих координат. Нарешті, четверта група відповідає за встановлення всіх параметрів з відповідними значеннями в моделі BIM.

3.4 Побудова та тестування створеного програмного забезпечення

Для побудови та тестування створеного програмного забезпечення необхідно використати реальний проект дизайну. Для дослідження було обрано проект приміського будинку.

Для використання моделі у системі необхідно моделювати з дотриманням суворих стандартів, щоб проводити всі необхідні дослідження і використовувати це дослідження як достовірне джерело документації для проекту. У цій роботі модель BIM була створена безпосередньо розробниками програмного забезпечення.

Моделювання цього проекту базується на планах CAD. Важливо підкреслити, що використовувана модель була розроблена спонтанно. З цієї

причини LOD проекту не відповідає найвищим стандартам. Більшість елементів цифрової моделі мають головну мету як джерело документації, в якому не потрібно мати елементи з точною геометрією.

Оскільки ця модель призначена для перевірки роботи програмного забезпечення, було обрано лише однин поверх з будинку.

Для реалізації програми необхідно враховувати ряд змінних, що впливають на робочий процес, наприклад, наявність моделі BIM, розмір проекту, рівень деталізації, та інші особливості, які можуть різнитися у кожному проекті. Наприклад, якщо проект не має моделі BIM, вся частина Revit є не обов'язковою, а додаток може бути запущений як незалежна програма.

Послідовність робочих процесів, розроблена для цього випадку, показана на рис.3.23. Коротко кажучи, геометрія елементів моделі BIM імпортується, а потім адаптується для використання у віртуальній реальності, згодом, вона реалізує елементи інтерактивності та створює вихідний файл. Коли файл готовий, відбувається перевірка дизайну, а пізніше зворотній зв'язок вставляється назад у модель BIM. Враховуючи, що дизайн компанії часто розробляють свої проекти за допомогою BIM, робочий процес, представлений у цьому розділі, є практичним робочим процесом, який потенційно може використовуватися для широкого кола проектів.

Рисунок 3.23 Послідовність процесів у моделі прототипу. [авторська розробка]

Перша дія, щоб розпочати підготовку програми, полягає у виборі площі бажаних елементів. Зазвичай у більшості компаній значна кількість компонентів,

що використовуються в моделі BIM, не оптимізовані для використання в віртуальній реальності. Як було сказано раніше, основними причинами цього є або висока складна геометрія елементів, таких як меблі зі складною геометрією, що потенційно може погіршити частоту кадрів в секунду або спровокувати втрату матеріалу на стадії експорт, у що робить модель непридатною для реалізації проекту віртуальної реальності.

Revit враховує при експорті файлу FBX елементи, які відображаються у вибраному 3D-перегляді. Тому одним із способів оптимізації експорту є вибір елементів, які необхідно експортувати, приховавши всі небажані елементи, як це можна побачити на малюнку.

Зокрема, за цією моделлю BIM, після тестування експорту FBX з елементами проекту, було зроблено висновок, що найбільш продуктивним шляхом, враховуючи специфікації проекту, крім приховування всіх інших рівнів також було приховано всі двері та вікна через втрату матеріалу, який робив їх непридатними для використання в Unity. Таким чином, процес експортування в Revit здійснюється шляхом виконання наступних кроків: Виберіть 3D-зображення → Приховати всі небажані елементи → Меню Revit → Експортувати FBX.

Після створення файлу FBX наступним кроком буде імпортувати всю геометрію до редактору Unity. Це робиться просто перетягуванням файлу FBX на екран редактора. Важливо, щоб імпортована геометрія не змінювалася, оскільки координати елементів з файлу Revit однакові в Unity, і, змінюючи геометрію, ця інформація втрачається.

Наступним кроком є адаптація моделі та додавання всіх бажаних деталей. Оскільки проект містить в основному геометрію стін та меблів, доречно додати всі інші об’єкти квартири, щоб модель проекту була прийнятною.

Існують 3 варіанти, як це описано раніше, для додавання елементів до сценарію віртуальної реальності: використання об’єктів з магазину об’єктів Unity, імпорт FBX або іншого 3D-формату файлів до проекту, а також створення об’єкта за допомогою інструментів Unity. Більшість об’єктів, використовуваних у даному

випадку, були завантажені з магазину об’єктів Unity, за деякими винятками, такими як двері та вікна.

