

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ МАТЕРІАЛЬНОГО
ІНТЕРФЕЙСУ КОРИСТУВАЧА ІГРОВОГО ЗАСТОСУНКУ
SOFTWARE-HARDWARE IMPLEMENTATION OF THE TANGIBLE
USER INTERFACE OF THE GAME APPLICATION

Виконав: студент 4 курсу, групи КНТ-112сп
Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ГРИЦЮК Д.В.

(ПРІЗВИЩЕ та ініціали)

Керівник ПАРХОМЕНКО А.В.

(ПРІЗВИЩЕ та ініціали)

Рецензент ЗЕЛЕНЬОВА І.Я.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ГРИЦЮКА Дениса Віталійовича
 (ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмно-апаратна реалізація матеріального інтерфейсу користувача ігрового застосунку. Software-Hardware Implementation of the Tangible User Interface of the Game Application

керівник проєкту (роботи) к.т.н., доц., ПАРХОМЕНКО Анжеліка Володимирівна,
 (науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “28” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 02 червня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Дослідження особливостей взаємодії через матеріальні інтерфейси користувача. 3. Проєктування програмно-апаратного забезпечення. 4. Конструювання програмно-апаратного забезпечення.

5. Перелік графічного матеріалу (з з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ПАРХОМЕНКО А.В., доцент		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання « 14 » квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдань роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури системи.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

_____ Денис ГРИЦЮК
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

_____ Анжеліка ПАРХОМЕНКО
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 86 с., 6 табл., 26 рис., 2 дод., 69 джерел.

ВІРТУАЛЬНА РЕАЛЬНІСТЬ, МАТЕРІАЛЬНИЙ ІНТЕРФЕЙС, ПРОГРАМНО-АПАРАТНИЙ КОМПЛЕКС, ІГРОВИЙ ЗАСТОСУНОК, ARDUINO, PYTHON.

Об'єкт дослідження – розробка інтерфейсу користувача з ігровим середовищем.

Предмет дослідження – методи та засоби розробки інтерфейсу користувача ігрового середовища.

Мета роботи – дослідження та розробка програмно-апаратного комплексу для покращення досвіду взаємодії користувача з ігровим середовищем через матеріальний інтерфейс.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, платформа Arduino, макетна плата, кнопки, мови програмування C++ та Python, середовища програмування Visual Studio та Arduino IDE, персональний комп'ютер з процесором Intel Core i7-5500U під управлінням операційної системи Microsoft Windows 10 Pro.

Результати. Створено програмно-апаратний комплекс, керований геймпадом на платформі Arduino, що забезпечує взаємодію з ігровим середовищем через матеріальний інтерфейс.

Висновки. Розробка та впровадження матеріальних інтерфейсів дозволяє покращити досвід взаємодії користувачів з середовищами віртуальної реальності.

Галузь використання – розробка ігрових застосунків та середовищ віртуальної реальності для сфери навчання та розваг.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 86 pages, 6 tables, 26 figures, 2 appendixes, 69 sources.

VIRTUAL REALITY, MATERIAL INTERFACE, SOFTWARE AND HARDWARE COMPLEX, GAME APPLICATION, ARDUINO, PYTHON.

Object of study is a process of user interaction with the gaming environment.

Subject of study are methods and tools for developing the user interface of a gaming environment.

The purpose of the work is research and development of a hardware-software complex for interaction with the gaming environment through a tangible interface.

Materials, methods, and tools: structured and object-oriented programming, Arduino platform, breadboard, buttons, C++ and Python programming languages, Visual Studio and Arduino IDE programming environments, personal computer with Intel Core i7-5500U processor running Microsoft Windows 10 Pro operating system.

Results. The hardware-software complex has been created, controlled by a gamepad based on the Arduino platform, providing interaction with the gaming environment through a tangible interface.

Conclusions. The development and implementation of tangible interfaces allows to improve user interaction experience with Virtual Reality environments.

The field of use is a development of gaming applications and Virtual Reality environments for education and entertainment.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	9
1.1 Класифікація середовищ віртуальної реальності	9
1.2 Особливості розробки програм віртуальної реальності	12
1.3 Розробка ігрових середовищ	17
1.4 Проєктування інтерфейсу користувача ігрових застосунків	23
1.5 Висновки до розділу 1	27
2 Дослідження особливостей взаємодії через матеріальні інтерфейси користувача.....	28
2.1 Досвід застосування.....	28
2.2 Матеріальний інтерфейс користувача	29
2.3 Огляд прикладів реалізації матеріальних інтерфейсів.....	32
2.4 Постановка завдань дипломної роботи.....	34
3 Проєктування програмно-апаратного забезпечення	36
3.1 Призначення та вимоги до системи	36
3.2 Модель взаємодії MVC.....	36
3.3 Вибір інструментарію розробки	39
3.4 Висновки до розділу 3	43
4 Конструювання програмно-апаратного забезпечення	44
4.1 Алгоритм генерації лабіринту	44
4.2 Програмно-апаратна реалізація геймпаду на платформі Arduino.....	48
4.3 Програмна реалізація ігрового середовища	52
4.4 Висновки до розділу 4	58
Висновки	59
Перелік джерел посилань	60
Додаток А Текст програми.....	68
Додаток Б Слайди презентації	79

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

AR	– Augmented Reality;
CLI	– Command Line Interface;
GUI	– Graphical User Interface;
HMD	– Head Mounted Device;
HUD	– Heads-Up Display;
ROS	– Robot Operating System;
SDK	– Software Development Kit;
TEI	– Tangible and Embodied Interaction;
TUI	– Tangible User Interface;
UI	– User Interface;
UX	– User eXperience;
VE	– Virtual Environment;
VUI	– Voice User Interface;
VR	– Virtual Reality;
ПЗ	– програмне забезпечення;
ПК	– персональний комп'ютер.

ВСТУП

Зі стрімким розвитком науки та техніки застосування технології віртуальної реальності (Virtual Reality, VR) стає все більш поширеним. Завдяки зануренню, інтерактивності та розширюваності технологія VR стала багатообіцяючою технологією з широкими перспективами застосування [1].

Сьогодні найбільші світові технологічні компанії (Google, Facebook, Microsoft, Apple) інвестують мільярди в технології розробки VR. Експерти кажуть, що ця революція VR буде більшою, ніж поява мобільних телефонів, і компанія, яка першою розробить споживчий продукт VR, прийнятий масами, виграє. За словами Брендана Ірібе: «Нарешті ми позбудемося 2D-монітора. Це було вікно у VR, у яке ми всі дивилися протягом 30–40 років». При цьому розробники мають забезпечити максимально природну та інтуїтивно зрозумілу взаємодію з платформами VR, тобто таку, що більше не потребує адаптації до обмежених форм-факторів комп'ютера (клавіатура, миша, сенсорний екран) [2].

Створення VR не надто відрізняється від процесу розробки веб або мобільного застосунку та включає особи користувача, концептуальні потоки, каркаси та модель взаємодії. Тим не менш, хоча технологія VR набирає обертів, однак бракує методів і найкращих практик її проєктування. Зокрема, актуальним є подальший розвиток технологій реалізації матеріального користувальницького інтерфейсу для кращої взаємодії з середовищем віртуальної реальності, які дозволять взаємодіяти з цифровим світом так само, як людина взаємодіє з фізичним світом.

Мета роботи – дослідження та розробка програмно-апаратного комплексу для покращення досвіду взаємодії користувача з ігровим середовищем через матеріальний інтерфейс.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Класифікація середовищ віртуальної реальності

VR - це середовище, створене комп'ютером, яке виглядає та здається реальним для людини, яка його відчуває. Це симуляція, яка дозволяє користувачам відчувати інше середовище в 3D, надсилаючи сигнали через носимі інтерактивні пристрої, зокрема окуляри, навушники, костюми, рукавички, тощо. У деяких випадках користувачі можуть навіть торкатися об'єктів у віртуальному просторі. Віртуальна реальність використовує рухи тіла, щоб контролювати те, що відображається на екрані користувача [3].

VR – це змодельований цифровий досвід, який може бути схожим на реальний світ або повністю відрізнятися від нього. Досвід VR створюється за допомогою комп'ютерних технологій і надається користувачеві через гарнітуру VR або наголовний дисплей (HMD), який створює захоплююче та інтерактивне тривимірне середовище. VR слід відчувати та взаємодіяти з ним так, ніби це віртуальне середовище було б реальним [4].

Існують наступні типи VR [3]:

- з повним зануренням - забезпечує реалістичну взаємодію з середовищем, надає користувачам симуляцію відчуття, що вони присутні в середовищі за допомогою апаратного забезпечення (HTC Vive, Oculus Rift або Google Cardboard);
- без занурення - користувач не взаємодіє безпосередньо з середовищем, але може контролювати діяльність за допомогою програмного забезпечення (ПЗ) (наприклад, відеоігри);
- з частковим зануренням - поєднує два попередні типи VR, користувачі можуть контролювати дії та відчувати реалістичну взаємодію;
- спільна віртуальна реальність - користувачі з різних місць спілкуються за допомогою аватарів і 3D, які використовуються в основному для співпраці та робочих просторів;

- доповнена реальність (Augmented Reality, AR) – передбачає інтеграцію цифрового вмісту з реальним середовищем користувача, зокрема додає цифрові елементи до перегляду в реальному часі за допомогою камери на смартфоні (наприклад, лінзи Snapchat);

- змішана реальність – досвід, який поєднує елементи як AR, так і VR, де взаємодіють реальні та цифрові об'єкти (HoloLens від Microsoft).

Основними галузями використання VR є [4]:

- навчання та освіта: VR забезпечує безпечне та контрольоване середовище для навчання в різних сферах, таких як медичні процедури, промислові процеси та реагування на надзвичайні ситуації. Це дозволяє практикуватися та вчитися на помилках без реальних наслідків;

- імерсивні розваги: VR надає можливості для захоплюючих розваг, таких як ігри, інтерактивне оповідання та віртуальний туризм. Це дозволяє користувачам стати активними учасниками своїх улюблених історій і досліджувати нові світи;

- віддалена співпраця та взаємодія: VR полегшує віддалену співпрацю та спілкування, також відоме як Social VR. Користувачі з різних місць можуть зустрічатися, працювати та спілкуватися разом у спільних віртуальних просторах, які пов'язані з ширшою концепцією метавсесвіту;

- здоров'я та благополуччя: віртуальна реальність є багатообіцяючою в терапевтичних цілях, включаючи консультування, терапію фобій, зниження стресу та лікування болю;

- дизайн і прототипування: VR може трансформувати дизайн продукту, оскільки дозволяє дизайнерам створювати віртуальні прототипи та взаємодіяти з ними.

До існуючих обмежень VR можна віднести такі [4]:

- вимоги до апаратного забезпечення: для високоякісного досвіду VR часто потрібне потужне та дороге обладнання, наприклад гарнітури VR високого класу та потужні персональні комп'ютери (ПК). Це може обмежити

доступність і прийняття, особливо для користувачів, які не можуть дозволити собі необхідне обладнання;

– негативні наслідки для здоров'я. Хвороба руху (також відома як кіберхвороба) є загальновідомою проблемою VR. Напруга очей і навіть судороги також відмічаються серед інших негативних наслідків для здоров'я. Саме хвороба та дискомфорт можуть перешкодити прийняттю VR [4].

Таким чином, класифікацію середовищ VR за типами та галузями застосування можна представити як наведено на рис. 1.1 [3]-[4].



Рисунок 1.1 – Класифікація середовищ VR за типами та галузями застосування [3]-[4]

1.2 Особливості розробки програм віртуальної реальності

Програми VR складаються з двох типів компонентів: середовища та інтерфейсів. Усі програми VR можна розташувати по двох осях відповідно до складності цих двох компонентів (рис. 1.2) [5].

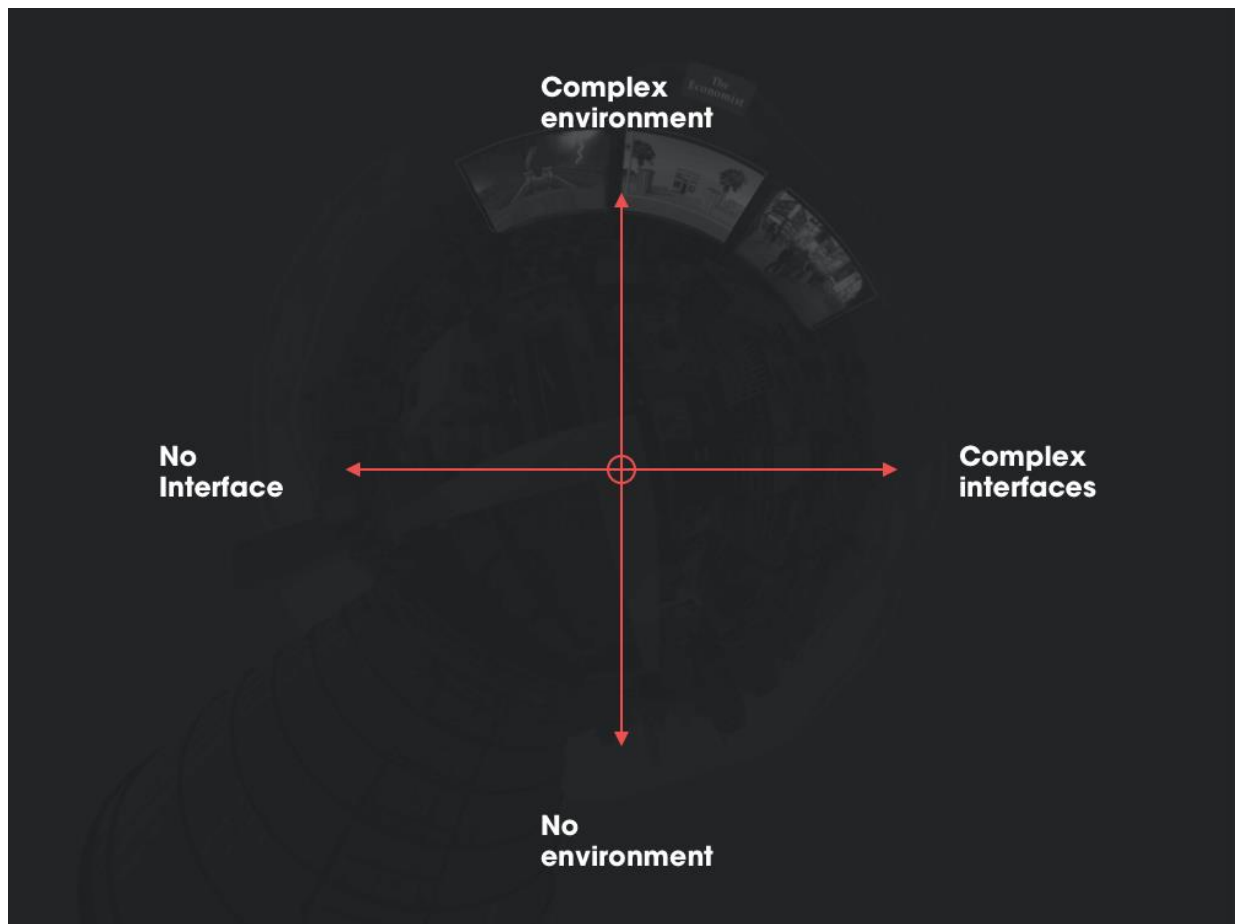


Рисунок 1.2 – Програми VR в залежності від складності компонентів [5]

У верхньому лівому квадранті знаходяться, наприклад, симулятори американських гірок, які мають повністю сформоване середовище, але взагалі не мають інтерфейсу та розраховані лише на поїздку гравця. У протилежному квадранті знаходяться застосунки, які мають розвинений інтерфейс, але мало або зовсім не мають середовища (наприклад, Samsung's Gear VR home screen) [5].

Інтерфейс – це набір елементів, з якими користувачі взаємодіють, щоб орієнтуватися в середовищі та контролювати свій досвід. Програми VR на основі інтерфейсу працюють відповідно до тієї ж базової динаміки, що й традиційні програми. Хоча процеси для розробки інтерфейсів VR ще не визначені, враховуючи схожість із традиційними програмами, випробувані робочі процеси розробки мобільних застосунків можна використовувати для створення інтерфейсів користувача VR. Вони містять наступні базові кроки.

Крок 1. Каркаси. Спочатку визначають взаємодії та загальний макет.

Крок 2. Візуальний дизайн. На цьому етапі функції та взаємодії затверджено. Керівні принципи бренду тепер застосовано до каркасів, і створено гарний інтерфейс.

Крок 3. План застосунка - основний орієнтир для розробників. Тут екрани організуються в потоки, створюються зв'язки між екранами та описується взаємодія для кожного екрана [5].

При проєктуванні для VR можна вибрати з трьох основних фреймворків [2]:

- Mozilla A-Frame призначена для Web VR і може використовуватися на таких платформах, як Google Cardboard, Samsung Gear і Oculus Rift;
- Daydream VR призначений для VR середнього класу та працює лише з мобільними телефонами, але призупинений компанією Google в 2019 році;
- Unity VR/Unreal SDK призначені для високоякісних гарнітур, зокрема Oculus Rift, HTC Vive та HoloLens (AR).

Основними вимогами до розробки є [2]:

- фізіологічний комфорт: забезпечення фіксованої точки відліку, такої як лінія горизонту або панель приладів, яка залишається з користувачем під час руху, допомагає контролювати користувача, особливо якщо у застосунку багато рухів — прискорення, масштабування, стрибки;

- екологічний комфорт: так само, як і у фізичному світі, люди можуть легко відчувати дискомфорт у маленьких, великих або високих приміщеннях (клаустрофобія/агорафобія/запаморочення), тому важливо розуміти масштаб середовища. Є багато способів провести користувачів у віртуальному середовищі. Особливо корисними можуть бути непросторові методи, включаючи звук і світло. Аудіо можна використовувати для позиціонування в просторі, а світло можна використовувати для виявлення шляху, яким треба слідувати [2];
- ергономіка: необхідно забезпечити комфортний діапазон рухових зон та дотримуватися рекомендованих схем (рис. 1.3) [2].

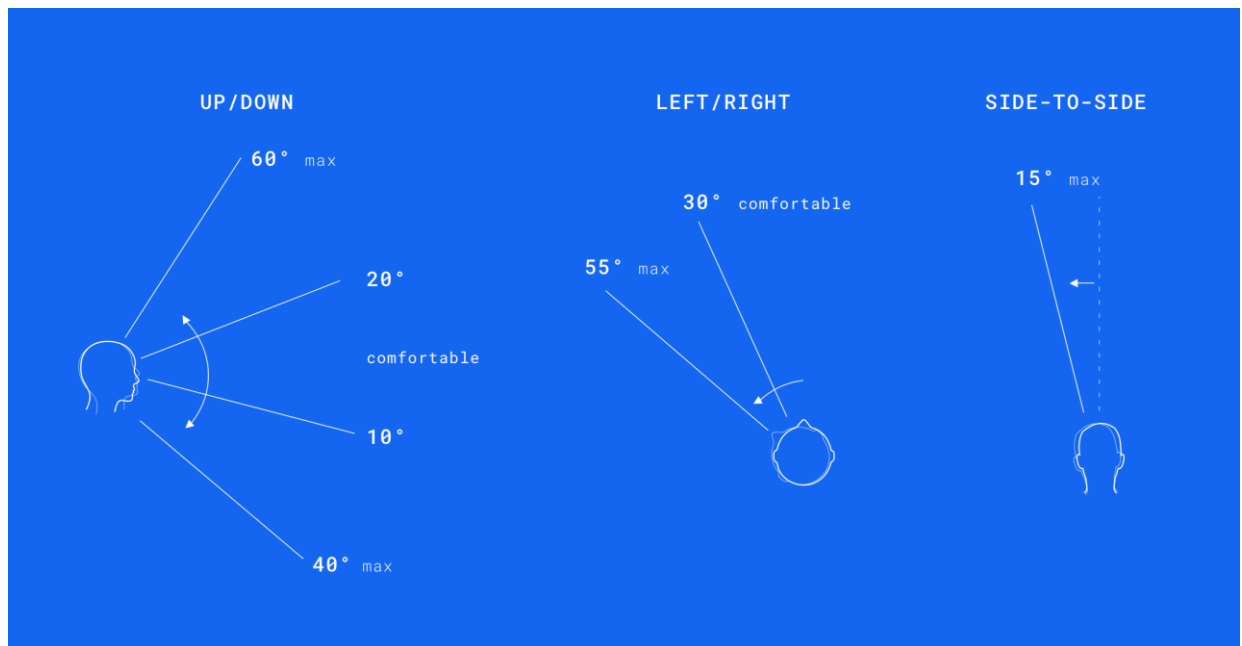


Рисунок 1.3 – Рекомендовані діапазони рухових зон [2]

Необхідно враховувати також існуючі методи введення за допомогою ручних контролерів VR, які мають бути придатними для використання в «сліпому режимі». Контролери покращують досвід VR, дозволяючи користувачам брати речі та фізично взаємодіяти з іншими [2].

