СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки

Кафедра інформаційних технологій та програмування

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної випускної роботи

освітній ступінь бакалавр
спеціальність 121 „Інженерія програмного забезпечення”
(шифр і назва спеціальності)

на тему „Гра для вивчення автоматизації та програмування”

Виконав: студент групи ІПЗ-20д _____ (підпис) І.О. Бандурко (ініціали і прізвище)

Керівник _____ (підпис) В.О. Лифар (ініціали і прізвище)

Завідувач кафедри _____ (підпис) О.І. Захожай (ініціали і прізвище)

Рецензент___ О.І. Захожай_____

Київ – 2024

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра інформаційних технологій та програмування
Освітній ступінь бакалавр
спеціальність 121 „Інженерія програмного забезпечення”
(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Захожай О.І.
“___” _____ 2024 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ ВИПУСКНУ РОБОТУ СТУДЕНТУ

Бандурко Ілля Олексійович
(прізвище, ім’я, по батькові)

1. Тема роботи: Гра для вивчення автоматизації та програмування

керівник роботи д.т.н., доцент В.О. Лифар
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджений наказом університету від “ 06 ” травня 2024 року №171/15.15-С

2. Строк подання студентом роботи 08.06.2024р.
3. Вихідні дані до роботи: Об'єктом даної роботи є процес розробки гри для вивчення автоматизації та програмування

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):) аналіз предметної області; вибір програмних засобів для розробки продукту; розробка продукту;

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслеників)

6. Консультанти розділів роботи

		Підпис, дата

Розділ	Прізвище, ініціали та посада консультанта	завдання видав	завдання прийняв

7. Дата видачі завдання 30.03.2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання кваліфікаційної випускної роботи	Строк виконання етапів	Примітка
1	Одержання завдання на виконання роботи	30.03.24	виконано
2	Укладання і погодження з керівником плану і етапів виконання роботи	06.04.24	виконано
3	Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної галузі»	13.04.24	виконано
4	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху	20.04.24	виконано
5	Укладання та тестування програмного продукту	27.04.24	виконано
6	Укладання, оформлення та погодження пояснювальної записки з керівником	04.05.24	виконано
7	Здача готової пояснювальної записки на кафедру	08.06.24	виконано
8	Укладання доповіді і презентації	10.06.24	виконано

Студент _____ І.О. Бандурко
підпис (ініціали і прізвище)

Керівник роботи _____ В.О. Лифар
підпис (ініціали і прізвище)

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ

дипломної роботи студента гр. ІПЗ-20д Бандурко І.О.

Науковий керівник:

Професор, д.т.н. _____ Лифар В.О.

Оцінка наукового керівника: _____

Рецензент _____

ПІБ, місто роботи, посада

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту:

_____

Голова ЕК

Професор кафедри ІТП

д.т.н. _____ Меняйленко О.С.

підпис

РЕФЕРАТ

Дипломна робота містить: 32 сторінок основного тексту, 14 сторінок додатків, 14 рисунків, 10 використаних джерел.

Метою випускної кваліфікаційної роботи є розробка гри для вивчення автоматизації та програмування, оптимізація методів її реалізації та підбір ефективних інструментів.

Проаналізовано існуючі навчальні ігри, їхні принципи роботи та взаємодія з користувачами. Особливу увагу приділено підбору продуктивного ігрового рушія та інструментів для проекту.

Результатом роботи є гра, що дозволяє користувачам вивчати основи автоматизації та програмування без попередніх навичок. Гра має модульну архітектуру на базі шаблону MVC, що забезпечує легке масштабування та розширення функціональності.

Система реалізована згідно з вимогами технічного завдання. Документ містить детальний опис процесу розробки та демонстрацію роботи готового продукту.

ЗМІСТ

ВСТУП

У сучасному світі автоматизація та програмування займають важливе місце в різних галузях життя, від побутових пристроїв до складних промислових систем. З розвитком технологій зростає потреба у фахівцях, які здатні створювати й управляти автоматизованими системами. Одним із ефективних способів навчання основам автоматизації та програмування є використання інтерактивних ігор, які дозволяють учням засвоювати складні концепції в доступній і зрозумілій формі.

Актуальність теми

Актуальність теми дипломної роботи полягає в тому, що навчальні ігри з елементами програмування сприяють розвитку логічного мислення, творчих здібностей і навичок розв'язання проблем у студентів. Існуючі рішення, такі як Scratch, вже довели свою ефективність, але завжди є можливість удосконалення та створення нових підходів, що ще більше стимулюватимуть інтерес до навчання та сприятимуть глибшому розумінню принципів автоматизації та програмування.

Мета дослідження

Метою цієї дипломної роботи є розробка інтерактивної гри для вивчення основ автоматизації та програмування, яка буде зрозумілою та доступною для широкої аудиторії. Програма має надати користувачам можливість управляти роботом, створювати алгоритми шляхом перетягування команд на ігрове поле та таким чином засвоювати базові принципи програмування і логіки.

Об'єкт дослідження

Об'єктом дослідження є процес навчання автоматизації та програмування за допомогою інтерактивних ігор. Зокрема, увага приділяється тому, як інтерактивні ігри можуть сприяти кращому засвоєнню матеріалу та розвивати навички у школярів.

Суб'єкт дослідження

Суб'єктом дослідження є студенти та учні, які навчаються основам програмування та автоматизації за допомогою інтерактивних освітніх ігор.

Особливий акцент робиться на молодшій аудиторії, для якої важливо створити сприятливі умови для розвитку інтересу до технологій і програмування.

Важливо відзначити, що інтерактивні ігри, які поєднують у собі навчання та розваги, здатні значно підвищити мотивацію до навчання. Такі ігри не тільки роблять навчальний процес більш захоплюючим, але й допомагають учням краще запам'ятовувати матеріал через активну взаємодію з ним. Крім того, використання ігрових елементів у навчанні сприяє розвитку критичного мислення, креативності та навичок співпраці, що є надзвичайно важливими у сучасному світі.

У наш час багато успішних програмістів починали свій шлях саме з ігор та інтерактивних програм, які пробуджували в них інтерес до кодування та технологій. Створення нової гри, яка допоможе розвиватися в цьому напрямку, є не лише корисним, але й надзвичайно захоплюючим завданням.

РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД

У даній частині роботи розглядається програмне забезпечення, яке використовується в проекті. Аналізується кожна програма, її переваги та недоліки, а також пояснюється вибір найкращої з них для даного проекту. Метою є показати, як вибір програмного забезпечення впливає на ефективність розробки.

1.1. Операційна система

Windows — популярна операційна система, яка використовується для розробки програмного забезпечення завдяки широкій підтримці різноманітних інструментів та середовищ розробки. Windows забезпечує зручний інтерфейс користувача, підтримку широкого спектра апаратного забезпечення, а також можливість запуску великої кількості програм, включаючи інструменти розробки як-от Visual Studio. Недоліками Windows є вартість ліцензії, потреба в регулярних оновленнях, що можуть впливати на стабільність роботи, а також можливі проблеми з безпекою.

Linux — операційна система з відкритим вихідним кодом, що пропонує високу гнучкість і налаштовуваність. Вона є безкоштовною і широко використовується для серверів та розробки програмного забезпечення завдяки стабільності та безпеці. Linux підтримує багато інструментів для розробки, таких як GCC, CMake, і IDE, як-от Eclipse. Недоліки включають складність для новачків, меншу підтримку деяких комерційних програм та можливі проблеми з драйверами для певного апаратного забезпечення.

macOS — операційна система від Apple, відома своєю стабільністю, безпекою та оптимізованим інтерфейсом користувача. macOS має інтеграцію з екосистемою Apple і забезпечує доступ до інструментів розробки, як-от Xcode, що робить її популярною серед розробників мобільних додатків для iOS. Основні недоліки macOS включають високу вартість апаратного забезпечення Apple, обмежену можливість налаштування системи та обмежену підтримку програмного забезпечення, яке не розроблено для екосистеми Apple.

Android — мобільна операційна система, розроблена Google, базується на ядрі Linux і орієнтована на мобільні пристрої. Android використовується для

розробки мобільних додатків і забезпечує доступ до Google Play Market. Розробка під Android здійснюється через Android Studio та Java/Kotlin. Переваги включають широкий ринок користувачів і велику кількість бібліотек для розробки. Основні недоліки — фрагментація платформи, що може ускладнювати тестування додатків, і менша стабільність порівняно з іншими платформами.

Висновок: Windows обрана для даного проекту завдяки її сумісності з різноманітними розробницькими інструментами та широкому спектру підтримуваного апаратного забезпечення. Незважаючи на недоліки, такі як вартість ліцензії та потреба в оновленнях, Windows забезпечує необхідну функціональність і зручність для розробки. Альтернативи, такі як Linux, macOS, і Android, мають свої переваги, проте Windows пропонує найкращий баланс між функціональністю, підтримкою інструментів та сумісністю з існуючим апаратним забезпеченням для невеликого навчального проекту.

1.2 Браузер

Google Chrome використовується для цього проекту через його високу швидкість роботи, широкий набір функцій та стабільність. Chrome підтримує велику кількість розширень, що розширюють його функціональність. Розробка та тестування веб-додатків на Chrome є зручними завдяки його потужним інструментам для розробників. Програма має часті оновлення безпеки, що робить її надійною.

Microsoft Edge також є швидким і стабільним браузером, який базується на тому ж рушії Chromium, що й Chrome. Він має вбудовані функції для підвищення продуктивності та безпеки, такі як блокування трекерів і інтеграція з Windows. Edge підтримує розширення з Chrome Web Store, що робить його майже настільки ж функціональним, як і Chrome. Однак, на відміну від Chrome, Edge може мати проблеми сумісності з деякими специфічними розширеннями та веб-додатками.

Opera пропонує унікальні функції, такі як вбудований VPN, блокування реклами і режим економії енергії. Opera також базується на Chromium, тому багато розширень для Chrome сумісні з ним. Незважаючи на це, Opera менш

популярний серед розробників, що може спричинити труднощі з тестуванням та оптимізацією додатків. Деякі користувачі відзначають менш стабільну роботу порівняно з Chrome та Edge.

Обрана програма, Google Chrome, хоча і не ідеальна, викликає менше проблем та трудозатрат через свою високу популярність серед розробників, багатий набір функцій та стабільність. Це робить його найбільш підходящим вибором для цього проекту.

1.3. Мова програмування

Python відомий своєю простотою та читабельністю коду, що робить його ідеальним для освітніх проектів. Він має багату бібліотеку модулів і пакетів, таких як PyQt5 для створення графічних інтерфейсів та Pygame для ігор. Python підтримує різні парадигми програмування, включаючи об'єктно-орієнтоване та функціональне програмування. Однак, його виконання може бути повільнішим у порівнянні з компільованими мовами.

Java - об'єктно-орієнтована мова програмування, яка відома своєю платформною незалежністю завдяки JVM (Java Virtual Machine). Вона має потужні інструменти для розробки, такі як IntelliJ IDEA, і добре підходить для великих проектів завдяки своїй надійності та масштабованості. Однак, Java може бути складнішою для початківців через більш громіздкий синтаксис та необхідність управління пам'яттю.

C++ - потужна мова програмування з високою продуктивністю. Вона дозволяє отримати детальний контроль над апаратним забезпеченням та використовується для розробки ігор і системного програмного забезпечення. C++ має складний синтаксис та потребує глибоких знань управління пам'яттю, що може ускладнити навчання. Вона менш підходить для швидкої розробки та прототипування в порівнянні з Python.

C# - мова програмування, розроблена Microsoft, відома своєю інтеграцією з .NET Framework. Вона підтримує сучасні парадигми програмування та має інструменти, такі як Visual Studio, що спрощують розробку. C# забезпечує високу продуктивність та безпеку коду, але вона

здебільшого орієнтована на Windows-платформу, що може бути обмеженням для крос-платформених проектів.

JavaScript - мова програмування, яка широко використовується для розробки веб-додатків. Вона підтримує інтерактивність і динамічність веб-сторінок та має потужні бібліотеки та фреймворки, такі як React і Node.js. JavaScript добре підходить для фронтенд-розробки, але менш придатний для розробки ігор та настільних додатків порівняно з Python. Синхронне програмування може бути складним для початківців.

Висновок: Python був обраний для цього проекту через його простоту, читабельність та багатий набір бібліотек, які ідеально підходять для навчального проекту. Незважаючи на можливі проблеми з продуктивністю, Python дозволяє швидко прототипувати та розробляти функціонал гри, що зменшує трудозатрати та спрощує навчання автоматизації та програмування.

1.4. Середа програмування

PyCharm — це середовище розробки (IDE) для мови програмування Python. PyCharm підтримує синтаксичне підсвічування, автозавершення коду, налагодження, рефакторинг та інтеграцію з системами контролю версій. PyCharm забезпечує вбудовану підтримку тестування, полегшує налаштування середовища розробки за допомогою віртуальних середовищ та має зручний графічний інтерфейс. Серед недоліків можна відзначити високу вимогливість до ресурсів комп'ютера, що може уповільнити роботу на менш потужних машинах.