Для організації об’єктів, хороша практика полягає в тому, щоб створити бібліотеку об’єктів (одну сцену з усіма елементами), як таку, що була створена для цього випадку, як показано на рисунку 3.24, з метою полегшення багаторазового використання об’єктів у майбутніх проектах.

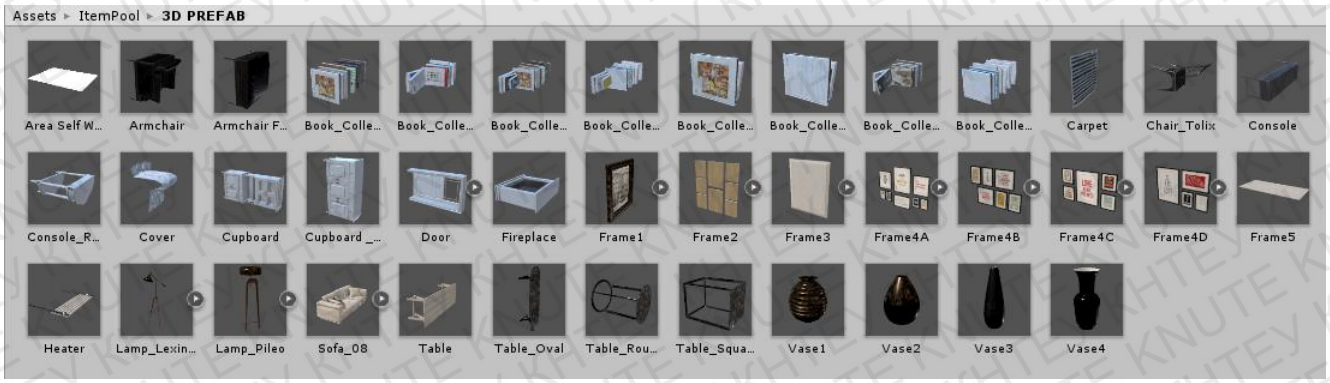

Рисунок 3.24 Бібліотека об’єктів у прототипі.[авторська розробка]

Наглядним є те, що з кожним проектом що розроблюється дизайнером, збільшується бібліотека об’єктів. Маючи розвинену бібліотеку, багаторазовість об’єктів значно підвищує продуктивність. Бібліотека може містити ряд елементів, таких як меблі, двері, вікна, , освітлення, матеріали, текстура, кольори, системи MEP тощо.

Unity має набір інструментів, які дозволяють користувачеві легко розташувати елементи у правильному положенні, легко переміщаючи об’єкт, обертаючи, змінюючи масштаб і змінюючи координати елемента. Використовуючи велику кількість елементів, інтегрованих у бібліотеку, беручи цей документ як довідковий матеріал, більша частина роботи, необхідної для сценарію віртуальної реальності, здійснюється шляхом розміщення елементів бібліотеки та, в кінцевому результаті, адаптації до сценарію шляхом зміни масштабу, обертання чи позиції.

Модель BIM, імпортована в даному випадку, не мала електричних світильників, світлових вимикачів, розеток, виходів Ethernet, телефонів та кабельного телебачення. Оскільки це важливі елементи, які потенційно можуть викликати проблеми з позиціонуванням (ергономіка) – їх додавали в ручному

режимі, зокрема тестуючи можливість додавання та позиціонування додаткових об’єктів.

Завершивши модель віртуальної реальності, необхідно помістити інтерактивні кнопки із запланованим стаціонарним. Цей пункт може суттєво відрізнятися відповідно до вимог та завдань проекту віртуальної реальності. У цьому дослідженні було вирішено, що фіксованими елементами, найбільш важливими для вміщення інтерактивної кнопки, є: меблі та нагрівальні елементи.

Щоб підвищити продуктивність, було створено цілий ряд елементів, які дозволяють дизайнерові просто перетягнути елементи, щоб включити їх до сцени віртуальної реальності. Одним з таких збірних елементів є інтерактивні кнопки, після розміщення інтерактивних кнопок на всіх бажаних елементах, необхідно заповнити інформацію про обладнання .

