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

Навчально-науковий
інститут електроенергетики
(інститут)
Факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня магістра

студента Чалої Карини Дмитрівни
(ПІБ)
академічної групи 123м-24з-1
(шифр)
спеціальності 123 Комп'ютерна інженерія
(код і назва спеціальності)
за освітньо-професійною програмою «Комп'ютерна інженерія»
(офіційна назва)
на тему «Обґрунтування структури та параметрів комп'ютерної системи Дніпровської початкової школи з інтегрованим розвиваючим ігровим застосунком»
(назва за наказом ректора)

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	проф. Цвіркун Л.І.			
розділів:				
синтез системи	доц. Ткаченко С.М.			
Розроблення програмного забезпечення	ас. Панферова Я.В.			

Рецензент				

Нормоконтролер	проф. Цвіркун Л.І.			

Дніпро
2025

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)
_____ В. В. Гнатушенко
(підпис) (ініціали, прізвище)

«_____» _____ 2025 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеню ___магістра_____
(бакалавра, спеціаліста, магістра)

здобувача вищої освіти ___Чалій К.Д.___ **академічної групи** _____123м-24з-1_____
(прізвище та ініціали) (шифр)

спеціальності _____123 Комп'ютерна інженерія_____
(код і назва спеціальності)

за освітньо-професійною програмою _____«Комп'ютерна інженерія»_____
(офіційна назва)

на тему «Обґрунтування структури та параметрів комп'ютерної системи Дніпровської початкової школи з інтегрованим розвиваючим ігровим застосунком» ,

затверджену наказом ректора НТУ «Дніпровська політехніка» від 13.10.2025 № 1166-с

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання	На основі матеріалів практик, інших науково-технічних джерел сформулювати наукове завдання, конкретизувати предмет та мету досліджень	10.10.2025
Теоретичний	Обґрунтувати теоретичні засади побудови комп'ютерних систем освітнього призначення та інтеграції розвиваючих ігрових застосунків у локальну мережу Дніпровської початкової школи.	24.10.2025
Синтез системи	Розробка структури та параметрів комп'ютерної системи Дніпровської початкової школи	14.11.2025
Розроблення програмного забезпечення	Розробка програмного забезпечення розвиваючого ігрового застосунку для комп'ютерної системи Дніпровської початкової школи.	28.11.2025
Експериментальний розділ	Тестування спроєктованої комп'ютерної системи та ігрового застосунку в умовах, наближених до реальної експлуатації.	05.12.2025

Завдання видано _____ _проф. Л.І. Цвіркун__
(підпис керівника) (ініціали, прізвище)

Дата видачі 05 вересня 2025 р

Дата подання до екзаменаційної комісії __19.12.2025 р._

Прийнято до виконання _____ _____К.Д. Чала___
(підпис здобувача вищої освіти) (ініціали, прізвище)

РЕФЕРАТ

Пояснювальна записка 76 стор., 10 рис., 2 табл., 1 дод., 12 джерел.

Об’єктом дослідження є комп’ютерна система Дніпровської початкової школи, що включає мережеві пристрої та інтегрований розвиваючий ігровий застосунок.

Метою магістерської роботи є обґрунтування структури та визначення параметрів комп’ютерної системи для забезпечення стабільної роботи ігрового застосунку та підвищення ефективності навчального процесу.

У роботі застосовано методи системного аналізу, моделювання та синтезу локальних обчислювальних мереж, аналіз навантаження на серверні й клієнтські компоненти, а також експериментальне моделювання та практичне тестування ігрового застосунку в умовах експлуатації.

У результаті дослідження розроблено та обґрунтовано структуру комп’ютерної системи початкової школи з інтегрованим ігровим застосунком, визначено оптимальні параметри мережевого обладнання, серверної підсистеми та клієнтських комп’ютерів. Наукова новизна полягає у поєднанні централізованої комп’ютерної інфраструктури з освітнім ігровим програмним забезпеченням, що забезпечує спільний доступ до ресурсів, збереження прогресу учнів та спрощення адміністрування системи.

Робота є логічним продовженням попередніх досліджень у галузі проєктування комп’ютерних мереж, серверних рішень та розроблення освітнього програмного забезпечення.

Висновки підтверджують, що обґрунтований вибір структури та параметрів комп’ютерної системи забезпечує стабільну роботу ігрового застосунку та надійне збереження даних.

Подальший розвиток системи передбачає розширення функціоналу ігрового застосунку, використання хмарних сервісів для збереження даних та впровадження сучасних мережевих технологій з метою підвищення масштабованості й гнучкості освітньої інфраструктури.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК, ОДИНИЦЬ І ТЕРМІНІВ

NAS – мережеве сховище даних, спеціалізований сервер для зберігання та спільного доступу до файлів у локальній мережі;

SMB – мережевий протокол для доступу до файлів, папок і принтерів у локальній мережі, широко використовується в ОС Windows та NAS-системах;

RAID1 – рівень RAID із дзеркальним копіюванням даних на два диски для підвищення надійності зберігання інформації;

HDD – жорсткий дисковий накопичувач, пристрій для довготривалого зберігання даних на магнітних носіях;

DDR – тип оперативної пам’яті з подвоєною швидкістю передачі даних за такт;

МБ – одиниця вимірювання обсягу інформації;

ACL – список контролю доступу, що визначає права користувачів або груп на доступ до ресурсів (файлів, папок, сервісів) у системі.

ВСТУП

Сучасний етап розвитку системи освіти характеризується активним впровадженням інформаційних технологій у навчальний процес. Комп'ютерні системи дедалі частіше використовуються не лише як допоміжний інструмент, а як повноцінний елемент освітнього середовища, що впливає на методи навчання, організацію роботи вчителя та пізнавальну активність учнів. Особливо важливою є роль комп'ютерних технологій у початковій школі, де формуються базові навчальні навички та закладаються основи логічного мислення.

Одним із перспективних напрямів використання комп'ютерної техніки у початковій школі є застосування розвиваючих ігрових програм. Такі програми дозволяють поєднати навчальний матеріал з елементами гри, що підвищує зацікавленість учнів, полегшує сприйняття інформації та сприяє індивідуалізації навчального процесу. Водночас ефективність використання розвиваючих ігрових застосунків значною мірою залежить від технічної реалізації комп'ютерної системи, у межах якої вони функціонують.

Практика експлуатації комп'ютерних класів у закладах початкової освіти свідчить, що комп'ютерні системи часто формуються без належного системного обґрунтування. Програмне забезпечення встановлюється локально на кожному робочому місці, що ускладнює його оновлення, призводить до різних версій програм та створює ризики втрати даних. За таких умов інтеграція розвиваючих ігрових застосунків не забезпечує стабільної та керованої роботи, особливо при одночасному використанні декількома учнями.

У зв'язку з цим виникає необхідність науково обґрунтованого підходу до побудови комп'ютерної системи початкової школи, яка б враховувала реальні умови експлуатації, обмежені апаратні ресурси та потребу в централізованому управлінні програмним забезпеченням. Особливо актуальним є синтез такої комп'ютерної системи, що дозволяє інтегрувати розвиваючий ігровий застосунок без необхідності його локальної інсталяції на кожному клієнтському

комп’ютері та забезпечує надійне збереження результатів навчальної діяльності учнів.

Таким чином, тема є актуальною, має практичну спрямованість і відповідає сучасним вимогам до організації освітнього процесу із застосуванням інформаційних технологій.

Мета дослідження – обґрунтування структури та визначення оптимальних параметрів комп’ютерної системи Дніпровської початкової школи для забезпечення ефективної та стабільної роботи інтегрованого розвиваючого ігрового застосунку.

Об’єкт дослідження – комп’ютерна система початкової школи як сукупність апаратних, мережевих і програмних засобів навчального призначення.

Предмет дослідження – структура та параметри комп’ютерної системи початкової школи і методи їх обґрунтування з урахуванням вимог інтеграції розвиваючого ігрового застосунку.

Методи дослідження. Системний аналіз комп’ютерних систем, методи моделювання та синтезу локальних обчислювальних мереж, аналіз навантаження на серверні та клієнтські компоненти, а також експериментальне моделювання та практичне тестування.

1 СТАН ПИТАННЯ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Стан питання та актуальність

На сучасному етапі розвитку освіти комп’ютерні системи відіграють важливу роль у забезпеченні навчального процесу, особливо в закладах початкової освіти. Використання інформаційних технологій у молодших класах спрямоване не лише на ознайомлення учнів з основами цифрової грамотності, а й на підтримку навчальної діяльності за допомогою інтерактивних та ігрових форм подання матеріалу.

Одним із найбільш поширених інструментів у цьому напрямі є розвиваючі ігрові застосунки, які дозволяють адаптувати навчальний матеріал до вікових особливостей учнів, забезпечують наочність та підвищують мотивацію до навчання. Проте ефективність таких застосунків значною мірою залежить від того, у межах якої комп’ютерної системи вони функціонують.

Аналіз існуючих підходів до організації комп’ютерних класів показує, що у більшості навчальних закладів програмне забезпечення встановлюється локально на кожному робочому місці. Такий підхід ускладнює оновлення програм, призводить до різних версій програмного забезпечення, а також створює проблеми із збереженням результатів роботи учнів. Крім того, локальна інсталяція підвищує вимоги до апаратних ресурсів кожного клієнтського комп’ютера.

Таким чином, актуальним є завдання обґрунтування структури комп’ютерної системи початкової школи, яка дозволяє інтегрувати розвиваючий ігровий застосунок у централізованому режимі з урахуванням реальних умов експлуатації та обмежених ресурсів навчального закладу.

1.2 Проблемна ситуація і протиріччя

У процесі експлуатації комп’ютерної інфраструктури початкової школи виникає низка проблем, пов’язаних із використанням навчального програмного забезпечення. Зокрема, при одночасній роботі декількох учнів виникають

труднощі з підтримкою однакових версій програм, збереженням індивідуального прогресу та стабільністю роботи системи.

Проблемна ситуація полягає в тому, що навчальний процес вимагає централізованого управління програмними ресурсами, тоді як наявна комп'ютерна інфраструктура зазвичай орієнтована на автономну роботу окремих клієнтських комп'ютерів. Це створює додаткове навантаження на технічний персонал та знижує надійність функціонування системи в цілому.

Основне протиріччя розвитку полягає у невідповідності між зростаючими вимогами до використання інтерактивних розвиваючих програм у початковій школі та можливостями традиційних способів організації комп'ютерних систем. Подолання цього протиріччя можливе шляхом застосування клієнт–серверного підходу з централізованим зберіганням програмних ресурсів і даних користувачів.

1.3 Характеристика галузі та об'єкта дослідження

Галузь дослідження охоплює сферу проєктування, експлуатації та розвитку комп'ютерних систем освітнього призначення. Зокрема, дослідження належить до напрямів комп'ютерної інженерії, що пов'язані з побудовою локальних обчислювальних систем, клієнт–серверних архітектур та інтеграцією програмного забезпечення у реальних умовах експлуатації.

Особливістю комп'ютерних систем освітньої галузі є поєднання технічних, організаційних та педагогічних вимог. З одного боку, такі системи повинні відповідати стандартним вимогам до надійності, продуктивності та інформаційної безпеки. З іншого боку, вони мають бути простими в адмініструванні, стабільними в роботі та адаптованими до користувачів, які не мають спеціальної технічної підготовки.

У галузі початкової освіти комп'ютерні системи використовуються переважно як інструмент підтримки навчального процесу. Тому ключовими вимогами є централізоване управління програмним забезпеченням, можливість одночасної роботи декількох користувачів, а також гарантоване збереження

навчальних результатів. Це зумовлює необхідність застосування системного підходу до синтезу комп’ютерних систем, орієнтованих саме на освітнє середовище.

Об’єктом дослідження є комп’ютерна система Дніпровської початкової школи, яка використовується для забезпечення навчального процесу з використанням інформаційних технологій.

Комп’ютерна система школи включає сукупність клієнтських робочих місць, мережевого обладнання та серверної підсистеми, об’єднаних у локальну обчислювальну мережу. Робочі місця представлені персональними комп’ютерами різних типів (ноутбуки, моноблоки), які використовуються учнями та педагогічними працівниками під час проведення навчальних занять.

Серверна частина системи виконує функції централізованого зберігання даних і програмних ресурсів. Саме на серверному вузлі розміщується розвиваючий ігровий застосунок, а також дані індивідуального прогресу учнів. Мережева підсистема забезпечує обмін інформацією між клієнтськими робочими місцями та сервером у межах локальної мережі школи.

Об’єкт дослідження функціонує в умовах обмежених апаратних ресурсів та постійного використання різними групами користувачів, що зумовлює підвищені вимоги до стабільності, надійності та простоти експлуатації системи.

1.4 Організаційна структура об’єкта дослідження

Організаційна структура Дніпровської початкової школи визначає порядок використання та обслуговування комп’ютерної системи в навчальному процесі. Безпосередніми користувачами системи є учні та вчителі, які застосовують комп’ютерну техніку під час проведення уроків та позакласних занять.

Адміністрування комп’ютерної системи здійснюється відповідальною особою (вчителем інформатики або системним адміністратором), до функцій якої належать налаштування мережевого обладнання, контроль доступу до серверних ресурсів та забезпечення працездатності системи в цілому. При

цьому кількість технічного персоналу є обмеженою, що підвищує вимоги до простоти та надійності обраного технічного рішення.

Організаційна структура об’єкта передбачає використання комп’ютерної системи в режимі колективного доступу, коли декілька користувачів одночасно працюють з одним програмним ресурсом. Це накладає додаткові вимоги на організацію інформаційних потоків, розмежування доступу та збереження індивідуальних результатів роботи.

Саме з урахуванням організаційної структури об’єкта дослідження обґрунтовується доцільність використання централізованої клієнт-серверної архітектури з запуском розвиваючого ігрового застосунку з серверного вузла.

1.5 Постановка завдання

Об’єкт дослідження: комп’ютерна інфраструктура початкової школи, що включає клієнтські робочі місця, локальну мережу Ethernet та серверний вузол з обмеженими апаратними ресурсами.

Потрібно визначити: структуру та параметри комп’ютерної системи, які забезпечують централізований запуск розвиваючого ігрового застосунку, оптимальну організацію інформаційних зв’язків між підсистемами та надійне збереження даних прогресу учнів.

Завдання має оптимізаційний характер і передбачає вибір технічного рішення, що забезпечує максимальну ефективність функціонування системи за умов заданих обмежень.