Atom — це текстовий редактор з відкритим кодом, створений GitHub. Atom підтримує встановлення пакетів, які додають функціональність для різних мов програмування, включаючи Python. Atom надає гнучкість налаштувань, синтаксичне підсвічування та автозавершення коду, а також інтеграцію з системами контролю версій. Недоліками є відносно повільна робота з великими файлами та обмежена функціональність порівняно з повноцінними IDE.

Visual Studio Code (VSCode) — це редактор коду, який підтримує численні розширення для різних мов програмування. VSCode надає

можливості автозавершення коду, налагодження, інтеграції з системами контролю версій, а також вбудованого терміналу. VSCode менш вимогливий до ресурсів комп'ютера та пропонує широку спільноту розробників для підтримки. Проте, деякі функції можуть вимагати встановлення додаткових розширень, що потребує додаткового часу на налаштування.

PyCharm обрано для цього проекту завдяки його глибокій інтеграції з Python, потужному налагодженню та підтримці тестування. Це робить PyCharm оптимальним вибором для розробки освітньої гри з використанням Python. Незважаючи на високі вимоги до ресурсів, його можливості спрощують процес розробки і зменшують кількість можливих проблем, що можуть виникнути при використанні менш інтегрованих рішень.

1.5. Модулі меню

PyQt5 — це модуль для створення графічного інтерфейсу користувача (GUI) на Python, що використовує бібліотеку Qt. PyQt5 підтримує багатий набір віджетів, дозволяє створювати складні інтерфейси з інтерактивними елементами та підтримує кросплатформенність. PyQt5 забезпечує потужні інструменти для розробки, включаючи інтеграцію з Qt Designer для візуального проектування інтерфейсу. Серед недоліків — складність у налаштуванні та вивченні через велику кількість функцій і специфічних деталей.

Tkinter — це стандартна бібліотека для створення GUI в Python, яка надає простий інтерфейс для створення базових віконних додатків. Tkinter підтримує основні віджети, такі як кнопки, мітки та поля введення, і дозволяє швидко створювати прості інтерфейси. Tkinter легко освоїти та налаштувати, однак його можливості обмежені порівняно з більш просунутими бібліотеками. Дизайн інтерфейсу в Tkinter може виглядати застарілим і не настільки привабливим.

wxPython — це бібліотека для створення кросплатформенних GUI-додатків, яка використовує нативні елементи управління. wxPython надає

широкий набір віджетів і дозволяє створювати інтерфейси, які виглядають природно на різних операційних системах. wxPython забезпечує високу продуктивність і підтримку складних інтерфейсів. Однак, wxPython складніше в освоєнні та має менш активну спільноту, що може ускладнити пошук допомоги та ресурсів.

Висновки: PyQt5 обрано для цього проекту через його потужні можливості та багатий набір віджетів, що дозволяють створювати складні та інтуїтивні інтерфейси для навчальної гри. Хоча PyQt5 має круту криву навчання, його функціональність і підтримка візуального проектування роблять його найкращим вибором для розробки GUI в цьому проекті, мінімізуючи потенційні проблеми та трудозатрати порівняно з іншими варіантами.

1.6 Модулі для створення ігор

Pygame — це бібліотека для створення 2D-ігор на Python. Вона забезпечує доступ до графіки, звуку, контролерів та інших компонентів, необхідних для розробки ігор. Pygame підтримує широкий спектр платформ і має простий синтаксис, що полегшує навчання і розробку. Серед недоліків можна відзначити обмежені можливості для 3D-графіки та відсутність деяких сучасних функцій, які присутні в більш нових бібліотеках.

Arcade — це сучасна бібліотека для розробки 2D-ігор на Python, яка надає потужні інструменти для роботи з графікою та звуком. Arcade підтримує сучасні графічні можливості, такі як використання OpenGL, і має простий та інтуїтивний інтерфейс. Вона легка в освоєнні та ідеально підходить для новачків і освітніх проектів. Проте, спільнота розробників менш активна порівняно з Pygame, що може ускладнити пошук допомоги та ресурсів.

Cocos2d — це бібліотека для розробки 2D-ігор, яка підтримує різні мови програмування, включаючи Python. Cocos2d надає потужні можливості для анімації, фізики та інших аспектів ігрової розробки. Вона забезпечує високу продуктивність і кросплатформенність, дозволяючи створювати складні ігри. Недоліком є висока складність освоєння і налаштування, а також необхідність вивчення специфічних концепцій та структур.

Ren'Py — це бібліотека, яка спеціалізується на створенні візуальних новел і інтерактивних історій. Ren'Py забезпечує простий інтерфейс для створення текстових і графічних елементів, підтримує анімацію, звук та інтерактивність. Вона ідеально підходить для розробки сюжетних ігор з акцентом на текст. Однак, Ren'Py не призначена для розробки загальних 2D-ігор, що обмежує її використання в інших жанрах ігрової розробки.

Pygame обрано для цього проекту через його широкі можливості для розробки 2D-ігор, простоту використання та активну спільноту. Це дозволяє швидко почати розробку та забезпечує доступ до великої кількості ресурсів і прикладів. Незважаючи на певні обмеження у сучасних графічних можливостях, Pygame є оптимальним вибором для освітнього проекту, оскільки мінімізує потенційні проблеми та трудозатрати, забезпечуючи при цьому необхідну функціональність для створення навчальної гри.

1.7 Модуль часу

Модуль time в Python забезпечує основні функції для роботи з часом. Він надає можливість вимірювати час, форматувати часові дані, виконувати затримки та отримувати поточний час. Основні функції модуля включають:

- time(): повертає поточний час в секундах з початку епохи (1 січня 1970 року).
- sleep(): затримує виконання програми на заданий інтервал часу.
- strftime(): форматування часових даних у рядковий формат.
- localtime(): перетворює час у місцевий час.
- gmtime(): повертає час у форматі UTC.

Модуль time дозволяє точно вимірювати продуктивність коду, синхронізувати процеси та керувати затримками в програмі, що є важливим для розробки програмного забезпечення, яке залежить від часових інтервалів або часових міток.

РОЗДІЛ 2: МОДЕЛЬ ПРОЕКТУ ТА ПОСТАНОВКА ЗАДАЧ

Вибір моделі розробки програмного забезпечення може мати вирішальне значення для складності виконання проекту та його успішності. Правильна модель може забезпечити ефективний процес розробки, гнучкість у відповідь на змінні вимоги та високу якість кінцевого продукту.

Каскадна модель

Каскадна модель, також відома як модель водоспаду, передбачає лінійний підхід до розробки програмного забезпечення, де кожен етап (вимоги, дизайн, реалізація, тестування, розгортання та підтримка) виконується послідовно. Ця модель має чітку структуру, де кожен етап має конкретні результати, що дозволяє легко відстежувати прогрес.

Каскадна модель забезпечує простоту управління проектом, оскільки кожен етап повинен бути завершений перед початком наступного. Це дозволяє уникнути непередбачуваних змін у процесі розробки. Проте, ця модель має недоліки у випадках, коли вимоги до проекту можуть змінюватися, оскільки такі зміни можуть вимагати повернення до попередніх етапів, що ускладнює процес.

Спіральна модель

Спіральна модель комбінує елементи дизайну та прототипування у повторюваному процесі, що дозволяє оцінювати ризики на кожному етапі. Ця модель включає кілька ітерацій (спіралей), кожна з яких охоплює планування, аналіз ризиків, реалізацію та оцінку.

Спіральна модель дозволяє враховувати нові вимоги та коригувати проект на кожному етапі розробки. Такий підхід забезпечує гнучкість і можливість адаптації до змін, що робить цю модель придатною для великих і складних проектів. Водночас, цей підхід є досить ресурсозатратним і складним для невеликих проектів, що може бути недоліком для навчальних цілей.

Інкрементна модель

Інкрементна модель передбачає розробку програмного забезпечення у вигляді серії інкрементів або часткових випусків. Кожен інкремент додає нову функціональність до попередньої версії програмного забезпечення. Це

дозволяє отримувати зворотній зв'язок на ранніх стадіях розробки та вносити зміни в наступні інкременти.

Інкрементна модель забезпечує гнучкість у процесі розробки, оскільки кожен інкремент є самостійним етапом з тестуванням і перевіркою. Це знижує ризики та дозволяє швидко реагувати на зміни вимог. Такий підхід є особливо корисним для невеликих проектів, де важлива можливість поступового додавання нових функціональностей.

Agile

Agile є методологією, яка ставить акцент на гнучкості, швидкому випуску робочого програмного забезпечення та активній взаємодії з користувачами. Вона включає в себе різні фреймворки, такі як Scrum, Kanban та інші, що дозволяє команді адаптуватися до змінних вимог та забезпечує часті випуски.

Основні принципи Agile включають інтенсивну співпрацю між розробниками та користувачами, гнучке реагування на зміни вимог, часті випуски програмного забезпечення та акцент на робочий продукт. Ця методологія сприяє підвищенню якості програмного забезпечення та задоволенню потреб користувачів.

Agile забезпечує високу ефективність команди та можливість швидко адаптуватися до нових вимог, що робить її придатною для різноманітних проектів, включаючи невеликі навчальні проекти. Часті випуски дозволяють отримувати зворотній зв'язок та покращувати продукт на кожному етапі розробки.

Використання моделей у навчальних проектах

При розгляді моделей розробки програмного забезпечення для невеликих навчальних проектів важливо враховувати специфічні потреби та обмеження студентів. Каскадна модель може забезпечити чітку структуру для новачків, проте її жорсткість може бути недоліком. Спіральна модель забезпечує гнучкість, але її складність може бути зайвою для невеликих проектів.

Інкрементна модель та Agile надають найбільшу гнучкість та можливість адаптації до змінних вимог, що є важливим фактором для навчальних проектів. Ці моделі дозволяють отримувати зворотній зв'язок на ранніх стадіях та поступово вдосконалювати продукт, що сприяє більш ефективному навчанню та кращому розумінню процесу розробки програмного забезпечення.

Саме тому, враховуючи всі аспекти та особливості різних моделей розробки програмного забезпечення, найоптимальнішим вибором для невеликого студентського проекту є Інкрементна модель. Цей підхід дозволяє починати з базового функціоналу та поступово розширювати його, забезпечуючи гнучкість у процесі розробки. Інкрементна модель надає можливість швидко реагувати на змінні вимоги та отримувати зворотній зв'язок на кожному етапі. Це сприяє зниженню ризиків і забезпечує поступове вдосконалення продукту. Також важливо відзначити, що така модель підходить для навчальних цілей, оскільки дозволяє краще розуміти та контролювати процес розробки програмного забезпечення, а також сприяє розвитку навичок роботи над реальними проектами. Саме тому для цього проекту було обрано Інкрементну модель.

План Розробки на Основі Інкрементної Моделі

Інкрементна модель передбачає поетапну розробку програмного забезпечення, де кожен етап додає нову функціональність до попередньої версії. Це дозволяє отримувати зворотний зв'язок на ранніх стадіях розробки та вносити зміни в наступні інкременти. Нижче наведено детальний план розробки для проекту "Гра для вивчення автоматизації та програмування" на основі інкрементної моделі.

Етап 1: Початковий Аналіз та Планування

- Визначення цілей проекту
- Встановлення вимог до системи
- Аналіз цільової аудиторії
- Розробка загальної архітектури системи
- Оцінка необхідних ресурсів та часу на розробку

Етап 2: Розробка Базового Функціоналу

- Створення основного інтерфейсу користувача за допомогою PyQt5
- Реалізація основних кнопок управління (рух вперед, повороти)
- Впровадження можливості перетягування кнопок для створення алгоритмів
- Інтеграція базових елементів ігрової сцени за допомогою Pygame
- Тестування базового функціоналу

Етап 3: Розширення Функціоналу

- Додавання циклів для створення повторюваних алгоритмів
- Реалізація системи збереження та завантаження алгоритмів
- Поліпшення інтерфейсу користувача для зручності роботи
- Впровадження системи підказок та навчальних матеріалів
- Тестування нових функцій та їх інтеграція з базовим функціоналом

Етап 4: Інтеграція Рівнів та Модулів Навчання

- Розробка меню вибору рівнів за допомогою PyQt5
- Створення декількох навчальних рівнів з різними завданнями
- Впровадження системи оцінки та прогресу користувача
- Тестування та відладка рівнів

Етап 5: Оптимізація та Поліпшення

- Оптимізація коду для підвищення продуктивності
- Поліпшення графічного інтерфейсу та дизайну
- Впровадження системи зворотного зв'язку від користувачів
- Розширення функціоналу з урахуванням зворотного зв'язку
- Проведення кінцевого тестування та відладки

Потенційні Завдання

- Розробка алгоритмів для складніших дій робота
- Інтеграція додаткових модулів для вивчення інших аспектів програмування
- Впровадження багатомовної підтримки
- Розробка системи досягнень та мотивації для користувачів

Перспективи Інтеграції Нових Можливостей

- Додавання підтримки різних мов програмування

- Розробка мобільної версії програми
- Впровадження штучного інтелекту для аналізу алгоритмів та надання рекомендацій
- Розширення функціоналу для підтримки складніших алгоритмів та проектів