Коли всі елементи сценарію віртуальної реальності готові, важливо, щоб всі кнопки сцен і входів були перевірені на помилки. Програмне забезпечення Unity має консоль, яка дозволяє дизайнеру перевірити та виправити помилки, якщо деякі функції працюють не належним чином. Для більшості скриптів, які були розроблені для цього випадку дослідження (представлені в доповненнях цієї роботи), були реалізовані налагоджувальні повідомлення, які дозволяють розробнику краще контролювати та розуміти, як працює сценарій віртуальної реальності в режимі реального часу. Як можна побачити на рисунку [картинка та її номер], кожного разу, коли користувач у програмі натискає інтерактивну кнопку та змінює значення, на дисплеї з’являється повідомлення про зміну значень, щоб розробник міг відслідковувати можливі помилки.

Перед створенням кінцевого виконуваного файлу для запропонованої програми важливо перевірити ряд умов, таких як:

- перевірити, чи правильно працює радіальне меню віртуальної реальності.
- перевірити, чи правильно працюють такі інструменти, як лінійка та створення коментарів

- перевірити, чи правильно працює розпізнавання мовлення з коментаря.
- перевірити, чи панель звітів оновлюється та функціонує належним чином.
- перевірити, чи кнопка експорту (створення файлу журналу) працює правильно.

Якщо програма працює, як це передбачено, на наступному кроці необхідно створити виконуваний файл. Щоб створити файл, необхідно вибрати сцени, які будуть включені в програму, і натиснути значок збірки (рис.3.25). Після завершення збірки – необхідно перевірити функціонування програми - понатискати на кожній інтерактивній кнопці, щоб побачити, чи вони відкривають панель з правильною відповідною інформацією.