2 ТЕОРЕТИЧНИЙ РОЗДІЛ

2.1 Загальна характеристика об’єкта дослідження як комп’ютерної системи

Об’єкт дослідження у даній роботі розглядається як комп’ютерна система освітнього призначення, що функціонує в умовах локальної мережі навчального закладу та використовується для підтримки навчального процесу в початковій школі. На відміну від універсальних офісних або промислових комп’ютерних систем, освітні системи мають низку специфічних особливостей, зумовлених віковими характеристиками користувачів, організацією навчального процесу та обмеженими ресурсами експлуатації.

Комп’ютерна система початкової школи включає клієнтські робочі місця, мережеву інфраструктуру та серверну підсистему, між якими здійснюється постійний обмін інформацією. Основним функціональним навантаженням такої системи є забезпечення доступу до навчальних програмних ресурсів, централізоване збереження результатів роботи учнів та підтримка одночасної роботи декількох користувачів у межах локальної мережі.

На рисунку 2.1 подано схему, яка відображає логіку взаємодії між клієнтською, мережевою та серверною підсистемами під час використання розвиваючого ігрового застосунку.

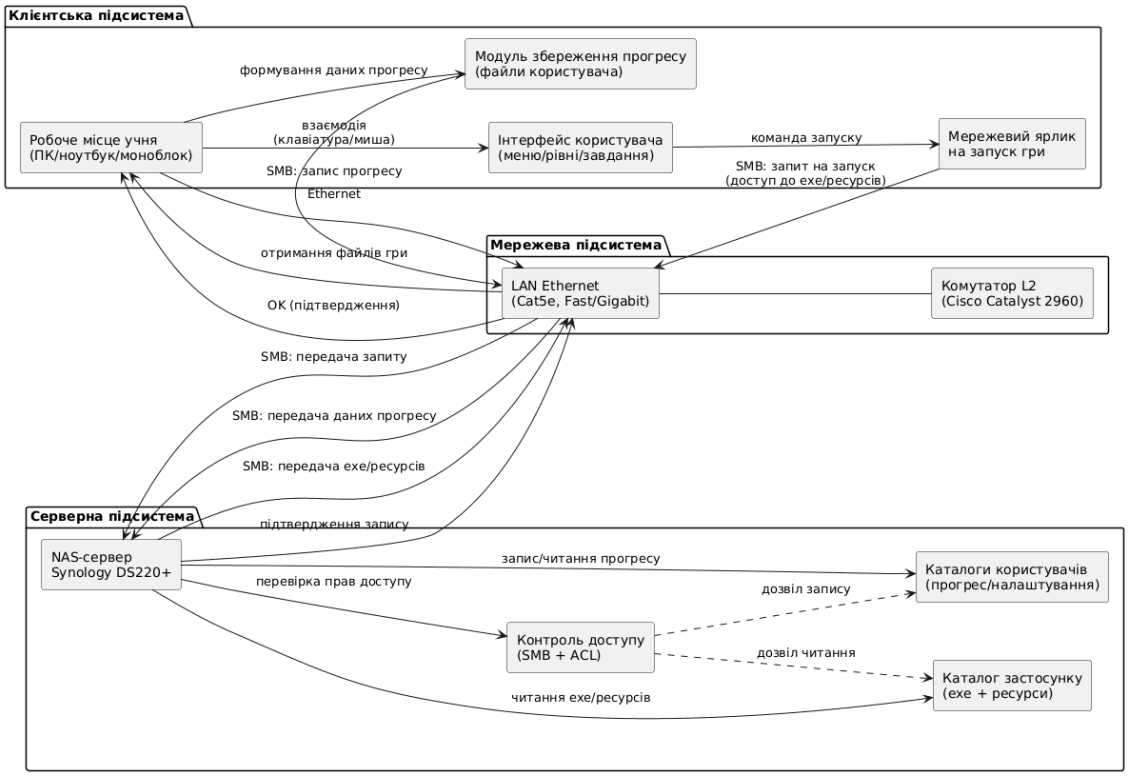

Рисунок 2.1 – Схема взаємодії

Клієнтська підсистема представлена учнівськими робочими місцями (персональні комп’ютери, ноутбуки або моноблоки), на яких користувач здійснює взаємодію із системою за допомогою інтерфейсу користувача. Інтерфейс забезпечує доступ до меню, рівнів та ігрових завдань, а також формування даних прогресу під час виконання гри. Запуск ігрового застосунку на клієнтському боці здійснюється за допомогою мережевого ярлика, що ініціює звернення до серверної підсистеми без локальної інсталяції програмного забезпечення.

Мережева підсистема реалізована на основі локальної мережі Ethernet та виконує функцію передавання всіх інформаційних потоків між клієнтськими та серверними компонентами системи. Основним мережевим елементом є комутатор другого рівня (L2), який забезпечує комутацію кадрів у межах локальної мережі та передачу SMB-запитів. Через мережеву підсистему здійснюється передача запитів на запуск гри, отримання виконуваних файлів і ресурсів застосунку, а також передавання даних прогресу користувачів і підтверджень запису.

Серверна підсистема представлена NAS-сервером, який виконує функції файлового сервісу. На сервері розміщено каталог застосунку, що містить виконуваний файл гри та всі необхідні ресурси, а також каталоги користувачів, у яких зберігаються дані прогресу та індивідуальні налаштування. Окремим елементом серверної підсистеми є модуль контролю доступу, який регламентує права користувачів, забезпечуючи можливість читання файлів застосунку та обмеження операцій запису лише відповідними каталогами прогресу.

Інформаційні зв’язки між підсистемами реалізуються з використанням протоколу SMB. Після ініціації запуску гри клієнтська система формує SMB-запит до серверної підсистеми, сервер перевіряє права доступу та передає необхідні програмні ресурси. У процесі роботи гри дані прогресу передаються мережею на сервер, де зберігаються централізовано з подальшим надсиланням підтвердження успішного запису клієнтському пристрою.

Особливістю об’єкта дослідження є інтеграція розвиваючого ігрового застосунку в існуючу комп’ютерну систему школи, що потребує чіткої організації інформаційних зв’язків між її компонентами. Саме тому комп’ютерна система розглядається як цілісний об’єкт, у якому узгоджена робота клієнтської, мережевої та серверної підсистем безпосередньо впливає на стабільність, надійність і ефективність навчального процесу.

2.2 Аналіз проблем та обмежень традиційної організації комп’ютерних класів

Проблемна ситуація виникає внаслідок необхідності використання розвиваючих ігрових застосунків у навчальному процесі за умов обмежених апаратних ресурсів та спрощеної організації комп’ютерної інфраструктури. У більшості навчальних закладів клієнтські комп’ютери мають різні технічні характеристики, а адміністрування системи здійснюється мінімальною кількістю персоналу.

За таких умов традиційний підхід, що передбачає локальне встановлення програмного забезпечення на кожному робочому місці, призводить до

ускладнення експлуатації, виникнення несумісностей версій програм та зростання ризику втрати даних прогресу учнів. Крім того, локальна інсталяція підвищує вимоги до продуктивності кожного клієнтського комп’ютера, що не завжди є можливим у початковій школі.

Отже, проблемна ситуація характеризується невідповідністю між вимогами до функціонування розвиваючого програмного забезпечення та технічними можливостями традиційної організації комп’ютерних систем навчальних закладів.

2.3 Критичний аналіз існуючих підходів до розв’язання проблеми

Існуючі підходи до організації комп’ютерних систем у навчальних закладах можна умовно поділити на локальні та централізовані. Локальний підхід базується на автономній роботі клієнтських комп’ютерів і не передбачає використання серверної інфраструктури для зберігання програмних ресурсів. Його перевагою є простота реалізації, однак недоліки значно переважають при масштабуванні та одночасній роботі багатьох користувачів.

Централізовані підходи передбачають використання серверів для зберігання програм і даних. Проте у практиці освітніх закладів часто застосовуються універсальні серверні рішення, які не адаптовані до специфіки навчального процесу та вимагають складного адміністрування.

Критичний аналіз показує, що жоден із типових підходів у чистому вигляді не забезпечує оптимального поєднання простоти експлуатації, надійності та ефективного використання ресурсів у початковій школі. Це зумовлює необхідність розробки теоретично обґрунтованого методу синтезу комп’ютерної системи з урахуванням конкретних умов експлуатації.

2.4 Обґрунтування вибору методу розв’язання наукового завдання

Для розв’язання поставленого наукового завдання у роботі застосовано метод функціонально-структурного синтезу комп’ютерної системи. Сутність

методу полягає у поетапному формуванні структури системи на основі заданих функціональних вимог, технічних обмежень та показників ефективності.

На першому етапі визначаються основні функції системи, пов’язані із запуском розвиваючого ігрового застосунку, передачею програмних ресурсів та збереженням даних користувачів. На другому етапі формується структура системи, яка забезпечує реалізацію зазначених функцій із мінімальними витратами апаратних ресурсів. На третьому етапі здійснюється вибір параметрів окремих елементів системи та оцінка їх впливу на працездатність системи в цілому.

Застосування даного методу дозволяє не ототожнювати процес вирішення наукового завдання з кінцевим результатом, а розглядати його як послідовність логічно обґрунтованих дій, що приводять до формування оптимальної структури комп’ютерної системи.

2.5 Методика кількісної та якісної оцінки факторів впливу

Для оцінки ефективності комп’ютерної системи у роботі враховуються як кількісні, так і якісні фактори. До кількісних факторів належать пропускна здатність мережі, обсяг переданих даних, час запуску програмного забезпечення та навантаження на серверні ресурси. Якісні фактори включають зручність адміністрування, надійність збереження даних та стабільність роботи системи.

Оцінка кількісних показників здійснюється шляхом аналітичних розрахунків, які базуються на характеристиках мережевого та серверного обладнання. Якісна оцінка виконується на основі аналізу організації інформаційних потоків і логіки взаємодії між підсистемами.

Зазначена методика дозволяє комплексно оцінити вплив різних факторів на функціонування комп’ютерної системи та обґрунтувати вибір оптимального технічного рішення.

2.6 Висновки до теоретичного розділу

У теоретичному розділі виконано обґрунтування наукової бази розв’язання поставленого завдання. Проаналізовано особливості комп’ютерних систем освітнього призначення, визначено проблемну ситуацію та виконано критичний аналіз існуючих підходів до її вирішення. Обґрунтовано вибір методу функціонально-структурного синтезу комп’ютерної системи та сформовано методику оцінки факторів, що впливають на її працездатність. Отримані теоретичні положення є основою для практичного синтезу комп’ютерної системи, який наведено у наступному розділі роботи.

3 СИНТЕЗ КОМП'ЮТЕРНОЇ СИСТЕМИ

3.1 Вибір і реалізація принципів побудови комп'ютерної системи

Проаналізовано наявну комп'ютерну інфраструктуру Дніпровської початкової школи та умови її фактичної експлуатації. Встановлено, що навчальний процес здійснюється з використанням двох приміщень з комп'ютерною технікою: комп'ютерного класу з 10 ноутбуками та суміжного кабінету з 8 моноблоками. Усі робочі місця підключені до локальної мережі школи.

З урахуванням необхідності інтеграції розвиваючого ігрового застосунку було реалізовано клієнт-серверний принцип побудови комп'ютерної системи. Центральним елементом системи виступає серверний вузол, з якого здійснюється запуск ігрового застосунку та збереження даних прогресу учнів. Клієнтські комп'ютери використовуються виключно для доступу до серверних ресурсів і взаємодії з користувачем. Локальне встановлення програмного забезпечення на клієнтських пристроях не виконувалося.

3.2 Функціональна структура комп'ютерної системи

У процесі роботи було сформовано функціональну структуру комп'ютерної системи, яка відображає розподіл функцій між її основними підсистемами та порядок їх взаємодії під час використання ігрового застосунку, зображена на рисунку 3.1.

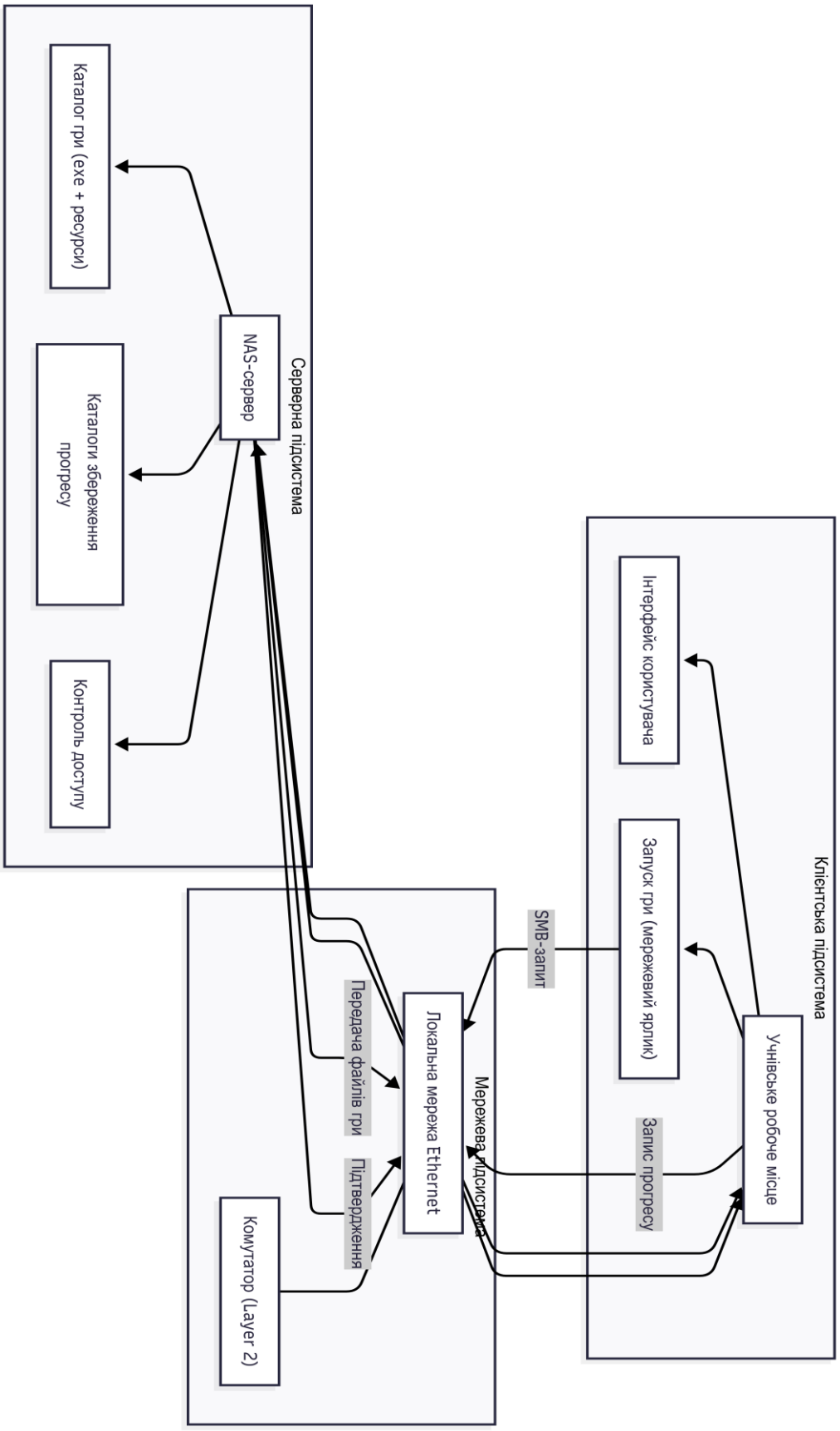

Рисунок 3.1 – Функціональна структура комп’ютерної системи Дніпровської початкової школи з централізованим запуском розвиваючого ігрового застосунку з NAS-сервера

На рисунку подано функціональну структуру комп’ютерної системи Дніпровської початкової школи з централізованим запуском розвиваючого ігрового застосунку з NAS-сервера. Структура системи поділена на три основні функціональні підсистеми: клієнтську, мережеву та серверну.