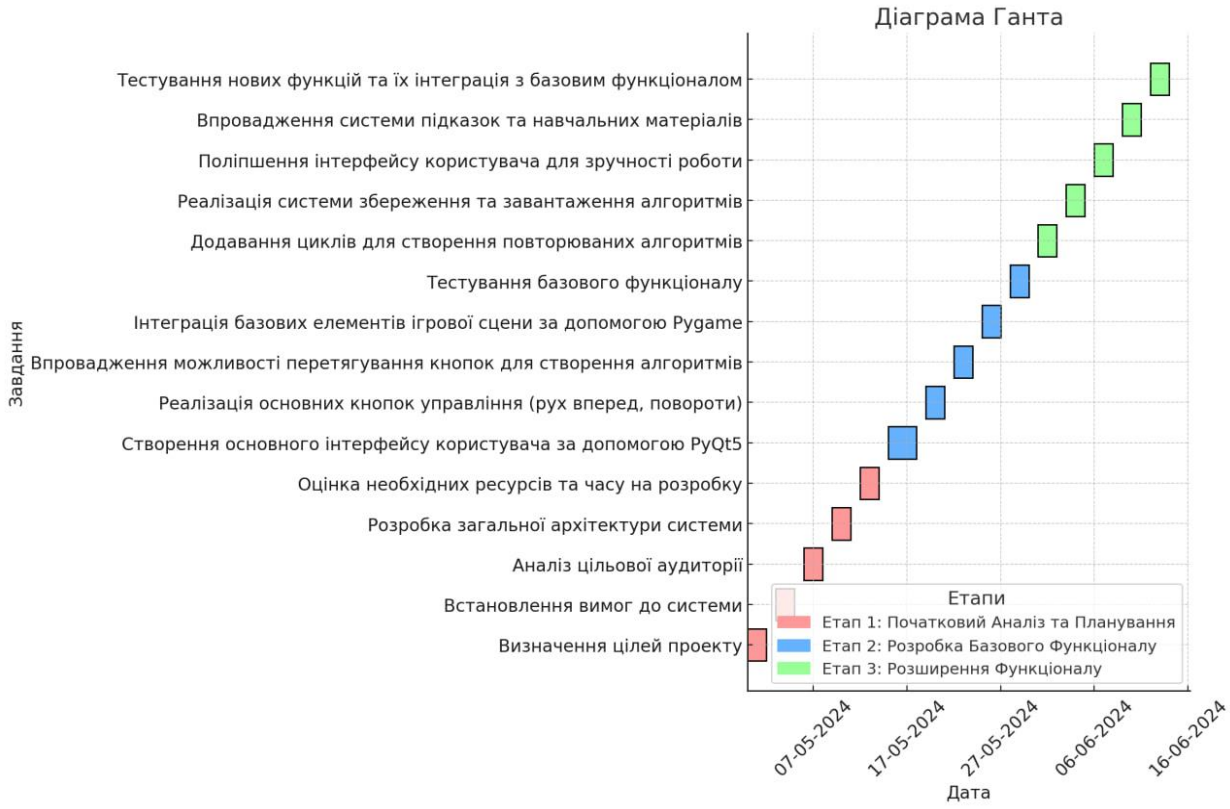

Рис. 2.1 діаграма Ганта

Етап 1: Початковий Аналіз та Планування

Етап початкового аналізу та планування є критично важливим для успішного старту проекту. Він включає в себе кілька ключових аспектів: визначення цілей проекту, встановлення вимог до системи, аналіз цільової аудиторії, розробку загальної архітектури системи та оцінку необхідних ресурсів і часу на розробку. Початок проекту заплановано на 15 квітня, а завершення основних робіт – на 13 червня.

Визначення цілей проекту

Цілі проекту визначаються його основною метою – створенням навчальної гри для початківців, яка допоможе їм вивчити основи автоматизації та програмування в цікавій та інтерактивній формі. Це передбачає забезпечення користувачів інтуїтивно зрозумілим інтерфейсом, де вони

можуть створювати алгоритми для управління роботом. Проект орієнтований на те, щоб навчання було не лише ефективним, але й захоплюючим, стимулюючи інтерес до програмування через гейміфікацію процесу навчання.

Завдання проекту включають:

- Створення інтуїтивно зрозумілого інтерфейсу користувача, який дозволить легко взаємодіяти з грою без попередніх знань у програмуванні.
- Забезпечення інтерактивного навчання через ігрові рівні та завдання, які поступово ускладнюються.
- Надання можливості користувачам візуально створювати алгоритми за допомогою простих блоків команд.
- Стимулювання інтересу до програмування та автоматизації за допомогою гейміфікації.

Встановлення вимог до системи

На цьому етапі важливо чітко визначити вимоги до системи, які поділяються на функціональні та нефункціональні.

Функціональні вимоги включають розробку основного меню користувача за допомогою PyQt5, реалізацію кнопок управління (рух вперед, повороти, цикли), можливість перетягування та компонування кнопок для створення алгоритмів, інтеграцію ігрової сцени за допомогою Pygame, інтеграцію навчальних матеріалів та підказок для користувачів.

Нефункціональні вимоги включають забезпечення стабільності та надійності системи, зручність у користуванні інтерфейсом, високу продуктивність та швидкість роботи, а також легкість у підтримці та оновленні.

Аналіз цільової аудиторії

Цільова аудиторія проекту – це початківці в програмуванні та автоматизації, зокрема студенти та школярі, які роблять перші кроки у цій сфері. Для цієї групи важливо забезпечити навчання у легкодоступній та зрозумілій формі. Аналіз цільової аудиторії дозволяє зрозуміти їхні потреби та очікування.

Початківці часто не мають попереднього досвіду у програмуванні, тому важливо створити інтерфейс, який буде зрозумілим на інтуїтивному рівні.

Вони також очікують, що навчання буде цікавим та захоплюючим, тому інтерактивні елементи гри, які дозволяють експериментувати з різними алгоритмами, є ключовими.

Цільова аудиторія потребує інструментів, які допоможуть їм зрозуміти основні концепції програмування без необхідності занурюватися у складну теорію. Це досягається через гейміфікацію навчального процесу, де користувачі можуть одразу бачити результати своїх дій і отримувати зворотний зв'язок.

Розробка загальної архітектури системи

Загальна архітектура системи визначається на основі вимог та аналізу цільової аудиторії. Архітектура включає наступні компоненти:

Інтерфейс користувача (UI) розроблений за допомогою PyQt5 і включає основне меню.

Ігрова сцена, панель управління, область створення алгоритмів та інші елементи, необхідні для взаємодії користувача з грою створені за допомогою Pygame. Вона є ключовим компонентом, де відображається робот і виконуються алгоритми, створені користувачами. Ігрова сцена повинна бути інтерактивною і надавати можливість легко відстежувати виконання команд робота. Важливо забезпечити, щоб інтерфейс був інтуїтивно зрозумілим і зручним у використанні.

Логіка управління відповідає за обробку команд користувача та виконання алгоритмів. Це включає інтерпретацію блоків команд, створених у інтерфейсі, і перетворення їх у дії робота на ігровій сцені.

Система збереження даних забезпечує збереження створених алгоритмів та прогресу користувача. Це дозволяє користувачам зберігати свою роботу і повертатися до неї у будь-який час.

Навчальні модулі включають інтегровані підказки та матеріали для навчання. Вони допомагають користувачам зрозуміти основні концепції програмування та автоматизації через практичні завдання та приклади.

Оцінка необхідних ресурсів та часу на розробку

Для успішної реалізації проекту необхідно оцінити ресурси та час, які знадобляться на кожному етапі розробки. Основні ресурси включають програмні ресурси, такі як PyQt5 для меню, Pygame для ігрової сцени та PyCharm для написання коду.

Людські ресурси передбачають одного розробника, який буде виконувати всі етапи розробки. Часовий план проекту передбачає початок робіт 30 квітня і завершення основних робіт 14 червня, з чітким розподілом часу на кожен етап розробки, що дозволить досягти запланованих результатів у встановлений термін.

Етап 2: Розробка Базового Функціоналу

Створення основного інтерфейсу користувача за допомогою PyQt5

На першому етапі розробки базового функціоналу було створено інтерфейс користувача (UI) за допомогою бібліотеки PyQt5. На скріншотах видно основне меню гри, де користувач може обрати рівні, змінювати мову та тестувати функціонал. Це меню є ключовою частиною UI, яке забезпечує користувачу можливість взаємодії з грою, вибираючи рівні для подальшого проходження. Приклад коду у додатках А - Д

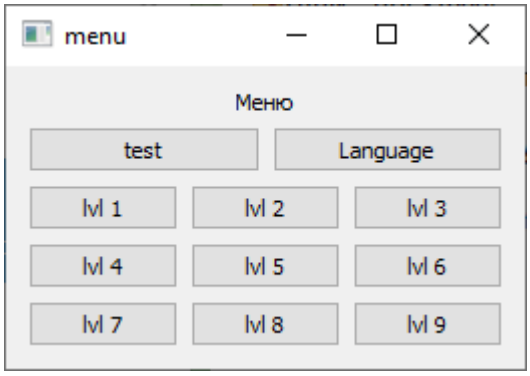

Рис 2.1 меню

Інтеграція базових елементів ігрової сцени за допомогою Pygame

Ігрова сцена та кнопки управління були створенні з використанням бібліотеки Pygame. На всіх скріншотах видно сітку, що представляє собою ігрове поле, де буде рухатися робот. Робот також присутній на ігровій сцені та може переміщатися по сітці згідно з алгоритмами, створеними користувачем. Це демонструє базову інтеграцію графічних елементів та функціоналу гри.

Реалізація основних кнопок управління (рух вперед, повороти)

На наступному етапі було реалізовано основні кнопки управління роботом, що включають рух вперед, повороти вправо та вліво. Ці кнопки розташовані на лівій стороні екрана та забезпечують користувачеві простий спосіб контролювати рухи робота. На скріншоті видно, що кнопки розташовані вертикально та готові до використання. Приклад коду у додатках Н та О

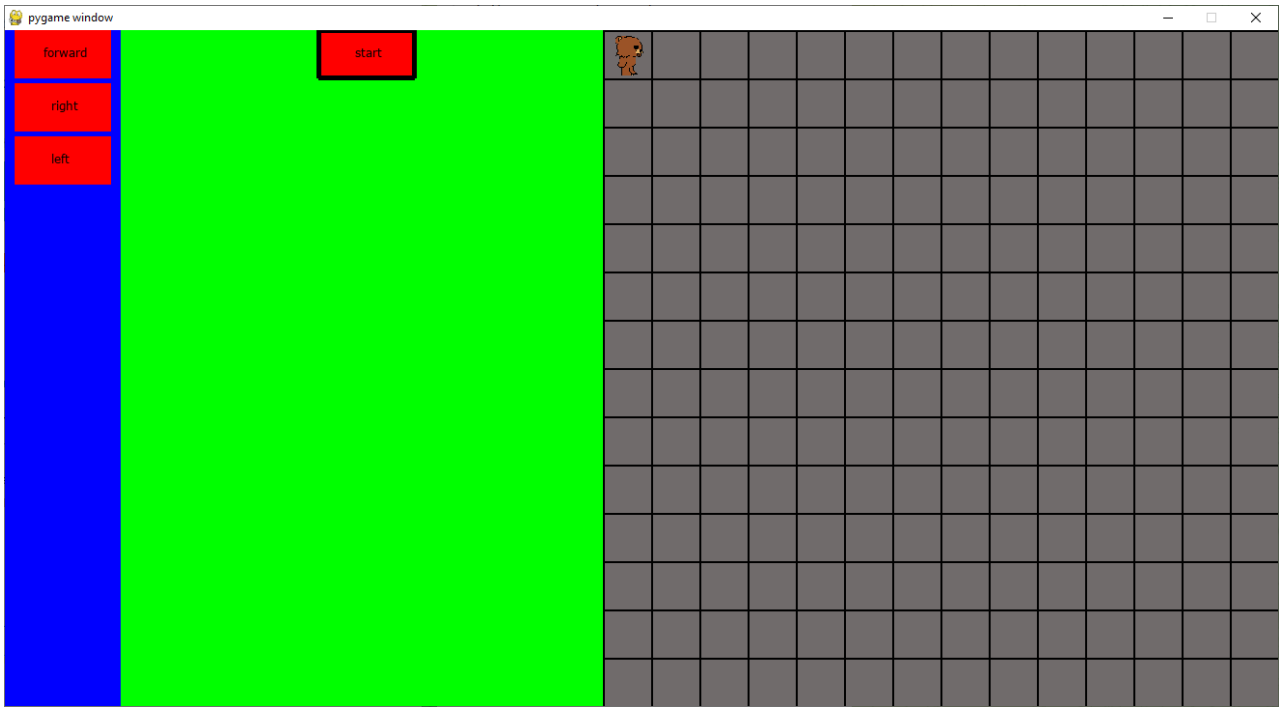

Рис2.2 Кнопки управління

Впровадження можливості перетягування кнопок для створення алгоритмів

Однією з ключових функцій проекту є можливість перетягування кнопок для створення алгоритмів руху робота. На другому скріншоті показано, як кнопка "forward" була перетягнута в центр робочої області, що дозволяє користувачеві створювати послідовності дій для робота. Ця функція додає інтерактивність і робить процес навчання програмуванню більш захоплюючим та ефективним.