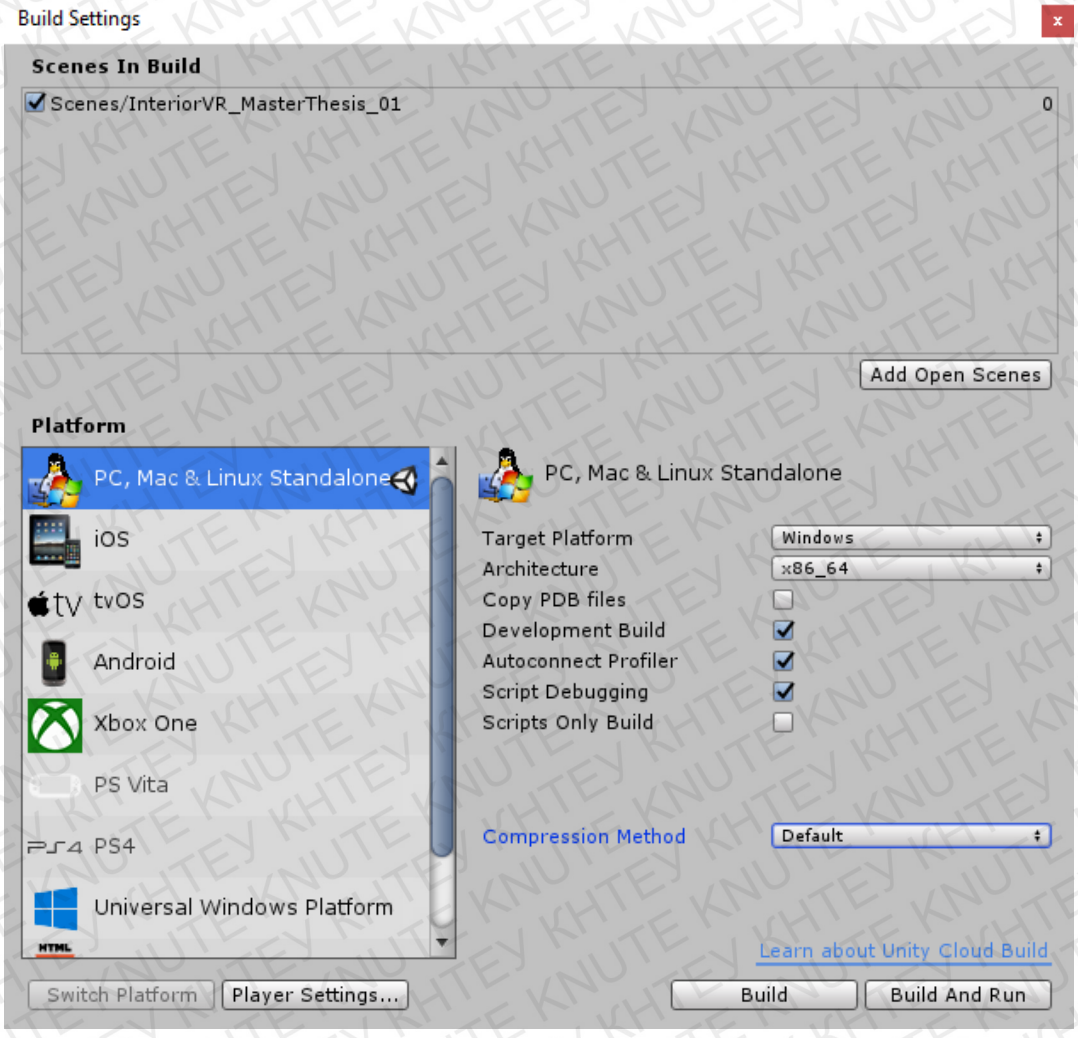

Рисунок 3.25 Створення exe файлу. [авторська розробка]

Тепер, коли файл програми готовий до використання, програма повинна бути протестована окремо або групою зацікавлених сторін проекту, щоб проаналізувати модель віртуальної реальності та перевірити функціонування основних функцій та створити коментарі до проекту.

Щоб довести роботоспроможність цієї програми, всі інтерактивні кнопки були перевірені, з додаванням декількох коментарів. Було створено файл журналу, його можна побачити на малюнку 3.24.

Було використано код Dynamo, описаний у розділі 2, для імпорту всієї інформації з файлу CSV (файл логу) у модель BIM. У файлі Revit було вставлено 17 загальних моделей за допомогою коду Dynamo, в яких 14 були фіксованими, а 3 були створені коментарями. Інформація що імпортується може бути використана в рамках моделі BIM як трекер проблем та побажань змін. Один з елементів, вставлених у файл Revit, вказує на те, що матеріал килима повинен бути змінений. Поки матеріал килимового покриття не переназначено, матеріал залишатиметься червоним через вказівку на тестування з параметром UNT_Valid як false.

Висновки до розділу 3

У даному розділі було розглянуто та скомбіновано найоптимальніший життєвий цикл, а саме у даній роботі обрано гібридну модель – у поданні Мунассара і Говардхана.

Було проаналізовано такі етапи розробки як побудова та тестування розробленого нами прототипу програмного забезпечення для дизайнерів за допомогою технологій віртуальної реальності.

Також ми описали процес побудови та створення прототипу програмного забезпечення для дизайнерів за допомогою технологій віртуальної реальності на базі системи Unity. Була проаналізована можливість сумісного двостороннього використання Revit – Dynamo – Unity. Даний аналіз був необхідний для перевірки життєспроможності використання Revit та Dynamo з аналогічними бібліотеками що використовує система Unity, для майбутньої побудови автономного додатку.

ВИСНОВКИ

Основна пропозиція цієї наукової роботи створення інтерактивного спільного інструменту, що використовує віртуальну реальність.

Дослідження, орієнтоване на будинок, допомогло зробити висновок, що реалізована програма може бути легко застосована з невеликими зусиллями до будь-якого типу дизайну інтер'єрів. Модель віртуальної реальності забезпечила надійний віртуальний прототип області вивчення, який може бути використаний як інструмент співпраці завдяки елементам інтерактивності та спільного використання з учасниками проекту, а також інтуїтивно зрозумілому застосуванню, що не вимагає глибокого вивчення програмного забезпечення.