Звук є невід’ємною частиною розробки VR. Оскільки у VR досі відсутній тактильний зворотний зв’язок, звук є чудовим способом забезпечити зворотний зв’язок, коли користувачі торкаються об’єктів [2].

Історично цифрові інтерфейси були створені відповідно до апаратних вимог 2D-екранів. Дизайнери вписували вміст і навігацію в рамки дисплеїв, перетворюючи реальний досвід на піктограми та інші елементи інтерфейсу користувача. Проектування для VR має означати не перенесення практики 2D у 3D, а пошук нової парадигми.

Існує багато типів інтерфейсів користувача (User Interface, UI), які використовуються в різних контекстах, зокрема [6]:

- графічний інтерфейс користувача (Graphical user interface, GUI): використовує графічні елементи, такі як піктограми та вікна, для полегшення взаємодії, які зазвичай зустрічаються в операційних системах і програмах;
- інтерфейс командного рядка (Command Line Interface, CLI): використовує текстові команди для взаємодії з користувачем, які часто використовуються розробниками та системними адміністраторами для точного контролю над операціями;
- голосовий інтерфейс користувача (Voice User Interface, VUI): забезпечує взаємодію за допомогою голосових команд і відповідей, популяризованих віртуальними помічниками, такими як Siri та Alexa;
- сенсорний інтерфейс користувача: дозволяє користувачам взаємодіяти з пристроями за допомогою сенсорних жестів, поширених у смартфонах, планшетах та інтерактивних кіосках;
- користувацький інтерфейс AR: інтегрує віртуальні елементи в природне середовище, які зазвичай можна побачити в програмах AR для навігації, ігор та освіти;
- інтерфейс користувача VR: створює захоплюючі віртуальні середовища для взаємодії з користувачем, які використовуються в іграх VR, моделюванні та навчальних програмах [6].

Дизайн UI відіграє важливу роль у створенні зручних для користувачів цифрових продуктів. Добре розроблений інтерфейс користувача може підвищити зручність використання, задоволеність користувачів і

утримання клієнтів, що призведе до конкурентної переваги, зниження витрат, збільшення прибутку та загального успіху програми [6].

Одна з головних цілей дизайну інтерфейсу користувача — передбачити, що може знадобитися зробити користувачам, і переконатися, що в інтерфейсі є елементи, до яких легко отримати доступ. В основі дизайну UI лежить мета забезпечити винятковий досвід користувача, який співпрацює з інтерфейсом користувача. Добре розроблений UI робить навігацію простою, залишає користувачів задоволеними та гарантує, що програмний продукт виділятиметься серед цифрового різноманіття [6].

Загалом успіх програми залежить від залучення користувачів, а добре розроблений інтерфейс користувача не лише приваблює користувачів спочатку, але й утримує їх з часом. Інтерфейс користувача зосереджується на естетиці та відіграє важливу роль в успіху будь-якого цифрового продукту [6].

Існує кілька основних елементів інтерфейсу користувача, зокрема інформаційна архітектура, інтерактивний дизайн і візуальний дизайн. UI зосереджується на якості взаємодії користувача з продуктом, тоді як користувацький досвід (User eXperience, UX) передбачає дослідження ринку та визначення потреб користувачів. Інтерфейс користувача має більш художні компоненти дизайну, які стосуються зовнішнього вигляду продукту, а UX зосереджується на визначенні потреб користувачів [6].

Дизайн UI поєднує в собі елементи дизайну взаємодії, візуального дизайну та інформаційної архітектури. Інтерфейс включає різні компоненти, такі як елементи керування введенням, навігаційні компоненти, інформаційні компоненти та контейнери [6].

Процес створення UI передбачає розробку зовнішнього вигляду та функціональності програми. Це включає дослідження брендингу та дизайну, адаптацію інтерфейсу до різних пристроїв, забезпечення інтерактивності та анімації, а також впровадження дизайну в дію. Дуже важливо переконатися,

що дизайн інтерфейсу користувача є зручним, візуально привабливим і простим у навігації, щоб користувачі могли мати бездоганний досвід [6].

Таким чином, у сфері UI існує потреба в тому, щоб середовище VR було естетично привабливим, відповідало та вирішувало потреби користувачів. А в сфері UX це дозволяє користувачам безперебійно виконувати процеси у віртуальній реальності.

1.3 Розробка ігрових середовищ

1.3.1 Тенденції розробки VR-ігор

Віртуальна реальність стає все більш поширеною в багатьох галузях, однак ігри все ще виділяються як лідери в цій технологічній еволюції [7].

Розробка ігор у віртуальній реальності – це процес, в якому використовуються апаратні засоби VR, зокрема гарнітура, яка імітує тривимірну реальність за допомогою стереоскопічних дисплеїв. За допомогою датчиків ці гарнітури можуть відстежувати положення гравців і рухи очей. Рухи гравця перетворюватимуться на ігрові дії та впливатимуть на його віртуальне середовище (Virtual Environment, VE). Розробка ігор VR також дозволяє гравцям відчувати 3D-звук і тактильний зворотний зв'язок контролера. Таким чином, VR-ігри занурюють гравця у вигадане середовище, надаючи гіперреалістичну перспективу від першої особи [7].

Переваги VR-ігор [7]:

- більше захоплюючих вражень;
- збільшення потенціалу для інновацій;
- більше соціального геймплея;
- більше фізичної активності.

Проблематика VR-ігор з точки зору гравців та розробників [7]:

- високий бар'єр для входу. Згідно зі статистичними даними Statista за 2023 рік, менше 2% людей у всьому світі регулярно грають у віртуальні ігри, у порівнянні з 45%, які грають у традиційні відеоігри. Таке

повільне впровадження можна частково пояснити високою ціною апаратного забезпечення;

- складні графіки розробки: графік розробки традиційних відеоігор є більш усталеним, точно налаштованим процесом. Для проєктів VR потрібно суттєво переглянути звичні робочі процеси, такі як кодування з подальшим відображенням сюжетної лінії та тестуванням. Розробка ігор у VR також передбачає додатковий етап вибору платформи для розробки, оскільки кожна суттєво відрізняється за потужністю обробки;

- підвищений потенціал проблем із графікою: VR-ігри мають багатші візуальні ефекти, що є перевагою для гравців, але ця якість має складні наслідки для розробників ігор. Оскільки гравці ще більше занурені в обстановку, ніж під час гри в традиційні відеоігри, згуртованість і точність набувають більшого значення;

- дискомфорт користувача: досвід VR може призвести до напруги очей і заколисування. Щоб боротися з цим, створюючи гру VR, розробники можуть зосередитися на скороченні часу обробки своєї гри та оптимізації частоти кадрів анімації. Вони можуть включати як активні моменти, так і нерухомі візуальні ефекти, щоб дати гравцям відпочити від руху. Постійні інновації в технології гарнітур VR також допомагають гарантувати, що гарнітури стають більш зручними та легкими [7].

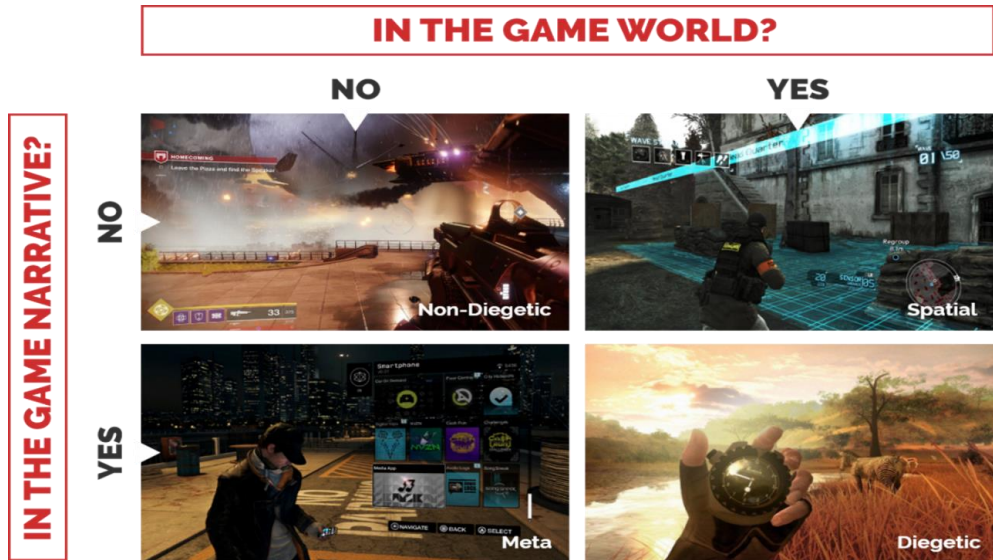
1.3.2 UI/ UX дизайн

Користувальницький інтерфейс (UI) зосереджується на візуальних елементах і тому, як гравці взаємодіють з грою. Користувацький досвід (UX) заглиблюється в загальний досвід користувача, охоплюючи відчуття гравців під час навігації грою [8].

Існує чотири типи візуального представлення, що стосується дизайну інтерфейсу користувача гри: дієгетичне, недієгетичне, просторове та мета (рис. 1.4) [9].



а)



б)

Рисунок 1.4 – Типи візуального представлення а) та приклади рішень б) [9]

Дієгетичні – це елементи інтерфейсу користувача, які може бачити або чути персонаж і які вписуються в контекст історії. Приклади: телефон із видимим екраном, голограми та інші футуристичні гаджети.

Недієгетичні – це елементи інтерфейсу користувача, які існують поза грою чи сюжетом і які бачить лише гравець. Приклади: екрани меню, вікна квестів і панелі здоров'я.

Просторові – це елементи інтерфейсу користувача, які представлені в ігровому просторі, але не видимі для персонажів. Приклади: контури

персонажів, що вказують на ворогів, стрілка, яка показує, куди впаде кинутий предмет, і текстові написи.

Мета – це елементи інтерфейсу користувача, які є контекстними для гри, але не представлені в ігровому просторі. Приклади: текст, що прокручується, або кольорові накладки, які вказують на зміну стану здоров'я гравця [9].

Основні елементи дизайну UI гри включають меню, HUD (Heads-Up Display) та елементи керування [8].

Меню – це основні інструменти навігації, які дозволяють гравцям починати гру, регулювати параметри та отримувати доступ до різних режимів гри. Меню мають бути інтуїтивно зрозумілими, з чіткою ієрархією, яка допомагає гравцям швидко знаходити те, що їм потрібно.

HUD – це накладання, яке відображає важливу ігрову інформацію, таку як здоров'я, боєприпаси, цілі та мапи. Добре продуманий HUD ненав'язливий, але інформативний, надає гравцям всю необхідну інформацію, не перевантажуючи їх [8].

Елементи керування – екранні кнопки для мобільних ігор, конфігурація контролера або клавіатури для консольних і комп'ютерних ігор. Компонування та швидкість відгуку елементів керування в грі є життєво важливими для того, щоб гравці могли легко взаємодіяти з ігровим світом.

Важливість інтуїтивно зрозумілих і візуально привабливих інтерфейсів неможливо переоцінити. Добре розроблений інтерфейс користувача не тільки покращує досвід гравця, але й підтримує його, зменшує розчарування та заохочує до тривалих ігрових сеансів. З іншого боку, захаращений або заплутаний інтерфейс може призвести до розчарування гравців, що зрештою змусить їх покинути гру [8].

UX-дизайн або дизайн користувацького досвіду в контексті ігор стосується процесу підвищення задоволеності гравців шляхом покращення зручності використання, доступності та задоволення від гри. На відміну від інтерфейсу користувача, який має справу з окремими візуальними

елементами, ігровий дизайн UX більше стосується всієї подорожі гравця в грі. Це стосується того, наскільки інтуїтивно зрозуміла гра, наскільки плавні переходи між різними частинами гри та наскільки задовольняє загальний досвід [8].

Ключові елементи ігрового дизайну UX включають подорож гравця, хід гри, зворотній зв'язок.

Подорож гравця - стосується загального шляху, який проходить гравець від моменту початку гри до завершення. Добре продумана подорож гравця веде гравців у гру таким чином, щоб відчувати себе природним і корисним [8].

Хід гри – це темп гри та те, як різні елементи представлені гравцеві. Хороший потік залучає гравців, гарантуючи, що вони постійно відчують виклики, але не перевантажені [8].

Зворотній зв'язок - його механізми, зокрема візуальні чи звукові підказки, необхідні для того, щоб повідомляти гравцям, як вони грають і що їм потрібно робити далі. Це може бути простим, як звуковий ефект, коли вони виконують завдання, або складним, як адаптивний рівень складності [8].

Класифікацію ключових елементів UI/UX дизайну наведено на рис. 1.5 [8].



Рисунок 1.5 – Ключові елементи ігрового UI/UX дизайну

Неможливо переоцінити важливість безперебійної та привабливої взаємодії з користувачем. Коли гравці стикаються з грою, яка є інтуїтивно зрозумілою та приємною для навігації, вони, швидше за все, залишаться зацікавленими, рекомендуватимуть гру іншим і продовжуватимуть грати довше. З іншого боку, погано розроблений UX може призвести до розчарування, зниження рівня утримання гравців і, зрештою, негативно вплинути на успіх гри [8].

1.3.3 Вибір апаратного забезпечення VR-ігор

Гарнітури віртуальної реальності використовуються для занурення користувача в ігрову або робочу обстановку. Деякі гарнітури можуть фіксувати рухи голови та очей і передавати їх у гру. Інші включають стереозвукові системи для проєктування аудіовмісту гри. Деякі поставляються просто як окуляри, які відображають віртуальний світ гри для користувача. Гарнітури VR не дозволяють користувачам бачити реальний зовнішній світ навколо них. Деякі гарнітури також мають вбудовані мікрофони [10].

Контролери дозволяють користувачам взаємодіяти з віртуальним світом. Користувачі можуть вибирати об'єкти, перетягувати об'єкти, вказувати, окреслювати межі та прокручувати вгору чи вниз за допомогою контролера VR. Деякі контролери мають детектори руху, які можуть сприймати рухи рук користувача, наприклад, кидки. Контролер відтворює ці дії в грі [10].

Рукавички VR роблять крок вперед у контролерах VR, виявляючи мікрорухи в руках і пальцях користувача та передаючи їх у гру. Це додає дрібну моторику у відеоігри VR і дозволяє користувачам хапати, брати та носити дрібні предмети під час гри. Деякі рукавички містять тактильну технологію, яка дозволяє користувачам відчувати віртуальний світ гри. Рукавички створюють вібрацію на кінчиках пальців, щоб імітувати

присутність предмета. Деякі також містять крихітні приводи, які зміщують шкіру користувача та застосовують значну силу опору [10].

Деякі інші допоміжні ігрові аксесуари VR:

- кермо та контролери акселератора для гоночних ігор;
- бігові доріжки, які визначають дистанцію та швидкість бігу;
- тактильні костюми для всього тіла;
- ложі, що імітують реальну зброю у VE;
- ручки для настільного тенісу;
- тактильні ножні контролери та взуття;
- базові станції, які виявляють і відстежують бездротові VR-об'єкти та включають їх у гру;
- тканинні чохли для підвищення комфорту гарнітури;
- портативні акумуляторні батареї для живлення складніших технологій VR [10].

1.4 Проєктування інтерфейсу користувача ігрових застосунків

На сьогоднішній день режими проєктування інтерфейсу користувача в основному поділяються на дизайн інтерфейсу на основі сценаріїв, дизайн глобального інтерфейсу та дизайн ручного інтерфейсу [11].

Режим проєктування інтерфейсу на основі сценарію полягає в інтеграції інтерфейсу та операційної інформації у віртуальний простір і сцену, яка є інтерфейсом прямого контакту між людиною та віртуальним простором [12].

Режим глобального дизайну інтерфейсу - це дизайн інтерфейсу з точки зору користувача. У цьому режимі інтерфейс користувача з'являється у вигляді зависання, відкриваючи користувачеві базовий функціональний інтерфейс, опис розташування, інструкції взаємодії тощо. Глобальний інтерфейс зазвичай накладається на інші дисплеї сцени віртуального

простору, тому він знаходиться ближче до очей користувача, що може викликати зорову втому та певною мірою віртуальне запаморочення [13].

Режим проєктування ручного інтерфейсу — це режим інтерфейсу, який слідує за руками користувача в жестах. Цей режим узгоджується з повсякденним життям користувача під час взаємодії з ПК, тому користувачеві потрібно лише підняти руку, щоб працювати [14]. Таким чином, він ефективно уникає складних операцій користувачами та не викликає втоми зору. Крім того, завдяки довгостроковій взаємодії рухів рук людини цей режим інтерфейсу дуже відповідає досвіду користувача та повсякденному досвіду, легко опановується користувачами, надаючи комфортний досвід [12].

Виходячи з того факту, що найважливішими характеристиками VR є ефект занурення та інтерактивність, то чи відповідає дизайн інтерфейсу в середовищі VR природному інтерфейсу користувача та як він впливає на досвід користувача можна судити з наступних аспектів (рис. 1.6) [15]:

- просторовий реалізм: чи сприймається користувачем середовище VR як реальне;
- безпека та комфорт: чи хороший фізичний та емоційний стан користувача в середовищі VR;
- мотивація: чи сильна мотивація користувача брати участь у середовищі;
- ефективність взаємодії: впливає на швидкість навчання новачків, рівень майстерності та концентрацію уваги [15].

Застосування різних режимів дизайну інтерфейсу створить різні позиційні відносини з користувачами, а зміни цих відносин безпосередньо визначатимуть розпізнавання та роботу користувача, таким чином впливаючи на відчуття користувача [16].

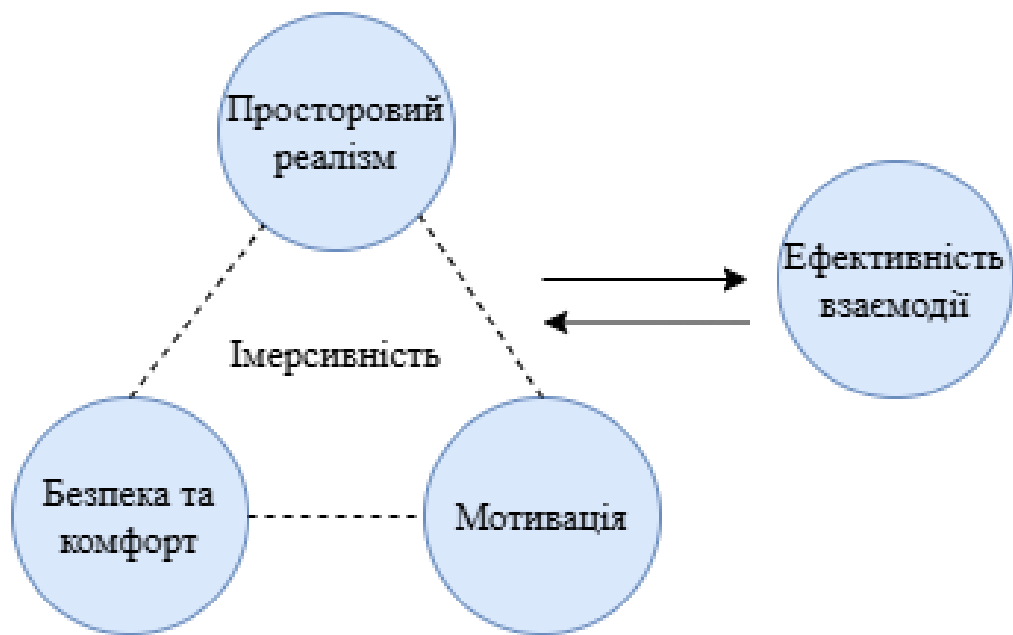


Рисунок 1.6 – Складові природного інтерфейсу

Тому від дизайнерів також вимагається розумна конфігурація інтерфейсу на початку, щоб краще допомогти користувачам завершити взаємодію в природному інтерфейсі користувача. Ці зв'язки можна в основному класифікувати як прив'язані, супутні, стаціонарні та вільної форми [17].