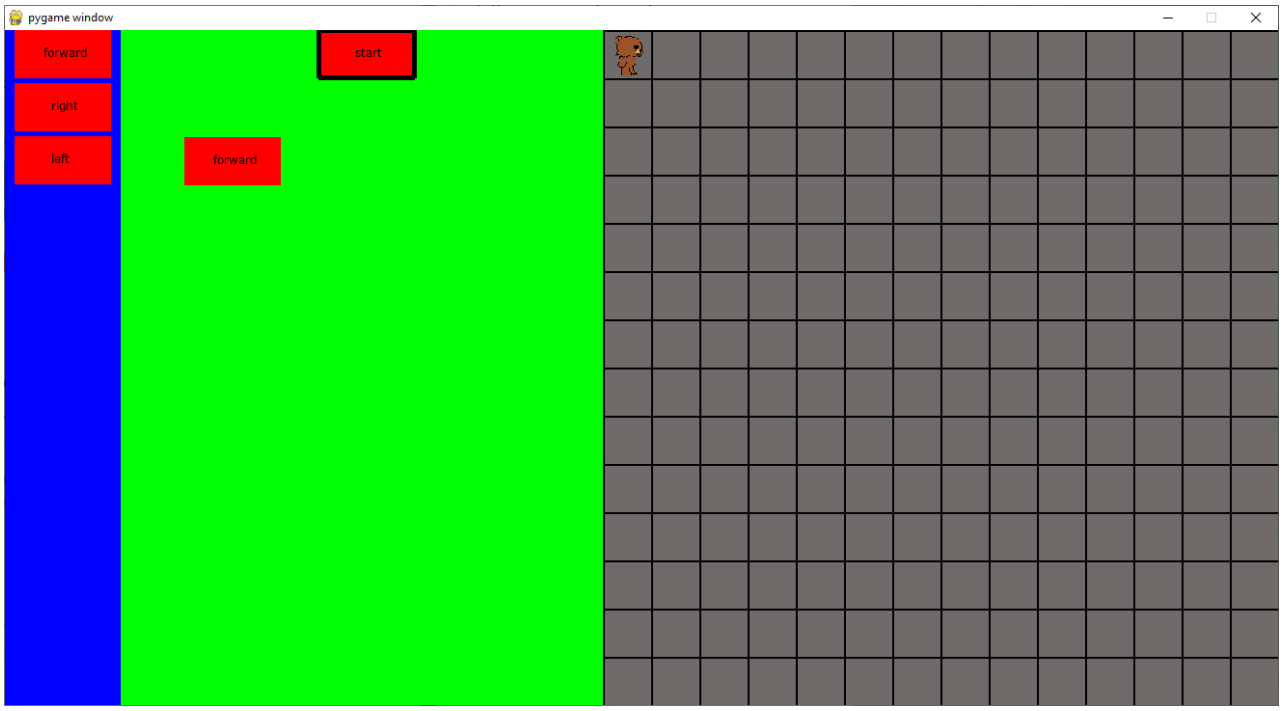

Рис 2.3 Перетягування кнопок

Тестування базового функціоналу

На останньому скріншоті робот знаходиться в іншій позиції на ігровому полі, що підтверджує працездатність базового функціоналу. Це свідчить про те, що кнопки управління та алгоритми дій працюють коректно, дозволяючи роботу пересуватися згідно з заданими командами. Тестування базового функціоналу є важливим етапом, що гарантує стабільність роботи програми та коректність виконання основних дій.

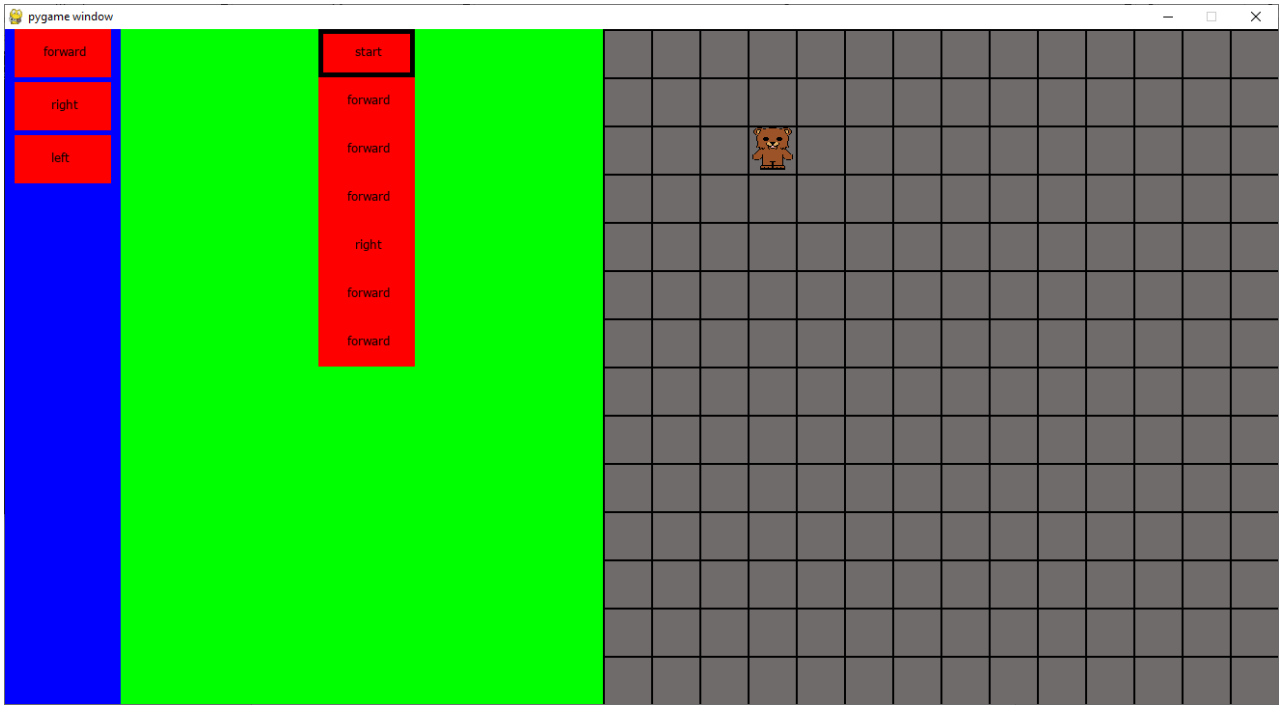

Рис 2.4 Виконання алгоритму

Етап 3: Розширення Функціоналу

На третьому етапі розробки проекту "Гра для вивчення автоматизації та програмування" було реалізовано кілька важливих оновлень, які значно покращують функціональність та зручність використання гри.

Додавання циклів

Основним нововведенням стало додавання циклів, що дозволяє створювати повторювані алгоритми. Це суттєво розширює можливості користувачів у програмуванні рухів робота. На скріншотах видно кнопки для додавання циклів та зміни їх кількості, що робить процес складання алгоритмів більш гнучким та динамічним. Приклад коду у додатку П

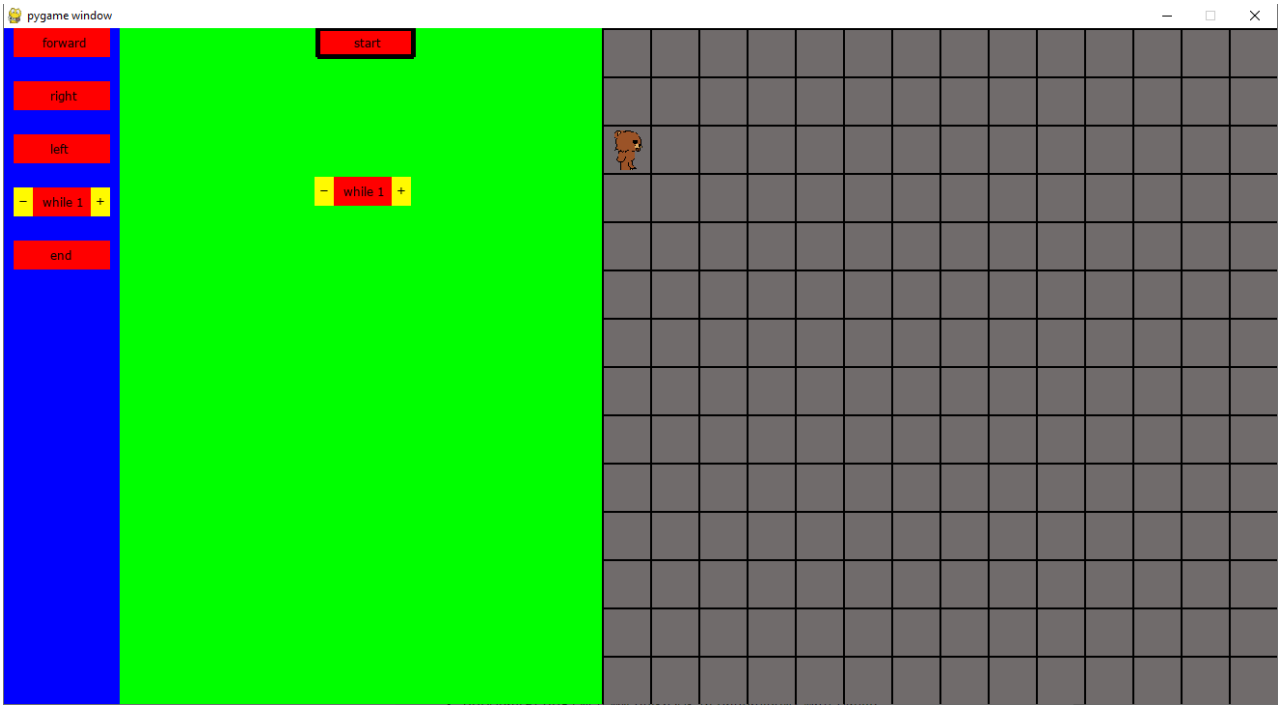

Рис 2.5 Клік по циклу

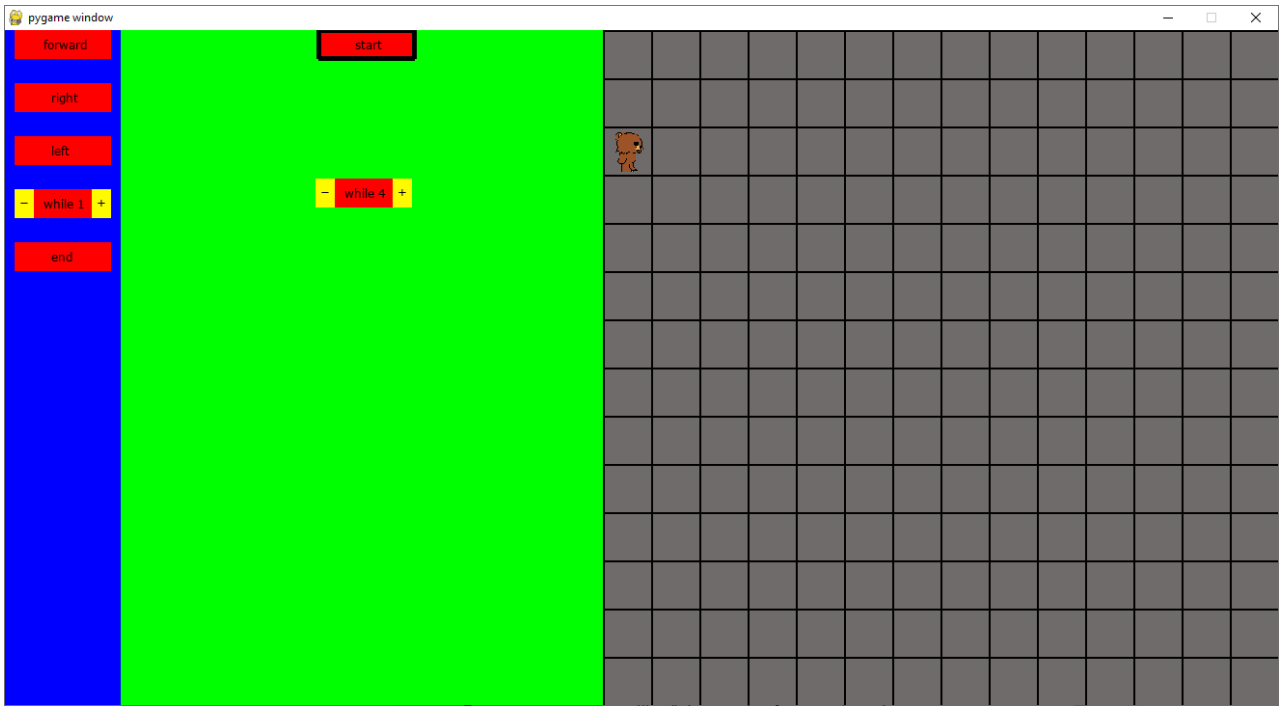

Рис 2.6 Клік по циклу

Інтерфейс користувача

Інтерфейс користувача було значно покращено для забезпечення більш інтуїтивного та зручного використання. Було додано межі карти, що чітко розділяють ігрове поле та область команд. Крім того, елементи інтерфейсу були перероблені для полегшення взаємодії з ними, як, наприклад, кнопка "очистити поле команд", яку видно на скріншотах зліва знизу екрану.