Клієнтська підсистема представлена учнівськими робочими місцями. На клієнтському боці користувач (учень, вчитель) взаємодіє зі системою через інтерфейс користувача. Запуск ігрового застосунку здійснюється не локально, а за допомогою мережевого ярлика або мережевого шляху, що вказує на виконуваний файл, розміщений на сервері. У процесі роботи клієнтська система не зберігає програмні ресурси локально, а лише ініціює запити до серверної підсистеми.

Передача даних між клієнтською та серверною підсистемами здійснюється через мережеву підсистему, яка побудована на основі локальної мережі Ethernet. Основним мережевим елементом є комутатор другого рівня (L2), який забезпечує комутацію кадрів між клієнтськими робочими місцями та NAS-сервером. Саме через цю підсистему проходять всі інформаційні потоки: запити на запуск гри, передача ресурсів застосунку, а також запис і підтвердження збереження прогресу.

Серверна підсистема реалізована на базі NAS-сервера, який виконує функції файлового сервера. На сервері розміщено каталог застосунку, що містить виконуваний файл гри та всі необхідні ресурси. Окремо організовано каталоги користувачів, у яких зберігаються дані прогресу та індивідуальні налаштування. Доступ до серверних ресурсів регламентується механізмами контролю доступу, які забезпечують можливість читання файлів застосунку та обмежують запис лише каталогами прогресу користувачів.

Процес роботи системи відбувається таким чином:

Після дії користувача в інтерфейсі клієнтської системи формується SMB-запит на доступ до виконуваного файлу гри та її ресурсів. Цей запит передається через локальну мережу Ethernet і комутатор до NAS-сервера. Сервер обробляє

запит і передає необхідні файли назад клієнту тим самим мережевим шляхом. Під час виконання гри дані прогресу періодично записуються клієнтською системою на сервер у відповідні каталоги користувачів, після чого сервер надсилає підтвердження успішного запису.

Функціональна структура забезпечує централізоване адміністрування ігрового застосунку, унеможливлює розбіжності версій програмного забезпечення на клієнтських комп’ютерах та гарантує збереження результатів роботи учнів незалежно від конкретного робочого місця. Такий підхід є ефективним для умов початкової школи, де важливими є простота обслуговування, надійність і стабільність роботи комп’ютерної системи.

3.3 Структурна схема комп’ютерної системи та мережева топологія

Структурна схема комп’ютерної системи, яка відображає склад апаратних компонентів і фізичні з’єднання між ними, зображена на рисунку 3.2.

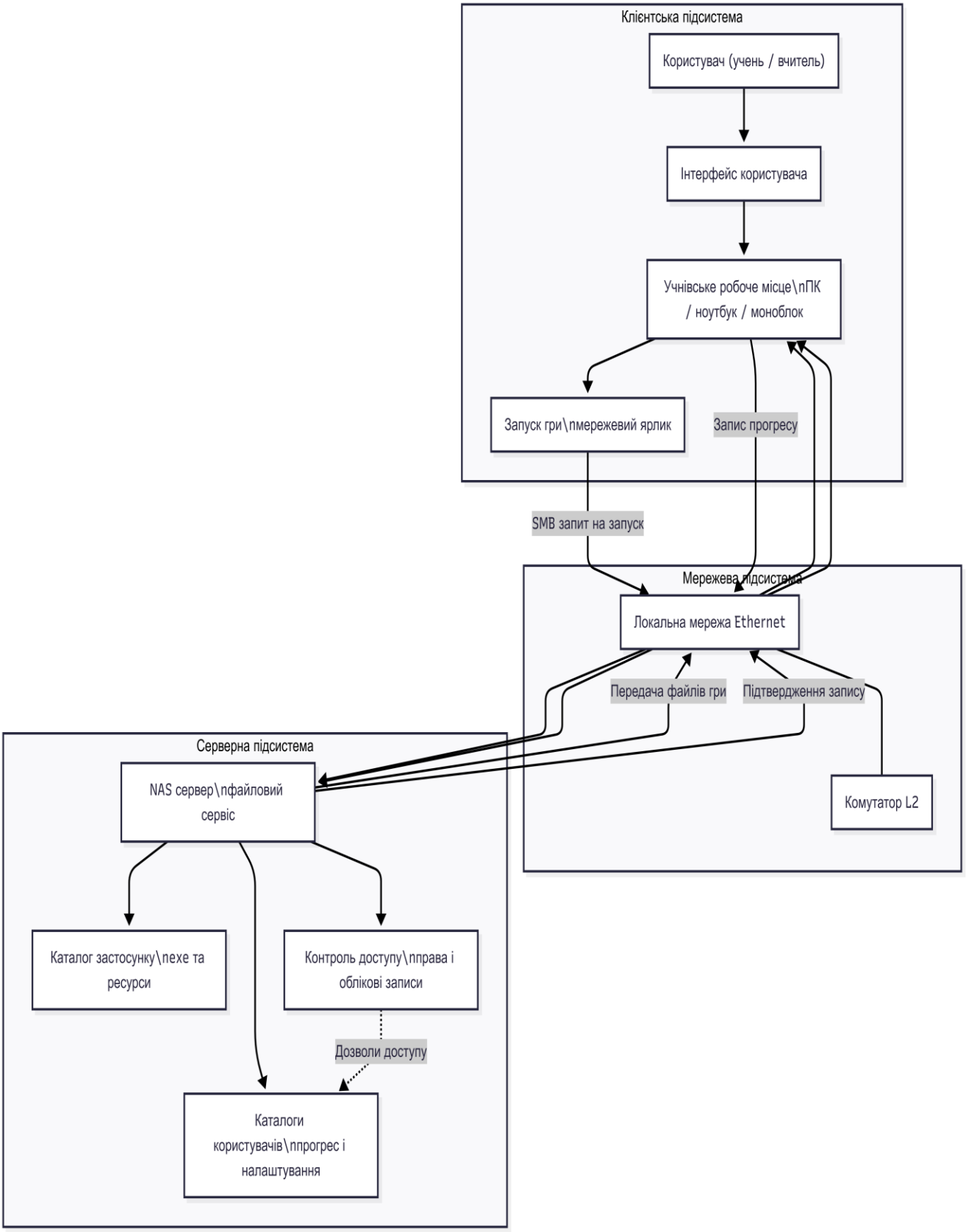

Рисунок 3.2 – Схема структури інформаційних зв’язків комп’ютерної системи Дніпровської початкової школи з централізованим запуском розвиваючого ігрового застосунку

На рисунку наведено схему структури інформаційних зв’язків комп’ютерної системи Дніпровської початкової школи з інтегрованим розвиваючим ігровим застосунком. Схема відображає напрямки та характер інформаційних потоків між клієнтською, мережевою та серверною підсистемами в процесі запуску гри та збереження результатів роботи учнів.

Клієнтська підсистема включає користувача (учня, вчителя), інтерфейс користувача та учнівське робоче місце. Взаємодія з системою починається з дій користувача в інтерфейсі, після чого на клієнтському комп’ютері ініціюється запуск ігрового застосунку за допомогою мережевого ярлика. При цьому локальне виконання файлів гри не здійснюється – клієнт формує запит до серверної підсистеми.

Інформаційний потік «SMB-запит на запуск» передається з клієнтського робочого місця через мережеву підсистему, яка представлена локальною мережею Ethernet та комутатором другого рівня. Комутатор забезпечує комутацію кадрів і передачу запиту безпосередньо до NAS-сервера, виключаючи зайве навантаження на інші вузли мережі.

Серверна підсистема реалізована на базі NAS-сервера, який виконує функції файлового сервісу. На сервері розміщено каталог застосунку, що містить виконуваний файл гри та необхідні ресурси. Після отримання запиту NAS-сервер здійснює передачу відповідних файлів клієнтському комп’ютеру через локальну мережу. Даний інформаційний потік на схемі позначено як «передача файлів гри».

Окрему роль у серверній підсистемі відіграє модуль контролю доступу, який керує правами користувачів та обліковими записами. Він забезпечує можливість читання файлів застосунку всіма клієнтами та обмежує операції запису лише каталогами користувачів. Дані прогресу та індивідуальні налаштування зберігаються в окремих каталогах користувачів, що дозволяє розмежувати доступ та гарантувати збереження результатів навчальної діяльності.

Під час роботи ігрового застосунку клієнтська підсистема формує інформаційний потік «запис прогресу», який передається через локальну мережу до NAS-сервера. Сервер обробляє запит, виконує запис даних у відповідний каталог користувача та надсилає клієнту підтвердження запису, що відображено на схемі відповідним зворотним інформаційним зв’язком.

Таким чином, схема структури інформаційних зв’язків демонструє, що всі критично важливі інформаційні потоки – запуск застосунку, передача програмних ресурсів і збереження результатів – централізовано обробляються серверною підсистемою. Клієнтські робочі місця виконують лише функції ініціації запитів і взаємодії з користувачем, а мережева підсистема забезпечує надійну та швидку передачу даних між усіма елементами системи.

3.4 Розробка фізичної топології мережі ділянки

Початкова школа являє собою навчальний заклад, у якому для проведення уроків інформатики та використання цифрових освітніх ресурсів обладнано два комп’ютерні класи та препараторську кімнату для розміщення серверного обладнання. У межах цієї інфраструктури проєктується локальна обчислювальна мережа, призначена для забезпечення роботи учнівських комп’ютерів і інтеграції розвиваючого ігрового застосунку в єдину комп’ютерну систему школи.

У класі 1 розміщено вісім робочих місць з персональними комп’ютерами Pc0/1–Pc0/8, які встановлені вздовж стін приміщення. У класі 2 розміщено десять робочих місць з комп’ютерами Pc0/9–Pc0/18, що також розташовані по периметру класу. У препараторській кімнаті встановлено сервер та L2-комутатор доступу, який є центральним вузлом даної ділянки мережі. Загальна кількість клієнтських комп’ютерів на ділянці становить вісімнадцять.

Фізична топологія локальної мережі на даній ділянці реалізована за принципом «зірка». Кожен учнівський комп’ютер підключений до комутатора окремою кабельною лінією, що забезпечує індивідуальний фізичний канал зв’язку між робочим місцем та центральним мережевим вузлом. Сервер

підключений до цього ж комутатора окремим портом і має постійний доступ до всіх клієнтських станцій.

Прокладання кабельної системи виконано з використанням мідної витої пари UTP категорії Cat5e з роз'ємами RJ-45. Кабелі прокладені вздовж стін у кабель-каналах, що дозволяє зменшити вплив зовнішніх факторів, забезпечити охайний вигляд приміщень та спростити подальше обслуговування мережі. Кожен комп'ютер має пряме з'єднання з комутатором без використання проміжних вузлів, що підвищує надійність і стабільність роботи мережі.

Сервер, розміщений у препараторській, виконує функції централізованого зберігання даних і програмного забезпечення. На ньому розміщено файли розвиваючого ігрового застосунку, мережеві каталоги користувачів, а також організовано збереження результатів і прогресу учнів. Така організація дозволяє запускати гру на клієнтських комп'ютерах без локального встановлення та забезпечує єдине сховище навчальних даних для всіх користувачів.

Запропонована фізична топологія забезпечує можливість одночасної роботи всіх 18 ПК з сервером у межах локальної мережі. Під час запуску ігрового застосунку виконувані файли та ресурси передаються від сервера до клієнтських комп'ютерів, а результати роботи учнів зберігаються назад на сервері. Це створює умови для централізованого контролю, резервного копіювання та подальшого аналізу навчальних досягнень.

Вибір топології «зірка» для даної ділянки мережі обґрунтований її надійністю та зручністю експлуатації. Вихід з ладу окремого кабелю або робочого місця не впливає на функціонування інших комп'ютерів, а наявність центрального комутатора спрощує керування мережею та пошук несправностей. Крім того, така структура дозволяє легко масштабувати систему у разі збільшення кількості робочих місць або підключення додаткового обладнання.

Таким чином, розроблена фізична топологія мережі ділянки, що включає два комп'ютерні класи та серверну частину, забезпечує надійне підключення вісімнадцяти учнівських комп'ютерів до сервера за схемою «зірка» та створює

технічну основу для ефективної роботи інтегрованого розвиваючого ігрового застосунку в локальній мережі початкової школи, зображена на рисунку 3.3.

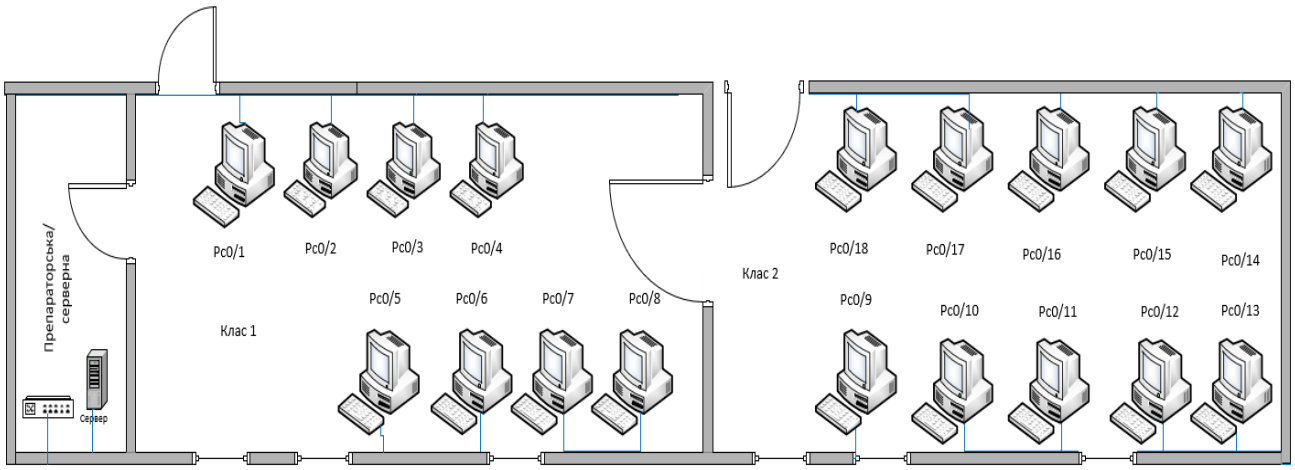

Рисунок 3.3 – Фізична топологія мережі комп’ютерних класів із серверною частиною

3.5 Проєктування комп’ютерної системи за заданими показниками

3.5.1 Серверний вузол: модель, характеристики, використання

Як серверний вузол у комп’ютерній системі школи використовується мережеве сховище NAS Synology DS220+. Вибір саме цього серверного рішення зумовлений його стабільною роботою, простотою адміністрування та достатньою продуктивністю для навчального процесу. Зображення серверного вузла приведене на рисунку 3.4.