Результати дослідження представлені в наступному списку:

- Проведено дослідження академічної літератури про віртуальну реальність;
- Створено двосторонній інформаційний зв'язок між Revit та Unity з мінімальними зусиллями для розробника;
- Розроблено прототип нового середовища для дизайнерів з використанням технологій віртуальної реальності;
- Розроблено ефективний користувальницький інтерфейс в віртуальній реальності;
- Досягнуто ефективний робочий процес для реалізації віртуальної реальності в проекті дизайну з гарним виглядом і оптимальною швидкістю роботи;

Таким чином, середовище віртуальної реальності, незважаючи на те, що воно повільно розвивається в плані використання на професійному рівні в галузі промисловості та дизайну, є перспективним та тенденція його застосування буде постійно зростати у значній кількості дизайнерських компаній через легку адаптацію з моделями BIM.

Оскільки ця робота є початковою розробкою прототипу інструменту для дизайнерів з використанням віртуальної реальності, існує два явних шляхи, які

слід розглядати як майбутні дослідження: створення повної програми віртуальної реальності через інше програмне забезпечення або вдосконалення програми, розробленої в цій роботі. Таким чином, наступний перелік представляє пропозиції, які можуть бути розроблені у майбутніх дослідженнях:

- Оптимізувати код програми та робочий процес, щоб бути ще більш ефективним;
- Розробити варіант моделювання проекту для інтеграції з дизайн проектами, які вже завершені, використовуючи такі технології, як лазерне сканування;
- Оновити програму для нових форм взаємодії під час занурення в віртуальну реальність, таких як: створення 3D-анотацій, переміщення елементів за допомогою фізичного контролера, створення розділів (для аналізу деталей конструктивності);, інтегрувати режим польоту, змінювати положення сонця (аналізувати природне освітлення), інструмент «віртуальна камера», щоб мати змогу робити скріншоти та надсилати їх зацікавленим сторонам;
- Інтеграція фактору часу в модель віртуальної реальності, щоб мати змогу переглядати та аналізувати проект протягом усього часу;
- Впровадження багатокористувацьких можливостей, що дозволяють більше ніж одному користувачеві одночасно взаємодіяти з тією ж проектною моделлю;
- Дослідити можливості програми до управління об’єктами - дозволяючи користувачеві інтуїтивно керувати та взаємодіяти з інформацією про дизайн під час занурення в віртуальну модель;
- Розробка програми, яка інтегрує доповнену реальність та віртуальну реальність в одну і ту ж віртуальну модель, з інструментами доповненої реальності, які будуть використовуватися на фізичному дизайн майданчику, та інструменти віртуальної реальності, які будуть використовуватися переважно дизайнерами та консультантами;

- Пристосувати один і той же робочий процес для інтеграції з різними програмними продуктами BIM, такими як Bentley Architecture, ArchiCAD, Vectorworks тощо.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Qiang Li, Qian Cao (2013). «The Novel Virtual Reality Fixture Design and Assembly System (VFDAS)» - Springer, Berlin, - ст. 337- 342.
2. Yu, K.M., Lam, T.W., Lee, A.H.C. (2013). «Immobilization check for fixture design» Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture (7), - ст. 499-512.
3. Onosato, M.; Teramoto, K. and Osaki, S.: Virtual manufacturing systems as advanced information infrastructure for integrated manufacturing resources and activity. 2011, CIRP, - ст. 335-338
4. Sherman, W.R.; Craig, A.B. Understanding Virtual Reality: Interface, Application, and Design; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2002
5. Zonghui Wang ; Xiaohong Jiang ; Jiaoying Shi. 2004. HIVE: a highly scalable framework for DVE, IEEE Virtual Reality 2004, Chicago, USA
6. Sherman, W.R.; Craig, A.B. Understanding Virtual Reality: Interface, Application, and Design; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2002.
7. Beier, K. (2008). Virtual Reality: A Short Introduction. Virtual Reality Laboratory, University of Michigan.
8. Helander, M.G. (Ed.) Handbook of Human-Computer Interaction; Elsevier: Amsterdam, The Netherlands, 2014.
9. Hartmann, T.; Wirth, W.; Vorderer, P.; Klimmt, C.; Schramm, H.; Böcking, S. Spatial Presence Theory: State of the Art and Challenges Ahead; Springer: Cham, Switzerland, 2015.
10. Harrell, F. Phantasmal Media. An Approach to Imagination, Computation and Expression; MIT Press: Cambridge, MA, USA, 2013.
11. Queirós, A.; Silva, A.; Alvarelhão, J.; Rocha, N.P.; Teixeira, A. Usability, accessibility and ambient-assisted living: A systematic literature review. Univers. Access Inf. Soc. 2015.