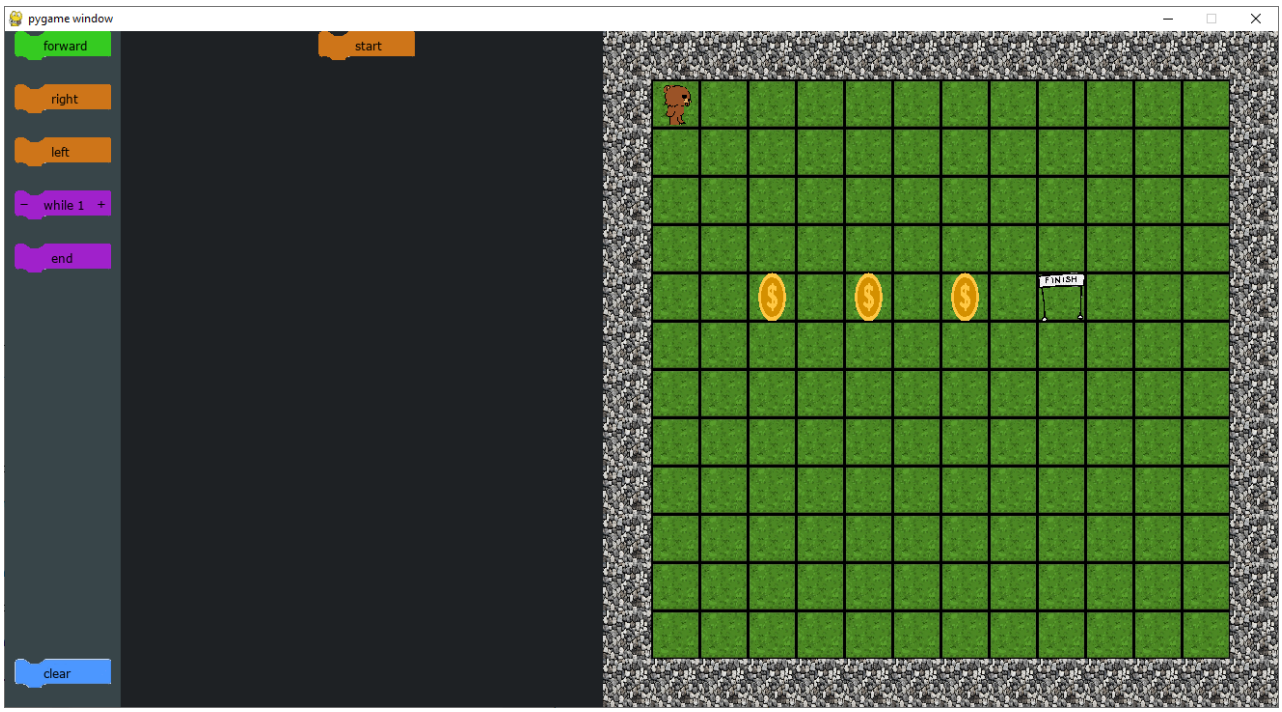

Рис 2.7 Інтерфейс

Система збереження та завантаження алгоритмів

Було проведено аналіз необхідності впровадження системи збереження та завантаження алгоритмів. В результаті було вирішено, що ця функція не є критично важливою для даної гри, оскільки використання збережених алгоритмів може заважати навчальному процесу. Постійне створення нових циклів і алгоритмів сприяє кращому запам'ятовуванню та розумінню основ програмування.

Впровадження системи підказок та навчальних матеріалів

Для полегшення процесу навчання було впроваджено систему підказок та навчальних матеріалів. Це допомагає користувачам швидше освоїти функціонал гри та розуміти, як використовувати нові елементи, такі як цикли. Підказки інтегровані у вигляді спливаючих вікон, що з'являються при наведенні курсора на відповідні елементи інтерфейсу.

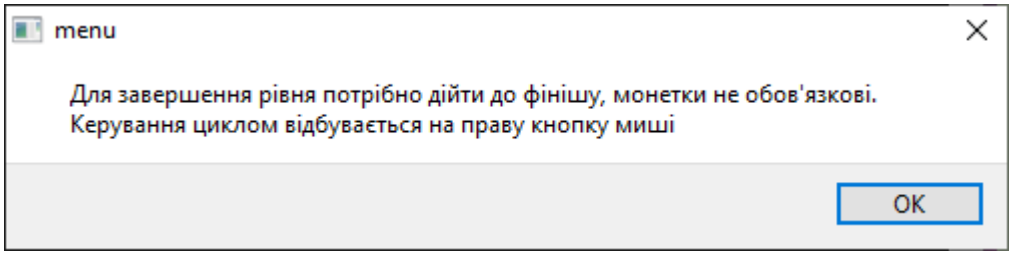

Рис 2.8 Підказка

Тестування нових функцій та їх інтеграція з базовим функціоналом

Всі нові функції пройшли ретельне тестування для забезпечення їх коректної роботи та сумісності з існуючим функціоналом гри. Було проведено кілька тестових сесій з залученням користувачів, що дозволило виявити та усунути можливі помилки. Після тестування нові функції були успішно інтегровані в основну систему гри.

Етап 4: Інтеграція Рівнів та Модулів Навчання

Розробка меню вибору рівнів за допомогою PyQt5

На даному етапі був розроблений інтерфейс меню вибору рівнів, використовуючи PyQt5. Це меню дозволяє користувачам обирати різні рівні гри, що робить навігацію більш зручною та інтуїтивно зрозумілою. На скріншоті видно, що меню містить кілька кнопок для вибору рівнів, кожна з яких відповідає за певний рівень. Впровадження меню передбачає поступове додавання нових рівнів, і кожна кнопка буде активована по мірі їх створення.

На даний момент меню не є повністю функціональним, однак воно вже забезпечує базову можливість вибору рівнів, що є важливим кроком у розвитку гри.

Приклад коду у додатках А - Д

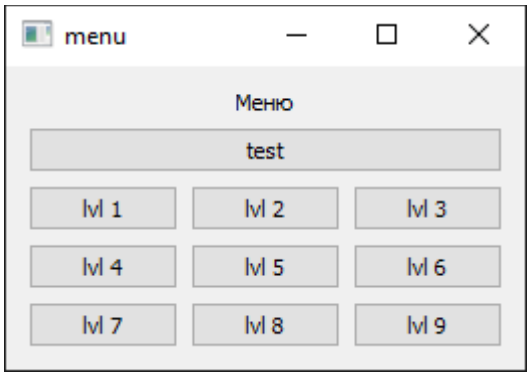

Рис 2.9 Оновлене меню

Створення декількох навчальних рівнів з різними завданнями

Було створено кілька навчальних рівнів, кожен з яких має унікальні завдання, спрямовані на розвиток навичок програмування у користувачів. На скріншотах видно два різних рівні, де метою є досягнення фінішної точки. Рівні відрізняються своїм дизайном та розташуванням елементів, таких як монети та фінішні точки. Монети на карті є додатковим заохоченням для користувачів, які можуть збирати їх для підвищення свого рейтингу. Ці монети не є обов'язковими для збору, але служать стимулом для більш ретельного проходження рівнів. Користувачі можуть розробляти алгоритми для управління рухом робота, використовуючи команди, доступні в інтерфейсі гри.

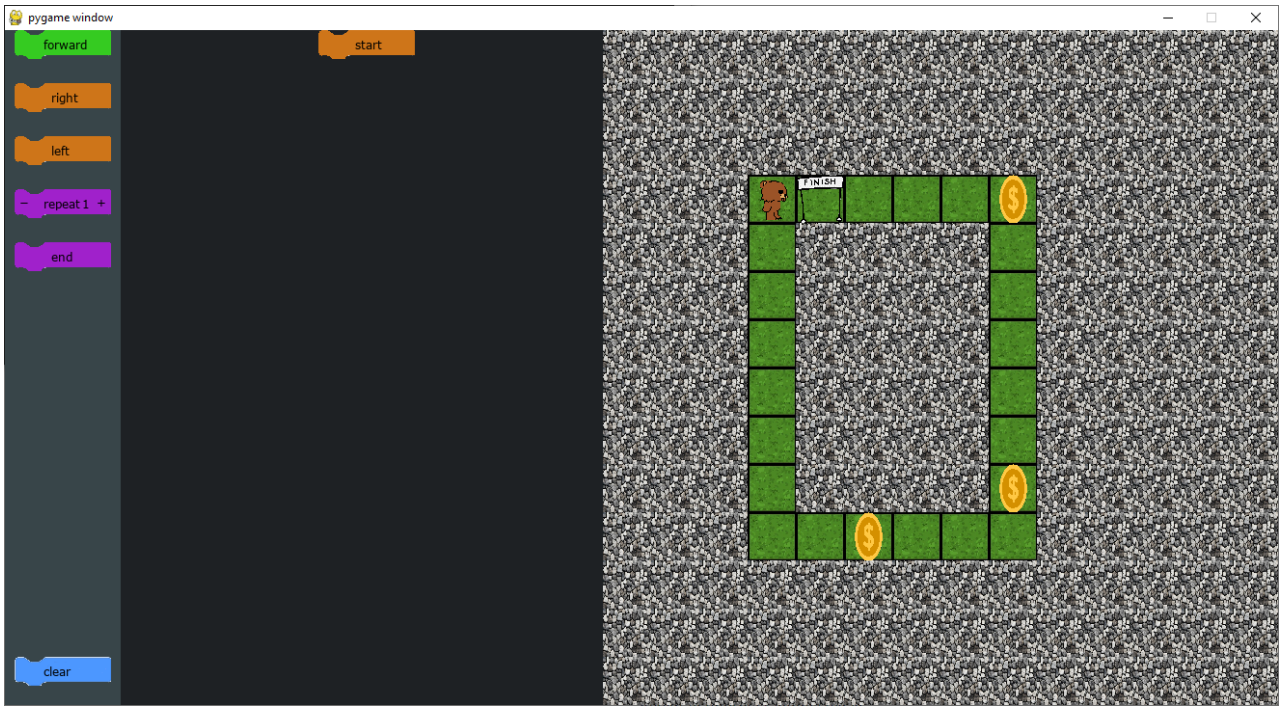

Рис 2.10 Рівень 1

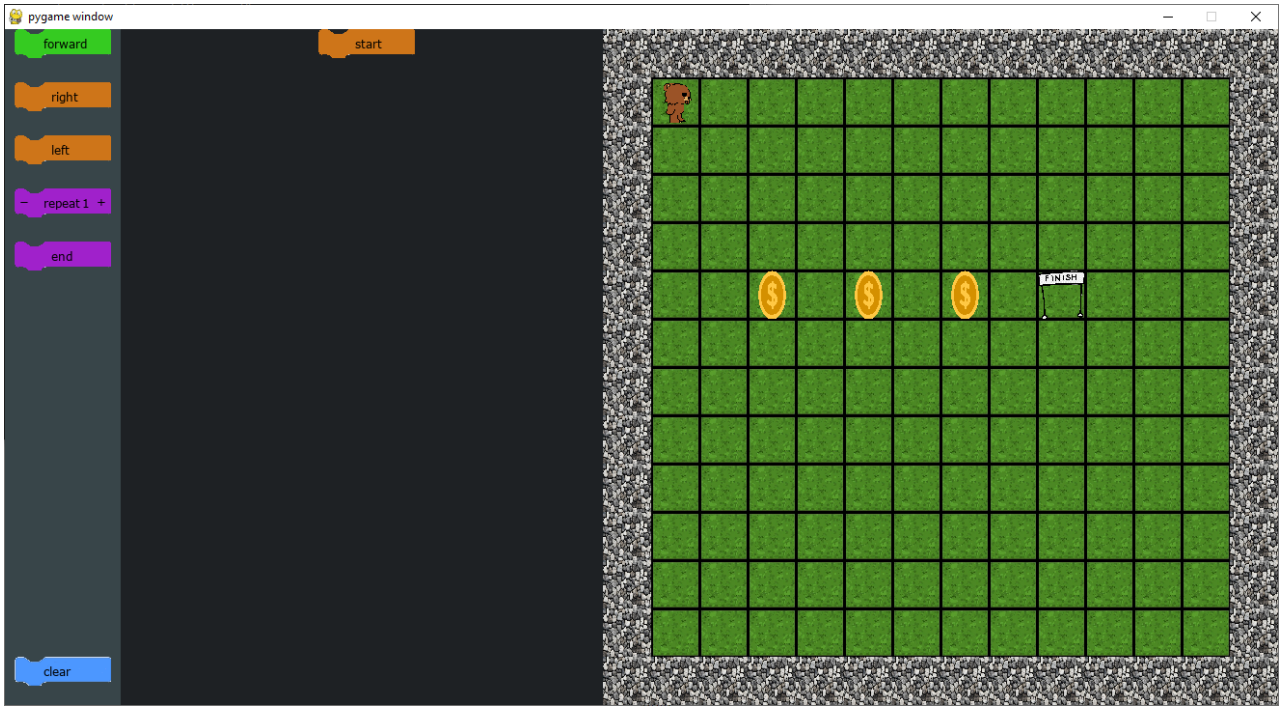

Рис 2.11 Рівень 0

Впровадження системи оцінки та прогресу користувача

Для мотивації та оцінки успіхів користувачів була розроблена система оцінки та прогресу. Після завершення кожного рівня, користувач отримує звіт про свою діяльність, включаючи інформацію про кількість зібраних монет. Ця система дозволяє користувачам бачити свої досягнення та прагнути до кращих результатів, повторюючи проходження рівнів для збору більшої кількості монет. Система оцінки також включає в себе візуальні індикатори прогресу, що

допомагає користувачам відстежувати свої успіхи та покращувати навички програмування.