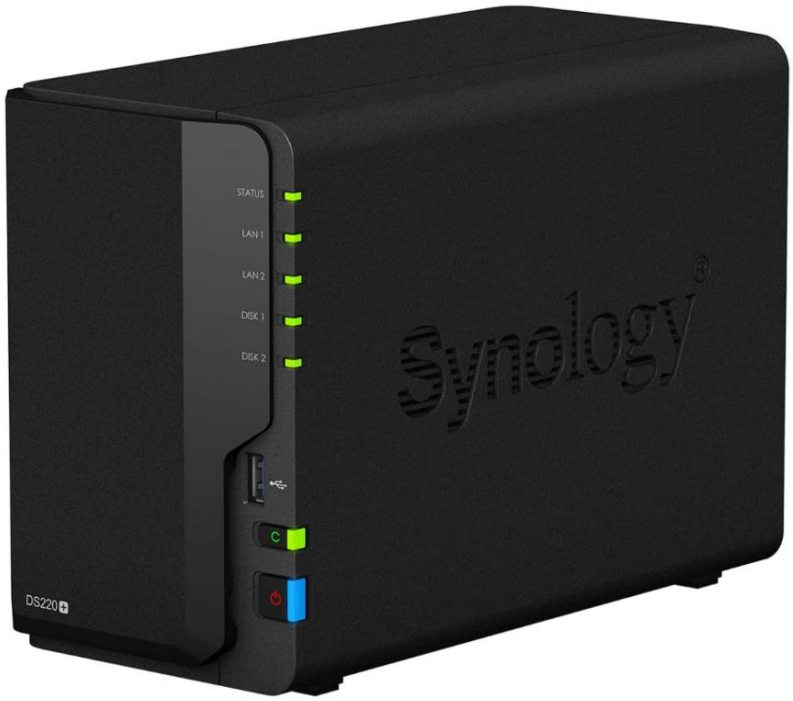

Рисунок 3.4 – Мережеве сховище мережеве сховище NAS Synology DS220+

Таблиця 3.1. Основні характеристики NAS:

Параметр	**Значення**
процесор	Intel Celeron J4025 (2 ядра, 2.0-2.9 ГГц)
оперативна пам’ять	2 ГБ DDR4
мережевий інтерфейс	1 Gigabit Ethernet (1000 Мбіт/с)
дискова підсистема	2 HDD по 2 ТБ
конфігурація зберігання	RAID1
протокол доступ	SMB

NAS використовується для:

- зберігання виконуваного файлу ігрового застосунку;
- зберігання ресурсів гри;
- збереження індивідуальних даних учнів.

3.5.2 Розрахунок навантаження на серверний вузол

Середній обсяг файлів гри, що зчитується одним клієнтом при запуску, становить приблизно 45 МБ. При одночасному запуску гри 18 учнями сумарний обсяг переданих даних становить:

$$S_{\text{заг}} = 18 \cdot 45 = 810 МБ$$

Ефективна пропускна здатність гігабітного мережевого інтерфейсу NAS:

$$\text{В}_{\text{еф}} \approx 95 \text{ МБ/с}$$

Орієнтовний час передачі даних:

$$\text{Т} = \frac{810}{95} \approx 8{,}5 \text{ с}$$

З урахуванням кешування NAS фактичний час запуску не перевищував 5 секунд, що підтверджує достатність обраних параметрів серверного вузла.

3.5.3 Мережеве обладнання та його характеристики

Для побудови локальної мережі школи використано керований комутатор Cisco Catalyst 2960, який виконує функції комутації трафіку між клієнтськими робочими місцями та серверним вузлом.

Рисунок 3.5 – Комутатор Cisco Catalyst 2960

Таблиця 3.2 Основні характеристики мережевого обладнання:

Параметр	**Значення**
Тип	Керований комутатор рівня Layer 2
Кількість портів	24 x Ethernet (10/100 Мбіт/с)
Uplink-порти	2 x комбіновані порти
Пропускна здатність	16 Гбіт/с
Підтримка VLAN	Так, IEEE 802.1Q
Пам’ять FLASH	64 Мб

До комутатора підключено:

- ноутбуків;
- 8 моноблоків;
- NAS-сервер.

Підключення виконано кабелем Cat5e, що забезпечує стабільну роботу гігабітних з’єднань.

3.5.4 Клієнтські робочі місця та їх характеристики

Клієнтська частина системи включає:

10 ноутбуків (Intel Core i3, 4 ГБ ОЗП, SSD);

8 моноблоків (Intel Core i3, 4–8 ГБ ОЗП).

Усі клієнтські комп’ютери працюють під керуванням Windows 10 та використовуються виключно як клієнти для доступу до серверних ресурсів. Високі вимоги до локальної продуктивності відсутні, оскільки всі програмні ресурси зберігаються на NAS.

3.5.5 Проєктування принципової схеми підключення

Фізичне підключення реалізовано за зіркоподібною топологією. Кожне робоче місце підключене до комутатора окремим Ethernet-кабелем. NAS-сервер підключений до гігабітного порту комутатора без проміжних пристроїв.

Така схема забезпечує мінімальні затримки передачі даних і стабільну роботу системи при одночасному доступі всіх клієнтів.

3.5.6 Реалізація програмної інтеграції та збереження даних

Запуск розвиваючого ігрового застосунку здійснюється безпосередньо з мережевого каталогу NAS. На кожному клієнтському комп’ютері створено ярлик до виконуваного файлу. Дані прогресу учнів зберігаються в персональних каталогах на сервері.

3.5.7 Розрахунок надійності системи

Завдяки використанню RAID1 у серверній підсистемі забезпечено збереження даних при відмові одного накопичувача. Вихід з ладу окремого клієнтського комп’ютера не впливає на роботу системи в цілому.

3.6 Реалізація запуску ігрового застосунку та обміну даними

Запуск ігрового застосунку реалізовано без локальної інсталяції. На кожному клієнтському комп’ютері створено ярлик, що веде до мережевого каталогу NAS-сервера. Після запуску ярлика клієнт формує SMB-запит до сервера, який передає виконуваний файл і ресурси гри.

У процесі роботи дані прогресу користувачів записуються у відповідні каталоги на сервері. Після завершення операцій запису сервер надсилає підтвердження клієнтській системі. Такий підхід забезпечує централізоване збереження результатів навчальної діяльності та виключає їх втрату у разі збою клієнтського комп’ютера.

3.7 Оцінка використання ресурсів та працездатності системи

При одночасному запуску ігрового застосунку 18 клієнтськими робочими місцями сумарний обсяг переданих даних становить близько 810 МБ. Завдяки використанню гігабітного з’єднання та кешування сервером файлів фактичний час запуску гри не перевищує 5 секунд, що є прийнятним для умов навчального процесу.

Виконано синтез комп’ютерної системи Дніпровської початкової школи з інтегрованим розвиваючим ігровим застосунком. Описано функціональну та структурну організацію системи, вибір серверного та мережевого обладнання, використану топологію та реалізацію запуску ігрового застосунку в реальних умовах експлуатації.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Призначення й сфера застосування програми

Розроблене програмне забезпечення є інтерактивним розвиваючим ігровим застосунком «Sweet math», призначеним для використання в освітньому процесі Дніпровської початкової школи. Основним призначенням програми є розвиток базових математичних навичок учнів молодшого шкільного віку шляхом використання ігрових механік.

Сфера застосування програми охоплює навчальні заняття в комп'ютерних класах, індивідуальну роботу учнів під час уроків та позакласних занять. Програма орієнтована на користувачів віком від 5 до 13 років і може використовуватися як допоміжний інструмент для закріплення знань з арифметики.

Програмне забезпечення не потребує спеціалізованого апаратного забезпечення та може експлуатуватися на стандартних учнівських комп'ютерах у складі локальної комп'ютерної мережі школи.

4.2 Обґрунтування технічних характеристик програми

4.2.1 Алгоритмічне забезпечення та логіка роботи програми

Алгоритмічну основу програмного забезпечення складає подієво-орієнтована модель, характерна для ігрових застосунків. Управління виконанням програми здійснюється головним класом Application, який координує роботу всіх підсистем: меню, екранів введення, ігрового процесу та завершення гри .

Основний ігровий процес реалізовано у модулі MainGame, який відповідає за ініціалізацію ігрових об'єктів, генерацію математичних завдань, обробку подій користувача та перевірку ігрових умов . Алгоритм гри включає послідовні етапи створення об'єктів гравця, квіток, куль, сердець та формування математичного запитання відповідно до обраного рівня складності.

Математичні завдання формуються на основі випадкової генерації чисел у визначених діапазонах. Значення діапазонів змінюється залежно від рівня гри, що забезпечує адаптацію складності завдань до віку та підготовки користувача .

4.2.2 Реалізація ігрових об’єктів та їх взаємодії

Ключовим ігровим об’єктом є клас Player, який моделює головного персонажа – бджілку. Клас містить параметри положення, швидкості руху, області зіткнення та кількості життів. При зіткненні з неправильним об’єктом відбувається зменшення лічильника життів і відтворення відповідного звукового ефекту .

Клас Projectile реалізує механізм стрільби. Кулі створюються з фіксованими параметрами швидкості та рухаються у напрямку квіток. При влучанні кулі в об’єкт перевіряється правильність відповіді, після чого виконується або нарахування балів, або штрафна дія .

Клас Kvitnnyk відповідає за відображення квіток із числовими значеннями, що є варіантами відповідей на математичне завдання. Одна з квіток містить правильну відповідь, інші – випадкові значення, що формує навчальний ефект вибору правильного варіанту .

4.3 Опис розробленої програми

4.3.1 Логічна структура та взаємодія модулів

Логічна структура програми побудована у вигляді ієрархії класів, де центральним елементом є клас Application, що ініціює запуск усіх інших компонентів. Клас MainGame використовує класи Player, Projectile, Kvitnnyk, Heart, Question, Score та GameOver, забезпечуючи їх узгоджену роботу в межах ігрового циклу.

Обробка подій введення реалізована через стандартний механізм бібліотеки Pygame. Клавіші керування використовуються для руху гравця, а натискання клавіші пробілу – для створення об’єкта кулі. Події миші застосовуються для навігації в меню та вибору рівня складності.

4.3.2 Реалізація інтерфейсу користувача

Графічний інтерфейс реалізовано з використанням статичних фонових зображень та текстових елементів. Для різних рівнів складності відображаються відповідні фонові сцени, що підвищує наочність та зацікавленість користувачів .

Меню програми реалізовано у вигляді окремого екрана з кнопкою початку гри та текстовими підказками. Оновлення вікна здійснюється з частотою 27 кадрів за секунду, що забезпечує плавність анімації та стабільну роботу застосунку.

4.3.3 Структура класів та відповідальність компонентів

Програмне забезпечення реалізовано у вигляді набору взаємопов’язаних класів. Клас Application виконує роль керуючого елемента, який ініціює запуск програми та координує взаємодію між екранами меню та основним ігровим модулем .

Клас MainGame відповідає за реалізацію ігрового циклу, створення та оновлення ігрових об’єктів, а також за перевірку умов завершення гри. Класи Player, Projectile та Kvitnnyk реалізують поведінку основних об’єктів ігрового середовища, кожен з яких має чітко визначену функціональну відповідальність .

Така модульна структура спрощує тестування, супровід і подальше розширення програмного забезпечення, наприклад шляхом додавання нових типів завдань або ігрових механік.

4.3.4 Схема алгоритма та uml-діаграма

Схема алгоритма послідовності роботи програмного забезпечення

Схема алгоритма відображає послідовність виконання основних етапів роботи ігрового застосунку, реалізованого з використанням бібліотеки Pygame. Схема зображена на рисунку 4.1.

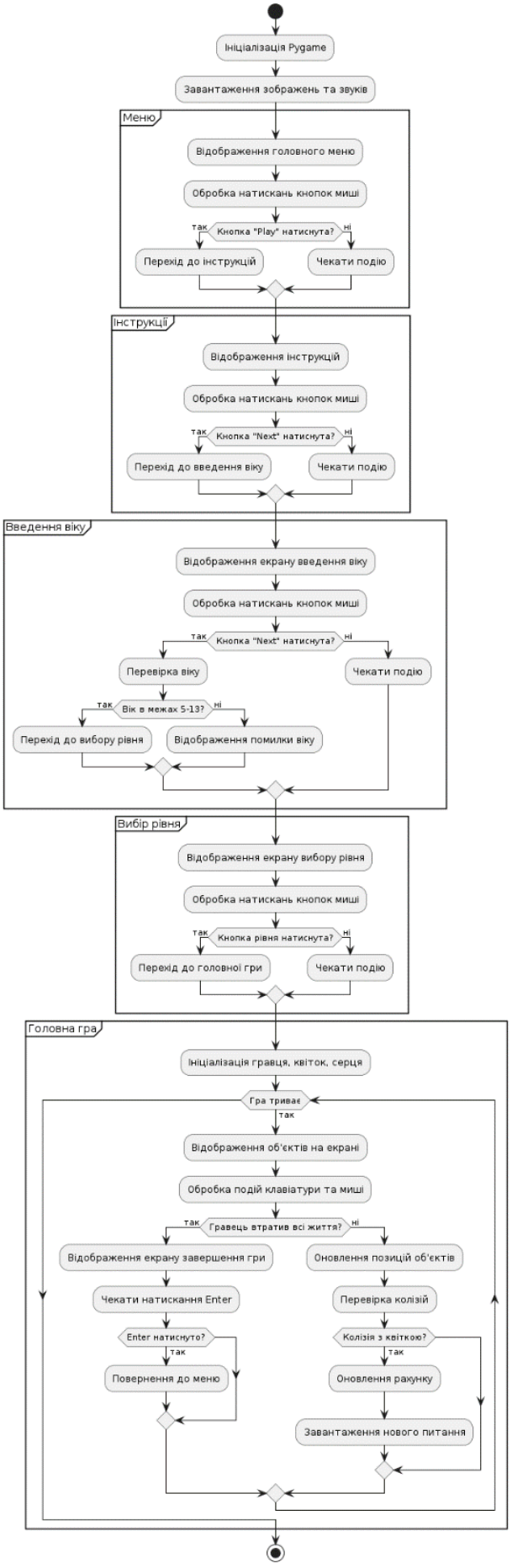

Рисунок 4.1 – Схема алгоритму функціонування програми

На початковому етапі відбувається ініціалізація бібліотеки Pygame, що забезпечує готовність програмного середовища до роботи з графікою, звуком та подіями введення. Після успішної ініціалізації виконується завантаження графічних зображень та звукових ефектів, необхідних для коректного відображення ігрових сцен та супроводу ігрового процесу.