Прив'язаний інтерфейс означає прив'язку до частини віртуальної моделі користувача, а його просторове розташування має тенденцію змінюватися відповідно до дій користувача [17].

Супутній інтерфейс часто залишається відносно нерухомим, тобто, куди б не йшов користувач, супутній інтерфейс також слідує і часто представляє подібний стан для очей користувача [17].

Стаціонарні інтерфейси часто можуть з'являтися як певна частина сцени VR, вони не змінюють своє просторове положення відповідно до пози користувача та інформації про положення, натомість вони завжди узгоджені зі структурою моделі сцени [17].

Інтерфейс вільної форми характеризується здатністю користувача розміщувати та регулювати відносне положення інтерфейсу в просторі за

бажанням, надаючи користувачеві більше можливостей для персоналізації, а також ускладнюючи процес роботи та збільшуючи ймовірність помилки [17].

Модель інтерфейсу користувача визначає взаємозв'язок між користувачем та інтерфейсом, і ці фактори також впливають на досвід користувача (табл. 1.1).

Таблиця 1.1 - Взаємозв'язок між користувачем та інтерфейсом у режимі інтерфейсу користувача та вплив на досвід користувача [1]

Режим проектування інтерфейсу	Інтерфейс роботи	Досвід користувача	Відносини між інтерфейсом і користувачем
Дизайн інтерфейсу на основі сценарію	Інтерфейс і процес роботи у віртуальних сценаріях	Відповідно до звичок реальної експлуатації, значно покращуючи занурення та досвід користувача під час досвіду	Стаціонарні
Глобальний дизайн інтерфейсу	Інтерфейс відображається на сцені як наведення	Користувачі повинні працювати незалежно у віртуальному світі, і якщо інтерфейс не налаштовано належним чином, легко спробувати зоруву втому та запаморочення, що спричиняє пригнічення досвіду користувача	Стаціонарні, супутні, вільної форми
Дизайн ручного інтерфейсу	Інтерфейс слідує за руками користувача	Відповідає реальній взаємодії з ПК, простий у управлінні та легкий для користувачів, щоб завершити взаємодію, сприяючи користувальницькому досвіду	Прив'язані, супутні

VR дозволяє відчувати VE та взаємодіяти з ним за допомогою різних методів взаємодії: за допомогою контролерів VR, або за допомогою жестів. Як правило, розглядаються чотири режими взаємодії: пересування, вибір, маніпулювання та масштабування, які допомагають орієнтуватися в VE,

взаємодіяти з об'єктами, змінювати їхню орієнтацію або форму та досліджувати об'єкт або деталі середовища відповідно [18].

Окрім взаємодії з 3D-віртуальними об'єктами та середовищами, багато комп'ютерних систем вимагають від користувачів вибору. Для здійснення вибору в 2D-комп'ютерній системі меню є типовим стилем взаємодії [19]. У той час як стилі взаємодії та дизайн користувацького інтерфейсу досить добре встановлені для 2D-систем, їх проєктування для 3D-застосунків і середовищ створює деякі додаткові проблеми, що призвело до вивчення різних методів і запропонованих таксономій [20].

Вибір меню та взаємодію для 3D-систем можна виконати шляхом безпосереднього відображення 2D-меню в 3D-простір. Наприклад, меню, що спливають та розкриваються, можна використовувати або закріпивши їх на вікні огляду камери, як на дисплеї (HUD), або розмістивши їх у 3D-просторі з попередньо визначеною позицією та орієнтацією [21]-[23]. В якості альтернативи можуть бути реалізовані колові меню [24].

Більш природною та захоплюючою альтернативою є використання дієгетичних інтерфейсів, де елементи інтерфейсу користувача існують як частина VE і видимі як для гравця, так і для віртуальних персонажів [25]. Такі інтерфейси в поєднанні з пасивним тактильним зворотним зв'язком можуть призвести до покращеного та навіть більш захоплюючого досвіду в різних застосунках VR. Цього можна досягти, зробивши інтерфейс користувача відчутним, або матеріальним.

1.5 Висновки до розділу 1

В цьому розділі виконано дослідження сфер застосування, типів та обмежень застосунків віртуальної реальності. Проведено аналіз особливостей розробки інтерфейсів ігрових застосунків та їх впливу на досвід користувача.

2 ДОСЛІДЖЕННЯ ОСОБЛИВОСТЕЙ ВЗАЄМОДІЇ ЧЕРЕЗ МАТЕРІАЛЬНІ ІНТЕРФЕЙСИ КОРИСТУВАЧА

2.1 Досвід застосування

Tangible and Embodied Interaction (TEI) та Tangible User Interface (TUI) не є новими поняттями [26]-[28]. Вони використовувалися в різних сферах, таких як освіта, розваги, візуалізація інформації, музика [29]-[30].

Автори [31] використали відчутні інтерфейси, які можна тримати та стискати в руках, відчуваючи серцебиття користувача під час тренування дихання. Їх система забезпечує варіабельність серцевого ритму, покращене керування диханням у тактильній модальності, яку можна використовувати в будь-який час і в будь-якому місці для релаксації та зняття стресу.

Система TANGAEON досліджує, як TEI можна використовувати для підтримки людей у практиці усвідомленості [32]. Система використовує інтерфейс на основі води у вигляді справжньої води в каністрі, розміщеній на планшеті, і дозволяє користувачам взаємодіяти з нею, таким чином отримуючи контроль над процесом розчинення думок.

TUI використовувалися для навчання та розваг [33]-[36] за допомогою обчислювально вдосконалених іграшок з інтерактивними датчиками та динамічною поведінкою як різні типи маніпуляцій, які сприяють новим ігровим можливостям і сприяють навчанню дітей.

TUI також може покращити продуктивність мальованих 3D-ескізів [37] і використовуватися в сценаріях спільної роботи [38].

Автор [39] показав, що використання фізичних аналогів віртуальних об'єктів значно збільшує присутність і покращує когнітивне картографування середовища. Ця концепція парного віртуального та фізичного середовища також відома як Substitution Reality [40]. Ці проксі можуть бути зроблені як низькоточні, щоб користувач бачив віртуальний об'єкт у VE та міг торкатися фізичного проксі, який забезпечує тактильний зворотний зв'язок на основі його форми та структури. Однак потрібно

звернути особливу увагу на мінімізацію невідповідності між фізичним об'єктом і його проксі, будь то форма, розмір, вага, текстура або щось інше, що стосується користувача та завдання [41].

Автори [40] показали, що проксі з низькою і навіть помірною невідповідністю, тобто подібністю матеріалу та варіаціями тактильного зворотного зв'язку, температури та ваги, мали позитивний вплив на досвід, тоді як такі ж зі значною невідповідністю мали негативний вплив на ілюзію.

2.2 Матеріальний інтерфейс користувача

VR — це технологія та засіб, який здатний надавати різноманітні враження за допомогою аудіовізуального комп'ютерного контенту, який відтворюється та зазвичай доставляється через гарнітуру VR [42].

Однією з найцікавіших парадигм інтерфейсу користувача VR є матеріальний або відчутний інтерфейс користувача [43].

Відчутний користувальницький інтерфейс — це такий, у якому користувач взаємодіє з цифровою системою шляхом маніпулювання фізичними об'єктами, пов'язаними з якістю зазначеної системи та безпосередньо представляючими її [43].

Ідея TUI полягає в тому, щоб мати прямий зв'язок між системою та тим, як користувач керує нею за допомогою фізичних маніпуляцій, маючи прямий зв'язок, який пов'язує фізичні маніпуляції з поведінкою, яку вони викликають у системі [43].

Це не просто питання наявності фізичного контролера для цифрової системи, але й переконання, що його впровадження має сенс для користувача та має додаткову цінність для більш природного та інтуїтивно зрозумілого керування проєктом. У певному сенсі інтерфейс стає практично невидимим, оскільки користувач має невід'ємні знання про такі маніпуляції, як захоплення та переміщення об'єктів, тому він може більше зосередитися на системі та поведінці, що запускається, ніж на тому, як їх активувати [43].

Phidgets - це набір датчиків і приводів, які можна використовувати для створення власних прототипів матеріальних інтерфейсів. Завдяки дуже гарній і повній колекції пристроїв, таких як акселерометри, датчики відстані та руху, світлодіоди, двигуни тощо, які, підключені до ПК, і з деяким кодуванням (підтримується багато різних мов), вони є дуже простими у використанні будівельними блоками для взаємодії дизайнерів, які хочуть експериментувати з фізичними інтерфейсами користувача [43].

Інтеграція відчутної взаємодії з віртуальними середовищами, які імітують ситуації реального світу, може надзвичайно збагатити досвід користувача. Дійсно, однією з головних перешкод віртуальним об'єктам із ефектом занурення є використання фіксованих контролерів із низькою доступністю. Доторкання, хапання, штовхання, стискання, якщо вони виконуються за допомогою звичайних контролерів, є неприродними, непрактичними та іноді викликають розчарування [44].

Щоб підвищити ефективність застосунків VR, з одного боку, зусилля спрямовані на розробку VE високої точності. Реалістичність середовища спирається не тільки на візуальну реконструкцію самого середовища, але й на мультисенсорну стимуляцію, яка може бути забезпечена, включаючи тактильний зворотний зв'язок для сприйняття сил і матеріальних характеристик об'єктів [45].

Дослідження свідчать, що дуже важко імітувати природну взаємодію в рамках досвіду VR. Жести завжди опосередковуються контролерами, що призводить до неприродних взаємодій (по відношенню до реального світу. Незважаючи на те, що різними виробниками HMD (HTC, Oculus, Samsung тощо) розроблено багато різних типів контролерів, пропоновані можливості взаємодії більш-менш однакові: кнопки та периферійні пристрої аналогового введення (стіки, панелі тощо) [44].

Цей вид введення не підходить, коли користувачеві доводиться виконувати завдання на основі взаємодії з певними пристроями. Деякими прикладами можуть бути: хірург, що працює за допомогою ланцета або

механік за допомогою електричної викрутки. У кожному з цих випадків розробка симуляції віртуальної реальності вводить неприродні взаємодії, оскільки контролер не має необхідних фізичних характеристик і можливостей фізичного пристрою. Такі якості як форма, вага, доступність матеріалу, поводження, повністю втрачаються, а весь контроль зводиться до використання віртуальних кнопок і пристроїв введення, які не імітують взаємодію з фізичним артефактом [44].

Наприклад, на рис. 2.1, а показано фізичну піпетку, яка використовується для завдання вилучення ДНК, а на рисунку 2.1, б показано контролер VR. Мета полягає в тому, щоб створити спеціальний контролер, який імітує фізичні властивості справжнього ДНК-екстрактора (форму, вагу, керування, кнопки тощо) і відповідні взаємодії для виконання завдання [44].

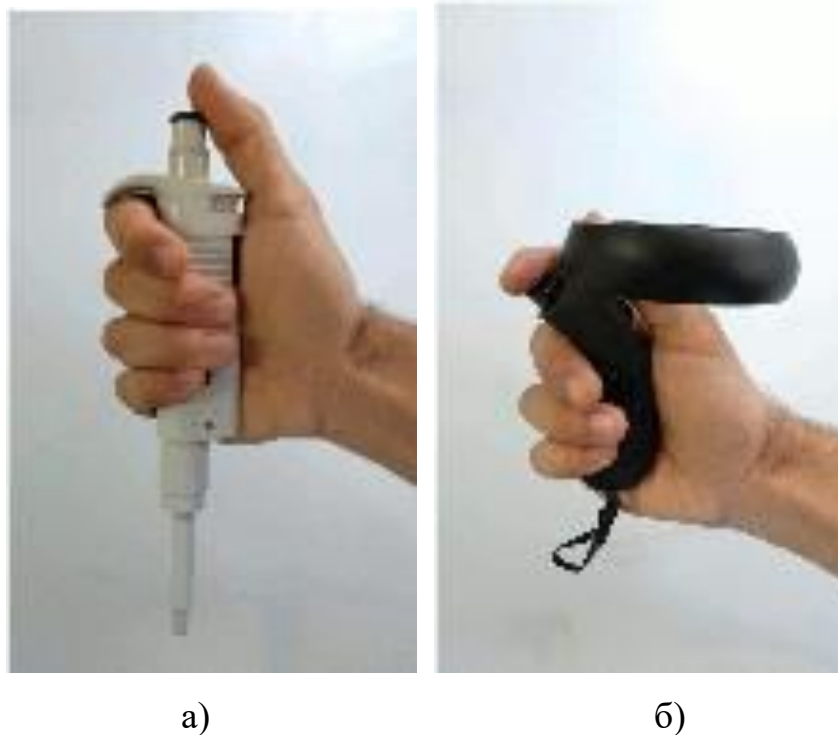


Рисунок 2.1 – Приклади реального обладнання а) та VR-контролера б) [44]

Таким чином, фактична можливість взаємодії, яку пропонують контролери, все ще недостатньо реалістична, щоб збільшити зануреність.

Чим більше ми відходимо від реальних об'єктів, намагаючись відобразити їхні матеріальні властивості у взаємодіях, які пропонують контролери, тим більше ми зменшуємо занурення. Зважаючи на це, вивчення можливостей, які пропонує змішаний підхід, який поєднує фізичні та віртуальні об'єкти, може призвести до визначення критеріїв і рекомендацій щодо проєктування, які можуть значно підвищити рівень занурення та залученість симуляції VR.

2.3 Огляд прикладів реалізації матеріальних інтерфейсів

ElasticVR [46] - це носимий пристрій, який забезпечує зворотний зв'язок за допомогою еластичної стрічки, серводвигунів і механічних замків. Він здатний забезпечувати як пасивний зворотний зв'язок за силою (наприклад, імітація опору та сили реакції), так і активний зворотний зв'язок за силою (наприклад, імітація віддачі та удару).

TacTiles - це гнучкий тактильний масив для забезпечення локалізованого тактильного зворотного зв'язку [47], який здатний відтворювати безперервний контакт з об'єктами та досліджувати їх поверхні та об'єми.

Haptic Revolver [48] - це портативний VR-контролер, який рендерить тактильні відчуття пальців під час взаємодії з віртуальними поверхнями.

З іншого боку, інтерактивність і, зокрема, відчутна взаємодія – тобто можливість маніпулювати об'єктами та здатність середовища реагувати на дії користувача – вважаються фундаментальним фактором порівняно з реалістичністю середовища [49]-[51].

Поточна тенденція дизайну зосереджена на інтуїтивно зрозумілих інтерфейсах шляхом впровадження природної взаємодії через розпізнавання жестів і відстеження рук для взаємодії з віртуальними об'єктами (наприклад, Oculus).

Хоча комерційні VR-пристрої оснащені власними контролерами, деякі автори досліджують нові концепції використання датчиків відстеження рук.

В роботі [52] порівнювали контролери HTC Vive із Leap Motion Camera у задачах маніпулювання об'єктами (наприклад, вибір, позиціонування та обертання). Автори виявили кращу продуктивність і меншу відчутну складність контролерів Vive завдяки їх стабільності, точності та меншій кривій навчання порівняно з тим, що вимагається від датчика Leap Motion. Тим не менш, з обома пристроями виникли інші труднощі, тому автори прийшли до висновку, що жоден з двох пристроїв не можна розглядати як остаточне вирішення проблем, пов'язаних з ефективними взаємодіями у віртуальному середовищі.

CLAW [53] являє собою доповнення традиційної функціональності контролера VR. Він забезпечує різні взаємодії: захоплення віртуальних об'єктів, торкання віртуальних поверхонь і спрацьовування і генерує вібрацію та силовий зворотний зв'язок у відповідь на дії користувача.

Абсолютно нову концепцію запропонували автори [54], які розробили Reconfigurable Tangible Device. Він складається з трьох або чотирьох рук, жорстко з'єднаних разом, які можна стискати або розтягувати, щоб маніпулювати формою та рухом віртуальних об'єктів.

Однією з головних перешкод у маніпулюванні об'єктами у VR є відсутність реалістичного тактильного зворотного зв'язку та контролерів із належною роздільною здатністю [55] для природного виконання завдань, пов'язаних із дотиком, стисканням, переміщенням, штовханням тощо, як це відбувається в реальному світі [56].

Зростаюча кількість застосунків у кількох сферах показує потенціал підходу матеріальної взаємодії у підтримці творчих здібностей користувачів, поглиблення досвіду взаємодії та надання повного контролю над інтерфейсом користувачеві. Але на відміну від додатків VR і 2D GUI, більшість сучасних TUI не мають можливості змінити свої фізичні властивості. Існуючі рішення, які дозволяють змінювати форму фізичних об'єктів [57] або їх положення [58], мають грубу роздільну здатність, високу затримку оновлення та потребують значного технічного обладнання. Ці

обмеження можуть ускладнити розробникам системи прийняти підхід і, таким чином, перешкоджати поширенню матеріальної взаємодії в реальних виробничих системах і застосунках кінцевих користувачів.

Інтеграція матеріальних інтерфейсів у віртуальне середовище викликає деякі проблеми, особливо оклюзію, фокусне зміщення та зміщення, які обмежують ілюзію інтегрованого віртуального середовища. Оклюзія виникає, якщо реальні об'єкти блокують огляд віртуальних об'єктів. Фокальне зміщення відбувається через різні рівні фокусів фізичних об'єктів (фокус на об'єкт під рукою) і віртуального середовища (фокус на площині проекції). Зсув спричинений неточним відстеженням положення та орієнтації матеріального об'єкта, що призводить до розбіжностей між спостережуваним положенням фізичного об'єкта та його впливом у віртуальному середовищі [59].

Тим не менш, поєднання технології VR і матеріальних інтерфейсів користувача є багатообіцяючим підходом до полегшення техніки взаємодії VR і підвищення гнучкості та програмованості TUI. Майбутні методи взаємодії VR, можливо, включатимуть розподілені фізичні об'єкти, а не покладатимуться на окремі пристрої взаємодії. Майбутні TUI отримають переваги від можливостей віртуальних середовищ, їх потужної графічної продукції та їхніх «віртуальних» законів природи. Можливі застосування в областях, які вимагають важких прямих маніпуляцій об'єктами та пропонують одночасне керування. Прикладами у сфері розробки віртуальних продуктів є програми на основі VR, такі як CAD; моделювання вільної форми, складання та швидке створення прототипів [60].

2.4 Постановка завдань дипломної роботи

Як показали проведені дослідження, існуючі проблеми розробки дизайну для ігр охоплюють надмірне ускладнення інтерфейсу великою кількістю елементів, погану навігацію, що не дає безперешкодно

переміщатися по грі, а також непослідовний дизайн, коли різні частини інтерфейсу гри мають різні стилі, кольори або елементи керування. Добре продуманий UI/UX дизайн не тільки підвищує естетичну привабливість гри, але й гарантує, що вона буде інтуїтивно зрозумілою, привабливою та приємною для гравців.

Зростаюча кількість застосунків VR в різних сферах показує потенціал підходу матеріальної взаємодії у підтримці творчих здібностей користувачів, поглиблення досвіду взаємодії та надання повного контролю над інтерфейсом користувачеві. Інтеграція матеріальних інтерфейсів користувача у віртуальне середовище є багатообіцяючим підходом до полегшення методів взаємодії з VR. Використання матеріального інтерфейсу забезпечує покращення відчутності у VR та пропонує величезні можливості для збагачення досвіду VR. Тому, тема роботи є актуальною.