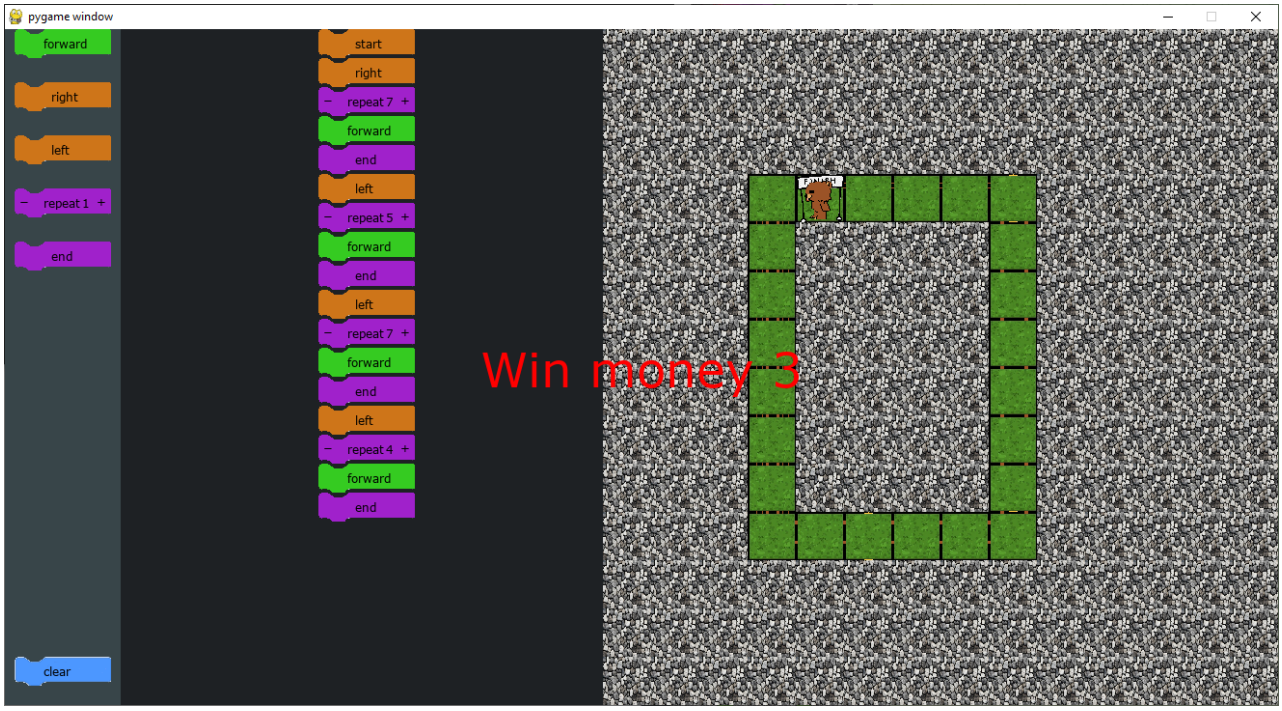

Рис 2.12 Відображення виграшу та кількості монет

Тестування та відладка рівнів

Після створення нових рівнів було проведено ретельне тестування та відладка для забезпечення їх коректної роботи. Тестування включало перевірку різних сценаріїв проходження рівнів, щоб виявити можливі помилки та неточності в алгоритмах гри. В ході тестування було враховано зворотній зв'язок від користувачів, що дозволило внести необхідні корективи та поліпшити якість гри. Відладка рівнів також включала перевірку сумісності нових функцій з існуючим функціоналом гри, що гарантує стабільність і надійність роботи програмного забезпечення.

Етап 5: Оптимізація та Поліпшення

Оптимізація коду для підвищення продуктивності

Рефакторинг коду став важливим етапом для забезпечення плавної роботи гри. Це включало перегляд існуючих алгоритмів для визначення можливих вузьких місць і їх оптимізацію. Наприклад, функції обробки подій були перегруповані для зменшення кількості перевірок, а рендеринг графічних елементів було зведено до мінімуму необхідного для поточного кадру. Також

були використані більш ефективні структури даних для зберігання і швидкого доступу до ігрових об'єктів.

Поліпшення графічного інтерфейсу та дизайну

На етапі вдосконалення графіки була проведена повна заміна тестових моделей на остаточні варіанти. Персонаж гри отримав вигляд робота, що включало більш детальне моделювання та текстурування. Монети також були покращені: тепер вони мають більш чіткі контури і привабливий дизайн, що робить їх більш помітними і естетично приємними. Вся ігрова сцена отримала нові текстури для підлоги і стін, створюючи більш реалістичне і приємне оточення для гравців.

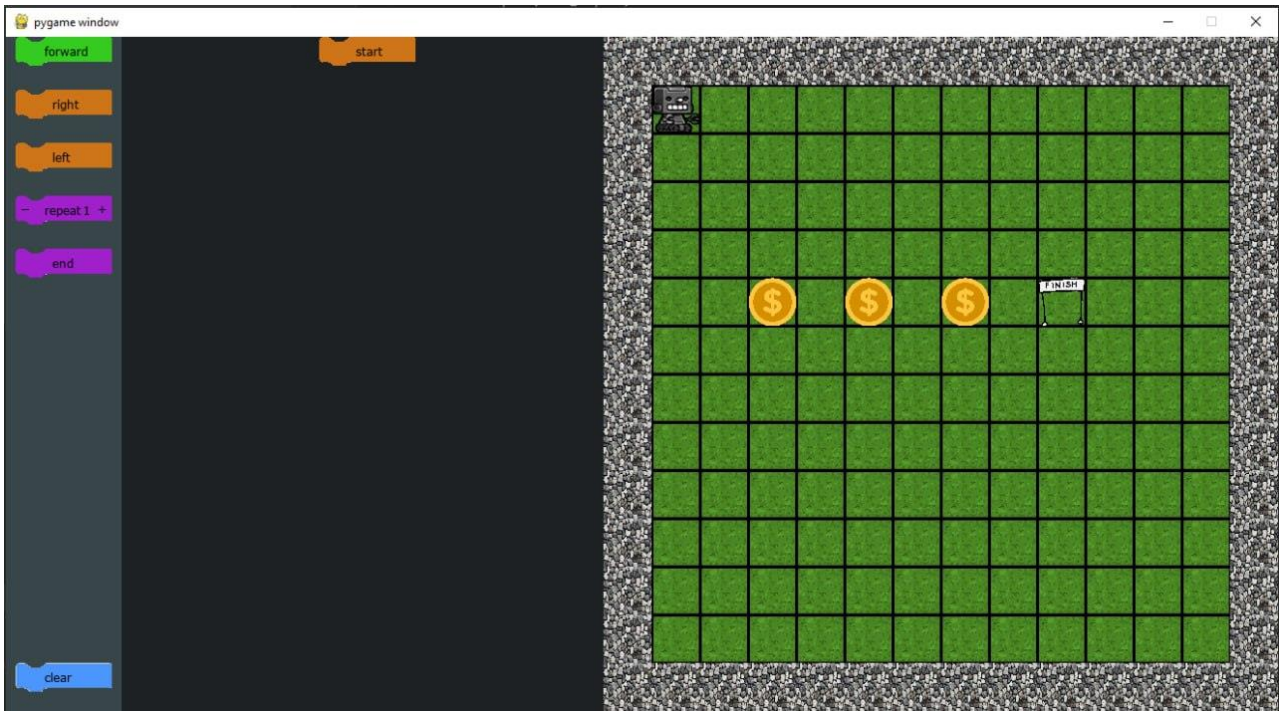

Рис 2.13 Оновлені текстури

Впровадження системи зворотного зв'язку від користувачів

Незважаючи на те, що проект невеликий, було вирішено залучити фокус-групу з одногрупників і друзів. Це дало можливість зібрати цінні відгуки про користувацький досвід. Учасники фокус-групи тестували гру, надаючи зворотний зв'язок щодо зручності інтерфейсу, логіки гри та загального враження. Виявлені проблеми були класифіковані за пріоритетністю і враховані у подальшій роботі.

Розширення функціоналу з урахуванням зворотного зв'язку

На основі отриманих відгуків було додано кілька нових функцій. Зокрема, меню виклику після завершення рівня стало ключовим оновленням, яке дозволяє гравцям зберігати прогрес, вибирати нові рівні або повертатися до головного меню без закриття гри. Це значно покращило зручність користування.

Проведення кінцевого тестування та відладки

Фінальне тестування включало кілька етапів, починаючи від тестування окремих функцій і закінчуючи інтеграційним тестуванням всієї гри. Було важливо переконатися, що всі додані функції працюють належним чином і не створюють нових проблем. Виявлені баги були виправлені, що включало корекцію логіки руху персонажа, належне відображення зібраних монет і правильне функціонування нових меню. Після завершення цих тестів гра досягла стабільного стану, готового до подальшого використання і можливого розширення в майбутньому.

Етап Потенційних Завдань

Хоча на поточному етапі багато ключових завдань було виконано, існує кілька потенційних напрямків для подальшого розвитку проекту. Ці завдання, хоча і не були виконані через брак часу та ресурсів, можуть бути реалізовані в майбутньому, якщо буде потреба в продовженні розвитку гри.

Розробка алгоритмів для складніших дій робота

Наразі робот виконує базові команди, такі як рух вперед, повороти і цикли. В майбутньому можна розробити складніші алгоритми, які дозволять роботу виконувати завдання, що потребують більшої гнучкості та інтелекту. Наприклад, можна додати алгоритми для обходу перешкод, прийняття рішень на основі умов, а також взаємодії з іншими об'єктами в ігровому середовищі. Такі розширення значно підвищать навчальну цінність гри, дозволяючи користувачам вивчати складніші концепції програмування та алгоритміки.

Інтеграція додаткових модулів для вивчення інших аспектів програмування

Поточна версія гри зосереджена на базових елементах алгоритмів і логіки. В майбутньому можна інтегрувати додаткові модулі, які охоплюватимуть інші аспекти програмування, такі як обробка даних, робота з файлами, мережеве програмування та інші. Це дозволить користувачам отримувати більш всебічне розуміння програмування, працюючи з різноманітними задачами та середовищами. Наприклад, можна додати модуль для роботи з базами даних, що навчить користувачів основам SQL і маніпуляціям з даними.

Впровадження мультимовної підтримки

Для розширення аудиторії гри варто розглянути можливість впровадження підтримки кількох мов. Це дозволить користувачам з різних країн легко користуватися грою на своїй рідній мові. Впровадження мультимовної підтримки включатиме переклад інтерфейсу гри, документації та навчальних матеріалів. Цей крок зробить гру більш доступною і привабливою для міжнародної аудиторії, збільшуючи її навчальну та комерційну цінність.

Розробка системи досягнень та мотивації для користувачів

Для підвищення мотивації користувачів і залучення до процесу навчання можна розробити систему досягнень. Ця система може включати різноманітні нагороди за виконання певних завдань, проходження рівнів або освоєння нових навичок. Наприклад, можна додати значки, сертифікати або інші віртуальні нагороди, які користувачі зможуть збирати і демонструвати. Система досягнень стимулюватиме користувачів до подальшого вивчення та вдосконалення своїх навичок, роблячи процес навчання більш цікавим і захоплюючим.

Хоча ці завдання не були виконані на поточному етапі через обмеженість часу та ресурсів, вони представляють собою перспективні напрямки для розвитку гри у майбутньому. Реалізація цих ідей дозволить значно розширити функціональність гри, підвищити її навчальну цінність та зробити її більш

привабливою для широкої аудиторії. Якщо виникне необхідність продовжити розвиток гри, ці завдання можуть стати основою для подальшої роботи.

Перспективи Інтеграції Нових Можливостей

Додавання підтримки різних мов програмування

На даний момент гра використовує одну мову програмування для створення алгоритмів. Проте, для розширення навчальних можливостей і залучення більшої аудиторії, можливо впровадити підтримку кількох мов програмування, таких як Python, JavaScript, C++, та інші. Це дозволить користувачам вибирати мову, яку вони хочуть вивчати або вже знають, і створювати алгоритми на ній. Інтеграція підтримки різних мов програмування включатиме створення багатомовного середовища кодування, де користувачі зможуть переключатися між мовами, а також адаптацію інтерфейсу та навчальних матеріалів для кожної мови. Це значно розширить можливості гри, роблячи її універсальним інструментом для навчання різним мовам програмування.

Розробка мобільної версії програми

Зараз гра доступна тільки на комп'ютерах, але розробка мобільної версії дозволить значно розширити аудиторію користувачів. Мобільна версія гри буде доступна на платформах iOS та Android, що дозволить користувачам навчатися програмуванню в будь-який час та в будь-якому місці. Розробка мобільної версії включатиме адаптацію інтерфейсу для сенсорного управління, оптимізацію продуктивності для мобільних пристроїв та забезпечення сумісності з різними роздільними здатностями екранів. Крім того, можна інтегрувати функції, специфічні для мобільних платформ, такі як пуш-повідомлення для нагадувань про навчання або досягнення.