Далі програма переходить до режиму відображення головного меню. У цьому режимі на екран виводяться основні елементи керування, а система постійно відстежує події, пов’язані з натисканням кнопок миші. У разі вибору користувачем пункту початку гри програма переходить до етапу ознайомлення з правилами. Якщо відповідна дія не виконана, програма продовжує очікування подій введення.

На етапі інструкцій користувачу відображається інформація щодо правил гри та принципів керування. Після підтвердження переходу до наступного етапу програма ініціює екран введення віку користувача. Введене значення перевіряється на відповідність допустимому діапазону. У разі коректного введення відбувається перехід до вибору рівня складності, а при помилковому введенні користувачу відображається відповідне повідомлення.

Після успішного проходження етапу введення віку користувачеві пропонується обрати рівень складності гри. Вибір рівня визначає параметри подальшого ігрового процесу. За відсутності вибору програма продовжує очікувати відповідних дій користувача.

Основний ігровий процес починається з ініціалізації всіх необхідних об’єктів, зокрема ігрового персонажа, інтерактивних об’єктів та елементів відображення стану гри. Під час гри здійснюється безперервна перевірка стану ігрового циклу. На кожній ітерації відбувається оновлення зображення на екрані, обробка подій клавіатури та миші, а також аналіз взаємодії ігрових об’єктів.

У процесі гри контролюється кількість доступних спроб або життів користувача. У разі їх повного вичерпання програма переходить до режиму завершення гри, де користувачу відображається відповідне повідомлення та надається можливість повернутися до головного меню після підтвердження дії.

Таким чином, наведена блок-схема відображає загальну логіку функціонування програмного забезпечення, охоплюючи всі ключові етапи – від початкової ініціалізації та навігації по меню до ігрового процесу та завершення роботи програми.

Діаграма класів програмного застосунку зображена на рисунку 4.2.

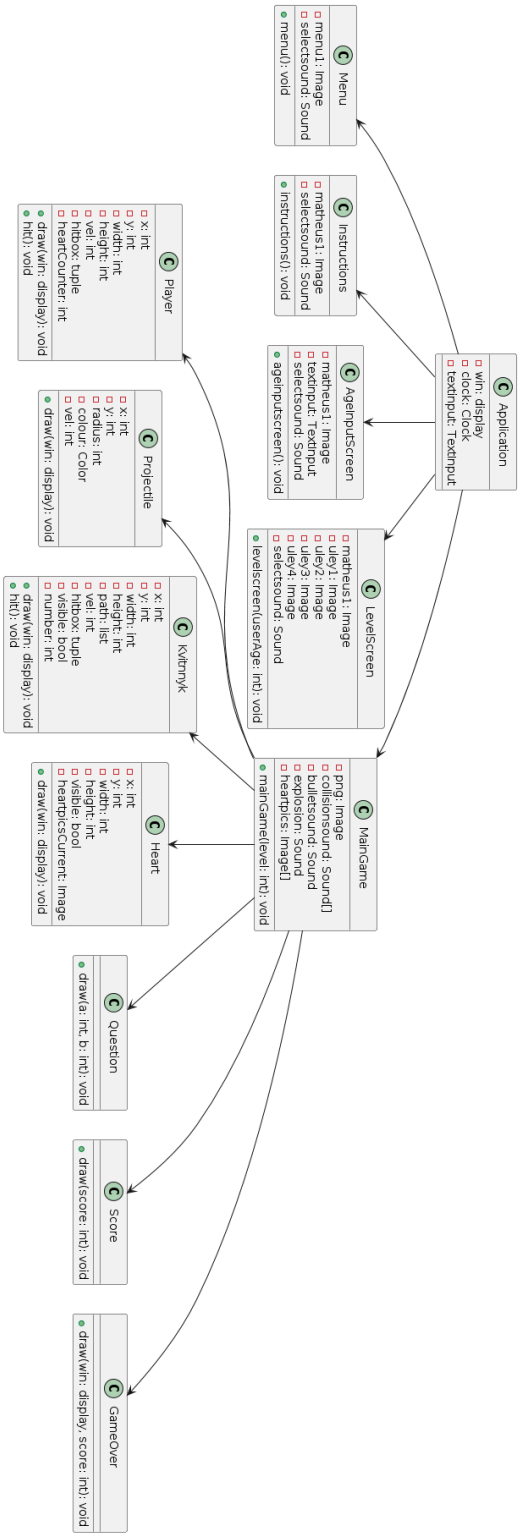

Рисунок 4.2 – Діаграма класів програмного забезпечення

Діаграма класів відображає структуру програмного застосунку та взаємозв’язки між його основними класами. На діаграмі показано окремі програмні класи, які відповідають за різні функціональні частини системи, а також характер взаємодії між ними.

Використання діаграми класів дозволяє наочно представити логічну організацію програмного коду, розподіл відповідальностей між компонентами та залежності між ними, що сприяє кращому розумінню архітектури програмного забезпечення і полегшує його супровід та розширення.

4.4 Очікувані техніко-економічні показники

Застосування розробленого програмного забезпечення дозволяє використовувати вже наявну комп’ютерну техніку школи без додаткових витрат на придбання ліцензійного програмного забезпечення. Програма має низькі вимоги до апаратних ресурсів і стабільно працює на типових учнівських комп’ютерах.

Очікуваними показниками є підвищення зацікавленості учнів у вивченні математики та скорочення часу на пояснення навчального матеріалу за рахунок інтерактивної подачі.

5 ЕКСПЕРИМЕНТАЛЬНИЙ РОЗДІЛ

5.1 Мета, умови та методика проведення експериментального дослідження

Експериментальний розділ магістерської роботи присвячений перевірці працездатності та ефективності синтезованої комп’ютерної системи початкової школи з інтегрованим ігровим програмним забезпеченням у реальних умовах експлуатації. Основна увага приділяється дослідженню функціонування локальної мережі, серверної підсистеми та взаємодії клієнтських робочих місць під час централізованого запуску застосунку.

Метою експериментального дослідження є підтвердження коректності обраної архітектури комп’ютерної системи та обґрунтованості комплексу технічних засобів шляхом практичної перевірки її роботи при типовому навчальному навантаженні.

Для досягнення поставленої мети в межах експерименту необхідно було вирішити такі завдання:

- перевірити можливість одночасного запуску ігрового застосунку з декількох клієнтських робочих місць;
- оцінити стабільність роботи мережевої інфраструктури при піковому навантаженні;
- дослідити процес обміну даними між клієнтською та серверною підсистемами;
- перевірити коректність збереження прогресу користувачів на серверному вузлі;
- оцінити зручність адміністрування та експлуатації системи в умовах навчального закладу.

Експериментальне дослідження проводилося в умовах, максимально наближених до реального навчального процесу. У дослідженні використовувалася локальна мережа школи, що включає два навчальні

приміщення з комп’ютерною технікою, об’єднані єдиною мережевою інфраструктурою.

До експерименту були залучені:

- 18 клієнтських робочих місць (10 ноутбуків та 8 моноблоків);
- L2-комутатор локальної мережі;
- NAS-сервер, на якому розміщено ігровий програмний застосунок та дані користувачів.

Усі клієнтські комп’ютери працювали в межах однієї локальної мережі та мали доступ до серверного вузла за файловим мережевим протоколом.

Методика експериментального дослідження ґрунтується на послідовному відтворенні типових сценаріїв використання комп’ютерної системи під час навчальних занять. Дослідження проводилося у декілька етапів:

1. Підготовчий етап. На серверному вузлі було розміщено виконуваний файл ігрового застосунку та створено окремі каталоги для збереження результатів користувачів. На клієнтських комп’ютерах налаштовано мережевий доступ до серверних ресурсів.
2. Етап запуску застосунку. Виконувалося одночасне ініціювання запуску ігрового застосунку з усіх клієнтських робочих місць з фіксацією стабільності роботи мережі та сервера.
3. Етап роботи під навантаженням. Під час виконання ігрових завдань здійснювалося періодичне збереження прогресу користувачів на сервері з метою перевірки коректності обміну даними.
4. Етап аналізу результатів. Проводився аналіз отриманих результатів, зокрема стабільності роботи системи, відсутності збоїв та збереження даних.

Застосування даної методики дозволяє комплексно оцінити працездатність синтезованої комп’ютерної системи та зробити обґрунтовані висновки щодо її ефективності в умовах реальної експлуатації.

5.2 Опис експериментального стенду та програмно-апаратного середовища

Для проведення експериментального дослідження було сформовано експериментальний стенд, який відтворює реальну комп’ютерну систему початкової школи з інтегрованим ігровим програмним забезпеченням. Стенд побудовано на основі синтезованої у попередніх розділах архітектури та включає апаратні, мережеві й програмні компоненти, що використовуються у повсякденному навчальному процесі.

Апаратна частина експериментального стенду включає клієнтські робочі місця, мережеве обладнання та серверний вузол.

Клієнтська частина представлена 18 комп’ютерними робочими місцями, з яких:

- 10 ноутбуків, розміщені у комп’ютерному класі;
- 8 моноблоків, встановлені у суміжному навчальному кабінеті.

Клієнтські пристрої використовуються для взаємодії користувачів із програмним застосунком, ініціювання мережевих запитів та відображення інтерфейсу користувача. Високі обчислювальні навантаження на клієнтські системи відсутні, оскільки програмні ресурси зберігаються на серверному вузлі.

Мережева частина експериментального стенду реалізована на основі керованого комутатора другого рівня (L2), який забезпечує комутацію трафіку між усіма клієнтськими робочими місцями та серверною підсистемою. Підключення здійснено за допомогою кабельної інфраструктури Ethernet категорії Cat5e зі швидкістю передачі даних до 1 Гбіт/с.

Серверна частина експериментального стенду реалізована на базі NAS-сервера, який виконує функції централізованого файлового сховища. На серверному вузлі розміщено:

- виконуваний файл ігрового застосунку;
- програмні ресурси (графіка, звукові файли);

– каталоги збереження індивідуального прогресу користувачів.

Усі клієнтські комп’ютери працюють під керуванням сучасної настільної операційної системи з підтримкою файлових мережевих протоколів. Доступ до серверних ресурсів здійснюється за допомогою стандартного файлового протоколу, що забезпечує прозорий доступ до мережевих каталогів без необхідності встановлення додаткового клієнтського програмного забезпечення.

Ігровий програмний застосунок розроблено мовою Python із використанням бібліотеки Pygame. Запуск застосунку здійснюється без локальної інсталяції — шляхом виконання файлу безпосередньо з мережевого каталогу NAS-сервера. Такий підхід дозволяє централізовано керувати версіями програмного забезпечення та спрощує процес оновлення.

Для збереження результатів навчальної діяльності використовується файлове збереження даних у персональних каталогах користувачів на серверному вузлі. Доступ до каталогів обмежується правами файлової системи, що підвищує рівень безпеки та впорядкованості даних.

Експериментальний стенд функціонує в межах єдиного локального мережевого сегмента. Усі клієнтські пристрої та серверний вузол мають постійне мережеве з’єднання, що дозволяє проводити експерименти з одночасним доступом декількох користувачів до серверних ресурсів.

Режим роботи стенду відповідає типовому навчальному сценарію: одночасний запуск застосунку декількома учнями, виконання ігрових завдань та періодичне збереження результатів на сервері. Це дозволяє адекватно оцінити працездатність і стабільність комп’ютерної системи під реальним навантаженням.

Таким чином, експериментальний стенд є повною апаратно-програмною реалізацією синтезованої комп’ютерної системи та забезпечує коректні умови для проведення експериментального дослідження її функціонування, продуктивності та надійності в умовах освітнього середовища.

5.2.1 Коротка характеристика розробленого ігрового програмного застосунку, що використовується в експерименті

У межах експериментального дослідження для перевірки працездатності синтезованої комп’ютерної системи використовується ігровий програмний застосунок навчального призначення, розроблений мовою програмування Python з використанням бібліотеки Pygame. Застосунок було створено в межах попередньої кваліфікаційної роботи та адаптовано для інтеграції у локальну комп’ютерну систему початкової школи.

Основним призначенням застосунку є розвиток базових математичних навичок дітей молодшого шкільного віку шляхом виконання інтерактивних ігрових завдань. При цьому в даній магістерській роботі застосунок розглядається не як основний об’єкт дослідження, а як програмне навантаження, що використовується для експериментальної перевірки роботи комп’ютерної системи та мережевої інфраструктури.

Ігровий застосунок реалізовано у вигляді настільної програми з графічним інтерфейсом користувача. Взаємодія з користувачем здійснюється за допомогою клавіатури та миші. Програма має модульну структуру та включає:

- головне меню застосунку;
- ігрові рівні з математичними завданнями;
- систему обліку та збереження результатів користувачів.

Рисунок 5.1 – Головне меню ігрового застосунку

Головне меню забезпечує навігацію між основними режимами роботи програми та дозволяє користувачу розпочати ігровий процес.

Під час ігрового процесу користувачеві пропонуються математичні завдання, що відповідають віковим особливостям учнів початкової школи. Завдання подаються у формі гри, що сприяє підвищенню мотивації та зацікавленості учнів.

Рисунок 5.2 – Рівні у вигляді вуликів

Результати виконання завдань (бали, рівень проходження, успішність) зберігаються у вигляді файлів даних. У межах даної магістерської роботи ці файли зберігаються у персональних каталогах користувачів на NAS-сервері, що дозволяє використовувати застосунок у клієнт-серверному режимі.