Мета роботи – дослідження та розробка програмно-апаратного комплексу для покращення досвіду взаємодії користувача з ігровим середовищем через матеріальний інтерфейс.

Завдання роботи:

- виконати аналіз вимог до програмно-апаратного забезпечення ігрового застосунку;
- розробити модель взаємодії з ігровим застосунком через матеріальний інтерфейс;
- обрати програмні та апаратні засоби розробки;
- виконати розробку прототипу геймпаду та програмного забезпечення для взаємодії через матеріальний інтерфейс;
- виконати програмну реалізацію ігрового середовища.

3 ПРОЄКТУВАННЯ ПРОГРАМНО-АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Призначення та вимоги до системи

Розроблюване ігрове середовище призначене для ілюстрації навичок ведення військових дій та підвищення рівня патріотичного виховання молоді. Ключовими об'єктами можуть виступати будь які рухомі платформи (наземні дрони, танки, бойові машини піхоти). Ігрова мапа має будуватися на основі генерації лабіринту та містити різні перешкоди, від яких під час бою можуть відбиватися та рикошетити снаряди. Рухомий об'єкт ворога вважається знищеним після трьох попадань в нього.

До програмних вимог можна віднести наступні:

- робота у 2D режимі;
- зручна файлова структура проєкту;
- зрозуміла структура та синтаксис взаємодії об'єктів, компонентів та інших елементів проєкту;
- підтримка релізу для ПК;
- підтримка популярних форматів моделей, звуків, текстур, матеріалів, скриптів;
- система контролю версій.

До апаратних вимог можна віднести:

- зручний спосіб розташування компонентів керування;
- низька ціна апаратного забезпечення для реалізації прототипу геймпаду.

3.2 Модель взаємодії MVC

Матеріальні інтерфейси надають цифровій інформації фізичну форму, використовуючи фізичні артефакти і як представлення, і як засоби керування обчислювальними носіями. Матеріальні інтерфейси користувача поєднують фізичні представлення (наприклад, фізичні об'єкти, якими можна керувати

просторово) з цифровими представленнями (наприклад, графікою та аудіо), утворюючи інтерфейси користувача, які опосередковані обчислювальними засобами, але, як правило, не ідентифікуються як «комп'ютери» як такі [61].

Модель взаємодії Model-View-Controller (MVC) служить технічною моделлю для розробки ПЗ графічного інтерфейсу користувача, а також надає інструмент для вивчення концептуальної архітектури графічних інтерфейсів і зв'язку цієї архітектури з підходом матеріального інтерфейсу. MVC підкреслює поділ графічного інтерфейсу між візуальним представленням (View), що забезпечується графічним дисплеєм, і можливостями керування (Controller), які здійснюються за допомогою миші та клавіатури (рис. 3.1).

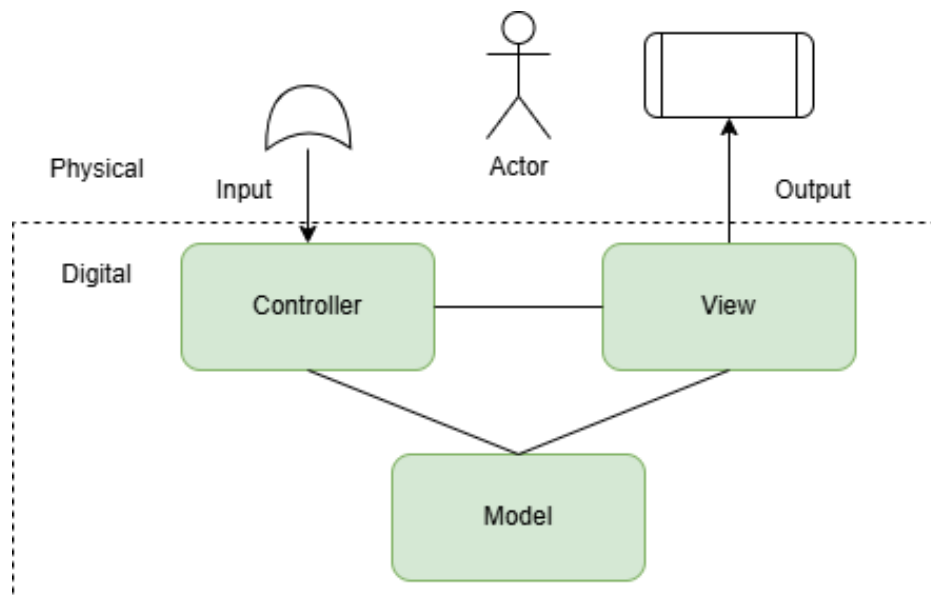


Рисунок 3.1 - Модель взаємодії «Model-View-Controller» [61]

На рис. 3.2 представлено модифіковану модель взаємодії для матеріальних інтерфейсів, в якій так само присутні елементи «Model» і «Controller», а елемент «View» розділено на два підкомпоненти: фізичне представлення Rep-p (фізично втілені елементи матеріальних інтерфейсів), і цифрове представлення Rep-d (елементи матеріальних інтерфейсів без втіленої фізичної форми, наприклад, відео та аудіо).

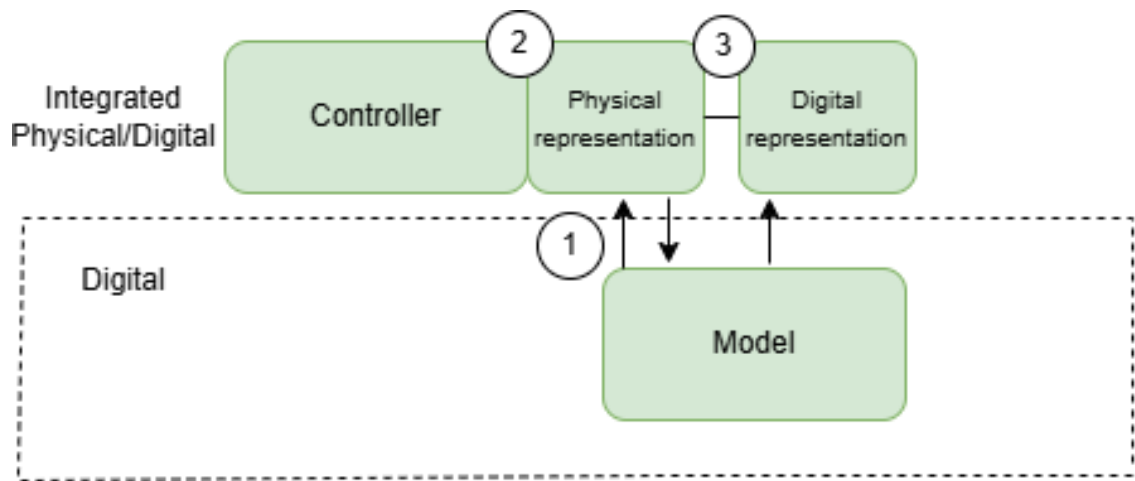


Рисунок 3.2 – Модифікована модель взаємодії для матеріального інтерфейсу

Таким чином, модель MVC на рис. 3.1 ілюструє різницю графічного інтерфейсу користувача між графічним представленням і керуванням, а модель на рис. 3.2 підкреслює інтеграцію фізичного представлення та керування TUI. Ця інтеграція присутня не лише на концептуальному рівні, але й у фізичному аспекті — артефакти TUI (або «матеріальні речі») фізично втілюють як контрольний шлях, так і центральний репрезентативний (інформаційний) аспект інтерфейсу.

Модель взаємодії MCRpd надає інструмент для вивчення кількох важливих властивостей матеріальних інтерфейсів. Зокрема, корисно розглянути три зв'язки, спільні для фізичних представлень («гер-р») TUI.

Як показано на рис. 3.2, модель виділяє наступні три ключові характеристики матеріальних інтерфейсів:

- основною характеристикою матеріальних інтерфейсів є зв'язок фізичних представлень з основною цифровою інформацією та обчислювальними моделями (1);
- фізичні представлення TUI служать одночасно як інтерактивні фізичні елементи керування. Матеріальні об'єкти можуть бути фізично інертними, рухатися лише тоді, коли ними безпосередньо маніпулюють руки користувача, або ж фізично приведені в дію за допомогою підходів із

зворотним зв'язком сили, що керуються двигуном, або за допомогою індукованих підходів, таких як вібраційні пластини (2);

– фізичні представлення перцептивно пов'язані з активно опосередкованими цифровими представленнями. Матеріальні інтерфейси покладаються на баланс між фізичними та цифровими представленнями. Хоча втілені фізичні елементи відіграють центральну, визначальну роль у представленні та контролі TUI, цифрові представлення, особливо графіка та аудіо, часто є опосередковувачем більшої частини динамічної інформації, що надається основною обчислювальною системою (3).

Для інтеграції фізичних об'єктів у 3D-середовище необхідні точне просторове відстеження та графічне відтворення. За бажанням матеріальний інтерфейс може бути оснащений інтерактивними елементами керування, такими як кнопки, повзунки, світлодіоди тощо.

Цей підхід спрямований на інтеграцію фізичних і віртуальних частин об'єктів. Він фокусується на:

- інтеграції простору дії та сприйняття (введення та виведення) для полегшення маніпулювання об'єктами;
- взаємодії з просторово розподіленими об'єктами замість окремих пристроїв введення, щоб збільшити пропускну здатність взаємодії;
- гібридних об'єктах, які пов'язують фізичні та віртуальні графічні об'єкти;
- інтеграції в робоче середовище з чіткими просторами взаємодії;
- легкій взаємодії для покращеної ергономіки.

3.3 Вибір інструментарію розробки

3.3.1 Вибір програмних засобів розробки

При виборі інструментарію розробки слід керуватися складністю гри, розміром команди розробників і часом проєктування.

Більшість команд розробників VR-ігор використовують такий інструментарій: ігровий рушій, набір інструментів для розробки програмного забезпечення (Software Development Kit, SDK), програмне забезпечення для проєктування, програмне забезпечення для контролю версій (рис. 3.3) [7].

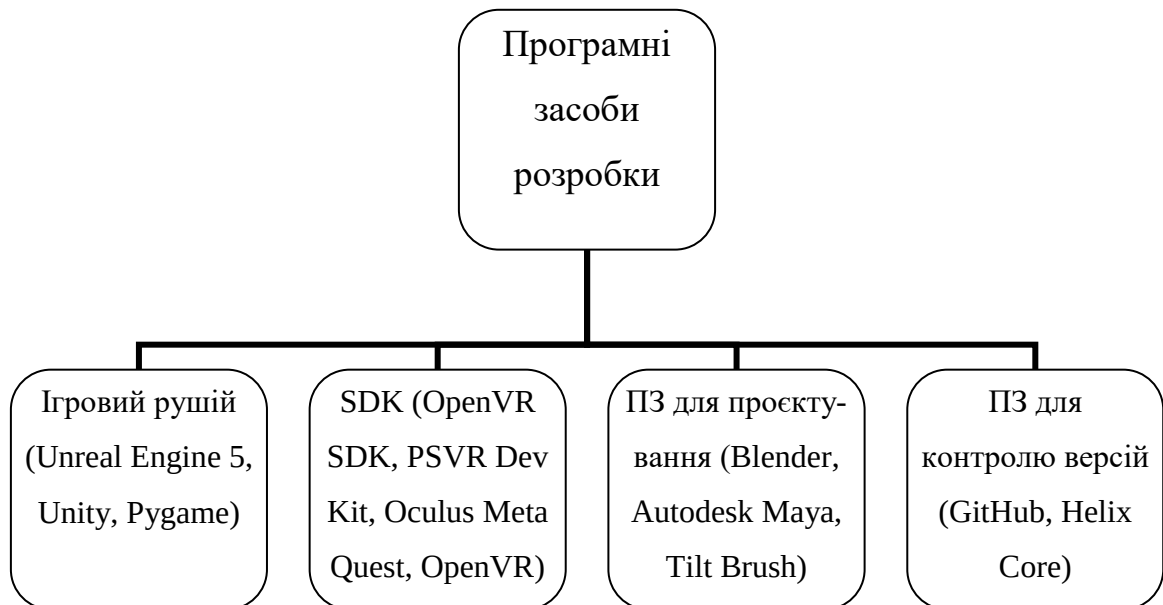


Рисунок 3.3 – Стек інструментів програмної розробки [7]

Ігрові рушії життєво важливі для розробки ігор VR. Використання ігрового рушія дозволяє централізувати ресурси та служить потужним редактором. Вони працюють разом із інструментами художника та дизайнера, а також програмним забезпеченням проєктування, щоб допомогти розробнику створювати реалістичні враження. При цьому, Unreal Engine 5 (UE5) і Unity є найчастіше використовуваними ігровими рушіями [7].

Unreal Engine 5 (UE5) пропонує командам потужний набір інструментів розробки програмного забезпечення VR. Він працює з багатьма VR-технологіями. UE5 забезпечує:

- доступ до сценаріїв у C++ та/або Python, щоб налаштувати інструмент відповідно до потреб;
- багатокористувацький фреймворк у поєднанні з гнучким редактором;

- візуальний сценарій Blueprint, який допомагає швидко створювати прототипи;
- тимчасова надроздільна здатність;
- віртуальні мапи тіней [7].

Unity добре відомий в сфері розробки ігор зі спеціалізацією на мобільних пристроях. Але його також можна використовувати для розробки рішень VR. Команди різних галузей використовували його для розробки тренінгів і моделювання, оскільки він добре працює з усіма популярними платформами VR. Unity забезпечує:

- потужний 3D редактор;
- вбудовані інструменти художника та дизайнера, а також підтримку інструментів САПР;
- функції співпраці, які спрощують процес розробки [7].

Pygame – це простий ігровий рушій, зосереджений на Python. Архітектура Pygame базується на бібліотеці SDL, яка обробляє графіку, звук і вхідні пристрої, пропонуючи прості інтерфейси для розробки ігор у Python. Бібліотеки Pygame включають модулі для графіки (`pygame.display`), обробки звуку (`pygame.mixer`), управління часом (`pygame.time`), зчитування вхідних даних (`pygame.event`) і роботи з шрифтами та текстами (`pygame.font`). Інструментарій Pygame охоплює функції створення вікон, малювання об'єктів, відтворення звуків, зчитування дій користувача та управління часовими інтервалами [62].

Порівняльну характеристику ігрових рушіїв наведено в табл. 3.1 [63].

Вибір мови програмування відіграє суттєву роль та має виконуватися відповідно до групи критеріїв (табл. 3.2) [64].

Таким чином, оскільки розроблювана гра буде відноситися до класу інді-гра, створюватися одним розробником за обмежений час, тож доцільно обрати прості засоби розробки, зокрема ігровий рушій Pygame та мову Python. Це дозволить реалізувати проєкт без значних складнощів, забезпечуючи доступ до ключових функцій і спрощуючи процес розробки.

Таблиця 3.1 – Порівняння ігрових рушіїв [63]

Рушій/ Критерії	Мульти- платформність	Мова сценаріїв	Поріг входження
Unreal Engine	Windows, OSX, Linux	C++, Blueprints	Не вимагає високого рівня навичок програмування
Unity	Windows, OSX, Linux	UnityScript, C#	Вимагає володіння навичками програмування
CryEngine	Windows, OSX, Linux	C++, Lua	Вимагає високого рівня навичок програмування
Pygame	Windows, macOS, Linux	Python, SDL	Не вимагає високого рівня навичок програмування
Panda3D	Windows, Linux, macOS, Android и iOS	Python, OpenGL	Вимагає високого рівня навичок програмування

Таблиця 3.2 – Порівняння мов програмування для розробки ігрових застосунків [64]

Критерії/Мови програмування	C++	Python	Javascript	C#
Продуктивність	+++	+	+	++
Тип гри	3D	2D	2D	3D
Ігровий рушій	Unity, Unreal Engine, CryEngine	Panda3D, Pygame	Phaser	Unity
Простота	-	+	+	-
Швидкодія	-	++	-	+
Поріг входження	+++	+	+	++

3.3.2 Вибір апаратних засобів розробки

Плати Arduino широко використовуються в якості основи різних проєктів, в тому числі при створенні зв'язку між ігровим рушієм та мікроконтролером при створенні віртуальних середовищ з матеріальним інтерфейсом [65].

В табл. 3.3 наведено порівняльний аналіз характеристик найбільш популярних плат Arduino [66].

Таблиця 3.3 – Порівняння популярних плат Arduino [66]

Плата Arduino	МК	Цифр. вх/вих	Вих. з ШІМ	Аналог. вх\вих	Flash пам'ять, Кб	ОЗУ, Кб	EEPROM-пам'ять, Кб	Габарити, мм	USB-рознім
Uno	Atmega328	14	6	6	32	2	1	69x53	USB A-B
Mega	Atmega2560	54	14	16	256	8	4	102x53	USB A-B
Nano	Atmega328	14	6	8	32	2	1	18x45	Mini- USB
Micro	Atmega32u4	20	7	12	32	2,5	1	48x18	Micro-USB
Mini	Atmega328	14	6	6	32	2	1	30x18	USB-Serial

Аналіз показав, що хоча Arduino Mega є достатньо складною у вивченні, має вищу вартість у порівнянні з іншими платами, але дозволяє підключати достатньо датчиків, актуаторів та інших пристроїв, зокрема Bluetooth-модулі, а також сумісна з усіма платами розширення. Тому, для створення прототипу геймпаду було обрано саме цю плату.

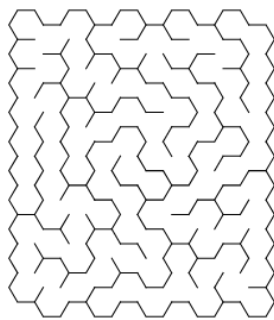
3.4 Висновки до розділу 3

В цьому розділі відповідно до мети та завдань роботи виконано аналіз вимог до програмно-апаратного комплексу, розроблено модель взаємодії з ігровим застосунком через матеріальний інтерфейс на основі MVC шаблону, виконано огляд та вибір програмних та апаратних засобів розробки.

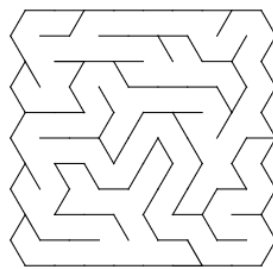
4 КОНСТРУЮВАННЯ ПРОГРАМНО-АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Алгоритм генерації лабіринту

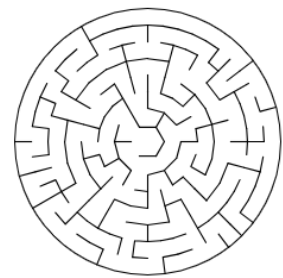
В сучасних ігрових середовищах часто зустрічаються лабіринтоподібні частини ігрової мапи (рис. 4.1) [67].



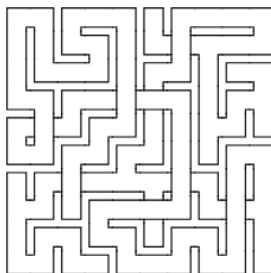
Hex (Sigma) Mazes



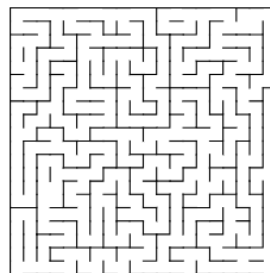
Triangle (Delta) Mazes



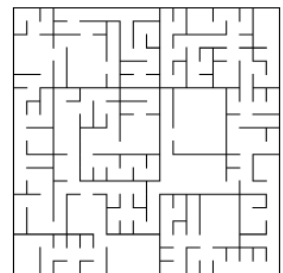
Circle (Theta) Mazes



Weave Mazes



Braid Mazes



Mazes with Rooms

Рисунок 4.1 – Типи лабіринтів [67]

Для побудови лабіринтів існує багато відомих алгоритмів, зокрема виділяються алгоритм Sidewinder [68], заснований на алгоритмі двійкового дерева, а також алгоритм Олдоса-Бродера, який генерує максимально заплутаний лабіринт, за рахунок великої частки застосування "випадків" в алгоритмі.

На рис. 4.2 представлено схему алгоритму Sidewinder. Важливим моментом алгоритму є очищення першого ряду стінок. Таким чином початковим рядом вважається ряд, наступний за першим.