Впровадження штучного інтелекту для аналізу алгоритмів та надання рекомендацій

Використання штучного інтелекту (ШІ) може значно підвищити якість навчання. ШІ можна інтегрувати для аналізу алгоритмів, створених користувачами, і надання рекомендацій щодо їх оптимізації. Наприклад, ШІ

може виявляти неефективні частини коду, пропонувати кращі практики та альтернативні рішення, що допоможе користувачам краще зрозуміти принципи програмування і вдосконалити свої навички. Крім того, ШІ може надавати персоналізовані поради на основі рівня знань і прогресу кожного користувача, що зробить процес навчання більш індивідуалізованим і ефективним.

Розширення функціоналу для підтримки складніших алгоритмів та проектів

Наразі гра зосереджена на базових концепціях програмування, але для більш просунутих користувачів можна розширити функціонал, включивши підтримку складніших алгоритмів і проектів. Це може включати додавання нових типів завдань, які вимагатимуть використання просунутих структур даних, таких як графи або дерева, а також реалізацію більш комплексних проектів, таких як міні-ігри або застосунки. Розширення функціоналу також передбачає впровадження інструментів для командної роботи, що дозволить користувачам спільно працювати над проектами, обмінюватися кодом та ідеями, тим самим розвиваючи навички колективного програмування і управління проектами.

РОЗДІЛ 3 ПРОБЛЕМИ РОЗРОБКИ ТА МЕТОДИ ЇХ ВИРІШЕННЯ

3.1 Відображення малюнків.

Кожне зображення на сцені відображається за допомогою модуля pygame. Раніше ігрова сцена відображалась одним малюнком(сіткою) і вручну створювались усі перешкоди та ігрові об’єкти. На той момент не існувало жодних заготовок карти

Приклад коду:

```
self.background =
pygame.transform.scale(pygame.image.load("сетка.png"),
(700, 700))
self.player =
pygame.transform.scale(pygame.image.load("право.png"),
(50, 50))

self.player =
Area(self.screen,620,100,50,50,None,lambda: None,)
self.player.setpic("право.png")
self.wall = Area(self.screen,720,100,50,50,None,lambda:
None,)
self.wall.setpic("wall.jpg")
self.fin = Area(self.screen,720,150,50,50,None,lambda
:None,)
self.fin.setpic("finish.png")

self.floor = Area(self.screen,720,50,50,50,None,lambda
:None,)
self.floor.setpic("пол.png")
self.map =[self.wall,self.fin,self.floor,self.player]

self.mone = Area(self.screen, 670, 150, 50, 50, None,
lambda: None, )
self.mone.setpic("монета.png")
```

Наразі карта зберігається у списку розміром
приклад знаходиться у додатку Б

Кожний елемент списку є об’єктом класу Map(Enum):

Приклад у додатку В

3.2 Проблема розробки класу Cycle.

Ця проблема полягала у складнощах з правильним ініціюванням та виконанням вкладених циклів у програмі. Основною трудністю було коректне розпізнавання ключового слова "end", що визначало кінець блоку циклу, та відповідне формування ієрархії циклів для їх послідовного виконання.

Для вирішення цієї проблеми використовувався рекурсивний алгоритм у методі parse, що дозволив ефективно виявляти та структурувати вкладені цикли у вхідному списку кнопок. Кожне входження ключового слова "repeat" викликало створення нового екземпляру класу Cycle, що забезпечувало правильне управління ітераціями циклу.
Приклад коду у додатку Р

3.3 Проблема кнопки циклу

У грі всі кнопки створювались за допомогою класів Label та Area але у результаті кнопка була самодостатнім об’єктом. Кнопка циклу це 3 повноцінні кнопки які треба об’єднати в один об’єкт. Для вирішення цієї проблеми було створено Whilebut який об’єднує ці 3 кнопки в 1.
приклад коду у додатку П

СПИСОК ВІКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Документація Python (https://docs.python.org/3/)
2. Документація Pygame (https://www.pygame.org/docs/)
3. "Game Programming Patterns" by Robert Nystrom
4. Офіційна документація PyQt (https://www.riverbankcomputing.com/static/Docs/PyQt5/)
5. Документація Qt (https://doc.qt.io/)
6. "Learning Python" by Mark Lutz
7. Beginning Game Development with Python and Pygame: From Novice to Professional (Beginning From Novice to Professional) by Will McGugan
8. Teach Your Kids to Code: A Parent-Friendly Guide to Python Programming by Bryson Payne
9. Scratch Programming Playground: Learn to Program by Making Cool Games by Al Sweigart
10. Beginning PyQt: A Hands-on Approach to GUI Programming by Joshua M. Willman

ДОДАТКИ

Додаток А

Підключення модулів до файлу меню

```
from enum import Enum
from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout,
QHBoxLayout, QPushButton, QLabel, QMessageBox
import game
from PyQt5.QtCore import Qt
```

Додаток Б

Клас для створення рівнів та один з рівнів

```
class Map(Enum):
    empty = "пол.png"
    coin = "монета.png"
    wall = "wall.jpg"
    fin = "finish.png"

# Визначення рівня 0
lvl0 = [
     # Карта рівня
     [Map.wall, Map.wall, Map.wall, Map.wall, Map.wall, Map.wall,
Map.wall, Map.wall, Map.wall, Map.wall, Map.wall, Map.wall,
Map.wall, Map.wall],
     [Map.wall, Map.empty, Map.empty, Map.empty, Map.empty,
Map.empty, Map.empty, Map.empty, Map.empty, Map.empty, Map.empty,
Map.empty, Map.empty, Map.wall],
     [Map.wall, Map.empty, Map.empty, Map.empty, Map.empty,
Map.empty, Map.empty, Map.empty, Map.empty, Map.empty, Map.empty,
Map.empty, Map.empty, Map.wall],
     [Map.wall, Map.empty, Map.empty, Map.empty, Map.empty,
Map.empty, Map.empty, Map.empty, Map.empty, Map.empty, Map.empty,
Map.empty, Map.empty, Map.wall],
     [Map.wall, Map.empty, Map.empty, Map.empty, Map.empty,
Map.empty, Map.empty, Map.empty, Map.empty, Map.empty, Map.empty,
Map.empty, Map.empty, Map.wall],
     [Map.wall, Map.empty, Map.empty, Map.coin, Map.empty,
Map.coin, Map.empty, Map.coin, Map.empty, Map.fin, Map.empty,
Map.empty, Map.empty, Map.wall],
     [Map.wall, Map.empty, Map.empty, Map.empty, Map.empty,
Map.empty, Map.empty, Map.empty, Map.empty, Map.empty, Map.empty,
Map.empty, Map.empty, Map.wall],
     [Map.wall, Map.empty, Map.empty, Map.empty, Map.empty,
Map.empty, Map.empty, Map.empty, Map.empty, Map.empty, Map.empty,
Map.empty, Map.empty, Map.wall],
     [Map.wall, Map.empty, Map.empty, Map.empty, Map.empty,
Map.empty, Map.empty, Map.empty, Map.empty, Map.empty, Map.empty,
Map.empty, Map.empty, Map.wall],
     [Map.wall, Map.empty, Map.empty, Map.empty, Map.empty,
Map.empty, Map.empty, Map.empty, Map.empty, Map.empty, Map.empty,
Map.empty, Map.empty, Map.wall],
     [Map.wall, Map.empty, Map.empty, Map.empty, Map.empty,
```

```
Map.empty, Map.empty, Map.empty, Map.empty, Map.empty, Map.empty,
Map.empty, Map.empty, Map.wall],
    [Map.wall, Map.empty, Map.empty, Map.empty, Map.empty,
Map.empty, Map.empty, Map.empty, Map.empty, Map.empty, Map.empty,
Map.empty, Map.empty, Map.wall],
    [Map.wall, Map.empty, Map.empty, Map.empty, Map.empty,
Map.empty, Map.empty, Map.empty, Map.empty, Map.empty, Map.empty,
Map.empty, Map.empty, Map.wall],
    [Map.wall, Map.wall, Map.wall, Map.wall, Map.wall, Map.wall,
Map.wall, Map.wall, Map.wall, Map.wall, Map.wall, Map.wall,
Map.wall, Map.wall],
]
```

Додаток В

Створення класу Menu та метод init

```
class Menue(QWidget):
    def __init__(self):
        super().__init__()

        # Основний вертикальний layout
        main_layout = QVBoxLayout()
        label_menu = QLabel("Меню")  # Заголовок меню
        layout1 = QHBoxLayout()  # Горизонтальні layout'и для
кнопок
        layout2 = QHBoxLayout()
        layout3 = QHBoxLayout()
        layout4 = QHBoxLayout()
        main_layout.addWidget(label_menu, alignment=Qt.AlignCenter
| Qt.AlignTop)

        main_layout.addLayout(layout4)

        # Створення кнопок для рівнів
        button1 = QPushButton("lvl 1")
        button2 = QPushButton("lvl 2")
        button3 = QPushButton("lvl 3")
        button4 = QPushButton("lvl 4")
        button5 = QPushButton("lvl 5")
        button6 = QPushButton("lvl 6")
        button7 = QPushButton("lvl 7")
        button8 = QPushButton("lvl 8")
        button9 = QPushButton("lvl 9")

        buttontest = QPushButton("test")
        buttonlang = QPushButton("Language")

        # Додавання кнопок до горизонтальних layout'ів
        layout1.addWidget(button1)
        layout1.addWidget(button2)
        layout1.addWidget(button3)
        layout2.addWidget(button4)
        layout2.addWidget(button5)
```

```
        layout2.addWidget(button6)
        layout3.addWidget(button7)
        layout3.addWidget(button8)
        layout3.addWidget(button9)

        layout4.addWidget(buttontest)
        # layout4.addWidget(buttonlang)  # Додати кнопку мови при
необхідності

        main_layout.addLayout(layout1)
        main_layout.addLayout(layout2)
        main_layout.addLayout(layout3)

        # Прив'язка подій до кнопок
        buttontest.clicked.connect(self.test)
        button1.clicked.connect(self.lvl1)
        self.setLayout(main_layout)
```

Додаток Г

Метод для кнопки start

```
def test(self):
    # Створення інформаційного вікна
    info = QMessageBox()
    info.setText("Для завершення рівня потрібно дійти до фінішу,
монетки не обов'язкові. Керування циклом відбувається на праву
кнопку миші")
    info.exec()
    self.hide()  # Приховування основного вікна

    Game = game.Game()  # Ініціалізація гри
    Game.drawmap(lvl0, 670, 50)  # Відображення карти рівня
    while Game.p:
        Game.loop()  # Основний ігровий цикл

    self.show()  # Показ основного вікна після завершення гри
```

Додаток Ґ

Метод для кнопки lvl1

```
def lvl1(self):
    # Створення інформаційного вікна
    info = QMessageBox()
    info.setText("Для завершення рівня потрібно дійти до фінішу,
монетки не обов'язкові. Керування циклом відбувається на праву
кнопку миші")
    info.exec()
    self.hide()  # Приховування основного вікна

    Game = game.Game()  # Ініціалізація гри
    Game.drawmap(lvl1, 770, 150)  # Відображення карти рівня
```

```
    while Game.p:
        Game.loop()  # Основний ігровий цикл

    self.show()  # Показ основного вікна після завершення гри
```

Додаток Д

Основний код для створення об’єкту класу та запуску меню

```
app = QApplication([])  # Ініціалізація додатку
w = Menu()  # Створення віджета гри
w.show()  # Показ віджета
app.exec_()  # Запуск основного циклу додатку
```

Додаток Е

модулі файлу гри

```
import random
import pygame
from pygame.locals import QUIT, USEREVENT, MOUSEBUTTONDOWN
import time
```

Додаток Є

Клас для створення прямокутників

```
gamewin = (1320, 700)
# Класс для створення прямокутників (хітбоксів)
class Area():
  def __init__(self, mw, x, y, width, height, color, function
=lambda:None, pic=None):
      self.rect = pygame.Rect(x, y, width, height) #прямоугольник
      self.fill_color = color
      self.mw = mw
      self.function = function
      self.pic = pic

  # метод для зміни кольорів
  def color(self, new_color):
      self.fill_color = new_color

  # метод для зафарбовування прямокутника
  def fill(self):
      pygame.draw.rect(self.mw, self.fill_color, self.rect)

  # створення контура
  def outline(self, frame_color, thickness): #обводка
существующего прямоугольника
      pygame.draw.rect(self.mw, frame_color, self.rect, thickness)

  # перевірка натискання мишкою
```

```
  def collidepoint(self, x, y):
      return self.rect.collidepoint(x, y)

  #перевірка дотику інших прямокутників
  def colliderect(self, rect):
      return self.rect.colliderect(rect)

  # відображення картинок у прямокутнику
  def drawpic(self):
      #self.rect = self.obj.get_rect()
      self.mw.blit(self.obj, (self.rect.x, self.rect.y))

  #встановлення картинки для прямокутника
  def setpic(self,pic,widht=50,height=50):
      self.pic = pic
      self.obj =
pygame.transform.scale(pygame.image.load(self.pic), (widht,
height))
```

Додаток Ж

Клас для створення надписів

```
class Label(Area):
  def set_text(self, text, fsize=12, text_color=(0, 0, 0),):
      self.text = text

      self.image = pygame.font.SysFont('verdana',
fsize).render(text, True, text_color)

  def draw(self, shift_x=0, shift_y=0):
      #self.fill()
      try:
        self.drawpic()
      except:
          pass
     self.mw.blit(self.image, (self.rect.x + shift_x, self.rect.y + shift_y))
```