Для проведення експерименту ігровий застосунок було адаптовано до запуску з мережевого файлового ресурсу без локальної інсталяції на клієнтських комп’ютерах. Такий підхід дозволив:

- централізувати зберігання виконуваного файлу та ресурсів;
- забезпечити використання єдиної версії програмного забезпечення;
- спростити оновлення та технічне обслуговування;
- перевірити працездатність комп’ютерної системи під реальним мережевим навантаженням.

Таким чином, розроблений ігровий програмний застосунок використовується як інструмент експериментального навантаження, що дозволяє на практиці оцінити ефективність синтезованої комп’ютерної системи та її готовність до використання в умовах навчального закладу.

5.3 Дослідження працездатності комп’ютерної системи при одночасному доступі користувачів

Дослідження працездатності комп’ютерної системи проводилося з метою оцінки її стабільності та коректності функціонування в умовах одночасного доступу декількох користувачів до серверних ресурсів. Основна увага приділялася перевірці роботи мережевої інфраструктури, серверної підсистеми та механізмів обміну даними між клієнтськими робочими місцями і NAS-сервером.

5.3.1 Сценарій експериментального навантаження

Експеримент проводився за сценарієм, що імітує типовий навчальний процес у комп’ютерному класі початкової школи. У межах експерименту одночасно використовувалися всі 18 клієнтських робочих місць, підключених до локальної мережі.

Сценарій навантаження включав такі етапи:

- одночасний запуск ігрового програмного застосунку з усіх клієнтських комп’ютерів;
- паралельну роботу користувачів у застосунку;
- періодичне збереження прогресу кожного користувача на серверному вузлі;
- завершення роботи та повторний запуск застосунку для перевірки коректності відновлення даних.

Такий сценарій дозволяє створити пікове навантаження на серверну та мережеву підсистеми, що є критичним з точки зору перевірки працездатності комп’ютерної системи.

5.3.2 Хід експерименту

На початковому етапі всі клієнтські комп’ютери були підключені до локальної мережі та мали доступ до спільного мережевого каталогу на NAS-сервері. Запуск застосунку здійснювався шляхом виконання файлу з мережевого ресурсу.

Під час одночасного запуску застосунку спостерігалося короткочасне збільшення мережевої активності, пов’язане з передачею виконуваного файлу та ресурсів гри. Після завершення етапу ініціалізації навантаження на мережу знижувалося та стабілізувалося.

У процесі виконання ігрових завдань користувачами відбувався регулярний обмін даними з сервером, пов’язаний із записом результатів ігрового

процесу. Збереження даних здійснювалося у персональні каталоги користувачів без конфліктів доступу або втрати інформації.

5.3.3 Спостереження та результати експерименту

У ході експериментального дослідження було зафіксовано такі результати:

- комп’ютерна система забезпечує стабільний одночасний запуск ігрового застосунку з 18 клієнтських робочих місць;
- відсутні збої у роботі мережевої інфраструктури при піковому навантаженні;
- серверна підсистема коректно обробляє паралельні запити на читання та запис даних;
- збереження прогресу користувачів відбувається без втрати або пошкодження інформації;
- повторний запуск застосунку підтверджує коректність збережених даних.

Також було встановлено, що обраний тип мережевої організації та використання L2-комутатора забезпечують достатню пропускну здатність для обслуговування всіх клієнтів без помітного зниження швидкодії.

5.3.4 Оцінка працездатності системи

Отримані результати свідчать про те, що синтезована комп’ютерна система є працездатною та відповідає вимогам експлуатації в умовах навчального закладу. Централізований підхід до зберігання програмного забезпечення та даних користувачів забезпечує надійність, зручність адміністрування та стійкість системи до типових навантажень.

Проведене дослідження підтвердило доцільність обраної архітектури та комплексу технічних засобів для інтеграції ігрового програмного забезпечення у локальну комп’ютерну систему початкової школи.

5.4 Аналіз результатів експериментального дослідження

У даному підрозділі наведено аналіз результатів експериментального дослідження працездатності синтезованої комп’ютерної системи початкової школи з інтегрованим ігровим програмним забезпеченням. Аналіз ґрунтується на спостереженнях, отриманих у процесі одночасного використання системи всіма клієнтськими робочими місцями в умовах, наближених до реального навчального процесу.

Результати експерименту показали, що обрана архітектура комп’ютерної системи забезпечує стабільну роботу при одночасному запуску ігрового застосунку з 18 клієнтських робочих місць. Короткочасне пікове навантаження на мережеву інфраструктуру під час ініціалізації застосунку не призвело до порушень зв’язку або відмови серверної підсистеми.

Після завершення етапу запуску спостерігалося зниження мережевої активності та стабілізація роботи системи, що свідчить про правильний розподіл навантаження між клієнтськими та серверними компонентами.

Серверна підсистема, реалізована на базі NAS-сервера, продемонструвала здатність коректно обробляти паралельні запити на читання та запис даних. Запис результатів ігрового процесу в індивідуальні каталоги користувачів відбувався без конфліктів доступу та втрати інформації.

Централізоване зберігання даних дозволило уникнути проблем, пов’язаних із локальним збереженням результатів на клієнтських пристроях, та забезпечило можливість відновлення прогресу незалежно від конкретного робочого місця користувача.

Використання L2-комутатора та топології типу «зірка» забезпечило достатню пропускну здатність локальної мережі для обслуговування всіх клієнтів. В експерименті не було зафіксовано втрат пакетів, затримок або деградації якості обміну даними, що могло б негативно вплинути на роботу програмного забезпечення.

Отримані результати підтверджують, що застосування мережі Ethernet зі швидкістю до 1 Гбіт/с є технічно обґрунтованим рішенням для умов шкільного комп’ютерного класу.

З точки зору експлуатації комп’ютерної системи, централізований запуск програмного забезпечення та збереження даних значно спрощують процес адміністрування. Оновлення застосунку здійснюється шляхом заміни файлів на серверному вузлі без необхідності втручання в налаштування клієнтських комп’ютерів.

Такий підхід знижує витрати часу на технічне обслуговування та мінімізує ймовірність помилок, пов’язаних з використанням різних версій програмного забезпечення.

Узагальнюючи результати експериментального дослідження, можна зробити висновок, що синтезована комп’ютерна система відповідає поставленим вимогам щодо стабільності, надійності та зручності використання в освітньому середовищі. Обрані технічні рішення забезпечують ефективну інтеграцію ігрового програмного забезпечення та створюють умови для його безперебійної експлуатації в умовах навчального закладу.

ВИСНОВКИ

Кваліфікаційна робота є завершеною науковою роботою, яка полягає в обґрунтуванні структури та параметрів комп’ютерної системи Дніпровської початкової школи з інтегрованим розвиваючим ігровим застосунком, що забезпечує централізований запуск програмного забезпечення, стабільний обмін даними та надійне збереження результатів навчальної діяльності учнів.

Основні висновки і результати роботи полягають у наступному:

1. Проаналізовані сучасні підходи до побудови комп’ютерних систем освітніх закладів і визначено вимоги до апаратної, мережевої та серверної складових.

2. Запропоновано клієнт-серверну архітектуру комп’ютерної системи з використанням локальної мережі Ethernet, L2-комутатора та NAS-сервера як центрального вузла зберігання програмних ресурсів і даних користувачів.

3. Обгрунтовано вибір комплексу технічних засобів, розроблено структурну та функціональну схеми системи, а також визначено принципи її функціонування в умовах навчального закладу. Запропоновані технічні рішення забезпечують зменшення витрат на адміністрування, спрощення оновлення програмного забезпечення та можливість масштабування системи при збільшенні кількості робочих місць.

4. Проведено перевірку працездатності синтезованої комп’ютерної системи в умовах, наближених до реального навчального процесу. Результати експериментальних досліджень підтвердили стабільну роботу системи при одночасному підключенні 18 клієнтських робочих місць, коректний обмін даними між клієнтами та серверною підсистемою, а також надійне збереження прогресу користувачів.

Наукове значення роботи полягає в обґрунтуванні підходів до побудови комп’ютерної системи освітнього закладу з інтегрованим програмним

навантаженням та у формуванні структурних і функціональних моделей, що можуть бути використані при проєктуванні аналогічних систем.

Практичне значення роботи полягає в можливості впровадження розробленої комп’ютерної системи у початкових школах для забезпечення ефективного використання навчальних ігрових застосунків, централізованого зберігання даних та спрощення процесів адміністрування.

Отримані результати підтверджують досягнення поставленої мети та доводять, що розроблена комп’ютерна система відповідає вимогам надійності, продуктивності та зручності експлуатації в умовах початкової школи.

ПЕРЕЛІК ПОСИЛАНЬ

1. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. – Київ : ДП «УкрНДНЦ», 2016. – 17 с.
2. Таненбаум Е., Везерол Д. Комп’ютерні мережі : підручник. – 5-те вид. – Київ : Вільямс, 2018. – 960 с.
3. Олійник В. В., Коваленко О. М. Комп’ютерні системи і мережі : навчальний посібник. – Київ : Кондор, 2019. – 344 с.
4. Гриценко В. І. Основи комп’ютерних систем та мереж : навчальний посібник. – Харків : ХНУРЕ, 2017. – 286 с.
5. Cisco Networking Academy. Introduction to Networks (ITN). – Cisco Systems, 2023. – URL: https://www.netacad.com
6. Cisco Systems. Catalyst 2960 Series Switches Data Sheet. – 2022. – URL: https://www.cisco.com
7. Synology Inc. Synology DS220+ Hardware Specification. – 2023. – URL: https://www.synology.com
8. Microsoft. SMB Protocol Overview. – 2023. – URL: https://learn.microsoft.com
9. Python Software Foundation. Python Documentation. – 2024. – URL: https://docs.python.org
10. Pygame Community. Pygame Documentation. – 2024. – URL: https://www.pygame.org/docs
11. Цвіркун Л.І. Комп’ютерні мережі. Методичні рекомендації до виконання лабораторних робіт студентами галузі знань 12 Інформаційні технології спеціальності 123 Комп’ютерна інженерія: у 2 ч. / Л.І. Цвіркун, Я.В. Панферова ; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». –Дніпро: НТУ «ДП», 2018. – Ч. 2. – 39 с.
12. Жалдак М. І., Рамський Ю. С. Інформаційні технології в освіті : навчальний посібник. – Київ : НПУ ім. М. П. Драгоманова, 2018. – 312 с.

ДОДАТОК А
ТЕКСТ ПРОГРАМИ РОЗВИВАЮЧОГО ІГРОВОГО ЗАСТОСУНКУ

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ РОЗВИВАЮЧОГО ІГРОВОГО ЗАСТОСУНКУ

Текст програми

804.02070743.25002-01 12 01

Листів 19

2025

АНОТАЦІЯ

Програмний код ігрового застосунку «Sweet Math» реалізує навчальну комп’ютерну гру з математики, призначену для учнів молодшого та середнього шкільного віку. Застосунок розроблено мовою програмування Python з використанням бібліотеки Pygame, що забезпечує створення графічного інтерфейсу, обробку подій користувача, керування ігровою логікою та відтворення мультимедійного контенту.

Основний програмний модуль відповідає за ініціалізацію середовища виконання, завантаження графічних і звукових ресурсів, організацію структури гри та керування переходами між екранами. У програмі реалізовано головне меню, екран інструкцій, форму введення віку користувача, вибір рівня складності та основний ігровий процес. Логіка гри побудована таким чином, що складність математичних завдань автоматично адаптується до віку гравця та обраного рівня, що підвищує навчальну ефективність застосунку.

Ігровий процес базується на взаємодії користувача з динамічними об’єктами, які представляють математичні завдання. Реалізовано систему життів, підрахунок балів, генерацію випадкових числових прикладів, перевірку правильності відповідей і реакцію на помилки. Програма також містить механізми контролю зіткнень, керування рухом ігрового персонажа, відображення поточного результату та обробку завершення гри з можливістю повернення до головного меню.

Допоміжний модуль забезпечує реалізацію текстового введення у графічному середовищі Pygame та використовується для коректного введення даних користувачем. Він відповідає за обробку натискань клавіш, керування курсором, валідацію введених даних і їх відображення на екрані. Загалом програмний код має модульну структуру, що спрощує його розширення та супровід, а також дозволяє інтегрувати ігровий застосунок у комп’ютерну систему навчального закладу як елемент освітнього середовища.