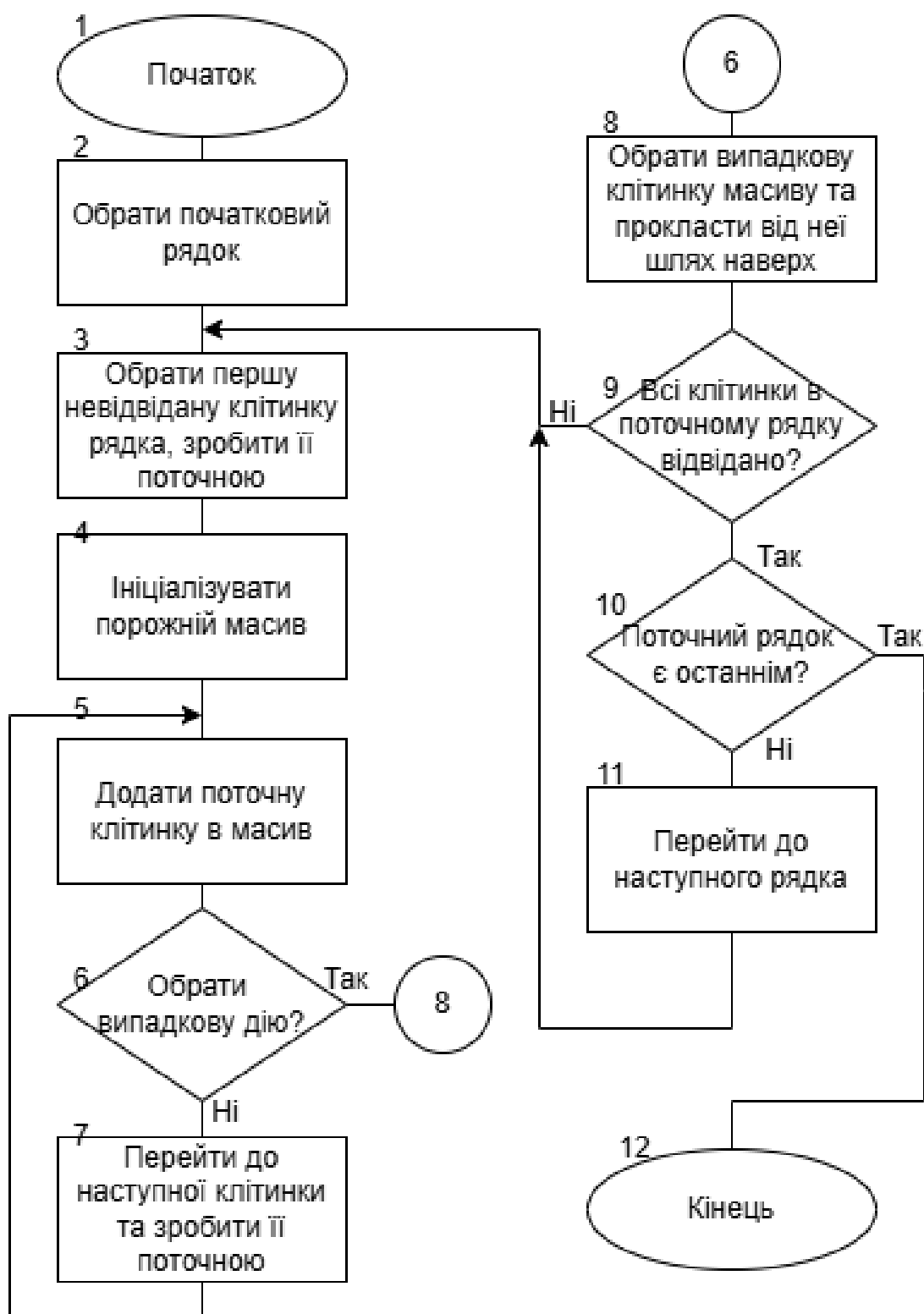


Рисунок 4.2 – Схема алгоритму Sidewinder

Алгоритм Sidewinder заснований на алгоритмі двійкового дерева, де шлях прокладається лише у двох напрямках. Під прокладанням шляху мається на увазі руйнування стінки між клітинами на полі, що складається з безлічі клітин та стінок між ними.

В алгоритмі Sidewinder використовується той самий принцип вибору одного з двох рішень, але при цьому вибір відбувається між «додати до деякого масиву ще одну клітину ряду» і «вибрати з цього масиву одну з клітин і з'єднати цей масив клітин з верхнім рядом через вибрану клітину». Так чи інакше, алгоритм є простим для розуміння і дозволяє створювати нескінченні лабіринти, при цьому мапа лабіринту буде достатньо складною.

На рис. 4.3 графічно представлено розбір алгоритму для прикладу поля розміром 6х6. Поле 1 відображає процес очищення першого ряду стінок. На полях 2-3 представлено двічі обрану дію «додати в масив ще одну клітину ряду». На полі 4 виконується вибір випадкової клітини з масиву, після чого всі стінки між клітинами масиву, а також верхня стінка випадково обраної клітини знищуються. На полі 5 представлено ініціалізацію масиву, додавання першої клітини масиву, після чого перехід до дії «вибрати з масиву одну з клітин і з'єднати цей масив клітин з верхнім рядом через обрану клітину». У цій ситуації руйнується лише верхня стінка над поточною клітиною. Поля 6-8 відображають той самий процес, що й поля 2-4.

На рис. 4.4 наведено приклад реалізації генератора лабіринту допомогою JavaScript і бібліотеки P5 на основі алгоритма Sidewinder [69].

Таким чином, це проста з точки зору програмування конструкція, що дозволяє створити достатньо складні лабіринти. Перший ряд лабіринту завжди залишається порожнім.

Лабіринти, створені таким чином, зазвичай легко пройти, рухаючись від низу лабіринту до верху, але складніше працювати зверху вниз.

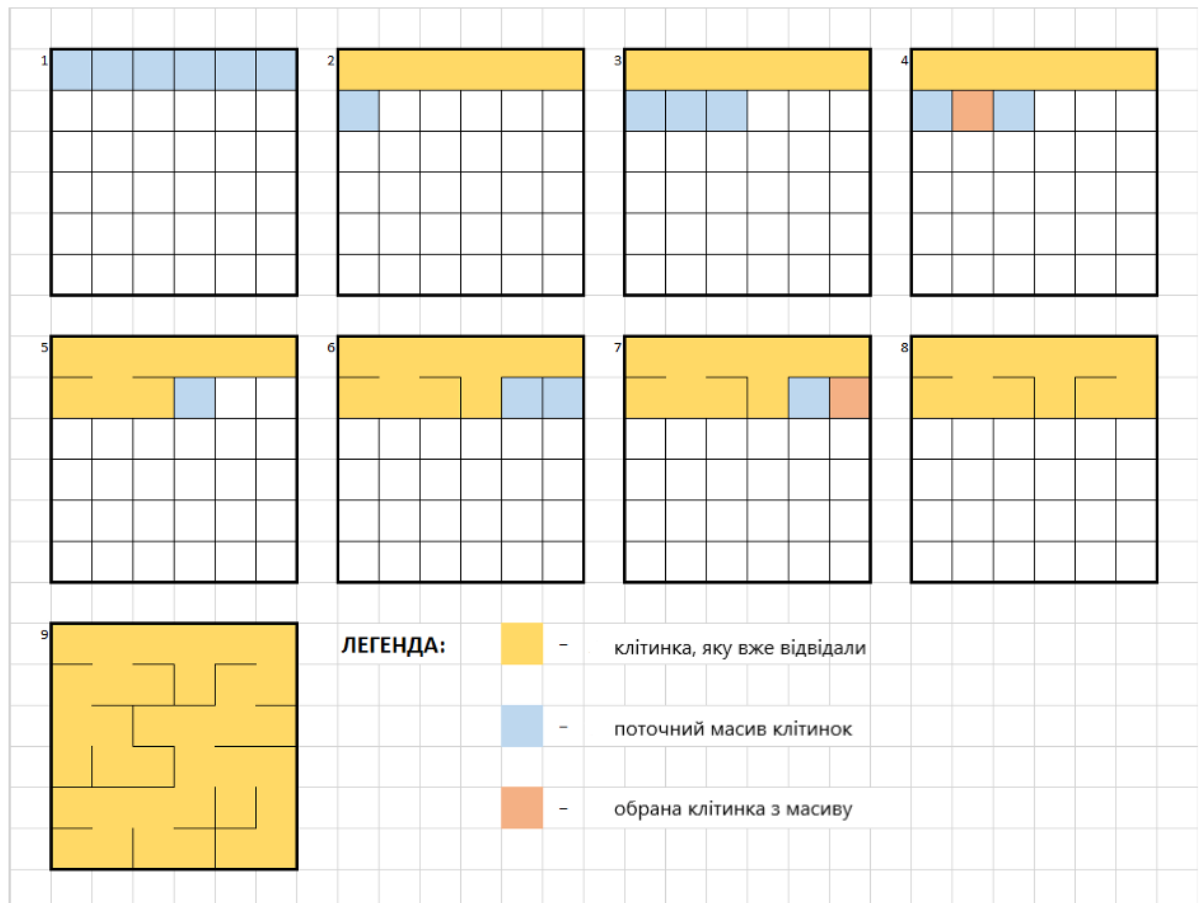


Рисунок 4.3 – Кроки алгоритму Sidewinder

Sidewinder Maze Generation

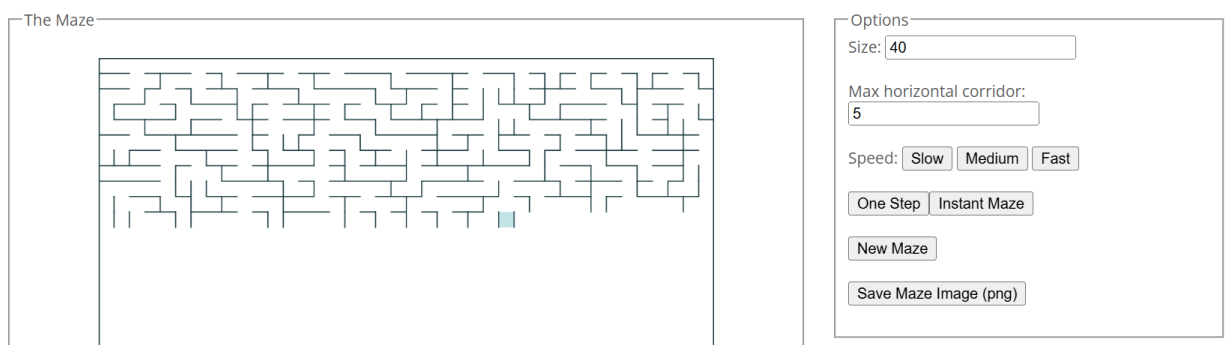


Рисунок 4.4 – Приклад реалізації генератора лабіринта [69]

4.2 Програмно-апаратна реалізація геймпаду на платформі Arduino

На рис. 4.5 зображено структурну схему програмно-апаратного комплексу, яка побудована на базі плати Arduino, а також геймпаду, які разом утворюють матеріальний інтерфейс для взаємодії з ігровим середовищем. Прототип геймпаду створений на основі макетної плати та кнопок (рис. 4.6).

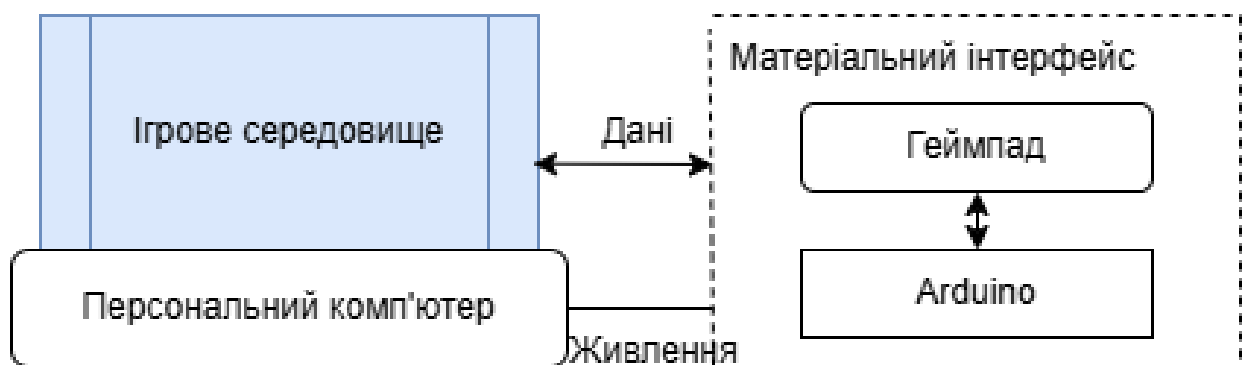


Рисунок 4.5 – Структурна схема програмно-апаратного комплексу

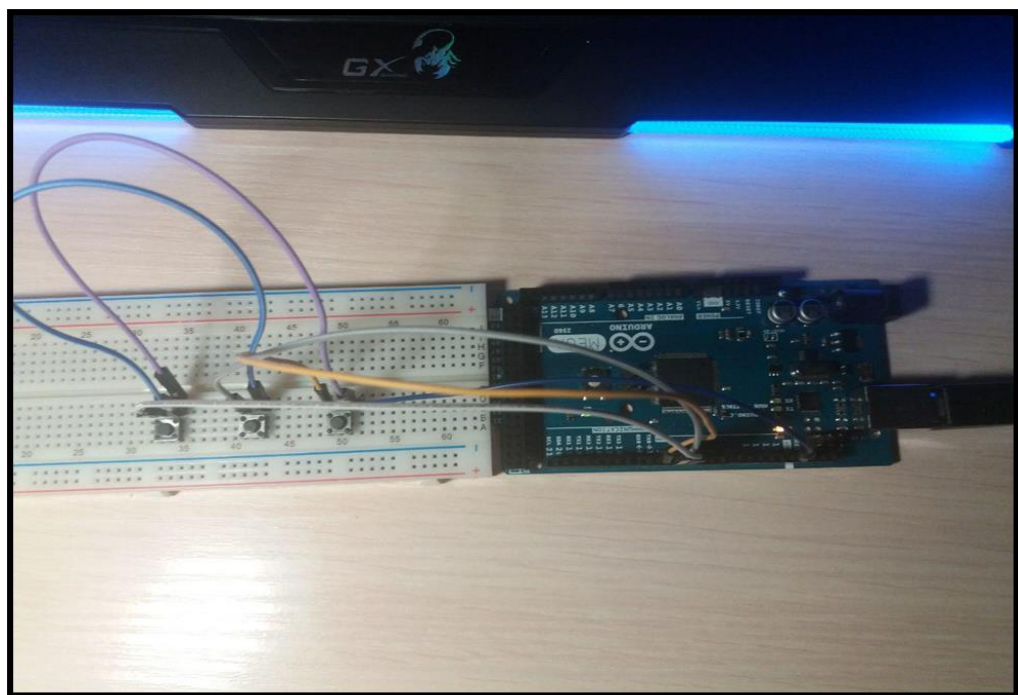
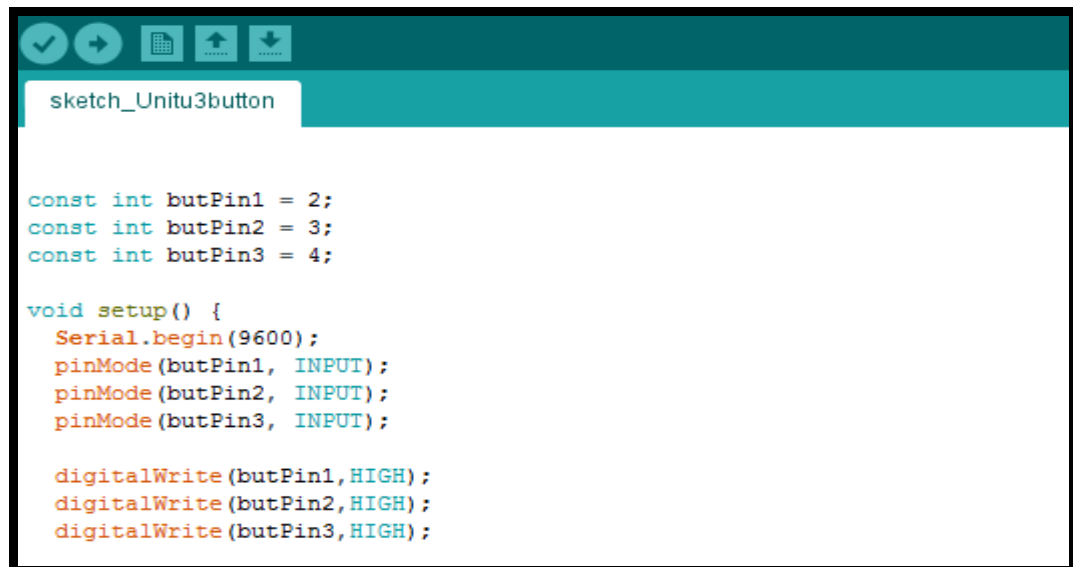


Рисунок 4.6 – Прототип геймпаду на базі Arduino Mega

Реалізація взаємодії ігрового середовища з матеріальним інтерфейсом включає наступні кроки.

Крок 1. Ініціалізація "пінів", та вказування порту у який буде записуватись інформація з клавіш (рис. 4.7).



```

const int butPin1 = 2;
const int butPin2 = 3;
const int butPin3 = 4;

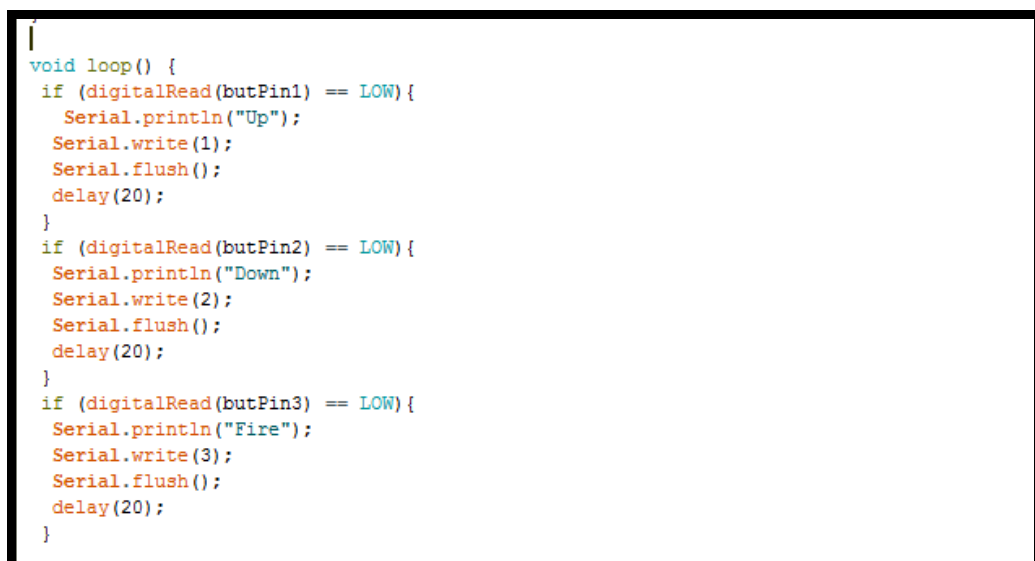
void setup() {
  Serial.begin(9600);
  pinMode(butPin1, INPUT);
  pinMode(butPin2, INPUT);
  pinMode(butPin3, INPUT);

  digitalWrite(butPin1, HIGH);
  digitalWrite(butPin2, HIGH);
  digitalWrite(butPin3, HIGH);
}
  
```

Рисунок 4.7 – Ініціалізація "пінів"

Крок 2. Перевірка чи натиснута кнопка у реальному часі і запис даних у порт (рис. 4.8).