Додаток З

Ініціалізація класу game

```
 def __init__(self):
    self.p = True
    pygame.display.init()
    pygame.font.init()
    self.clock = pygame.time.Clock()
    self.screen = pygame.display.set_mode(gamewin)
    self.money = 0
    #self.background =
pygame.transform.scale(pygame.image.load("сетка.png"), (700, 700))

    self.background_left = self.rect = pygame.Rect(0, 0, 120,
```

```
gamewin[1])
    self.background_middle = self.rect = pygame.Rect(120, 0, 500,
gamewin[1])



    self.direction = 90

    self.startbtn = Label(self.screen, 325, 0, 100, 30, (255, 0,
0), self.start)
    self.startbtn.setpic("оранжевый.png",100,30)
    self.startbtn.set_text("  start")

    self.button_forward =
Label(self.screen,10,0,100,30,None,self.forward,)
    self.button_forward.set_text("forward")
    self.button_forward.setpic("зелёный.png",100,30)

    self.button_right =
Label(self.screen,10,55,100,30,(255,0,0),self.rotate)
    self.button_right.set_text("  right")
    self.button_right.setpic("оранжевый.png", 100, 30)

    self.button_left =
Label(self.screen,10,110,100,30,(255,0,0),self.rotate)
    self.button_left.set_text("  left")

    self.button_left.setpic("оранжевый.png", 100, 30)

    self.button_while = Label(self.screen, 10, 165, 100, 30, (255,
0, 0))
    self.button_while.set_text("repeat")
    self.button_while.setpic("фиолетовый.png", 100, 30)

    self.button_whiletxt = Label(self.screen,
self.button_while.rect.x+40, self.button_while.rect.y, 10, 10,
self.button_while.fill_color )
    self.button_whiletxt.set_text("1")

    self.button_whileup = Label(self.screen,
self.button_while.rect.x+80, self.button_while.rect.y, 20, 30,
(255, 250, 0))
    self.button_whileup.set_text("+")

    self.button_whiledown = Label(self.screen,
self.button_while.rect.x, self.button_while.rect.y, 20, 30, (255,
250, 0)  )
    self.button_whiledown.set_text("-")

    self.button_end = Label(self.screen, 10, 220, 100, 30, (255,
0, 0) )
    self.button_end.set_text("  end")
    self.button_end.setpic("фиолетовый.png",100,30)

    self.button_clear = Label(self.screen, 10, gamewin[1]-50, 100,
```

```
30, (255, 0, 0), self.clear)
    self.button_clear.set_text("clear")
    self.button_clear.setpic("синий.png",100,30)


    self.moving = False
    self.movingobj = None
    self.buttons
=[self.button_forward,self.button_right,self.button_left,self.butt
on_while,self.button_end,self.button_whiletxt, self.button_clear]
    self.buttonswhile =[self.button_whileup,self.button_whiledown]
    self.additional_buttons = [self.startbtn]
    self.monee = []
    self.map = []
    self.wall =[]
```

Додаток И

Основний ігровий цикл – метод класу game

```
def loop(self):
    # перевірка натискання кнопки закриття гри
    for event in pygame.event.get():
        if event.type == QUIT:
            self.p = False
            #pygame.quit()


        if event.type == MOUSEBUTTONDOWN and event.button == 3:
            x,y = event.pos
            for button in self.additional_buttons:
                if button.__class__.__name__ == "Whilebut":
                    if button.button_whileup.collidepoint(x, y):
                        button.button_whileup.function()
                    elif button.button_whiledown.collidepoint(x,
y):
                        button.button_whiledown.function()

        if event.type == MOUSEBUTTONDOWN and event.button ==1:
            x, y = event.pos
            if self.button_forward.collidepoint(x, y):
                button_fw = Label(self.screen, 0, 0, 100, 30,
(255, 0, 0), self.forward)
                button_fw.set_text("forward")
                button_fw.setpic("зелёный.png",100,30)
                self.additional_buttons.append(button_fw)

            elif self.button_right.collidepoint(x, y):
                button_r = Label(self.screen, 0, 55, 100, 30,
(255, 0, 0), lambda: self.rotate(90))
```

```
                button_r.set_text("  right")
                button_r.setpic("оранжевый.png",100,30)
                self.additional_buttons.append(button_r)

            elif self.button_left.collidepoint(x, y):
                button_l = Label(self.screen, 0, 110, 100, 30,
(255, 0, 0), lambda: self.rotate(-90))
                button_l.set_text("  left")
                button_l.setpic("оранжевый.png", 100, 30)
                self.additional_buttons.append(button_l)

            elif self.startbtn.collidepoint(x,y):
                self.start()

            elif self.button_while.collidepoint(x,y):
                button_while = Whilebut(self.screen, 10, 165, 100,
30, (255, 0, 0), lambda: None)
                button_while.set_text("repeat")
                button_while.setpic("фиолетовый.png", 100, 30)
                self.additional_buttons.append(button_while)


            elif self.button_end.collidepoint(x,y):
                button_end = Label(self.screen, 10, 220, 100, 30,
(255, 0, 0), lambda: None)
                button_end.set_text("  end")
                button_end.setpic("фиолетовый.png", 100, 30)

                self.additional_buttons.append(button_end)
            elif self.button_clear.collidepoint(x,y):
                self.button_clear.function()


            for button in self.additional_buttons:
                if button != self.startbtn:
                    if button.collidepoint(x,y):
                        self.moving = True
                        self.movingobj = button
                        break
        if event.type == pygame.MOUSEMOTION:
            if self.moving:
                if not(self.movingobj == self.startbtn):
                    x_new, y_new = event.rel
                    self.movingobj.rect.x = self.movingobj.rect.x
+ x_new
                    self.movingobj.rect.y = self.movingobj.rect.y
+ y_new
        if event.type == pygame.MOUSEBUTTONUP and event.button ==
1:

            for button in self.additional_buttons:
                if
not(button.colliderect(self.background_middle)):
                    self.additional_buttons.remove(button)
                    del button
```

```
            if self.moving:
                buttons =[]
                buttons.append(self.startbtn)
                buttons.extend(self.conectblock())
                for i in buttons:
                    if self.movingobj.colliderect(i):
                        button=buttons[-1]
                        self.movingobj.rect.x = button.rect.x
                        self.movingobj.rect.y = button.rect.bottom

            self.moving = False

    self.update()
```

Додаток І

Метод для очищення сцени команд

```
def clear(self):
    self.additional_buttons =[self.startbtn]
```

Додаток Ї

Метод для оновлення стану екрану

```
def update(self):


pygame.draw.rect(self.screen,(30,33,36),self.background_middle)

    for button in self.additional_buttons:
        button.draw(30,7)

    #self.screen.blit(self.background, (620, 0))

    pygame.draw.rect(self.screen, (56, 69, 73),
self.background_left)

    for button in self.buttons:
        button.draw(30,7)
    for button in self.buttonswhile:
        button.draw(5, 5)
    #self.screen.blit(self.player, (self.rect.x, self.rect.y))
    for obj in self.map:
        obj.drawpic()
    for obj in self.wall:
        obj.drawpic()
    for money in self.monee:
        money.drawpic()
```

```
    self.fin.drawpic()
    self.player.drawpic()

    self.clock.tick(40)
    pygame.display.update()
```

Додаток Й

Метод перевірки програшу та виграшу

```
def check(self):
    for obj in self.wall:
        if self.player.colliderect(obj):
            self.player.rect.x = self.x
            self.player.rect.y = self.y
            message =["Спробуй знову","Ти можеш краще","Давай ще
раз",]
            self.show_message(message[random.randint(0,2)])
            time.sleep(2)  # Задержка на 2 секунды
            self.update()  # Обновляем экран, чтобы убрать
сообщение
            self.direction = 0
            self.rotate(90)
            return True

    if self.player.colliderect(self.fin) and self.go:
        self.show_message("Win money "+str(self.money))
        time.sleep(2)  # Задержка на 2 секунды
        self.player.rect.x = self.x
        self.player.rect.y = self.y
        self.direction = 0
        self.rotate(90)
        self.update()  # Обновляем экран, чтобы убрать сообщение

        return True
    for money in self.monee:
        if len(self.monee )!=0 and self.player.colliderect(money):
            self.money+=1
            self.monee.remove(money)
            del money
```

Додаток К

Метод для відображення тексу перемоги та програшу

```
def show_message(self, text):
    # Создаём поверхность с сообщением
    font = pygame.font.SysFont('verdana', 50)
    message = font.render(text, True, (255, 0, 0))  # Красный
текст
    text_rect = message.get_rect(center=(gamewin[0] / 2,
gamewin[1] / 2))
```

```
    # Определяем размеры и позицию прямоугольника для фона
    padding = 20  # Отступы вокруг текста
    background_rect = pygame.Rect(
        text_rect.left - padding,
        text_rect.top - padding,
        text_rect.width + 2 * padding,
        text_rect.height + 2 * padding
    )

    # Рисуем прямоугольник с фоном
    background_color = (255, 255, 255)  # Черный цвет фона
    pygame.draw.rect(self.screen, background_color,
background_rect)

    # Отображаем текст поверх прямоугольника
    self.screen.blit(message, text_rect)
    pygame.display.update()
```

Додаток Л

Метод для створення списку об’єднаних користувачем блоків

```
def conectblock(self):
    algoritm = []
    block1 = self.additional_buttons[0]
    for i in range(len(self.additional_buttons)):
        flag = True
        for block2 in self.additional_buttons:
            if block1.rect.bottom == block2.rect.top:
                algoritm.append(block2)
                block1 =
self.additional_buttons[self.additional_buttons.index(block2)]
                flag = False
        if flag:
            return algoritm
    return  algoritm
```

Додаток М

Метод який починає запуск користувацького алгоритму

```
def start(self):
    self.go = False
    algoritm =  self.conectblock()

    x = Cycle(iter(algoritm),1)
    x.function()
    self.go = True
    self.check()
```

Додаток Н

Рух робота прямо

```
def forward(self):
    if self.direction % 360 == 90:
        x = 1
        y = 0
    elif self.direction % 360 == 180:
        x = 0
        y = 1
    elif self.direction % 360 == 270:
        x = -1
        y = 0
    elif self.direction % 360 == 0:
        x = 0
        y = -1
    for i in range(50):
        self.player.rect.x += x
        self.player.rect.y += y
        self.update()
    self.check()
    time.sleep(0.3)
```

Додаток О

Метод поворотів робота

```
def rotate(self,direction ):
    self.direction += direction
    if self.direction % 360 == 90:
        #self.player =
pygame.transform.scale(pygame.image.load("право.png"), (50, 50))
        self.player.setpic("роботправо.png")
    elif self.direction % 360 == 180:
        #self.player =
pygame.transform.scale(pygame.image.load("низ.png"), (50, 50))
        self.player.setpic("роботниз.png")
    elif self.direction % 360 == 270:
        #self.player =
pygame.transform.scale(pygame.image.load("лево.png"), (50, 50))
        self.player.setpic("роботлево.png")
    elif self.direction % 360 == 0:
        #self.player =
pygame.transform.scale(pygame.image.load("верх.png"), (50, 50))
        self.player.setpic("роботверх.png")
    self.update()
```

Додаток П

Клас для створення і керування кнопкою циклу

```
class Whilebut(Label):
    def __init__(self, mw, x, y, width, height, color, function,):
        super().__init__( mw, x, y, width, height, color,
function)
        self.count = 1
        self.button_whiletxt = Label(self.mw, x + 35, y, 10,
10,color, None, )
        self.button_whiletxt.set_text("1")

        self.button_whileup = Label(self.mw, x + 80, y, 20,
30,(255, 250, 0), self.plus, )
        self.button_whileup.set_text("+")

        self.button_whiledown = Label(self.mw, x, y, 20,30, (255,
250, 0), self.minus, )
        self.button_whiledown.set_text("-")

    def plus(self):
        self.count +=1
        self.button_whiletxt.set_text(str(self.count))
    def minus(self):
        self.count -=1
        if self.count <=0:
            self.count = 1
        self.button_whiletxt.set_text(str(self.count))
    def draw(self, shift_x=0, shift_y=0):
        super().draw(shift_x,shift_y)

        self.button_whiletxt.rect.x = self.rect.x + 35
        self.button_whiletxt.rect.y = self.rect.y
        self.button_whileup.rect.x = self.rect.x + 80
        self.button_whileup.rect.y = self.rect.y
        self.button_whiledown.rect.x = self.rect.x
        self.button_whiledown.rect.y = self.rect.y


        self.button_whiletxt.draw(35,7)
        self.button_whileup.draw(5,5)
        self.button_whiledown.draw(5,5)
```

Додаток Р

Клас для опрацювання користувацьких циклів

```
class Cycle():
    def __init__(self, buttons, iters):
        self.iters = iters
        self.res = self.parse(buttons)
    def function(self):
        for j in range(self.iters):
            for i in self.res:
                i.function()
    def parse(self, buttons):
        res = []
```

```
        for i in buttons:
            if i.text == "repeat":
                x = Cycle(buttons, i.count)
                res.append(x)

            elif i.text == "  end":
                return res
            else:
                res.append(i)
        return res
```