ЗМІСТ

1 ПРОГРАМА ІГРОВОГО ЗАСТОСУНКУ

1.1 Реалізація main.py

```
import pygame
import random
import pygame_textinput

pygame.init()

win = pygame.display.set_mode((1000, 550))


pygame.display.set_caption("Sweet math")


pngRaw = pygame.image.load('Images/png.png')
png = pygame.transform.scale(pngRaw, (150, 90))
kvitkaRaw = pygame.image.load('Images/kvitka.png')
kvitkaPic = pygame.transform.scale(kvitkaRaw, (120, 120))
kvitka1Pic = pygame.transform.scale(kvitkaRaw, (170, 120))
heart0 = pygame.transform.scale(pygame.image.load('Images/Heart0.png'), (90,
40))
heart1 = pygame.transform.scale(pygame.image.load('Images/Heart1.png'), (130,
50))
heart2 = pygame.transform.scale(pygame.image.load('Images/Heart2.png'), (130,
50))
heart3 = pygame.transform.scale(pygame.image.load('Images/Heart3.png'), (130,
50))
heartpics = [heart1, heart2, heart3, heart0]
menu1raw = pygame.image.load('Images/menu.jpg')
menu1 = pygame.transform.scale(menu1raw, (1000, 550))
matheus1raw = pygame.image.load('Images/matheus1.jpg')
matheus1 = pygame.transform.scale(matheus1raw, (1000, 550))
matheus2raw = pygame.image.load('Images/matheus2.jpg')
matheus2 = pygame.transform.scale(matheus2raw, (1000, 550))
matheus3raw = pygame.image.load('Images/matheus3.jpg')
matheus3 = pygame.transform.scale(matheus3raw, (1000, 550))
matheus4raw = pygame.image.load('Images/matheus4.jpg')
matheus4 = pygame.transform.scale(matheus4raw, (1000, 550))
uley1raw = pygame.image.load('Images/uley.png')
uley1 = pygame.transform.scale(uley1raw, (150, 150))
uley2raw = pygame.image.load('Images/uley.png')
uley2 = pygame.transform.scale(uley2raw, (150, 150))
uley3raw = pygame.image.load('Images/uley.png')
uley3 = pygame.transform.scale(uley3raw, (150, 150))
uley4raw = pygame.image.load('Images/uley.png')
uley4 = pygame.transform.scale(uley4raw, (150, 150))
```

```
# імпортування всіх звуків
collisionsound = [pygame.mixer.Sound('Audio/1.wav'),
pygame.mixer.Sound('Audio/2.wav'), pygame.mixer.Sound('Audio/3.wav'),
                  pygame.mixer.Sound('Audio/4.wav'),
pygame.mixer.Sound('Audio/5.wav'), pygame.mixer.Sound('Audio/6.wav'),
                  pygame.mixer.Sound('Audio/7.wav'),
pygame.mixer.Sound('Audio/8.wav'), pygame.mixer.Sound('Audio/9.wav')]
bulletsound = pygame.mixer.Sound('Audio/Gunshot.wav')
selectsound = pygame.mixer.Sound('Audio/selectsound.wav')
explosion = pygame.mixer.Sound('Audio/explosion.wav')
music = pygame.mixer.music.load('Audio/BgMusic.mp3')
pygame.mixer.music.play(-1)

clock = pygame.time.Clock()
textinput = pygame_textinput.TextInput()


def menu():
    def redrawGameWindow():
        win.blit(menu1, (0, 0))
        screenText()
        pygame.display.update()

    def screenText():

        font = pygame.font.Font('ARCADECLASSIC.TTF', 150)
        text = font.render('Sweetmath', 1, (150, 0, 24))
        win.blit(text, (500 - ((text.get_width() / 2) - 20), 40))




        font2 = pygame.font.Font('ARCADECLASSIC.TTF', 50)
        text2 = font2.render('Play', 1, (150, 0, 24))
        win.blit(text2, (500 - ((text2.get_width() / 2) - 8), 262))


    run = True
    while run:
        clock.tick(27)

        keys = pygame.key.get_pressed()

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                quit()

            if event.type == pygame.MOUSEBUTTONDOWN:
                if mouse[0] >= 430 and mouse[0] <= 570 and mouse[1] >= 280 and
mouse[1] <= 320:
                    selectsound.play()
                    instructions()
```

```
        mouse = pygame.mouse.get_pos()

        redrawGameWindow()


def instructions():
    def redrawGameWindow():
        win.blit(matheus1, (0, 0))
        screenText()
        pygame.display.update()

    def screenText():


        font = pygame.font.Font('ARCADECLASSIC.TTF', 150)
        text = font.render('Sweetmath', 1, (150, 0, 24))
        win.blit(text, (500 - ((text.get_width() / 2) - 20), 40))


        font2 = pygame.font.Font('ARCADECLASSIC.TTF', 60)
        text2 = font2.render('Instructions', 1, (150, 0, 24))
        win.blit(text2, (500 - (text2.get_width() / 2), 150))


        font3 = pygame.font.Font('ARCADECLASSIC.TTF', 30)
        text3 = font3.render((
                'Welcome' + '   ' + 'to' + '   ' + 'Sweetmath' + '   ' + 'Your'
+ '   ' + 'mission' + '   ' + 'is' + '   ' + 'to' + '   ' + 'collect'),
            1, (231, 84, 128))
        text4 = font3.render((
                'all' + '   ' + 'the' + '   ' + 'honey' + '   ' + 'by' + '   ' +
'answering' + '   ' + 'math' + '   ' + 'questions'),
            1, (231, 84, 128))
        text5 = font3.render((
                'Use' + '   ' + 'the' + '   ' + 'up' + '   ' + 'and' + '   ' +
'down' + '   ' + 'arrow' + '   ' + 'keys' + '   ' + 'to' + '   ' + 'move' +
'   ' + 'the' + '   ' + 'bee'),
            1, (231, 84, 128))
        text6 = font3.render((
                'and' + '   ' + 'the' + '   ' + 'space' + '   ' + 'bar' + '   '
+ 'to' + '   ' + 'shoot' + '   ' + 'Press' + '   ' + 'space ' + '   ' + 'bar'+
'   ' + 'to ' + '   ' + 'shoot'),
            1, (231, 84, 128))
        text7 = font3.render((
                'Good' + '   ' + 'luck' + '   ' + 'friend'),
            1, (231, 84, 128))
        win.blit(text3, (500 - (text3.get_width() / 2), 220))
        win.blit(text4, (500 - (text4.get_width() / 2), 240))
        win.blit(text5, (500 - (text5.get_width() / 2), 260))
        win.blit(text6, (500 - (text6.get_width() / 2), 280))
        win.blit(text7, (500 - (text7.get_width() / 2), 300))
```

```
        pygame.draw.rect(win, (0, 100, 200), [(500 - (140 / 2)), 440, 130, 40])


        font4 = pygame.font.Font('ARCADECLASSIC.TTF', 50)
        text4 = font4.render('Next', 1, (150, 0, 24))
        win.blit(text4, (500 - (text4.get_width() / 2), 435))

    run = True
    while run:
        clock.tick(27)
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                quit()
            if event.type == pygame.MOUSEBUTTONDOWN:
                if mouse[0] >= 430 and mouse[0] <= 570 and mouse[1] >= 440 and
mouse[1] <= 480:
                    selectsound.play()
                    ageinputscreen()

        mouse = pygame.mouse.get_pos()

        redrawGameWindow()

def ageinputscreen():
    def redrawGameWindow():
        win.blit(matheus1, (0, 0))
        screenText()
        win.blit(textinput.get_surface(), (450, 293))
        ageChecker()
        pygame.display.update()

    def screenText():


        font = pygame.font.Font('ARCADECLASSIC.TTF', 150)
        text = font.render('Sweetmath', 1, (150, 0, 24))
        win.blit(text, (500 - ((text.get_width() / 2) - 20), 40))


        font2 = pygame.font.Font('ARCADECLASSIC.TTF', 80)
        text2 = font2.render('Enter your age', 1, (150, 0, 24))
        win.blit(text2, (500 - (text2.get_width() / 2), 180))

        pygame.draw.rect(win, (0, 100, 200), [(500 - (140 / 2)), 440, 130, 40])
        pygame.draw.rect(win, (255, 255, 255), [(500 - (200 / 2)), 300, 200,
80])


        font4 = pygame.font.Font('ARCADECLASSIC.TTF', 50)
        text4 = font4.render('Next', 1, (150, 0, 24))
        win.blit(text4, (500 - (text4.get_width() / 2), 435))
```

```
    def ageChecker():
        if ageVar == 1:
            font4 = pygame.font.Font('ARCADECLASSIC.TTF', 50)
            text4 = font4.render('You are too young!', 1, (231, 84, 128))
            win.blit(text4, (500 - (text4.get_width() / 2), 380))

        if ageVar == 2:
            font4 = pygame.font.Font('ARCADECLASSIC.TTF', 50)
            text4 = font4.render('You are too old!', 1, (231, 84, 128))
            win.blit(text4, (500 - (text4.get_width() / 2), 380))

        if ageVar == 3:
            font4 = pygame.font.Font('ARCADECLASSIC.TTF', 50)
            text4 = font4.render('Please   enter   a   valid   age!', 1, (255,
255, 255))
            win.blit(text4, (500 - (text4.get_width() / 2), 380))

    ageVar = 0
    run = True
    while run:
        clock.tick(27)
        events = pygame.event.get()
        for event in events:
            if event.type == pygame.QUIT:
                quit()
            if event.type == pygame.MOUSEBUTTONDOWN:
                if mouse[0] >= 430 and mouse[0] <= 570 and mouse[1] >= 440 and
mouse[1] <= 480:
                    selectsound.play()
                    try:
                        userAge = int(textinput.get_text())

                        if userAge >= 5 and userAge <= 13:
                            levelscreen(userAge)
                        else:
                            if userAge < 5:
                                ageVar = 1
                            if userAge > 13:
                                ageVar = 2

                    except ValueError:
                        ageVar = 3

        textinput.update(events)

        mouse = pygame.mouse.get_pos()

        redrawGameWindow()

def levelscreen(userAge):
```

```
    def redrawGameWindow(userAge):
        win.blit(matheus1, (0, 0))
        win.blit(uley1, (105, 323))
        win.blit(uley2, (315, 323))
        win.blit(uley3, (530, 323))
        win.blit(uley4, (740, 323))
        screenText(userAge)
        pygame.display.update()

    def screenText(userAge):


        font = pygame.font.Font('ARCADECLASSIC.TTF', 150)
        text = font.render('Sweetmath', 1, (150, 0, 24))
        win.blit(text, (500 - ((text.get_width() / 2) - 20), 40))


        font2 = pygame.font.Font('ARCADECLASSIC.TTF', 80)
        text2 = font2.render('Select Level', 1, (150, 0, 24))
        win.blit(text2, (500 - (text2.get_width() / 2), 180))


        font3 = pygame.font.Font('ARCADECLASSIC.TTF', 80)
        text3 = font3.render('1', 1, (150, 0, 24))
        text3a = font3.render('2', 1, (150, 0, 24))
        text3b = font3.render('3', 1, (150, 0, 24))
        text3c = font3.render('4', 1, (150, 0, 24))
        win.blit(text3, (180 - (text3.get_width() / 2), 400 -
(text3.get_height() / 2)))
        win.blit(text3a, (393.3 - (text3a.get_width() / 2), 400 -
(text3.get_height() / 2)))
        win.blit(text3b, (606.6 - (text3b.get_width() / 2), 400 -
(text3.get_height() / 2)))
        win.blit(text3c, (819.9 - (text3c.get_width() / 2), 400 -
(text3.get_height() / 2)))

        if userAge >= 5 and userAge <= 7:
            k = 100
        if userAge >= 8 and userAge <= 10:
            k = 307
        if userAge >= 11 and userAge <= 12:
            k = 517
        if userAge == 13:
            k = 730

        font2 = pygame.font.Font('ARCADECLASSIC.TTF', 20)
        text2 = font2.render('Recommended level', 1, (231, 84, 128))
        win.blit(text2, (k, 300))

    run = True
    while run:
        clock.tick(27)
```

```
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                quit()


            if event.type == pygame.MOUSEBUTTONDOWN:
                if mouse[0] >= 137.5 and mouse[0] <= 222.5 and mouse[1] >= 357.5
and mouse[1] <= 442.5:
                    level = 1
                    selectsound.play()
                    mainGame(level)
                if mouse[0] >= 350.8 and mouse[0] <= 435.8 and mouse[1] >= 357.5
and mouse[1] <= 442.5:
                    level = 2
                    selectsound.play()
                    mainGame(level)
                if mouse[0] >= 564.1 and mouse[0] <= 649.1 and mouse[1] >= 357.5
and mouse[1] <= 442.5:
                    level = 3
                    selectsound.play()
                    mainGame(level)
                if mouse[0] >= 777.4 and mouse[0] <= 862.4 and mouse[1] >= 357.5
and mouse[1] <= 442.5:
                    level = 4
                    selectsound.play()
                    mainGame(level)

        mouse = pygame.mouse.get_pos()

        redrawGameWindow(userAge)


def mainGame(level):
    class player(object):
        def __init__(self, x, y, width, height):
            self.x = x
            self.y = y
            self.width = width
            self.height = height
            self.vel = 10
            self.hitbox = (self.x + 10, self.y, 100, 90)
            self.heartCounter = 3

        def draw(self, win):
            win.blit(png, (self.x, self.y))
            self.hitbox = (self.x + 10, self.y, 100, 90)


        def hit(self):
            (random.choice(collisionsound)).play()
            pygame.display.update()
            bee.heartCounter -= 1
```

```
            if bee.heartCounter == 2:
                heartInGame.heartpicsCurrent = heartpics[1]
            if bee.heartCounter == 1:
                heartInGame.heartpicsCurrent = heartpics[0]
            if bee.heartCounter == 0:
                heartInGame.heartpicsCurrent = heartpics[3]
                print("game over")


            i = 0
            while i < 3:
                pygame.time.delay(10)
                i += 1
                for event in pygame.event.get():
                    if event.type == pygame.QUIT:
                        i = 301
                        pygame.quit()

class projectile(object):
    def __init__(self, x, y, radius, colour):
        self.x = x
        self.y = y
        self.radius = radius
        self.colour = colour
        self.vel = 70

    def draw(self, win):
        pygame.draw.circle(win, self.colour, (self.x, self.y), self.radius)

class kvitnnyk(object):
    def __init__(self, x, y, width, height, end):
        self.x = x
        self.y = y
        self.width = width
        self.height = height
        self.path = [x, end]
        self.vel = 8
        self.hitbox = (self.x, self.y + 10, 120, 100)
        self.visible = True
        self.number = 0

    def draw(self, win):
        if self.visible:
            if level == 1:
                win.blit(kvitkaPic, (self.x, self.y))
            if level == 2 or level == 3 or level ==4:
                win.blit(kvitka1Pic, (self.x, self.y))
            self.hitbox = (self.x, self.y + 10, 120, 100)

            font2 = pygame.font.Font('ARCADECLASSIC.TTF', 80)
```

```
                text = font2.render((str(self.number)), 1, (255, 255, 255))
                win.blit(text, (self.x + 30, self.y + 30))


        def hit(self):
            self.visible = False

    class heart(object):
        def __init__(self, x, y, width, height):
            self.x = x
            self.y = y
            self.width = width
            self.height = height
            self.visible = True
            self.heartpicsCurrent = heartpics[2]

        def draw(self, win):
            if self.visible:
                win.blit(self.heartpicsCurrent, (self.x, self.y))

    class question(object):
        def draw(self, a, b):
            self.a = a
            self.b = b
            font2 = pygame.font.Font('ARCADECLASSIC.TTF', 80)
            text = font2.render((str(self.a) + " x " + str(self.b)), 1, (255,
255, 255))
            win.blit(text, (500 - (text.get_width() / 2), 450))

    class score(object):
        def draw(self, score):
            self.score = score
            font2 = pygame.font.Font('ARCADECLASSIC.TTF', 40)
            text = font2.render(("Score " + str(score)), 1, (255, 255, 255))
            win.blit(text, (830, 5))
            text2 = font2.render(('Exit to menu'), 1, (255, 255, 255))
            win.blit(text2, (10, 5))

    class gameover(object):
        def draw(self, win, score):
            if bee.heartCounter == 0:
                font3 = pygame.font.Font('ARCADECLASSIC.TTF', 120)
                text = font3.render(('GAME OVER'), 1, (231, 84, 128))
                win.blit(text, ((500 - text.get_width() / 2), 50))
                font4 = pygame.font.Font('ARCADECLASSIC.TTF', 60)
                text2 = font4.render(('SCORE ' + str(score)), 1, (231, 84, 128))
                win.blit(text2, ((500 - text2.get_width() / 2), 150))
                text3 = font4.render(
                    ('Press' + '   ' + 'enter' + '   ' + 'to' + '   ' + 'go' +
'   ' + 'to' + '   ' + 'menu'), 1,
                    (231, 84, 128))
```

```
            win.blit(text3, ((500 - text3.get_width() / 2), 300))


def redrawGameWindow(a, b, score, scorequestionVisible, level):


    if level == 1:
        win.blit(matheus1, (0, 0))
    if level == 2:
        win.blit(matheus2, (0, 0))
    if level == 3:
        win.blit(matheus3, (0, 0))
    if level == 4:
        win.blit(matheus4, (0, 0))

    bee.draw(win)
    kvitkaTop.draw(win)
    kvitkaMiddle.draw(win)
    kvitkaBottom.draw(win)
    heartInGame.draw(win)


    if scorequestionVisible == True:
        questionGame.draw(a, b)
        scoreGame.draw(score)

    gameoverGame.draw(win, score)

    for bullet in bullets:
        bullet.draw(win)

    pygame.display.update()


bee = player(104, 235, 100, 100)
kvitkaTop = kvitnnyk(800, 40, 100, 100, 300)
kvitkaMiddle = kvitnnyk(800, 190, 100, 100, 300)
kvitkaBottom = kvitnnyk(800, 340, 100, 100, 300)
heartInGame = heart(30, 490, 30, 30)
questionGame = question()
scoreGame = score()
gameoverGame = gameover()
bullets = []
run = True


if level == 1:
    number1 = random.randint(1, 3)
    number2 = random.randint(1, 3)
if level == 2:
    number1 = random.randint(1, 6)
```

```
    number2 = random.randint(1, 6)
if level == 3:
    number1 = random.randint(1, 9)
    number2 = random.randint(1, 9)
if level == 4:
    number1 = random.randint(1, 19)
    number2 = random.randint(1, 19)

scorenumber = 0
whichKvitochka = random.randint(1, 3)


if whichKvitochka == 1:
    kvitkaTop.number = number1 * number2
    kvitkaMiddle.number = random.randint(1, 30)
    kvitkaBottom.number = random.randint(1, 30)
if whichKvitochka == 2:
    kvitkaTop.number = random.randint(1, 30)
    kvitkaMiddle.number = number1 * number2
    kvitkaBottom.number = random.randint(1, 30)
if whichKvitochka == 3:
    kvitkaTop.number = random.randint(1, 30)
    kvitkaMiddle.number = random.randint(1, 30)
    kvitkaBottom.number = number1 * number2
w = 1
scorequestionVisible = True

while run:
    clock.tick(27)

    if bee.heartCounter == 0:
        if kvitkaTop.x == 1200:
            kvitkaTop.vel = 0
            kvitkaMiddle.vel = 0
            kvitkaBottom.vel = 0
        bee.vel = 0
        if keys[pygame.K_KP_ENTER] or keys[pygame.K_RETURN]:
            menu()
        scorequestionVisible = False

    kvitkaTop.x -= kvitkaTop.vel
    kvitkaMiddle.x -= kvitkaMiddle.vel
    kvitkaBottom.x -= kvitkaBottom.vel

    if kvitkaTop.x == -176:
        kvitkaTop.x = 1200
        kvitkaTop.visible = True

        if level == 1:
            number1 = random.randint(1, 3)
            number2 = random.randint(1, 3)
        if level == 2:
```

```
            number1 = random.randint(1, 6)
            number2 = random.randint(1, 6)
        if level == 3:
            number1 = random.randint(1, 9)
            number2 = random.randint(1, 9)
        if level == 4:
            number1 = random.randint(1, 19)
            number2 = random.randint(1, 19)

        whichKvitochka = random.randint(1, 3)


if kvitkaMiddle.x == -176:
    kvitkaMiddle.x = 1200
    kvitkaMiddle.visible = True
if kvitkaBottom.x == -176:
    kvitkaBottom.x = 1200
    kvitkaBottom.visible = True

    if whichKvitochka == 1:
        kvitkaTop.number = number1 * number2

        if level == 1:
            kvitkaMiddle.number = random.randint(1, 9)
            kvitkaBottom.number = random.randint(1, 9)
        if level == 2:
            kvitkaMiddle.number = random.randint(1, 40)
            kvitkaBottom.number = random.randint(1, 40)
        if level == 3:
            kvitkaMiddle.number = random.randint(1, 90)
            kvitkaBottom.number = random.randint(1, 90)
        if level == 4:
            kvitkaMiddle.number = random.randint(1, 400)
            kvitkaBottom.number = random.randint(1, 400)

    if whichKvitochka == 2:
        kvitkaMiddle.number = number1 * number2
        if level == 1:
            kvitkaTop.number = random.randint(1, 9)
            kvitkaBottom.number = random.randint(1, 9)
        if level == 2:
            kvitkaTop.number = random.randint(1, 40)
            kvitkaBottom.number = random.randint(1, 40)
        if level == 3:
            kvitkaTop.number = random.randint(1, 90)
            kvitkaBottom.number = random.randint(1, 90)
        if level == 4:
            kvitkaTop.number = random.randint(1, 400)
            kvitkaBottom.number = random.randint(1, 400)

    if whichKvitochka == 3:
        kvitkaBottom.number = number1 * number2
```

```
                if level == 1:
                    kvitkaMiddle.number = random.randint(1, 9)
                    kvitkaTop.number = random.randint(1, 9)
                if level == 2:
                    kvitkaMiddle.number = random.randint(1, 40)
                    kvitkaTop.number = random.randint(1, 40)
                if level == 3:
                    kvitkaMiddle.number = random.randint(1, 90)
                    kvitkaTop.number = random.randint(1, 90)
                if level == 4:
                    kvitkaMiddle.number = random.randint(1, 400)
                    kvitkaTop.number = random.randint(1, 400)

        if kvitkaTop.visible == True:
            if bee.hitbox[1] < kvitkaTop.hitbox[1] + kvitkaTop.hitbox[3] and
bee.hitbox[1] + \
                    bee.hitbox[
                        3] > kvitkaTop.hitbox[1]:
                if bee.x == kvitkaTop.x:
                    bee.hit()

        if kvitkaMiddle.visible == True:
            if bee.hitbox[1] < kvitkaMiddle.hitbox[1] + kvitkaMiddle.hitbox[3]
and bee.hitbox[1] + \
                    bee.hitbox[3] > kvitkaMiddle.hitbox[1]:
                if bee.x == kvitkaMiddle.x:
                    bee.hit()

        if kvitkaBottom.visible == True:
            if bee.hitbox[1] < kvitkaBottom.hitbox[1] + kvitkaBottom.hitbox[3]
and bee.hitbox[1] + \
                    bee.hitbox[3] > kvitkaBottom.hitbox[1]:
                if bee.x == kvitkaBottom.x:
                    bee.hit()

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                quit()

            if event.type == pygame.MOUSEBUTTONDOWN:
                if mouse[0] >= 10 and mouse[0] <= 240 and mouse[1] >= 10 and
mouse[1] <= 50:
                    selectsound.play()
                    menu()
        mouse = pygame.mouse.get_pos()

        for bullet in bullets:

            if whichKvitochka == 1:
                if kvitkaTop.visible == True:
```

```
                    if bullet.y - bullet.radius < kvitkaTop.hitbox[1] +
kvitkaTop.hitbox[
                            3] and bullet.y + bullet.radius > \
                                kvitkaTop.hitbox[1]:
                            if bullet.x + bullet.radius > kvitkaTop.hitbox[0] and
bullet.x - bullet.radius < \
                                        kvitkaTop.hitbox[0] + \
                                        kvitkaTop.hitbox[2]:
                                bullets.pop(bullets.index(bullet))
                                scorenumber += 1
                                explosion.play()
                                kvitkaTop.hit()
            if whichKvitochka == 2:
                if kvitkaMiddle.visible == True:
                    if bullet.y - bullet.radius < kvitkaMiddle.hitbox[1] +
kvitkaMiddle.hitbox[
                            3] and bullet.y + bullet.radius > \
                                kvitkaMiddle.hitbox[1]:
                            if bullet.x + bullet.radius > kvitkaMiddle.hitbox[0] and
bullet.x - bullet.radius < \
                                        kvitkaMiddle.hitbox[
                                            0] + kvitkaMiddle.hitbox[2]:
                                bullets.pop(bullets.index(bullet))
                                scorenumber += 1
                                explosion.play()
                                kvitkaMiddle.hit()

            if whichKvitochka == 3:
                if kvitkaBottom.visible == True:
                    if bullet.y - bullet.radius < kvitkaBottom.hitbox[1] +
kvitkaBottom.hitbox[
                            3] and bullet.y + bullet.radius > \
                                kvitkaBottom.hitbox[1]:
                            if bullet.x + bullet.radius > kvitkaBottom.hitbox[0] and
bullet.x - bullet.radius < \
                                        kvitkaBottom.hitbox[
                                            0] + kvitkaBottom.hitbox[2]:
                                bullets.pop(bullets.index(bullet))
                                scorenumber += 1
                                explosion.play()
                                kvitkaBottom.hit()

            if bullet.x < 1000 and bullet.x > 0:
                bullet.x += bullet.vel
            else:
                bullets.pop(bullets.index(bullet))


        keys = pygame.key.get_pressed()

        if keys[pygame.K_SPACE]:
            if len(bullets) < 1:
```

```
                bullets.append(
                    projectile(round(bee.x + bee.width + 10 // 2),
                               round(bee.y + bee.height // 2), 7, (117, 237,
255)))  # creates bullet
                bulletsound.play()

        if keys[pygame.K_UP] and bee.y > 50:
            bee.y -= bee.vel
        elif keys[pygame.K_DOWN] and bee.y < 400:
            bee.y += bee.vel

        redrawGameWindow(number1, number2, scorenumber, scorequestionVisible,
level)  # викликає функцію перемальовування


menu()
pygame.quit()
```