Крок 3. Компіляція та завантаження скетчу на плату (рис. 4.9, 4.10).



```

void loop() {
  if (digitalRead(butPin1) == LOW) {
    Serial.println("Up");
    Serial.write(1);
    Serial.flush();
    delay(20);
  }
  if (digitalRead(butPin2) == LOW) {
    Serial.println("Down");
    Serial.write(2);
    Serial.flush();
    delay(20);
  }
  if (digitalRead(butPin3) == LOW) {
    Serial.println("Fire");
    Serial.write(3);
    Serial.flush();
    delay(20);
  }
}
  
```

Рисунок 4.8 – Перевірка натиснення кнопок

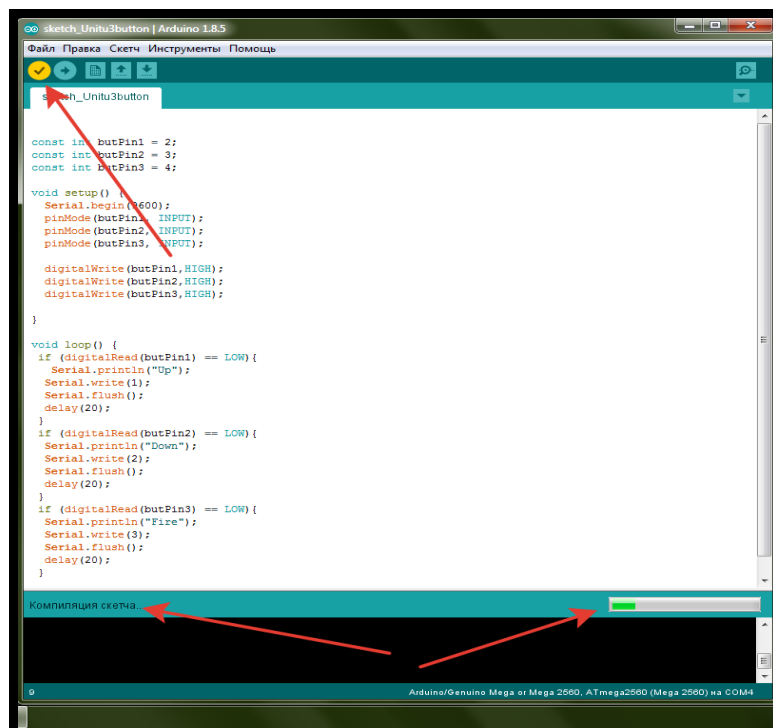


Рисунок 4.9 – Перевірка натиснення кнопок

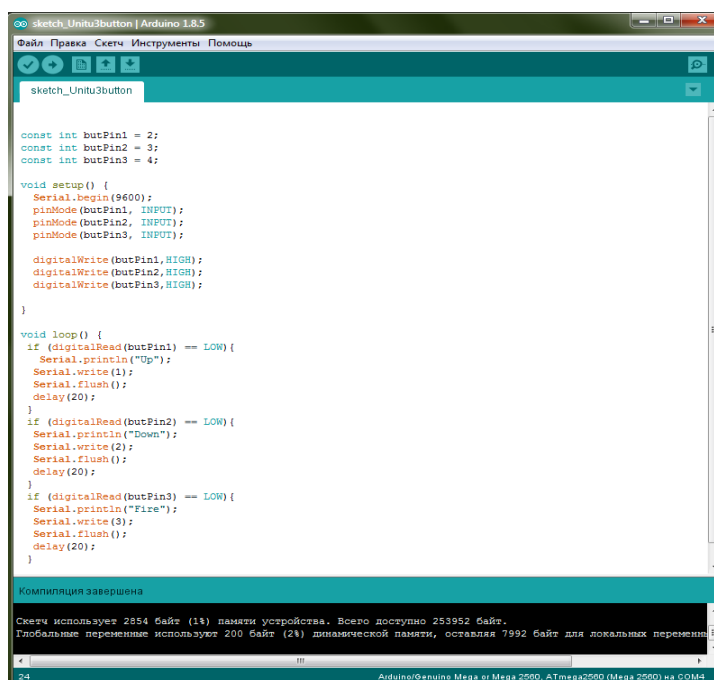


Рисунок 4.10 – Завантажений скетч на плату

До роботи алгоритму зі сторони ігрового рушія можна віднести:

- перевірка наявності всіх компонентів за адресами які були визначені;

- підключення всіх необхідних бібліотек;
- ініціалізація порту для роботи з Arduino (`SerialPort sp = new SerialPort("COM4", 9600) public bool SerialWork = false`);
- відкриття порту для роботи з ним (`sp.Open()` `sp.ReadTimeout = 1`);
- зчитування сигналів клавіш і інтерпретація їх значень для того щоб виконати одну з декількох дій, пересування вгору або вниз, постріл у реальному часі (рис. 4.11);
- завдання швидкості переміщення танку гравця;
- випадковим чином створюємо ворогів на ігровому полі і задаємо їм швидкість та кількість життів;
- у разі попадання у ворога, відбувається перевірка кількості життів, перевіряємо кількість життів і віднімаємо їх, пропорційно урону пострілу, якщо кількість життів >0 , то видаляємо ворога з ігрового поля, а на місці смерті запускаємо анімацію вибуху, додаємо кількість очок від 100 до 1000 гравцю в залежності від сили ворога;
- у разі попадання ворогом у гравця, перевіряємо кількість життів і віднімаємо їх, пропорційно урону пострілу, якщо кількість життів >0 , то видаляємо гравця з ігрового поля, на місці смерті запускаємо анімацію вибуху, виводимо на екран кількість очок та повідомлення що він програв.

```

        if (sp.IsOpen)
        {
            try
            {
                SerialWork = true;
                if (sp.ReadByte() == 1)
                {
                    transform.position += new Vector3(0,
0.5f);
                }
                if (sp.ReadByte() == 2)
                {
                    transform.position += new Vector3(0, -
0.5f);
                }
                if (sp.ReadByte() == 3)
                {
                    FindObjectOfType<Shipcontrol>().Shoot();
                }
            }
            catch (System.Exception)
            {
            }
        }

```

Рисунок 4.11 – Зчитування сигналів клавіш

При розробці було використано модифікований код Joystick (рис. 4.12). Було додано ініціалізацію клавіш керування, змінено модуль контролю гравця на більш простий і зрозумілий.

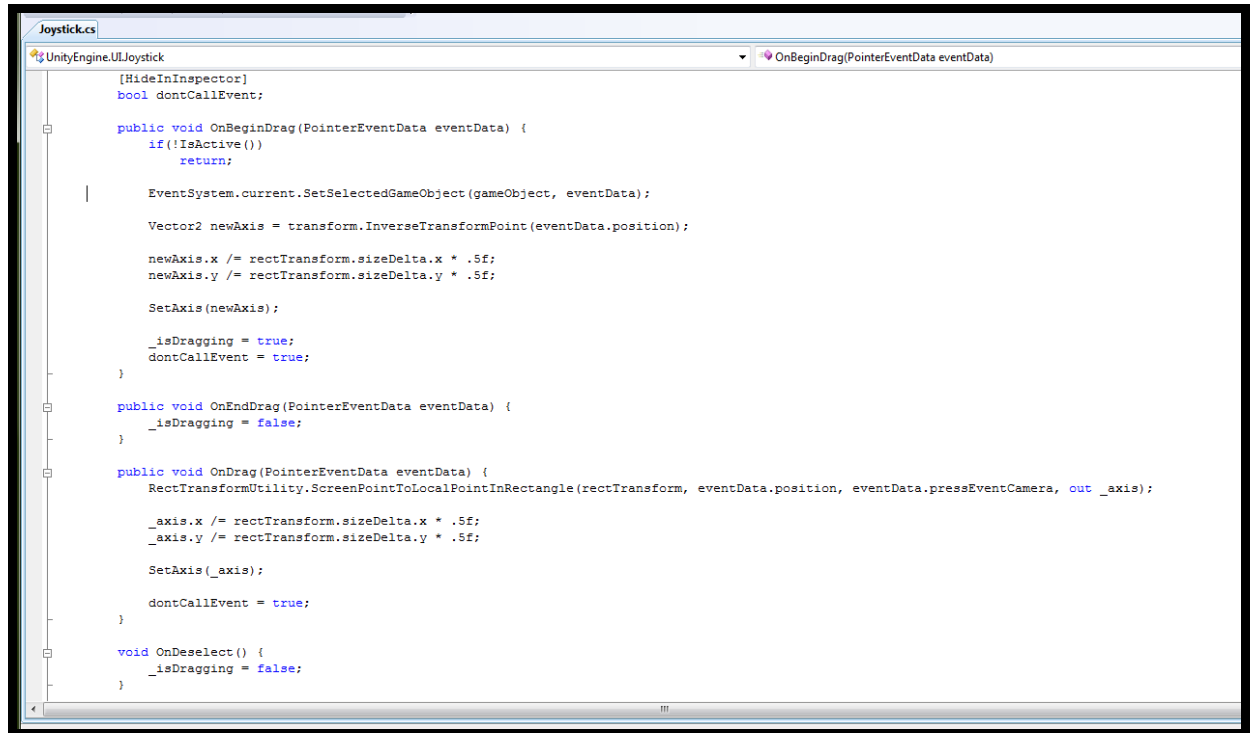


Рисунок 4.12 – Модифікований код Joystick

4.3 Програмна реалізація ігрового середовища

Файлову структуру проєкту представлено на рис. 4.13:

- lab_3.py – головний файл, де реалізовано запуск гри, логіку подій масштабування та завершення гри, а також головний програмний цикл з відмальовуванням та оновленням головних об’єктів;
- screen.py – файл, що містить набір глобальних змінних: параметри ігрової мапи, змінну статусу гри, змінну часу запуску гри, змінну графічного вікна гри тощо; а також функції для завантаження звуків та отримання часу відображення одного кадру;
- base.py – файл, що містить реалізовані класи GraphicObj та Bullet;

- tank.py – файл, що містить реалізовані класи Explosion та Player, а також їх об’єкти;
- Gmap.py – файл, що містить реалізовані класи Block та Map, а також їх об’єкти.

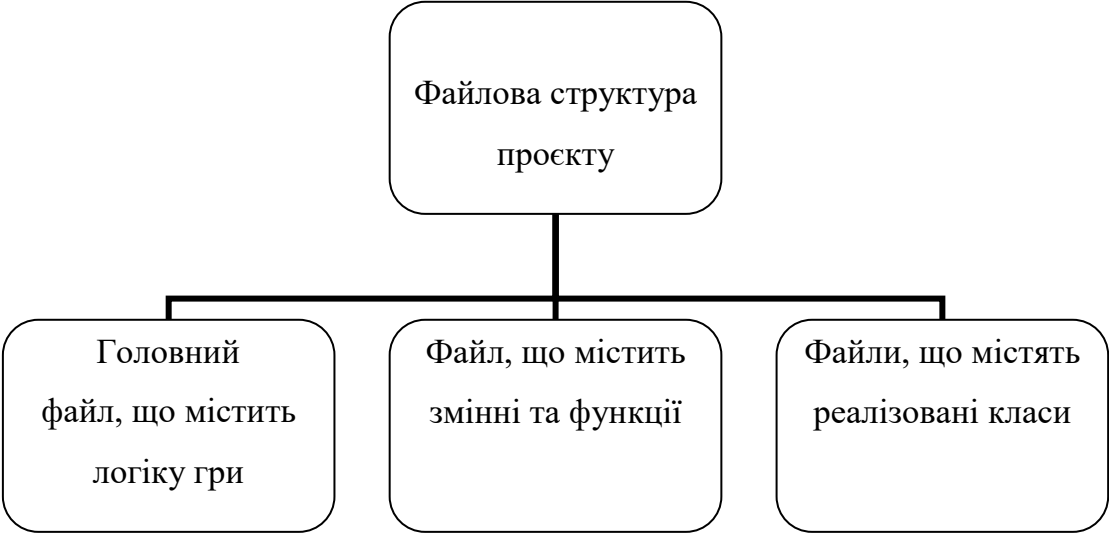


Рисунок 4.13 – Файлова структура проєкту

У процесі розробки було створено такі класи (табл. 4.1). Опис об’єктів та атрибутів наведено в табл. 4.2.

Таблиця 4.1 – Опис розроблених класів

Клас	Опис
1	2
GraphicObj	наслідує вбудований клас Sprite бібліотеки pygame. Є основою для будь-яких графічних об’єктів. Містить методи для відображення окремого графічного об’єкта, його масштабування
Bullet	наслідує GraphicObj. Є основою для таких об’єктів, як снаряди ворогів та гравця. Містить методи для визначення параметрів окремого снаряду та його переміщення

Кінець табл. 4.1

1	2
Explosion	наслідує GraphicObj. Є основою для таких об'єктів, як вибухи різних типів. Містить методи для визначення параметрів окремого вибуху та контролю тривалості його існування
Player	наслідує GraphicObj. Є основою для такого об'єкта, як танк гравця. Містить методи для реалізації пострілу, переміщення ігровою мапою, перевірки колізій тощо
Block	наслідує GraphicObj. Є основою для таких об'єктів, як блоки ігрової мапи різних типів. Містить методи для визначення та зміни параметрів окремого блоку
Map	є основою для такого об'єкта, як ігрова мапа. Містить методи для генерації ігрової мапи, її масштабування, відображення тощо.

Таблиця 4.2 – Опис об'єктів та атрибутів

Клас	Об'єкт	Опис об'єкта	Атрибути
1	2	3	4
Player	G_player	являє собою танк гравця	plbullets – об'єкт вбудованого класу Group бібліотеки pygame. Містить у собі набір об'єктів класу Bullet (випущені снаряди гравця); explosions – об'єкт вбудованого класу Group бібліотеки pygame. Містить у собі набір об'єктів класу Explosion (виниклі вибухи).

Кінець табл. 4.2

1	2	3	4
Map	G_map	являє собою ігрову мапу	blockmap – перелік об’єктів класу Block (блоки, що складають ігрову мапу); walls – об’єкт вбудованого класу Group бібліотеки rугame. Містить у собі набір об’єктів класу Block (стіни ігрової мапи); crates – об’єкт вбудованого класу Group бібліотеки rугame. Містить у собі набір об’єктів класу Block; enemies – об’єкт вбудованого класу Group бібліотеки rугame. Містить у собі набір об’єктів класу Block; enbullets – об’єкт вбудованого класу Group бібліотеки rугame. Містить у собі набір об’єктів класу Bullet.
GraphicObj	victory	являє собою графічний банер для інформування гравця про перемогу	
	defeat	являє собою графічний банер для інформування гравця про поразку	

Концепцію комп'ютерної гри можна описати як “танкова битва”. Гравець керує синім танком та рухається ігровою мапою, що за своєю структурою нагадує лабіринт (рис. 4.14).



Рисунок 4.14 – Ігрова мапа

Ігрова мапа генерується випадково та складається з блоків різних типів: стін, ящиків та трави. Танк гравця може рухатись блоками трави, в той час як стіни та ящики будуть йому перешкоджати. Кожен окремий ящик можна знищити одним пострілом гравця, тоді ж як стіни здатні відбивати кожен снаряд до трьох разів. Снаряди танка гравця не здатні нашкодити йому ж. Керування танком гравця відбувається за допомогою набору клавіш клавіатури (КК) та кнопок матеріального інтерфейсу (КМІ):

- КМІ 1 – рух у прямому напрямку;
- КМІ 2 – рух у зворотному напрямку;
- КМІ 3 – виконання пострілу;
- КК D або стрілка вправо – поворот за годинниковою стрілкою;
- КК A або стрілка вліво – поворот проти годинникової стрілки.

Окрім гравця на ігровій мапі знаходяться вороги – червоні танки. Вони не здатні рухатись ігровою мапою, проте можуть обертатися та

стріляти. Снаряди ворогів не здатні відбиватись від стін, знищувати ящики або шкодити іншим червоним танкам, проте вони здатні нашкодити танку гравця. Кожен з танків (в тому числі танк гравця) має три “життя” – після третього попадання снаряду танк вибухає. Якщо танк гравця вибухає, гра закінчується поразкою (рис. 4.15). Якщо ж усі ворожі танки знищені, гра закінчується перемогою (рис. 4.16).



Рисунок 4.15 – Поразка



Рисунок 4.16 – Перемога

4.4 Висновки до розділу 4

В цьому розділі відповідно до мети та завдань дипломної роботи було програмно реалізовано алгоритм побудови ігрової мапи на основі лабіринту, механіки гри та взаємодію через матеріальний інтерфейс. Тестування підтвердило працездатність розробки.

ВИСНОВКИ

В результаті виконання дипломної роботи було розроблено ігровий застосунок з матеріальним інтерфейсом.

Показано, що впровадження матеріальних інтерфейсів дозволяє уникнути проблем дизайну складних графічних інтерфейсів користувача та зробити досвід використання ігрового середовища більш цікавим.

Оскільки розроблювана гра відноситься до класу інді-гра, створювалася одним розробником за обмежений час, тож було обрано прості засоби розробки, зокрема ігровий рушій Pygame та мову Python. Це дозволило реалізувати проєкт без значних складнощів, забезпечуючи доступ до ключових функцій і спрощуючи процес розробки.

Використання плати Arduino дозволило створити недорогий прототип геймпаду, що надає три кнопки керування, але може бути удосконалений шляхом додавання додаткових інтерактивних елементів керування, зокрема кнопок, а також зумерів та світлодіодів для зворотнього зв'язку та нотифікації.

Тестування підтвердило працездатність розробки та виконання запланованого функціоналу.

В подальшій роботі можливо створити 3D варіант гри, з використанням іншого ігрового рушія, зокрема Unity/Unreal Engine/Godot. Також можливо реалізувати використання інших технологій для розширення функцій матеріального інтерфейсу та реалізації бездротового зв'язку геймпаду з ігровим середовищем. Також можливо, використовувати `ros2_arduino` або `micro_ros_arduino`. Зокрема, використання пакету `roserial_arduino` дозволить використовувати ROS безпосередньо з Arduino IDE. Це дозволить Arduino бути повноцінним вузлом ROS, який може безпосередньо публікувати та підписуватися на повідомлення ROS, публікувати перетворення координат і отримувати системний час ROS.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Xie, H. The applications of interface design and user experience in Virtual Reality / H. Xie // Highlights in Science, Engineering and Technology. – 2023. – Vol. 44. – P. 189-198.
2. Designing for VR: a beginners guide [Electronic resource]. – Access mode: <https://marvelapp.com/blog/designing-vr-beginners-guide/>.
3. Balarabe, H. S. Virtual Reality (VR) the emerging trend in user interface and user experience (UI/UX) design [Electronic resource] / H. S. Balarabe. – Access mode: <https://medium.com/design-bootcamp/virtual-reality-vr-the-emerging-trend-in-user-interface-and-user-experience-ui-ux-design-f6f1ae1b87ce>.
4. Virtual Reality (VR) guide [Electronic resource]. – Access mode: <https://www.interaction-design.org/literature/topics/virtual-reality#:~:text=Spatial%20Interaction%3A%20In%20VR%2C%20UI,unlike%20traditional%20D%20UI%20design>.
5. Deruette, A. Getting started with VR UI interface design [Electronic resource] / A. Deruette. – Access mode: <https://marvelapp.com/blog/getting-started-vr-interface-design/>.
6. What is a user interface (UI): types & examples [Electronic resource]. – Access mode: <https://daph.com/insights/user-interface-ui>.
7. What is Virtual Reality (VR) game development? Trends, examples, and essential tools for creating a VR game [Electronic resource]. – Access mode: <https://www.perforce.com/blog/vcs/virtual-reality-game-development#:~:text=Through%20sensors%2C%20these%20headsets%20can,sound%20and%20haptic%20controller%20feedback>.
8. Kruchkov, A. What is UI UX design in games: 10 differences or how to avoid a terrible mistake [Electronic resource] / A. Kruchkov. – Access mode: <https://kreonit.com/idea-generation-and-game-design/ui-ux-design-in->

games/#:~:text=While%20UI%20focuses%20on%20the,and%20eager%20to%20c
ontinue%20playing.

9. What is game UI? A complete beginner's guide to game interface design [Electronic resource]. – Access mode: <https://www.sketch.com/blog/game-ui-design/>.