12. Beier, K. (2008). Virtual Reality: A Short Introduction. Virtual Reality Laboratory, University of Michigan.
13. Chryssolouris, G., V. Karabatsou, D. Mavrikios, D. Fragos, K. Pistiolis, Petrakou, H. (2000). «A virtual environment for assembly design and training». Proceedings of the 33 rd International CIRP Seminar on Manufacturing Systems, Stockholm, Sweden, (June 2000), pp. 326-330.
14. Rossmann, J.; Buecken, A.; Hoppen, M.; Priggemeyer, M. Integrating Virtual Reality, Motion Simulation and a 4D GIS. Res. Urban. Ser. 2016,
15. OpenGL Documentation. [Електронний ресурс]. – Режим доступу: https://www.opengl.org/documentation/
16. IEEE 1003.9-1992 - IEEE Standard for Information Technology - POSIX(R) FORTRAN 77 Language Interfaces - Part 1: Binding for System Application Program Interface (API). IEEE. 24 November 2018.
17. Eastgate, R. (2001). «The Structured Development of Virtual Environments: Enhancing Functionality and Interactivity ». PHD Thesis, University of Nottingham
18. Donalek, C.; Djorgovski, S.G.; Cioc, A.; Wang, A.; Zhang, J.; Lawler, E.; Yeh, E.; Mahabal, A.; Graham, M.; Drake, A.; et al. Immersive and collaborative data visualization using virtual reality platforms. In Proceedings of the IEEE International Conference on Big Data - Washington, DC, USA - 2014; - ст. 609 - 614.
19. Franco, A.O.; González, J.F. Realidad virtual: Un medio de comunicación de contenidos. Aplicación como herramienta educativa y factores de diseño e implantación en museos y espacios públicos. Rev. - ст. 185–211.
20. William R. Sherman and Alan B. Craig, «Understanding Virtual Reality – Interface, Application, and Design 2nd Edition» - Morgan Kaufmann Publishers, USA - ст. 6.
21. William R. Sherman and Alan B. Craig, 2018, «Understanding Virtual Reality – Interface, Application, and Design 2nd Edition» - Morgan Kaufmann Publishers, USA - ст. 9.

22.William R. Sherman and Alan B. Craig, 2018, «Understanding Virtual Reality – Interface, Application, and Design 2nd Edition» - Morgan Kaufmann Publishers, USA - ст. 284.

23.Laurel, B., Computers as Theatre (2nd Edition). 2003 - Menlo Park - Ca: Addison Wesley.

24.Merriam-Webster Incorporate. [Електронний ресурс]. – Режим доступу: http://www.m-w.com/cgi-bin/dictionary?book=Dictionary&va=Manipulation.

25.Munassar, N., & Govardhan, A. Comparison between five models of software. International Journal of Computer Science – 2010 - ст. 94–101.

26.Massey, V., & Satao, K. Comparing various SDLC models and the new proposed model on the basis of available methodology. International Journal of Advanced Research in Computer Science and Software Engineering - 2012 - ст. 170–177.

27.Boehm, B. Spiral development: experience, principles and refinements. Technical I CMU/SEI-2000-SR-008, Carnegie Mellon University, Software Engineering Institute, burgh (2000).

28.Balaji, S., Murugaiyan, M., 2012. Waterfall vs. V-Model vs. Agile: A comparative study on SDLC. International Journal of Information Technology and Business Management, ст. 26–30.

29.Mathur, S., & Malik, S. 2010. Advancements in the V-Model. International Journal of Computer Applications IJCA, ст. 30–35.

30.Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. 2001. Manifesto for Agile software development. [Електронний ресурс]. – Режим доступу: http://agilemanifesto.org

31.Sabale, R., & Dani, A. 2012. Comparative study of prototype model for software engineering with system development Life Cycle. IOSR Journal of Engineering, ст. 21–24.