1.2 Реалізація pygame_textinput.py

```
import os.path

import pygame
import pygame.locals as pl

pygame.font.init()


class TextInput:

    def __init__(
            self,
            initial_string="",
            font_family="ARCADECLASSIC.TTF",
            font_size=100,
            antialias=True,
            text_color=(0, 0, 0),
            cursor_color=(0, 0, 1),
            repeat_keys_initial_ms=400,
            repeat_keys_interval_ms=35,
            max_string_length=-1,
            password=False):



        self.antialias = antialias
        self.text_color = text_color
        self.font_size = font_size
        self.max_string_length = max_string_length
        self.password = password
        self.input_string = initial_string

        if not os.path.isfile(font_family):
            font_family = pygame.font.match_font(font_family)

        self.font_object = pygame.font.Font(font_family, font_size)


        self.surface = pygame.Surface((1, 1))
        self.surface.set_alpha(0)


        self.keyrepeat_counters = {}
        self.keyrepeat_intial_interval_ms = repeat_keys_initial_ms
        self.keyrepeat_interval_ms = repeat_keys_interval_ms


        self.cursor_surface = pygame.Surface((int(self.font_size / 20 + 1),
self.font_size))
        self.cursor_surface.fill(cursor_color)
```

```
        self.cursor_position = len(initial_string)
        self.cursor_visible = True
        self.cursor_switch_ms = 500  # /|\
        self.cursor_ms_counter = 0

        self.clock = pygame.time.Clock()

    def update(self, events):
        for event in events:
            if event.type == pygame.KEYDOWN:
                self.cursor_visible = True


                if event.key not in self.keyrepeat_counters:
                    if not event.key == pl.K_RETURN:
                        self.keyrepeat_counters[event.key] = [0, event.unicode]

                if event.key == pl.K_BACKSPACE:
                    self.input_string = (
                        self.input_string[:max(self.cursor_position - 1, 0)]
                        + self.input_string[self.cursor_position:]
                    )


                    self.cursor_position = max(self.cursor_position - 1, 0)
                elif event.key == pl.K_DELETE:
                    self.input_string = (
                        self.input_string[:self.cursor_position]
                        + self.input_string[self.cursor_position + 1:]
                    )

                elif event.key == pl.K_RETURN:
                    return True

                elif event.key == pl.K_RIGHT:

                    self.cursor_position = min(self.cursor_position + 1,
len(self.input_string))

                elif event.key == pl.K_LEFT:

                    self.cursor_position = max(self.cursor_position - 1, 0)

                elif event.key == pl.K_END:
                    self.cursor_position = len(self.input_string)

                elif event.key == pl.K_HOME:
                    self.cursor_position = 0

                elif len(self.input_string) < self.max_string_length or
self.max_string_length == -1:
```

```
                    self.input_string = (
                        self.input_string[:self.cursor_position]
                        + event.unicode
                        + self.input_string[self.cursor_position:]
                    )
                    self.cursor_position += len(event.unicode)

            elif event.type == pl.KEYUP:

                if event.key in self.keyrepeat_counters:
                    del self.keyrepeat_counters[event.key]


        for key in self.keyrepeat_counters:
            self.keyrepeat_counters[key][0] += self.clock.get_time()


            if self.keyrepeat_counters[key][0] >=
self.keyrepeat_intial_interval_ms:
                self.keyrepeat_counters[key][0] = (
                    self.keyrepeat_intial_interval_ms
                    - self.keyrepeat_interval_ms
                )

                event_key, event_unicode = key, self.keyrepeat_counters[key][1]
                pygame.event.post(pygame.event.Event(pl.KEYDOWN, key=event_key,
unicode=event_unicode))


        string = self.input_string
        if self.password:
            string = "*" * len(self.input_string)
        self.surface = self.font_object.render(string, self.antialias,
self.text_color)


        self.cursor_ms_counter += self.clock.get_time()
        if self.cursor_ms_counter >= self.cursor_switch_ms:
            self.cursor_ms_counter %= self.cursor_switch_ms
            self.cursor_visible = not self.cursor_visible

        if self.cursor_visible:
            cursor_y_pos =
self.font_object.size(self.input_string[:self.cursor_position])[0]

            if self.cursor_position > 0:
                cursor_y_pos -= self.cursor_surface.get_width()
            self.surface.blit(self.cursor_surface, (cursor_y_pos, 0))

        self.clock.tick()
        return False
```

```
    def get_surface(self):
        return self.surface

    def get_text(self):
        return self.input_string

    def get_cursor_position(self):
        return self.cursor_position

    def set_text_color(self, color):
        self.text_color = color

    def set_cursor_color(self, color):
        self.cursor_surface.fill(color)

    def clear_text(self):
        self.input_string = ""
        self.cursor_position = 0



if __name__ == "__main__":
    pygame.init()


    textinput = TextInput()

    screen = pygame.display.set_mode((1000, 200))
    clock = pygame.time.Clock()

    while True:
        screen.fill((225, 225, 225))

        events = pygame.event.get()
        for event in events:
            if event.type == pygame.QUIT:
                exit()


        textinput.update(events)

        screen.blit(textinput.get_surface(), (10, 10))

        pygame.display.update()
        clock.tick(30)
```