10. Lutkevich, B. Virtual Reality gaming (VR gaming) [Electronic resource] / B. Lutkevich. – Access mode: <https://www.techtarget.com/whatis/definition/virtual-reality-gaming-VR-gaming#:~:text=Virtual%20reality%20environments%20are%20created,and%20laptops%2C%20and%20gaming%20consoles.>

11. Kim, Y.M. A systematic review of a Virtual Reality system from the perspective of user experience / Y. M. Kim, I. Rhiu, M. H. Yun // International Journal of Human–Computer Interaction. – 2020. – Vol. 36(10). - P. 893-910.

12. Shi, P. Analysis of interface design patterns in the Virtual Reality context / P. Shi // China Academic Journal Electronic Publishing House. – 2018. – Vol. 28. – P. 1-12.

13. Exploring head-based mode-switching in Virtual Reality // [R. Shi, N. Zhu, H.-N. Liang et al.] // Mixed and Augmented Reality: 2021 IEEE International symposium, Bari, Italy, 4-8 October, 2021: proceedings. – Los Alamitos: IEEE, 2021. - P. 118-127.

14. Wang, Y. An experimental investigation of menu selection for immersive virtual environments: fixed versus handheld menus / Y. Wang, Y. Hu, Y. Chen // Virtual Reality. – 2020. - Vol. 25. - P. 409–419.

15. Daniele, R. Virtual Reality applications: guidelines to design natural user interface / R. Daniele, C. Rizzi, A. Vitali // Computers and Information in Engineering Conference: proceedings. – American Society of Mechanical Engineers. - 2018. – P 1-15.

16. Kharoub, H. 3D user interface design and usability for immersive VR / H. Kharoub, M. Lataifeh, N. Ahmed // Applied Sciences. – 2019. - Vol. 9, № 22. - P. 1-29.

17. T. Zhu, Research on the application of human-computer interaction interface based on VR technology/ T. Zhu // Southeast University, 2019. – 299 p.
18. Tangible user interfaces for enhancing user experience of Virtual Reality cultural heritage applications for utilization in educational environment / [V. Hulusic, L. Gusia, N. Luci at. al] // Journal on Computing and Cultural Heritage. – 2023. -Vol. 16, № 2. – P. 1-24.
19. Mine, M. R. Virtual environment interaction techniques / M. R. Mine // Technical Report. University of North Carolina at Chapel Hill, USA. - 1995. – 30 p.
20. Dachzelt, R. Three-dimensional menus: a survey and taxonomy / R. Dachzelt, A. Hübner // Computers and Graphics. – 2007. – Vol. 31, № 1. – P. 53–65.
21. Bowman, D. A. Design and evaluation of menu systems for immersive virtual environments / D. A. Bowman, C. A. Wingrave // Virtual Reality: IEEE conference, Yokohama, Japan, 13-17 March 2001: proceedings. – Los Alamitos: IEEE, 2001. – P. 149–156.
22. Coninx, K. A hybrid 2D/3D user interface for immersive object modeling / K. Coninx, F. Van Reeth, E. Flerackers // International Computer Graphics: IEEE conference, Hasselt and Diepenbeek, Belgium, 23-27 June 1997: proceedings. – Los Alamitos: IEEE, 1997. – P. 47–55.
23. Jacoby, R. H. Using virtual menus in a virtual environment / R. H. Jacoby, S. R. Ellis // Electronic Imaging: Science and Technology: Symposium, San Jose, CA, USA, 9-14 February 1992: proceedings. – SPIE, 1992. - Vol. 1661. – P. 39–49.
24. Gerber, D. The spin menu: a menu system for virtual environments / D. Gerber, D. Bechmann // Virtual Reality: IEEE Conference, Bonn, Germany, 12-16 March 2005: proceedings. – Los Alamitos: IEEE, 2005. – P. 271–272.
25. Improving accessibility to intangible cultural heritage preservation using Virtual Reality/ [E. Selmanović, S. Rizvic, C. Harvey et.al] // Journal on Computing and Cultural Heritage, 2020. - Vol. 13, № 2. – P. 1–19.

26. Dourish, P. Where the action is: the foundations of embodied interaction / P. Dourish. – MIT Press, 2004. – 233 p.
27. Hornecker, E. The role of physicality in tangible and embodied interactions / E. Hornecker // *Interactions*, 2011. – Vol. 18, №2. – P. 19–23.
28. Ishii, H. Tangible bits: towards seamless interfaces between people, bits and atoms/ H. Ishii, B. Ullmer // *Human factors in computing systems: ACM SIGCHI Conference, Atlanta, Georgia, USA, March 22 - 27: proceedings.* – 1997. – P. 234–241.
29. Ishii, H. The tangible user interface and its evolution / H. Ishii // *Communications of the ACM*, 2008. – Vol. 51, №6. – P. 32–36.
30. Shaer, O. Tangible user interfaces: past, present, and future directions / O. Shaer, E. Hornecker // *Foundations and Trends in Human–Computer Interaction*, Vol. 3. - 2009.- P. 1-137.
31. ViBreathe: Heart rate variability enhanced respiration training for workaday stress management via an eyes-free tangible interface / [B. Yu, P. An, S. Hendriks et al.] // *International Journal of Human-Computer Interaction*, 2021. – Vol. 37, №16. – P. 1551–1570.
32. Vianello, A. TANGAEON: tangible interaction to support people in a mindfulness practice / A. Vianello, L. Chittaro, A. Matassa // *International Journal of Human-Computer Interaction*, 2019. – Vol. 35, №12. – P. 1086–1101.
33. A review on complementary natures of tangible user interfaces (TUIs) and early spatial learning/ [G. E. Baykal, I. V. Alaca, A. E. Yantaç et al.] // *International Journal of Child-Computer Interaction*, 2018. – Vol. 16. – P. 104–113.
34. Fernandes, R. Interactive design vs. design for interaction: developing interactive play tools that promote interactions between children / R. Fernandes, T. Yamanaka // *Interactive Multimedia.* - London: IntechOpen, 2019. – 228 p.
35. Comparing the use of tangible and graphical programming languages for informal science education / [M. S. Horn, E. T. Solovey, R. J. Crouser et al.] //

Human Factors in Computing Systems: SIGCHI conference, Boston, MA, USA, 4 - 9 April 2009: proceedings. – 2009. - P. 975–984.

36. Zuckerman, O. Extending tangible interfaces for education: digital montessori-inspired manipulatives / O. Zuckerman, S. Arida, M. Resnick // Human Factors in Computing Systems: ACM SIGCHI Conference, Portland Oregon USA, 2 - 7 April 2005: proceedings. – 2005. - P. 859–868.

37. Experimental evaluation of sketching on surfaces in VR / [R. Arora, R. H. Kazi, F. Anderson et al.] // Human Factors in Computing Systems: ACM SIGCHI Conference, New York, NY, USA, 6-11 May 2017: proceedings. – 2017. – P. 5643–5654.

38. On the shoulder of the giant: a multi-scale mixed reality collaboration with 360 video sharing and tangible interaction / [T. Piumsomboon, G. A. Lee, A. Irlitti et al.] // Human Factors in Computing Systems: ACM SIGCHI Conference, Glasgow Scotland Uk, 4 - 9 May 2019: proceedings. – 2019. – P. 1–17.

39. Insko, B. E. Passive haptics significantly enhances virtual environments/ B. E. Insko. - The University of North Carolina at Chapel Hill, 2001. – 200 p.

40. Simeone, A. L. Substitutional reality: using the physical environment to design Virtual Reality experiences / A. L. Simeone, E. Velloso, H. Gellersen // Human Factors in Computing Systems: 33rd Annual ACM Conference, Seoul Republic of Korea, 18 - 23 April 2015: proceedings. – 2015. – P. 3307–3316.

41. Kwon, E. Effects of sizes and shapes of props in tangible augmented reality / E. Kwon, G. J. Kim, S. Lee // Mixed and Augmented Reality: 8th IEEE International Symposium, Orlando, FL, USA, 19-22 October 2009: proceedings. – Los Alamitos: IEEE, 2009. – P. 201–202.

42. Slater, M. Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments / M. Slater // Philosophical Transactions of the Royal Society B: Biological Sciences, 2009. – Vol. 364. – P. 3549–3557.

43. Caicedo, D. G. What are tangible user interfaces? [Electronic resource] / D. G. Caicedo. – Access mode: <https://medium.com/@bluehair.co/what-are-tangible-user-interfaces-ce6a5aa5c912>.
44. Tangible interactions in Virtual Reality environments / [M. Sirizzottia, S. Guerciob, F. Lampusa et al.] // European Tangible Interaction Studio, Siena, Italy, 16-20 November 2020: proceedings. – 2020. – P.1-7.
45. Gobbetti, E. Virtual Reality: past, present and future / E. Gobbetti, R. Scateni // Studies in health technology and informatics, 1998. – Vol. 58. – P. 3–20.
46. Tsai, H.-R. ElasticVR: Providing multi-level active and passive force feedback in Virtual Reality using elasticity / H.-R. Tsai, J. Rekimoto // Human Factors in Computing Systems: CHI Conference, Montreal QC Canada, 21 - 26 April 2018: proceedings. – 2018. - P. 1–4.
47. TacTiles: dual-mode low-power electromagnetic actuators for rendering continuous contact and spatial haptic patterns in VR / [V. Vechev, J. Zarate, D. Lindlbauer et al.] // Virtual Reality and 3D User Interfaces (VR): IEEE conference, Osaka Japan, 23-27 March 2019: proceedings. – 2019. – P. 312–320.
48. Haptic revolver: touch, shear, texture, and shape rendering on a reconfigurable Virtual Reality controller / [E. Whitmire, H. Benko, C. Holz et al.] // Human Factors in Computing Systems: CHI Conference, Montreal QC Canada, 21 - 26 April 2018: proceedings. – 2018. – P. 1–12.
49. Sanchez-Vives, M. From presence to consciousness through Virtual Reality / M. Sanchez-Vives, M. Slater // Nature reviews Neuroscience, 2005. – Vol. 6. – P. 332–339.
50. Zahorik, P. Presence as being-in-the-world / P. Zahorik, R. L. Jenison // Presence: Teleoperators and Virtual Environments, 1998. – Vol. 7. – P. 78–89.
51. Slater, M. Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments / M. Slater // Philosophical Transactions of the Royal Society B: Biological Sciences, 2009. - Vol. 364. - P. 3549–3557.
52. Caggianese, G. The vive controllers vs. Leap Motion for interactions in virtual environments: a comparative evaluation / G. Caggianese, L. Gallo, P.

Neroni // Smart Innovation, Systems and Technologies, 2019. – Vol. 98. - P. 24–33.

53. Claw: A multifunctional handheld haptic controller for grasping, touching, and triggering in Virtual Reality/ [I. Choi, E. Ofek, H. Benko et al.] // Human Factors in Computing Systems: CHI Conference, Montreal QC Canada, 21 - 26 April 2018: proceedings. – 2018. - P. 1–13.

54. Aguerreche, L. Reconfigurable tangible devices for 3d virtual object manipulation by single or multiple users / L. Aguerreche, T. Duval, A. Lécuyer // Virtual Reality Software and Technology: 17th ACM Symposium, Hong Kong, 22 - 24 November 2010: proceedings. – 2010. – P. 227–230.

55. Schorr, S. B. Fingertip tactile devices for virtual object manipulation and exploration / S. B. Schorr, A. M. Okamura // Human Factors in Computing Systems: 2017 CHI Conference, Denver Colorado USA, 6 – 11 May 2017: proceedings. – 2017. – P. 3115–3119.

56. Bermejo, C. A survey on haptic technologies for mobile augmented reality/ C. Bermejo, P. Hui // ACM Computing Surveys (CSUR), 2017. – Vol. 54, Issue 9. – P. 1-35.

57. Project FEELEX: adding haptic surface to graphics / [H. Iwata, H. Yano, F. Nakaizumi et al.] // Computer graphics and interactive techniques: 28th annual conference, Los Angeles, California, USA, 12-17 August 2001: proceedings. – 2001. – P. 469 – 476.

58. Pangaro, G. The actuated workbench: computer controlled actuation in tabletop tangible interfaces / G. Pangaro, D. Maynes-Aminzade, H. Ishii // User Interface Software and Technology (UIST '02): Symposium, Paris France, 27-30 October 2002: proceedings. – 2002. – P. 181-190.

59. Israel, J. H. Tangible interfaces in virtual environments / J. H. Israel // Informatics for people: 36th Annual Conference of the German Society for Informatics, Dresden Germany, 2-6 October 2006: proceedings. – 2006. – P. 392-395.

60. Israel, J. H. Tangible interfaces for Virtual Reality based product development / J. H. Israel // Design Computing and Cognition: Second International Conference, Eindhoven, Netherlands, 10-12 July 2006: proceedings. – 2006. – P. 14 – 20.
61. Ullmer, B. Emerging frameworks for tangible user interfaces / B. Ullmer, H. Ishii // IBM Systems Journal, 2000. - Vol. 39, no. 3.4. - P. 915-931.
62. PyGame: a primer on game programming in python [Electronic resource]. – Access mode: <https://realpython.com/pygame-a-primer/>.
63. Яценко, О. Огляд переваг та недоліків ігрових рушіїв [Електрон. ресурс] / О. Яценко. – Режим доступу: <http://eprints.zu.edu.ua/40326/1/4444.pdf>.
64. Як обрати мову програмування для розробки ігор [Електрон. ресурс]. – Режим доступу: <https://pnn.com.ua/ua/blog/detail/how-to-select-programming-languages-for-game-development>.
65. Kucera, E. Connection between 3D engine unity and microcontroller Arduino: a virtual Smart House / E. Kucera, O. Haffner, S. Kozak // Cybernetics & Informatics (K&I): IEEE conference, Lazy pod Makytou Slovakia, 31 January-3 February 2018: proceedings. – Piscataway, NJ : IEEE, 2018. – P. 1–8.
66. Огляд плат Arduino [Електрон. ресурс]. – Режим доступу: https://distancionka.com/ua/blog/arduino/obzor-plat-arduino/?srsltid=AfmBOoqdwPO-A0_XXWJHHua3WHrncWB8wwTI-SLASW5FjJmGfaaHQBf-.
67. Buck, J. Mazes for programmers: code your own twisty little passages/ J. Buck //1st EditionPragmatic Bookshelf, 2015. - 288 p.
68. Maze generation: Sidewinder algorithm [Electronic resource]. – Access mode: <https://weblog.jamisbuck.org/2011/2/3/maze-generation-sidewinder-algorithm.html>.
69. Williams, L. K. Sidewinder maze generation [Electronic resource] / L. K. Williams. - Access mode: <https://www.integral-domain.org/lwilliams/Applets/algorithms/sidewinder.php>.

ДОДАТОК А
Текст програми

A.1 Код файла lab_3.py

```

import pygame
import screen
import time
import base
from Gmap import G_map as M
from tank import G_player as P

pygame.init()
pygame.mixer.init()

screen.startT = time.time()

victory = base.GraphicObj('textures/victory.png', 9.0, 5.0)
defeat = base.GraphicObj('textures/defeat.png', 9.0, 5.0)

pygame.display.set_caption("Tank Battle")

screen.loadSounds()

def GameResize():
    screen.G_screen =
    pygame.display.set_mode((screen.G_screen.get_width(),
    screen.G_screen.get_width() * screen.map_H / screen.map_W),
    pygame.RESIZABLE)
    screen.block_S = screen.G_screen.get_width() / screen.map_W
    P.transfigureP()
    M.transfigureM()
    victory.transfigure(9.0, 5.0)
    defeat.transfigure(9.0, 5.0)

G_run = True

GameResize()

while G_run:
    if screen.G_state == "":
        screen.clock.tick()
        screen.FPS = screen.clock.get_fps()

    M.drawM()

    if screen.G_state == "": M.updateM()

    P.drawP()

    if screen.G_state == "": P.updateP()

    if screen.G_state == "W":
        victory.draw_obj()
```

```

if screen.G_state == "L":
    defeat.draw_obj()

pygame.display.update()

for event in pygame.event.get():
    if event.type == pygame.QUIT:
        G_run = False
    if event.type == pygame.VIDEORESIZE:
        GameResize()

pygame.quit()

```

A.2 Код файла screen.py

```

import pygame

pygame.init()

def loadSounds():
    global defeatSnd, explodeSnd, hitSnd, musicSnd, shootSnd,
    victorySnd
    defeatSnd = pygame.mixer.Sound("sounds/defeat.wav")
    explodeSnd = pygame.mixer.Sound("sounds/explode.wav")
    hitSnd = pygame.mixer.Sound("sounds/hit.wav")
    shootSnd = pygame.mixer.Sound("sounds/shoot.wav")
    victorySnd = pygame.mixer.Sound("sounds/victory.wav")
    pygame.mixer.music.load("sounds/music.wav")
    pygame.mixer.music.play(-1)
    #####
    pygame.mixer.music.set_volume(0.2)
    defeatSnd.set_volume(1)
    explodeSnd.set_volume(1)
    hitSnd.set_volume(1)
    shootSnd.set_volume(1)
    victorySnd.set_volume(1)

G_state = ""
startT = 0

clock = pygame.time.Clock()
FPS = 0
def get_FPS():
    if FPS == 0:
        return 1
    return 1/FPS

block_S_0 = 90.0
block_S = 90.0
map_W = 18
map_H = 10

```

```
G_screen = pygame.display.set_mode((block_S * map_W / 2, block_S
* map_H / 2), pygame.RESIZABLE)
pygame.display.set_icon(pygame.image.load('textures/ptank.png'))
```

A.3 Код файла base.py

```
import pygame
import math
import screen

pygame.init()

class GraphicObj(pygame.sprite.Sprite):
    def __init__(self, texture, sX, sY):
        pygame.sprite.Sprite.__init__(self)
        self.base_image =
pygame.image.load(texture).convert_alpha()
        self.image = pygame.image.load(texture).convert_alpha()
        self.rect = self.image.get_rect()
        self.set_hitbox(sX, sY)
        self.angle = 0

    def set_hitbox(self, pX, pY):
        pX *= screen.block_S
        pY *= screen.block_S
        self.rect.center = (pX, pY)

    def draw_obj(self):
        cX, cY = self.rect.center
        screen.G_screen.blit(self.image, (cX - 0.5 *
self.image.get_width(), cY - 0.5 * self.image.get_height()))

    def transfigure(self, rX, rY, Rangle = 0):
        self.image = pygame.transform.scale(self.base_image, (1
+ self.base_image.get_width() * screen.block_S /
screen.block_S_0, 1 + self.base_image.get_height() *
screen.block_S / screen.block_S_0))
        self.rect.width, self.rect.height =
self.image.get_size()
        self.set_hitbox(rX, rY)
        self.angle += Rangle
        if self.angle > 360:
            self.angle -= 360
        if self.angle < 0:
            self.angle += 360
        self.image = pygame.transform.rotate(self.image,
self.angle)

class Bullet(GraphicObj):
    def __init__(self, stX, stY, Sangle, Stexture, Sspeed, stHP
= 0):
        GraphicObj.__init__(self, Stexture, stX, stY)
```

```

        self.pX = stX
        self.pY = stY
        self.Pspeed = Sspeed
        self.transfigure(stX, stY, Sangle)
        self.pHP = stHP

    def fly(self):
        self.pX -= math.sin(self.angle * 3.14159265 / 180) *
self.Pspeed * screen.get_FPS()
        self.pY -= math.cos(self.angle * 3.14159265 / 180) *
self.Pspeed * screen.get_FPS()
        self.set_hitbox(self.pX, self.pY)
        if (self.pX < 0.1 or self.pX > screen.map_W - 0.1 or
self.pY < 0.1 or self.pY > screen.map_H - 0.1):
            self.kill()

```

A.4 Код файла tank.py

```

import pygame
import screen
import base
import math
import time
from Gmap import G_map as M

pygame.init()

class Explosion(base.GraphicObj):
    def __init__(self, Stexture, sX, sY, sTime, level):
        base.GraphicObj.__init__(self, Stexture, sX, sY)
        self.eX = sX
        self.eY = sY
        self.eTime = sTime
        self.transfigure(sX, sY)
        if level == "H": screen.hitSnd.play()
        if level == "E": screen.explodeSnd.play()
    def eCheck(self):
        if (time.time() - self.eTime >= 0.4):
            self.kill()

class Player(base.GraphicObj):
    def __init__(self):
        base.GraphicObj.__init__(self, 'textures/ptank.png',
0.5, screen.map_H - 0.5)
        self.tX = 0.5
        self.tY = screen.map_H - 0.5
        self.Tspeed = 1
        self.tHP = 3
        self.plbullets = pygame.sprite.Group()
        self.last_Pshoot = time.time()
        self.explosions = pygame.sprite.Group()