32.Rahmany, N. (2012). The differences between life cycle models—Advantages and disadvantages. [Електронний ресурс]. – Режим доступу:

http://narbit.wordpress.com/2012/06/10/the-differences-between-life-cyclemodels-advantages-and-disadvantages/

33. Munassar, N., & Govardhan, A. (2010). Hybrid model for software development processes. Proceedings of the 11th International Arab Conference on Information Technology.
34. Ma, Z., D. Zhang, and J. Li, A dedicated collaboration platform for Integrated Project Delivery. Automation in Construction – 2018 - ст. 199-209.
35. Scozzari, Collaborating to increase Value and Efficiency, while reducing waste. [Електронний ресурс]. – Режим доступу: http://www.vjscozzariandsons.com/what-we-do/integrated-project-delivery/
36. Autodesk, Built for Bim. 2017. [Електронний ресурс]. – Режим доступу: https://www.autodesk.pt/products/revit-family/overview
37. Dynamo, Dynamo BIM. 2017. [Електронний ресурс]. – Режим доступу: http://dynamobim.org/
38. Dynamo Primer, What's a Code Block? 2017. [Електронний ресурс]. – Режим доступу: http://dynamoprimer.com/en/07_CodeBlock/7-1_what-is-a-code-block.html
39. The Next Web, This engine is dominating the gaming industry right now. 2016. [Електронний ресурс]. – Режим доступу: https://thenextweb.com/gaming/2016/03/24/engine-dominating-gaming-industry-rightnow/
40. Monteiro, M., Desenvolvimento de interfaces tridimensionais para aplicações móveis a partir da tecnologia BIM. Masters Thesis, Faculty of Engineering of University of Porto, 2013.
41. Autodesk, Subscribe to 3DS Max. 2018. [Електронний ресурс]. – Режим доступу: https://www.autodesk.pt/products/3dsmax/subscribe/
42. Shin, D.-H., The role of affordance in the experience of virtual reality learning: Technological and affective affordances in virtual reality, 2017. 34(8): ст. 1826-1836.

43. Unity, User interfaces for VR. 2013. [Електронний ресурс]. – Режим доступу: https://unity3d.com/pt/learn/tutorials/topics/virtualreality/user-interfaces-vr
44. Microsoft, Voice Input in Unity. 2017. [Електронний ресурс]. – Режим доступу: https://developer.microsoft.com/enus/windows/mixed-reality/voice_input_in_unity
45. Oculus, Locomotion. 2017. [Електронний ресурс]. – Режим доступу: https://developer.oculus.com/design/latest/concepts/bplocomotion/
46. Vizard VR. About. [Електронний ресурс]. – Режим доступу: https://www.worldviz.com/vizard
47. Iris VR. About. [Електронний ресурс]. – Режим доступу: https://irisvr.com/
48. Arq VR. About. [Електронний ресурс]. – Режим доступу: http://arqvr.com/
49. The Third Fate. About. [Електронний ресурс]. – Режим доступу: http://www.thethirdfate.com/front/
50. HTC Vive. About. [Електронний ресурс]. – Режим доступу: https://www.vive.com/ru/

ДОДАТКИ

Додаток А

Програмний код інструменту «Лінійка»

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

public class ToolRuler : MonoBehaviour {

   public  GameObject      avatar;
   private GameObject[]    _textFields;
   private Material        _lineRenderMat;
   private bool[]          _isRayCasted;
   private float           _duration = 10.0f;
   private float           _distance;

   private GameObject      _text;
   private int             _lNum;
   private Vector3[]       _positions;
   private Color           _color = Color.green;

   void Update ()
   {

      if (_textFields[0].activeSelf == true)
      {

         if (!_isRayCasted[0] && Input.GetMouseButtonDown(0))
         {

            bool cast = _castRayAndStoreData();

            if (cast)
            {
               _textFields[1].SetActive(true);
            }
         }
         else if (!_isRayCasted[1] && Input.GetMouseButtonDown(0))
         {
```

```
            bool cast = _castRayAndStoreData(1);
        }
    }

    if (_isRayCasted[0] && _isRayCasted[1])
    {

        _distance = Vector3.Distance(_positions[0], _positions[1]);

        GameObject tempLine = new GameObject();

        tempLine.name = "Measure " + _lNum.ToString();
        tempLine.transform.position = _positions[0];
        tempLine.AddComponent<LineRenderer>();

        LineRenderer lineRenderer = tempLine.GetComponent<LineRenderer>();
        lineRenderer.material = _lineRenderMat;
        lineRenderer.widthMultiplier = 0.01f;

        Gradient grad = new Gradient();
        float alphaChanel = 1.0f;
        grad.SetKeys(
            new GradientColorKey[]
            {
                    new GradientColorKey(_color, 0.0f),
                    new GradientColorKey(_color, 1.0f)
            },
                new GradientAlphaKey[] {
                    new GradientAlphaKey(alphaChanel, 1.0f),
                    new GradientAlphaKey(alphaChanel, 1.0f)
            }
        );


        _positions[0] = Vector3.MoveTowards(_positions[0], avatar.transform.position, 0.10f);
        _positions[1] = Vector3.MoveTowards(_positions[1], avatar.transform.position, 0.10f);

        lineRenderer.colorGradient = grad;
        lineRenderer.SetPosition(0, _positions[0]);
        lineRenderer.SetPosition(1, _positions[1]);

        List<GameObject> rendCubes = new List<GameObject>;
```

```
        foreach (var pos in _positions)
        {
            GameObject rendCube = GameObject.CreatePrimitive(PrimitiveType.Cube);
            rendCube.name = "Cube " + pos.ToString();
            rendCube.transform.localScale = new Vector3(0.02f, 0.02f, 0.02f);
            rendCube.transform.position = pos;
            rendCubes.Add(rendCube);
        }

        GameObject tempText = new GameObject();

        tempText.name = "Ruler Text";
        tempText.AddComponent<AlwaysFaceCamera>();

        tempText.transform.position = (_positions[0] + _positions[1]) / 2;
        Vector3 facingDir = _positions[0] - _positions[1];
        tempText.transform.Rotate(-facingDir.x, facingDir.y, facingDir.z);
        tempText.transform.localScale = new Vector3(0.01f, 0.01f, 0.01f);

        TextMesh textMesh = tempText.AddComponent<TextMesh>();

        textMesh.fontSize = 60;
        textMesh.fontStyle = FontStyle.Bold;
        textMesh.text = _distance.ToString("F2") + "m";

        tempText.transform.parent = tempLine.transform;
        foreach (var rendCube in rendCubes)
        {
            rendCube.transform.parent = tempLine.transform;
        }

        GameObject.Destroy(tempLine, _duration);

        _isRayCasted[0] = false;
        _isRayCasted[1] = false;
        _textFields[0].SetActive(false);
        _textFields[1].SetActive(false);
    }
}


private bool _castRayAndStoreData(int rayNum = 0)
{
    Ray        theRay = Camera.main.ScreenPointToRay(Input.mousePosition);
```

```
        RaycastHit  theHit;

        if (Physics.Raycast(theRay, out theHit, 30))
        {

            if (theHit.collider == null)
            {

                Debug.LogWarning("No collision");
                return false;
            }

            _isRayCasted[rayNum] = true;
            _positions[rayNum] = theHit.point;

            return true;
        }

        return false;
    }
}
```

Додаток Б

Програмний код розпізнавання мови використовуючи Windows API

```
using UnityEngine;
using UnityEngine.Windows.Speech;
using UnityEngine.UI;

public class ToolSpeechRecog : MonoBehaviour
{
    public InputField commentInput;
    private DictationRecognizer _dRecognizer;

    void Start()
    {
        _init();
    }

    private void _init()
    {
        _dRecognizer = new DictationRecognizer();
        _dRecognizer.DictationResult += onRecResult;
        _dRecognizer.DictationHypothesis += onRecPredict;
        _dRecognizer.DictationComplete += onRecComplete;
        _dRecognizer.DictationError += onRecError;
        _dRecognizer.Start();
    }

    void onRecResult(string text, ConfidenceLevel confidence)
    {
        commentInput.text = text;
    }

    void onRecPredict(string text)
    {
        // Unused for now
    }

    void onRecComplete(DictationCompletionCause cause)
    {
        if (cause != DictationCompletionCause.Complete)
            return;
    }

    void onRecError(string error, int result)
```

```
    {
        Debug.LogErrorFormat("Error: {0}; Result = {1}.", error, result);
    }
}
```