```

```

def pastCollide(self, bulletPst, wallPst, pstX, pstY):
    colRect = bulletPst.rect.copy()
    vX = bulletPst.pX + pstX * math.sin(bulletPst.angle *
3.14159265 / 180) * bulletPst.Pspeed * screen.get_FPS()
    vY = bulletPst.pY + pstY * math.cos(bulletPst.angle *
3.14159265 / 180) * bulletPst.Pspeed * screen.get_FPS()
    colRect.center = (vX * screen.block_S, vY *
screen.block_S)
    return colRect.colliderect(wallPst.rect)

def ricochet(self, bulletRch, wallRch):
    if self.pastCollide(bulletRch, wallRch, 1, 1):
        bulletRch.kill()
    elif self.pastCollide(bulletRch, wallRch, 0, 1):
        bulletRch.angle = 360.0 - bulletRch.angle
        bulletRch.transfigure(bulletRch.pX, bulletRch.pY)
    elif self.pastCollide(bulletRch, wallRch, 1, 0):
        bulletRch.angle = 180.0 - bulletRch.angle
        bulletRch.transfigure(bulletRch.pX, bulletRch.pY)
    else: bulletRch.kill()

def plbullCollision(self):
    for bullet in self.plbullets:
        bullet_hit = False
        for group in [M.enemies, M.crates]:
            for obj in group:
                if bullet.rect.colliderect(obj.rect):
                    if obj.bHP == 1:

self.explosions.add(Explosion('textures/explode.png', obj.bX,
obj.bY, time.time(), "E"))
                    obj.kill()
                    else:

self.explosions.add(Explosion('textures/hit.png', bullet.pX,
bullet.pY, time.time(), "H"))
                    obj.bHP -= 1
                    bullet.kill()
                    bullet_hit = True
                    break
                if bullet_hit: break
            if bullet_hit: continue
        for wall in M.walls:
            if bullet.rect.colliderect(wall.rect):
                if bullet.pHP == 0:
                    bullet.kill()
                else:
                    bullet.pHP -= 1
                    self ricochet(bullet, wall)
                    break

def enbullCollision(self):
    for bullet in M.enbullets:

```

```

        if bullet.rect.colliderect(self.rect):
            bullet.kill()
            self.tHP -= 1
            if self.tHP == 0:

self.explosions.add(Explosion('textures/explode.png',      self.tX,
self.tY, time.time(), "E"))
            break
        else:

self.explosions.add(Explosion('textures/hit.png',          bullet.pX,
bullet.pY, time.time(), "H"))
            continue
        #####
        bullet_hit = False
        for group in [M.walls, M.crates]:
            for obj in group:
                if bullet.rect.colliderect(obj.rect):
                    bullet.kill()
                    bullet_hit = True
                    break
            if bullet_hit: break
        #####

def drawP(self):
    self.draw_obj()
    for bullet in self.plbullets:
        bullet.draw_obj()
    for explode in self.explosions:
        explode.draw_obj()

def updateP(self):
    self.controls()
    self.plbullCollision()
    self.enbullCollision()
    if len(M.enemies) == 0:
        pygame.mixer.music.stop()
        screen.victorySnd.play()
        screen.G_state = "W"
        return
    if self.tHP == 0:
        pygame.mixer.music.stop()
        screen.defeatSnd.play()
        screen.G_state = "L"
        return
    for bullet in self.plbullets:
        bullet.fly()
    for explode in self.explosions:
        explode.eCheck()

def transfigureP(self):
    self.transfigure(self.tX, self.tY)
    for bullet in self.plbullets:

```

```

        bullet.transfigure(bullet.pX, bullet.pY)
    for explode in self.explosions:
        explode.transfigure(explode.eX, explode.eY)

    def moveCollision(self, vX, vY):
        colRect = self.rect.copy()
        vX = (self.tX + vX) * screen.block_S
        vY = (self.tY + vY) * screen.block_S
        colRect.center = (vX, vY)
        for group in [M.walls, M.crates, M.enemies]:
            if any(colRect.colliderect(obj.rect) for obj in
group):
                return True
        return False

    def move(self, dX, dY):
        dX *= self.Tspeed * screen.get_FPS()
        dY *= self.Tspeed * screen.get_FPS()
        if (dX != 0 and self.tX + dX > 0.45 and self.tX + dX <
screen.map_W - 0.45):
            if (not self.moveCollision(dX, 0)):
                self.tX += dX
                self.set_hitbox(self.tX, self.tY)
        if (dY != 0 and self.tY + dY > 0.45 and self.tY + dY <
screen.map_H - 0.45):
            if (not self.moveCollision(0, dY)):
                self.tY += dY
                self.set_hitbox(self.tX, self.tY)

    def shootP(self):
        curr_time = time.time()
        if curr_time - self.last_Pshoot >= 0.7:
            screen.shootSnd.play()
            self.last_Pshoot = curr_time
            shX = self.tX - math.sin(self.angle * 3.14159265 /
180) * 0.5
            shY = self.tY - math.cos(self.angle * 3.14159265 /
180) * 0.5
            self.plbullets.add(base.Bullet(shX, shY, self.angle,
'textures/plbullet.png', 3, 3))

    def controls(self):
        pkey = pygame.key.get_pressed()
        if pkey[pygame.K_s] == True or pkey[pygame.K_DOWN] ==
True:
            dltX = math.sin(self.angle * 3.14159265 / 180)
            dltY = math.cos(self.angle * 3.14159265 / 180)
            self.move(dltX, dltY)
        if pkey[pygame.K_w] == True or pkey[pygame.K_UP] ==
True:
            dltX = -math.sin(self.angle * 3.14159265 / 180)
            dltY = -math.cos(self.angle * 3.14159265 / 180)
            self.move(dltX, dltY)

```

```

        if pkey[pygame.K_a] == True or pkey[pygame.K_LEFT] ==
True:
            self.transfigure(self.tX, self.tY, self.Tspeed * 50
* screen.get_FPS())
            if pkey[pygame.K_d] == True or pkey[pygame.K_RIGHT] ==
True:
                self.transfigure(self.tX, self.tY, -self.Tspeed * 50
* screen.get_FPS())
                if pkey[pygame.K_SPACE] == True:
                    self.shootP()

G_player = Player()

```

A.5 Код файлу Gmap.py

```

import pygame
import screen
import base
import random
import time
import math

pygame.init()

class Block(base.GraphicObj):
    def __init__(self, Stexture, sX, sY, Stype = "", sHP = 1,
Sangle = 0):
        base.GraphicObj.__init__(self, Stexture, sX, sY)
        self.type = Stype
        self.bX = sX
        self.bY = sY
        self.bHP = sHP
        self.transfigure(sX, sY, Sangle)

    def change(self, Stexture, Stype):
        self.base_image =
pygame.image.load(Stexture).convert_alpha()
        self.image = pygame.image.load(Stexture).convert_alpha()
        self.type = Stype

class Map:
    def __init__(self):
        self.blockmap = []
        self.walls = pygame.sprite.Group()
        self.crates = pygame.sprite.Group()
        self.enemies = pygame.sprite.Group()
        self.enbullets = pygame.sprite.Group()
        self.last_Eshoot = time.time()
        self.generate_maze()

```

```

def drawM(self):
    for X in range(screen.map_W):
        for Y in range(screen.map_H):
            self.blockmap[X][Y].draw_obj()
    for crate in self.crates:
        crate.draw_obj()
    for enemy in self.enemies:
        enemy.draw_obj()
    for bullet in self.enbullets:
        bullet.draw_obj()

def shootM(self):
    curr_time = time.time()
    if curr_time - self.last_Eshoot >= 1.4:
        screen.shootSnd.play()
        self.last_Eshoot = curr_time
        for enemy in self.enemies:
            shX = enemy.bX - math.sin(enemy.angle *
3.14159265 / 180) * 0.5
            shY = enemy.bY - math.cos(enemy.angle *
3.14159265 / 180) * 0.5
            self.enbullets.add(base.Bullet(shX, shY,
enemy.angle, 'textures/enbullet.png', 2))

def updateM(self):
    self.shootM()
    for bullet in self.enbullets:
        bullet.fly()
    if time.time() - screen.startT >= 0.2:
        for enemy in self.enemies:
            enemy.transfigure(enemy.bX, enemy.bY, 50 *
screen.get_FPS())

def transfigureM(self):
    for X in range(screen.map_W):
        for Y in range(screen.map_H):
            self.blockmap[X][Y].transfigure(X + 0.5, Y +
0.5)
    for crate in self.crates:
        crate.transfigure(crate.bX, crate.bY)
    for enemy in self.enemies:
        enemy.transfigure(enemy.bX, enemy.bY)
    for bullet in self.enbullets:
        bullet.transfigure(bullet.pX, bullet.pY)

def generate_maze(self):
    for X in range(screen.map_W):
        Mrow = []
        for Y in range(screen.map_H):
            Sblock = Block('textures/wall.png', X + 0.5, Y +
0.5, "Wall")
            Mrow.append(Sblock)
        self.walls.add(Mrow)

```

```

        self.blockmap.append(Mrow)

        self.walls.remove(self.blockmap[0][9])
        self.blockmap[0][9].change('textures/grass.png',
"Grass")
        stack = [(0, 9)]

        directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]
        while stack:
            x, y = stack.pop()
            random.shuffle(directions)
            for dx, dy in directions:
                nx, ny = x + dx, y + dy
                if 0 <= nx < screen.map_W and 0 <= ny <
screen.map_H and self.blockmap[nx][ny].type == "Wall":
                    walls_count = sum(
                        1 for ddx, ddy in directions
                        if 0 <= nx + ddx < screen.map_W and 0 <=
ny + ddy < screen.map_H and self.blockmap[nx + ddx][ny +
ddy].type == "Wall"
                    )
                    if walls_count >= 2:
                        self.walls.remove(self.blockmap[nx][ny])

self.blockmap[nx][ny].change('textures/grass.png', "Grass")
                        stack.append((nx, ny))

        Eamnt = 0
        while (Eamnt != 7):
            for X in range(screen.map_W):
                for Y in range(screen.map_H):
                    if self.blockmap[X][Y].type == "Grass" and
(X != 0 or Y != 9):
                        if (random.random() > 0.95):
                            self.blockmap[X][Y].type = "Enemy"

self.enemies.add(Block('textures/etank.png', X + 0.5, Y + 0.5,
sHP = 3, Sangle = random.random() * 360.0))
                            Eamnt += 1
                            if Eamnt == 7: break
            if Eamnt == 7: break

            for X in range(screen.map_W):
                for Y in range(screen.map_H):
                    if self.blockmap[X][Y].type == "Grass" and (X !=
0 or Y != 9):
                        if (random.random() > 0.85):

self.crates.add(Block('textures/crate.png', X + 0.5, Y + 0.5))
G_map = Map()

```

Додаток Б
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дипломна кваліфікаційна робота бакалавра

ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ МАТЕРІАЛЬНОГО ІНТЕРФЕЙСУ КОРИСТУВАЧА ІГРОВОГО ЗАСТОСУНКУ

Виконав
ст. гр. КНТ-112сп

Денис ГРИЦЮК

Керівник
к.т.н., доцент

Анжеліка ПАРХОМЕНКО

Рисунок Б.1 – Слайд 1



Рисунок Б.2 – Слайд 2

МЕТА ТА ЗАВДАННЯ РОБОТИ

Мета роботи – дослідження та розробка програмно-апаратного комплексу для покращення досвіду взаємодії користувача з ігровим середовищем через матеріальний інтерфейс.

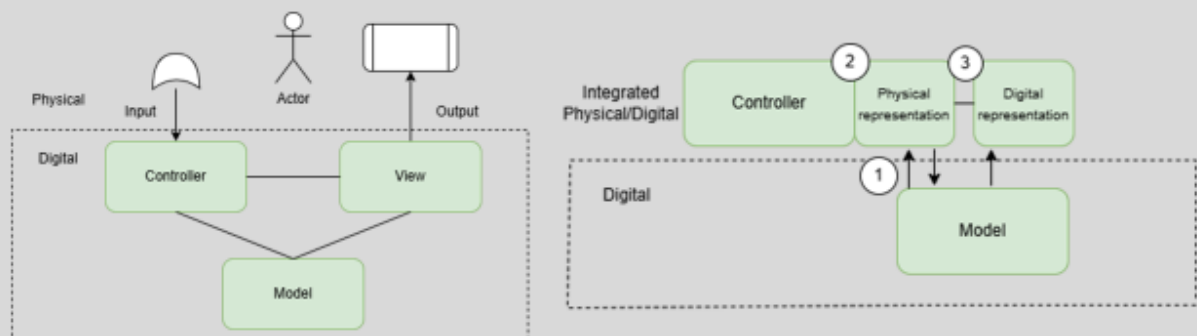
Завдання роботи:

- виконати аналіз вимог до програмно-апаратного забезпечення ігрового застосунку;
- розробити модель взаємодії з ігровим застосунком через матеріальний інтерфейс;
- обрати програмні та апаратні засоби розробки;
- виконати розробку прототипу геймпаду та програмного забезпечення для взаємодії через матеріальний інтерфейс;
- виконати програмну реалізацію ігрового середовища.

3

Рисунок Б.3 – Слайд 3

МОДЕЛЬ ВЗАЄМОДІЇ MVC



Базова та модифікована моделі взаємодії «Model-View-Controller»

4

Рисунок Б.4 – Слайд 4



Рисунок Б.5 – Слайд 5

АНАЛІЗ ПРОГРАМНИХ ЗАСОБІВ РОЗРОБКИ

Рушій/ Критерії	Мульти- платформність	Мова сценаріїв	Поріг входження
Unreal Engine	Windows, OSX, Linux	C++, Blueprints	Не вимагає високого рівня навичок програмування
Unity	Windows, OSX, Linux	UnityScript, C#	Вимагає володіння навичками програмування
CryEngine	Windows, OSX, Linux	C++, Lua	Вимагає високого рівня навичок програмування
Pygame	Windows, macOS, Linux	Python, SDL	Не вимагає високого рівня навичок програмування
Panda3D	Windows, Linux, macOS, Android и iOS	Python, OpenGL	Вимагає високого рівня навичок програмування

Вибір рушія

6

Рисунок Б.6 – Слайд 6

АНАЛІЗ ПРОГРАМНИХ ЗАСОБІВ РОЗРОБКИ

Критерії / Мови програмування	C++	Python	Javascript	C#
Продуктивність	+++	+	+	++
Тип гри	3D	2D	2D	3D
Ігровий рушій	Unity, Unreal Engine, CryEngine	Panda3D, Pygame	Phaser	Unity
Простота	-	+	+	-
Швидкодія	-	++	-	+
Поріг входження	+++	+	+	++

Вибір мови програмування

7

Рисунок Б.7 – Слайд 7

АНАЛІЗ АПАРАТНИХ ЗАСОБІВ РОЗРОБКИ

Плата Arduino	МК	Цифр. вх/вих	Вих. з ШІМ	Аналог. вх/вих	Flash пам'ять, Кб	ОЗУ, Кб	EEPROM-пам'ять, Кб	Габарити, мм	USB-роз'єм
Uno	Atmega328	14	6	6	32	2	1	69x53	USB A-B
Mega	Atmega2560	54	14	16	256	8	4	102x53	USB A-B
Nano	Atmega328	14	6	8	32	2	1	18x45	Mini- USB
Micro	Atmega32u4	20	7	12	32	2,5	1	48x18	Micro-USB
Mini	Atmega328	14	6	6	32	2	1	30x18	USB-Serial

Вибір плати Arduino

8

Рисунок Б.8 – Слайд 8

КОНСТРУЮВАННЯ ПРОГРАМНО-АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ

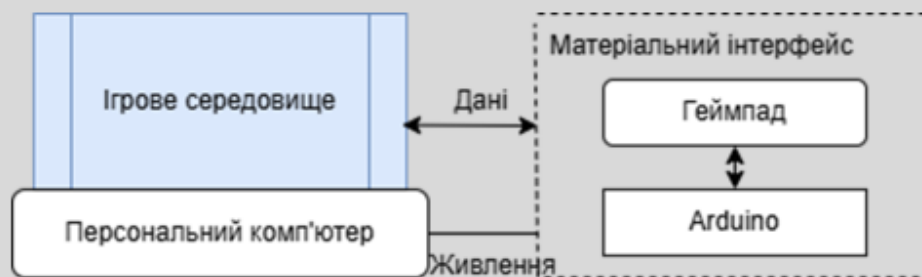


Схема алгоритму та реалізація побудови лабіринту для ігрової мапи

9

Рисунок Б.9 – Слайд 9

КОНСТРУЮВАННЯ ПРОГРАМНО-АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ



Структурна схема програмно-апаратного комплексу

10

Рисунок Б.10 – Слайд 10

КОНСТРУЮВАННЯ ПРОГРАМНО-АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ



Файлова структура проекту

Створені класи:

- GraphicObj – клас, що є основою для будь-яких графічних об'єктів;
- Bullet – клас, що є основою для таких об'єктів, як снаряди ворогів та гравця;
- Explosion – клас, що є основою для таких об'єктів, як вибухи різних типів;
- Player – клас, що є основою для такого об'єкта, як танк гравця;
- Block – клас, що є основою для таких об'єктів, як блоки ігрової мапи різних типів;
- Map – клас, що є основою для такого об'єкта, як ігрова мапа.

11

Рисунок Б.11 – Слайд 11

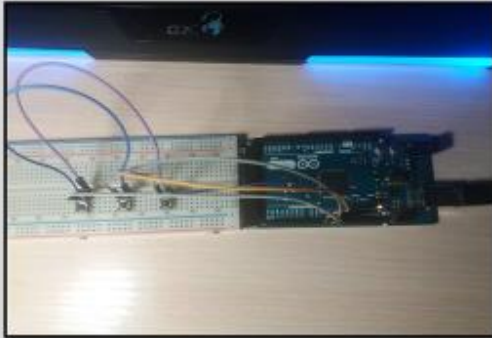
ГРАФІЧНИЙ ІНТЕРФЕЙС ІГРОВОГО ЗАСТОСУНКУ



12

Рисунок Б.12 – Слайд 12

МАТЕРІАЛЬНИЙ ІНТЕРФЕЙС ІГРОВОГО ЗАСТОСУНКУ



Прототип геймпаду на базі Arduino Mega

```

if (sp.IsOpen)
{
    SerialPort = true;
    if (sp.ReadByte() == 1)
    {
        transform.position += new Vector3(0,
0.5f);
    }
    if (sp.ReadByte() == 2)
    {
        transform.position += new Vector3(0, -
0.5f);
    }
    if (sp.ReadByte() == 3)
    {
        FindObjectOfType<ShipController>().Shoot();
    }
    catch (System.Exception)
    {
    }
}

```

Зчитування сигналів кнопок

13

Рисунок Б.13 – Слайд 13

ВИСНОВКИ

- В результаті виконання дипломної роботи було розроблено ігровий застосунок з матеріальним інтерфейсом, що дозволило уникнути проблем дизайну складних інтерфейсів користувача та зробити досвід використання ігрового середовища більш цікавим.
- Використання плати Arduino дозволило створити недорогий прототип геймпаду, що надає три кнопки керування, але може бути удосконалений шляхом додавання додаткових інтерактивних елементів керування, зокрема кнопок, а також зумерів та світлодіодів для зворотнього зв'язку та нотифікації.
- В подальшій роботі можливо створити 3D варіант гри, з використанням іншого ігрового рушія, зокрема Unity/Unreal Engine/Godot. Також можливо реалізувати використання додаткових технологій для розширення функцій матеріального інтерфейсу та реалізації бездротового зв'язку геймпаду з ігровим середовищем.

14

Рисунок Б.14 – Слайд 14